

AD-A269 945



REPORT DOCUMENTATION PAGE		Form Approved MAY 1962 EDITION	
1. AGENCY USE ONLY (Leave blank)		2. REPORT DATE	REPORT TYPE AND DATES COVERED
			FINAL/01 DEC 91 TO 31 MAR 93
4. TITLE AND SUBTITLE		5. AUTHOR(S)	
UUGM CODE DEVELOPMENT (U)		6775/DARPA F49620-89-C-0087	
6. AUTHOR(S)		7. PERFORMING ORGANIZATION	
Professor S. Eidelman		Science Applications International Corp 1710 Goodridge Drive, MS 2-3-1 McLean VA 22102	
9. SPONSORING MONITORING AGENCY NAME(S) AND ADDRESS(ES)		10. SPONSORING MONITORING AGENCY REPORT NUMBER	
AFOSR/NM 110 DUNCAN AVE, SUITE B115 BOLLING AFB DC 20332-0001		AEOSR-TR-0002 F49620-89-C-0087	
11. SUPPLEMENTARY NOTES			
<p>S DTIC ELECTE D E SEP 28 1993</p>			
12a. DISTRIBUTION AVAILABILITY STATEMENT		12b. DISTRIBUTION CODE	
APPROVED FOR PUBLIC RELEASE: DISTRIBUTION IS UNLIMITED		UL	
13. ABSTRACT (Maximum 2000 words)			
<p>The primary objective of SAIC was to develop an unstructured grid algorithm and code that dynamically adapts to the computed solution of the time dependent Euler equations of gasdynamics in two and three spatial dimensions. Important requirements that were imposed on the algorithm were: robustness, accuracy, efficiency, flexibility, and adaptability. The main research and code development effort was focused on achieving these objectives; extensive testing and code validation effort was undertaken to demonstrate the code's performance for realistic CFD problems. The method is accurate in all flow regimes from subsonic to hypersonic. The main achievement was the development of the AUGUST code (Adaptive Unstructured Grid Upwind Second Order for Triangles). AUGUST is implemented for solution of Euler's equations on dynamically adaptive triangular or tetrahedral grids. The code fully implements the Second-Order Godunov method, allowing accurate and robust numerical solution of Euler equations of gas dynamics. A new method was developed for Direct Dynamic Grid Refinement (DDR). The AUGUST code was also implemented for multiphase, multicomponent flows. A combined structured/unstructured version of the AUGUST code was also developed. The AUGUST code was extensively validated for a wide range of problems and has proven to be a robust tool.</p>			
14. SUBJECT TERMS		15. NUMBER OF PAGES	
		255	
		16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	SAR (SAME AS REPORT)

93-22349



UUGM CODE DEVELOPMENT



Science Applications International Corporation
An Employee-Owned Company

SAIC is a leading provider of
defense and intelligence

AIR FORCE
...
...
...
...
...

UUGM CODE DEVELOPMENT



Science Applications International Corporation
An Employee-Owned Company

Accession For	
NTIS CRA&I	<input checked="" type="checkbox"/>
DTIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By	
Distribution /	
Availability Codes	
Dist	Avail and/or Special
A-1	

DTIC QUALITY INSPECTED 3

UUGM CODE DEVELOPMENT

SAIC Final Report #SAIC-93/1152

Final Report for work accomplished under
AFOSR Contract #F49620-89-C-0087 during period
15 October 1990 through 30 November 1992

Contributors:
Shmuel Eidelman
William Grossmann
Isaac Lottati
Xiaolong Yang
Marty Fritts
Adam Drobot
Michael Kress
Aaron Friedman

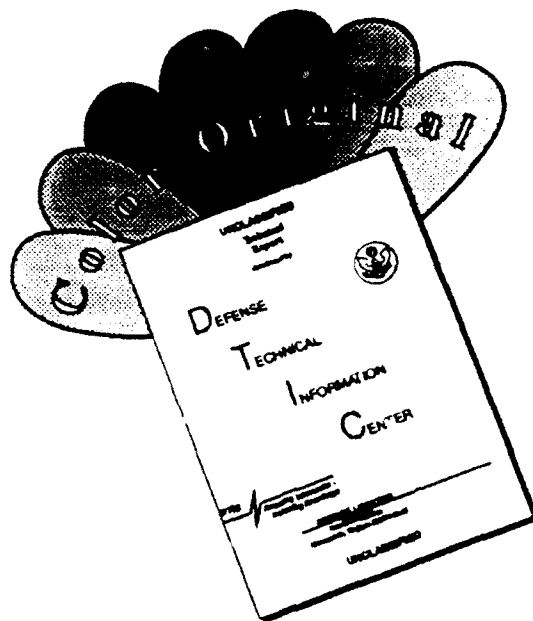
Prepared by:
Science Applications International Corporation
Applied Physics Operation
1710 Goodridge Drive, MS 2-3-1
McLean, VA 22102

Sponsored by
Defense Advanced Research Projects Agency
DARPA Order No. 6775
Monitored by AFOSR Under Contract No. F49620-89-C-0087

July 26, 1993

The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Defense Advanced Research Projects Agency or the U.S. Government.

DISCLAIMER NOTICE



THIS DOCUMENT IS BEST QUALITY AVAILABLE. THE COPY FURNISHED TO DTIC CONTAINED A SIGNIFICANT NUMBER OF COLOR PAGES WHICH DO NOT REPRODUCE LEGIBLY ON BLACK AND WHITE MICROFICHE.

EXECUTIVE SUMMARY

This progress report documents the effort conducted at SAIC from 15 October 1990 through 31 May 1993, under DARPA and AFOSR contract #F49620-89-C-0087 entitled "UUGM Code Development".

Scope of Research

The primary objective of SAIC was to develop an unstructured grid algorithm and code that dynamically adapts to the computed solution of the time dependent Euler equations of gasdynamics in two and three spatial dimensions. Important requirements that were imposed on the algorithm were: robustness, accuracy, efficiency, flexibility, and adaptability. The main research and code development effort was focused on achieving these objectives; extensive testing and code validation effort was undertaken to demonstrate the code's performance for realistic CFD problems. The method is accurate in all flow regimes from subsonic to hypersonic.

Achievements

The main achievement was the development of the AUGUST code (Adaptive Unstructured Grid Upwind Second Order for Triangles). AUGUST is implemented for solution of Euler's equations on dynamically adaptive triangular or tetrahedral grids. The code fully implements the Second-Order Godunov method, allowing accurate and robust numerical solution of Euler equations of gas dynamics.

A new method was developed for Direct Dynamic Grid Refinement (DDR). This method allows grid refinement in arbitrary regions of the computational domain, using only one level of undirectness in the logical data structure. The DDR is an integral part of the AUGUST solver and allows manipulation of the grid as a part of the solution. The adapted grid is not only more refined in the adaptation regions of the flow but is also improved structurally due to a refinement algorithm.

The AUGUST code was also implemented for multiphase, multicomponent flows. We used a multiple-fluid description, where a separate set of conservation laws is used to describe every flow component. In our approach Lagrangian tracers are used to describe sparse or discrete flow components that do not form a continuum. Use of unstructured triangular grids allows adjustment of the grid resolution to the accuracy requirements in the flow subdomains.

A combined structured/unstructured version of the AUGUST code was also developed. Following this approach the unstructured adaptive grid is used only in the flow regions requiring adaptation or description of the complex geometry elements. The structured grid is used to simulate the larger part of the computational domain. This approach has allowed us to capitalize on the advantages of both structured and unstructured grid approaches. Using the structured/unstructured grid version of the

AUGUST code, we simulated the shock wave focusing problem for the reflector used for extracorporeal shock-wave lithotripsy. In this simulation, we showed that the solution smoothly transits through the interfaces between the grids, maintaining the same accuracy and resolution.

The AUGUST code was extensively validated for a wide range of problems and has proven to be a robust tool. The code was initiated at the start of the UUGM project and has now evolved into a production code that is used for many applied problems. The list of applications includes potential flow past an ellipse, hypersonic flow past a flat plate, shock diffraction over single and double wedges, mine explosions under vehicles, pulsed detonation engines, shock focusing in air, and nonideal airburst in multiphase media. The code has shown the required robustness and insensitivity to the initial user specified grid. The number of nodes required to obtain a high-quality solution is significantly smaller than for structured grid codes. This is particularly true for transient problems with complicated flows having discontinuities.

It is important to note that the AUGUST code obtains a high resolution solution with no "knobs." The various flow regimes, except those requiring a different definition of boundary and initial conditions, were simulated using the same code.

TABLE OF CONTENTS

1. INTRODUCTION	1
1.1 RECENT CFD DEVELOPMENT	1
1.2 UNSTRUCTURED MESHES IN COMPLEX GEOMETRIES	1
2. UUGM: UNIVERSAL CFD SIMULATION ENVIRONMENT	2
2.1 MATHEMATICAL MODEL AND INTEGRATION ALGORITHM	3
2.2 MULTIPHASE MULTICOMPONENT REACTIVE FLOW	9
2.3 DIRECT DYNAMIC REFINEMENT METHOD FOR UNSTRUC- TURED TRIANGULAR GRIDS	13
2.4 STRUCTURED/UNSTRUCTURED COMPOSITE GRIDS	16
2.5 THREE-DIMENSIONAL CAPABILITY	19
3. APPLICATIONS	22
3.1 POTENTIAL FLOW OVER AN ELLIPSE	23
3.2 HYPERSONIC FLOW PAST A FLAT PLATE	24
3.3 SHOCK ON WEDGE WITH ADAPTIVE GRIDDING	26
3.4 MINE EXPLOSION UNDER VEHICLE	30
3.5 PULSED DETONATION ENGINE	36
3.6 SHOCK FOCUSING IN AIR	36
3.7 NONIDEAL AIRBURST IN MULTIPHASE MEDIA	39
3.8 FLOW IN THE SARL WIND TUNNEL	39
3.9 SHOCK ON DOUBLE WEDGE	42
3.10 SUPERSONIC SPRAY COATING DEVICES	46
3.11 DUSTY FLOW OVER A CYLINDER	49
3.12 IMAGE PROCESSING	52
3.13 DETONATION IN A MULTIPHASE MEDIUM	55
4. CONCLUSIONS	60
REFERENCES	61
APPENDIX A: Code Description	
APPENDIX B: Listings	
APPENDIX C: Copies of Publications	

LIST OF ILLUSTRATIONS

Figure 2.1.1	Representative triangular cell in the mesh showing fluxes and projected values.	5
Figure 2.1.2	Density profile comparison between analytical results and results obtained by applying the second-order Godunov algorithm using structured or unstructured grids.	8
Figure 2.3.1	Illustration of the grid refinement process.	14
Figure 2.3.2	Illustration of the grid coarsening process.	15
Figure 2.4.1	A possible candidate configuration for hybrid structured/unstructured domain decomposition.	18
Figure 2.4.2	Hybrid structured/unstructured grid used to simulate ellipsoidal reflector, showing adaptation to curved boundaries.	18
Figure 2.5.1	An elongated tetrahedron can be refined using smaller tetrahedra that are nearly regular.	20
Figure 2.5.2	Points used to define structure of vehicle.	21
Figure 2.5.3	Tetrahedral grid generated by Finite Octree method.	21
Figure 3.1.1	The grid used for simulating the flow over an ellipse.	24
Figure 3.2.1a	Grid for simulation of hypersonic flow over a flat plate.	25
Figure 3.2.1b	Second order solution for a flat plate, pressure contours. Mach = 32: 5509 grid vertices: $P_{\max} = 5.0 \times 10^4$ Pa, $P_{\min} = 98.7$ Pa.	26
Figure 3.3.1a	Density contours at early time for shock in planar channel (M = 8.7, wedge angle = 27°).	27
Figure 3.3.1b	Grid at early time for shock in planar channel (M = 8.7, wedge angle = 27°).	28
Figure 3.3.2a	Density contours at intermediate time for shock in planar channel (M = 8.7, wedge angle = 27°).	28
Figure 3.3.2b	Grid at intermediate time for shock in planar channel (M = 8.7, wedge angle = 27°).	29
Figure 3.3.3a	Density contours at late time for shock in planar channel (M = 8.7, wedge angle = 27°).	29
Figure 3.3.3b	Grid at late time for shock in planar channel (M = 8.7, wedge angle = 27°).	30
Figure 3.4.1	Two views of interaction between mine blast and M925 cargo truck: pressure contours at $t = 0.574$ msec.	31
Figure 3.4.2	Blast - plow interaction: pressure contours in initial stage.	33
Figure 3.4.3	Blast - plow interaction: pressure contours in advanced stage.	33
Figure 3.4.4	Structural response of the plow to blast load: a) $t = 0$; b) $t = 200$ msec; c) $t = 400$ msec; d) $t = 600$ msec.	34
Figure 3.5.1	Pulsed detonation engine simulation: flow tracers.	35
Figure 3.6.1a	Hybrid structured/unstructured grid used for numerical simulation of ellipsoidal reflector.	37
Figure 3.6.1b	A schematic drawing of the center cross section of the ellipsoidal reflector.	37

Figure 3.6.2	Pressure contours at time $t = 1.21 \times 10^{-6}$ sec showing the incident wave as reflected from the reflector wall.	38
Figure 3.6.3	Pressure contours at time $t = 8.41 \times 10^{-4}$ sec showing the stage at which the maximum focused pressure is obtained.	38
Figure 3.7.1	Formation of a radiative cloud. Multiphase simulation.	40
Figure 3.8.1	The unstructured grid used to simulate the SARL wind tunnel.	41
Figure 3.8.2	The pressure contours from the simulation of the SARL wind tunnel.	41
Figure 3.9.1	Experimental interferogram of a shock hitting a 45° corner at $M = 2.85$.	42
Figure 3.9.2	Interaction of a Mach 8.7 planar shock wave with a 27° double ramp: Mach reflection stage.	43
Figure 3.9.3	Interaction of Mach 8.7 planar shock wave with a 27° double ramp: start of the diffraction stage.	44
Figure 3.9.4	Interaction of Mach 8.7 planar shock wave with a 27° double ramp: shock diffraction stage.	45
Figure 3.10.1	The figure shows the initial computational grid for the jet spray simulation demonstration. Shown are the nozzle, injection region and target surface depicted as a flat plate with perforations, oriented perpendicular to the mean spray flow. The boundary conditions used for the sample simulation were: $V_g = 1000$ m/sec, $\rho_g = 0.1$ kg/m ³ , $T_g = 3500$ K at the inlet of the reactor nozzle; $V_g = 1500$ m/sec, $\rho_g = 0.3$ kg/m ³ , $T_g = 1500$ K, $V_p = 1500$ m/sec, $T_p = 1500$ K, $N_p = 2000$ at the inlet of the reactor nozzle.	46
Figure 3.10.2	Lagrangian marker particles are shown in color representing the evolution of injected particle temperature as a function of particle position and time in the jet spray stream.	47
Figure 3.10.3	Gas temperature contours in the jet spray stream. The maximum temperature is 3500°K and the minimum is 600°K .	48
Figure 3.10.4	Gas density contours in the jet spray stream. The injected stream and the main flow mix poorly. The diamond patterns describe the shock wave pattern resulting from the flow's overexpansion.	48
Figure 3.10.5	Pressure contours in the jet spray stream. The diamond patterns show that supersonic flow is maintained near the vicinity of the target surface.	48
Figure 3.11.1	Comparison for $M_s = 2.8$ pure-gas flow: (a) interferogram from experiment; (b) density contours from present calculation.	50
Figure 3.11.2	Density contours for the case $M_s = 2.8$, $\rho_p = 0.25$ kg/m ³ , $r_p = 10\mu\text{m}$ at two different times: (a) particle density at t_1 , (b) gas density at t_1 ; (c) particle density at t_2 , (d) gas density at t_2 .	51
Figure 3.11.3	Density contours for the case $M_s = 2.8$, $\rho_p = 0.76$ kg/m ³ , for two different particle sizes: (a) particle density and (b) gas density for $r_p = 10\mu\text{m}$; (c) particle density and (d) gas density for $r_p = 50\mu\text{m}$.	51
Figure 3.12.1	Edge enhancement for a sinusoidal distribution without noise.	56
Figure 3.12.2	Edge enhancement for a sinusoidal distribution with 10% intensity random noise.	54

Figure 3.12.3	Edge enhancement for a sinusoidal distribution with 50% intensity random noise.	55
Figure 3.12.4	Edge enhancement for a sinusoidal distribution with 100% intensity random noise.	55
Figure 3.13.1	Computational domain and boundary conditions.	57
Figure 3.13.2	Explosive initially localized in 2.5-cm layer at constant density of 100 kg/m ³ . Density in the cloud is 0.75 kg/m ³ . (a), (b), and (c) are gas pressure, gas density, and particle density at 66 μsec, respectively.	58
Figure 3.13.3	Particle density distributed in layer in accordance with the fourth power of height. Gas pressure, temperature, and particle density at 55 μsec, respectively.	59

1. INTRODUCTION

1.1 RECENT CFD DEVELOPMENT

Computational fluid dynamics (CFD) development over the past twenty years has truly been outstanding. The recent CFD developments that are particularly important are: 1) advances in flow solvers in all the regimes of fluid flow (very low speed and subsonic flows, transonic flow, supersonic and hypersonic flows), 2) advances in unstructured adaptive gridding techniques and, 3) advances in chemical and particle kinetic modeling for fluid flows. Developments in graphics and visualization, construction of graphical user interfaces (GUIs) and advances in large database management have also played an important role in the scale and complexity of problems that can now be realistically simulated by CFD techniques. SAIC has been involved in all aspects of these developments and is on the forefront of CFD technology development.

DARPA, NASA, DNA and DOE have for the most part been the largest benefactors of CFD development, and each agency today is actively pursuing CFD applications to real problems. Full 3-D unsteady flows about military and commercial aircraft are routinely simulated to assess aerodynamic performance characteristics, and where it used to require several hundred hours of CRAY CPU time it now takes minutes to an hour on a supercomputer or a like time on workstations, depending on the specifics of the problem being solved. The U.S. Marine Corps' latest initiative in the development of blast (due to land mines) resistant vehicles is being pursued successfully with the aid of full 3-D CFD simulations of land mine blast effects on truck configurations. The CFD technology developed in SAIC's UUGM contract is playing a leading role in this Marine Corps effort (see Section 3.4). Many other such examples of improvements in CFD performance exist. In view of this, it is quite appropriate to begin to transition CFD technology into other disciplines that can take advantage of realistic CFD based simulation.

1.2 UNSTRUCTURED MESHES IN COMPLEX GEOMETRIES

Current emphasis in CFD calls for solutions of applied physical problems for complex realistic geometries.¹ In addition to the inherent difficulties in describing the details of the complex three-dimensional geometry, the flow fields usually have an inhomogeneous structure. Regions of rapid change of the flow functions and chemical reactions will be embedded in regions where the flow gradients are relatively small. Accurate simulations of flows in regions with strong gradients is key to the overall accuracy of physical, chemical and biological simulations. For this reason most of the software and hardware computational resources are defined by the accuracy requirements of these flow regions and geometry of the computational domain.

Early CFD research was almost entirely concerned with the formulation the mathematical models of the flow and methods of solution. Mesh generation was regarded as secondary and meshes were developed for specific cases. During this early period very

significant improvements were made in the methods of integration of the partial differential equations of gasdynamics. Presently, as a result of steady improvement in the various integration techniques, the advantages which could be gained by using better flow solvers have become limited. On the other hand substantial progress is anticipated in the areas of grid generation and algorithm development.²

Currently, most numerical simulations employ structured meshes composed of quadrilaterals in two dimensions or hexahedra in three dimensions. However, it has become evident that the quadrilateral structured grids cannot satisfy the requirements of large scale numerical simulations over complex geometries in three dimensions. The physics of the flow about a complete aircraft is extremely complex. Yet the flow in many distinct regions and regimes may be represented by fairly well-known physical theories. Vortices shed by lifting surfaces are confined to fairly thin wake regions. Exhaust plumes can be initially approximated by regular bounding surfaces. Flow disturbances due to shocks are confined to thin discontinuities. Boundary layers are restricted to near-wall regions. Each of these flow regions requires different theories, different resolution and different numerical algorithms. This diversity of computational requirements cannot be satisfied by the quadrilateral structured grids.

Recently proposed alternatives to quadrilateral grids use triangles in two dimensions and tetrahedra in three dimensions. For these grids the mesh will generally lose its structure, allowing a new degree of flexibility in treating complex geometries.^{3,4} Unstructured grids can relatively easily be adapted to follow flow features, thereby increasing the solution accuracy. The result has been the development of adaptive refinement techniques which have been used with great success for two dimensional simulations on unstructured triangular grids. These methods have resulted in the resolution of previously difficult details in the evolving flows over complex configurations.

However, it is not a trivial task to adapt this approach to three-dimensional simulations. One of the problems is the generation of the adaptive grid. Since the grid is constructed from the volume elements (tetrahedra) the moving front is made up of a surface of triangular faces. It should be noted that this moving front can and will change its shape during the computation as time evolves. It is necessary to take care when determining the intersections of planar faces, and to ensure that no overlapping of tetrahedra occurs.

2. UUGM: UNIVERSAL CFD SIMULATION ENVIRONMENT

The Ultimate Unstructured Grid Method (UUGM) represents a new approach to the computational domain discretization. The principal advantage of the method is most apparent for simulations of complicated flow regimes with physical and chemical processes over bodies with complex geometries in three dimensions.

The usual technique employed in regridding is called hierarchical dynamic refinement (H refinement). The idea here is to retain a history of the original grid and the

subdivisions needed to change it into the current grid, so that it is always possible to retrace these steps and get back to previous grids. While this feature is useful in modeling reversible processes, it is generally unnecessary, and it increases overhead costs. Our implementation (Direct Dynamic Refinement) is Markovian, in the sense that the way regridding is done depends only on the current grid and flow conditions.

The other distinguishing feature is the use of the Second-order Godunov method to solve the Euler equations of gasdynamics. The philosophy behind it is to treat the local values of the dependent variables at every point on the grid as initial conditions for a Riemann problem, and to use the resultant solution of that problem to calculate the fluxes of material, momentum, and energy from one cell to the rest. Previous implementations of this method were confined to structured meshes.

2.1 MATHEMATICAL MODEL AND INTEGRATION ALGORITHM

We consider a system of two-dimensional Euler equations written in conservation law form as

$$\frac{\partial U}{\partial t} + \frac{\partial F}{\partial x} + \frac{\partial G}{\partial y} = 0, \quad (2.1)$$

where

$$U = \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ e \end{pmatrix}, \quad F = \begin{pmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ u(e+p) \end{pmatrix}, \quad G = \begin{pmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ v(e+p) \end{pmatrix}.$$

Here u, v are the x, y velocity vector components, p is the pressure, ρ is the density and e is total energy of the fluid. We assume that the fluid is an ideal gas and the pressure is given by the equation of state,

$$p = (\gamma - 1) \left[e - \left(\frac{\rho}{2} \right) (u^2 + v^2) \right], \quad (2.2)$$

where γ is the ratio of specific heats and is typically taken as 1.4 for air. It is assumed that an initial distribution of the fluid parameters is given at $t = 0$, and the boundary conditions defining a unique solution are specified for the computational domain.

The system of governing equations in Eq. (2.1) can be written as

$$\frac{\partial U}{\partial t} + \nabla \cdot Q = 0, \quad (2.3)$$

where Q represents the convective flux vector. Integrating Eq. (2.3) over space and using Gauss' theorem produces the expression

$$\frac{\partial}{\partial t} \int_{\Omega} U dA + \oint_{\partial\Omega} Q \cdot dl = 0, \quad (2.4)$$

where $dl = n dl$, n is the unit normal vector in the outward direction, and dl is the element of length on the boundary of the domain. Here Ω is the domain of computation and $\partial\Omega$ is the boundary of this domain.

We seek a solution to the system of Eq. (2.1) in the computational domain, which is decomposed in part into triangles with arbitrary connectivity and in part into rectangles using a logically structured grid. We use the advantage of the unstructured grid (Refs. 5-8) to describe the curved boundary of the computational domain and areas that need increased local resolution; this covers a small part of the total computational domain. The largest area of the computational domain is decomposed by the structured grid. The numerical technique for solving Euler's equation on an unstructured grid is described in Refs. 9-11, and the technique for the structured grid is described in Ref. 9. These numerical techniques apply some of the ideas that were introduced in Refs. 13-14. The structured and unstructured codes apply the center-based formulation, i.e., the primitive variables are defined in the center of the cell, which makes the cell the integration volume, while the fluxes are computed across the edges of the cell. The basic algorithmic steps of the Second-Order Godunov method can be defined as follows:

1. Find the value of the gradient at the baricenter of the cell for each gasdynamic parameter U_i ;
2. Find the interpolated values of U at the edges of the cell using the gradient values;
3. Limit these interpolated values based on the monotonicity condition;¹³
4. Subject the projected values to the characteristics constraints;¹⁴
5. Solve the Riemann problem applying the projected values at the two sides of the edges;
6. Update the gas dynamic parameter U according to the conservation equations (1) applying to the fluxes computed and the current timestep.

As was recommended in Ref. 11, we prefer the version based on triangle centers over the vertex-based version of the code. For the same unstructured grid, a center-based algorithm will result in smaller control volumes than a vertex-based. In addition, for the Second-Order Godunov solver, implementation of the boundary conditions is more straightforward and accurate for the center-based algorithm than in the vertex-based version. These two factors, along with the effects of grid connectivity, strongly affect the algorithm accuracy and performance and are the main reasons for the superiority of the center-based version over the vertex version.

Equation (2.4) can be discretized for each element (cell) in the domain

$$\frac{(U_i^{n+1} - U_i^n)}{\Delta t} A_i = \sum_{j=1}^M Q_j^n \cdot n_j \Delta L_j, \quad (2.5)$$

where A_i is the area of the cell; Δt is the marching timestep; U_i^n and U_i^{n+1} are the primitive variables at the center of the cell at time n and at the updated $(n + 1)$ st timestep; Q_j is the value of the fluxes across the boundaries on the circumference of the cell where n_j is the unit normal vector to the boundary edge j , and ΔL_j is the length of the boundary edge j . The fluxes Q_j^n are computed applying the Second-Order Godunov algorithm, and Eq. (2.5) is used to update the physical primitive variables u_i according to computed fluxes for each marching timestep Δt . The marching timestep is subjected to the Courant-Friedrichs-Lewy (CFL) constraint.

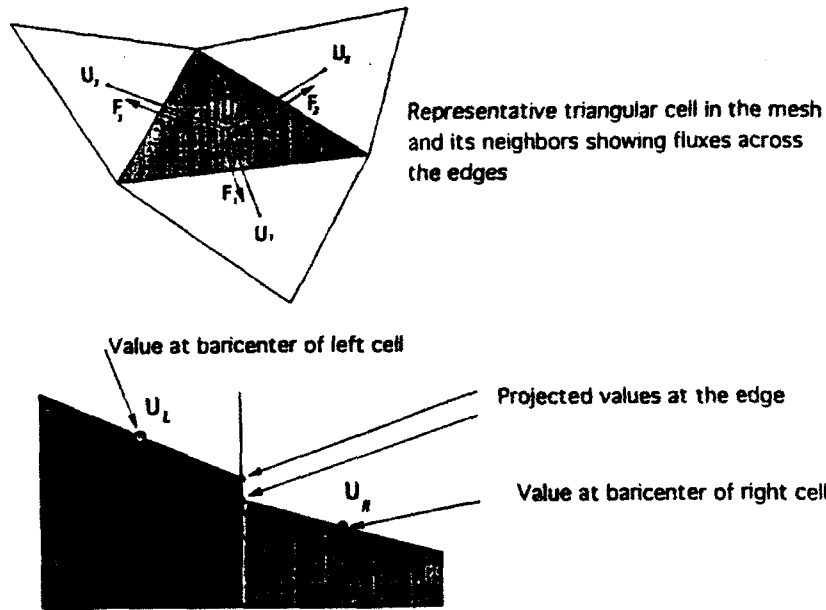


Figure 2.1.1 Representative triangular cell in the mesh showing fluxes and projected values

To obtain second-order spatial accuracy, the gradient of each primitive variable is computed in the baricenter of the cell. This gradient is used to define the projected values of primitive variables at the two sides of the cell edge, as shown in Fig. 2.1.1. The gradient is approximated by a path integral

$$\int_{\Omega} \nabla U_i^{\text{cell}} dA = \oint_{\partial\Omega} U_i^{\text{edge}} dl. \quad (2.6)$$

The notation is similar to the one used for Eq. (2.5), except that the domain Ω is a single cell and U_i and U_i are values at the baricenter and on the edge respectively. The gradient is estimated as

$$\nabla U_i^{\text{cell}} = \frac{1}{A} \sum_{j=1}^3 U_i^{\text{edge}} n_j \Delta L_j, \quad (2.7)$$

where U_j^{edge} is an average value representing the value of primitive variable for edge j .

The gradients that are computed at each baricenter are used to project values for the two sides of each edge by piecewise linear interpolation. The interpolated values are subjected to monotonicity constraints.³ The monotonicity constraint ensures that the interpolated values do not create new extrema.

The monotonicity limiter algorithm can be written in the following form

$$U_{\text{proj}}^{\text{edge}} = U_i^{\text{cell}} + \phi \nabla U_i \cdot \Delta \mathbf{r} \quad (2.8)$$

where $\Delta \mathbf{r}$ is the vector from the baricenter to the point of intersection of the edge with the line connecting the baricenters of the cells over the two sides of this edge. Here ϕ is the coefficient that limits the gradient ∇U_i .

First we compute the maximum and minimum values of the primitive variable in the i th cell and its three neighboring cells that share common edges (see Fig. 2.1.1):

$$\left. \begin{aligned} U_{\text{cell}}^{\text{max}} &= \max (U_k^{\text{cell}}) \\ U_{\text{cell}}^{\text{min}} &= \min (U_k^{\text{cell}}) \end{aligned} \right\} k = i, 1, 2, 3. \quad (2.9)$$

The limiter can be defined as

$$\phi = \min \{1, \phi_k^{lr}\}, k = 1, 2, 3 \quad (2.10)$$

where the superscript lr stands for left and right of the three edges (6 combinations altogether); ϕ_k^{lr} is defined by

$$\phi_k^{lr} = \frac{[1 + \text{sgn}(\Delta U_k^{lr})] \Delta U_{\text{cell}}^{\text{max}} + [1 - \text{sgn}(\Delta U_k^{lr})] \Delta U_{\text{cell}}^{\text{min}}}{2 \Delta U_k^{lr}}, k = 1, 2, 3 \quad (2.11)$$

where $\Delta U_k^{lr} = \nabla U_i^{lr} \cdot \Delta \mathbf{r}_k$ and

$$\left. \begin{aligned} \Delta U_{\text{cell}}^{\text{max}} &= U_{\text{cell}}^{\text{max}} - U_i^{\text{cell}} \\ \Delta U_{\text{cell}}^{\text{min}} &= U_{\text{cell}}^{\text{min}} - U_i^{\text{cell}} \end{aligned} \right\} \quad (2.12)$$

To obtain second-order accuracy in time and space, we subject the projected values of the left and right side of the cell edge to characteristic constraints following Ref. 4. The one-dimensional characteristic predictor is applied to the projected values at the half timestep $t^n + \Delta t / 2$. The characteristic predictor is formulated in the local system of coordinates for the one-dimensional Euler equation. We illustrate the implementation of

the characteristic predictor in the direction of the unit vector \mathbf{n}_c . The Euler equations for this direction can be written in the form

$$W_i + A(W)W_{nc} = 0; \quad (2.13)$$

$$W = \begin{Bmatrix} \tau \\ u \\ p \end{Bmatrix}; \quad A(W) = \begin{pmatrix} u & -\tau & 0 \\ 0 & u & \tau \\ 0 & \rho c^2 & u \end{pmatrix}, \quad (2.14)$$

where $\tau = \rho^{-1}$, ρ denotes density, u , p are the velocity and pressure. The matrix $A(W)$ has three eigenvectors ($l\#, r\#$) (l for left and r for right, where $\#$ stands for $+$, 0 , $-$) associated with the eigenvalues $\lambda^+ = u + c$, $\lambda^0 = u$, $\lambda^- = u - c$.

An approximation of the value projected to an edge, accurate to second order in space and time, can be written as

$$\begin{aligned} W_{i+\Delta r}^{n+1/2} &\approx W_i^n + \frac{\Delta t}{2} \frac{\partial W}{\partial t} + \Delta r \frac{\partial W}{\partial r_{nc}} \\ &\approx W_i^n + \left[\Delta r - \frac{\Delta t}{2} A(W_i) \right] \frac{\partial W}{\partial r_{nc}}. \end{aligned} \quad (2.15)$$

An approximation to $W_{i+\Delta r}^{n+1/2}$ can be written as

$$W_{i+\Delta r}^{n+1/2} = W_i + \left(\Delta r_i - \frac{\Delta t}{2} (M_x M_n) n_c \right) \cdot \nabla W_i, \quad (2.16)$$

where

$$(M_x M_n) = \begin{cases} \max(\lambda_i^+, 0) & \text{for the cell on the left of the edge} \\ \min(\lambda_i^-, 0) & \text{for the cell on the right of the edge.} \end{cases} \quad (2.17)$$

The gradients applied in the process of computing the projected values at $t^n + \Delta t/2$ are subjected to the monotonicity limiter.

Following the characteristic predictor described above, the full Riemann problem is solved at the edge. The solution of the Riemann problem defines the flux $Q^{n+1/2}$ through the edge. The fluxes through the edges of triangles are then integrated (Eq. 2.5), thus giving an updated value of the variables at t^{n+1} . One of the advantages of this algorithm is that calculation of the fluxes is done over the largest loop in the system (the loop over edges) and can be vectorized or parallelized. This leads to an efficient algorithm.

We have carried out an extensive and painstaking series of tests in the course of developing and implementing the algorithm. Most of these used a standard benchmark, the exploding diaphragm or "Sod problem" (Fig. 2.1.2). In this problem two regions containing an ideal gas at different densities and pressure are separated by an infinitely thin interface (the diaphragm). A shock wave, a rarefaction wave, and a contact discontinuity propagate away from that point at different speeds when this diaphragm is instantaneously removed. The Riemann solution yields an analytical solution in terms of simple waves which can be compared with the numerical approximation.

We used this problem as a testbed to compare structured vs. unstructured grids, first-order vs. second-order Godunov schemes, schemes with and without limiters, etc. For example, Fig. 2.1.2 shows that the solution obtained with an unstructured grid is noticeably better than that obtained with a structured grid.

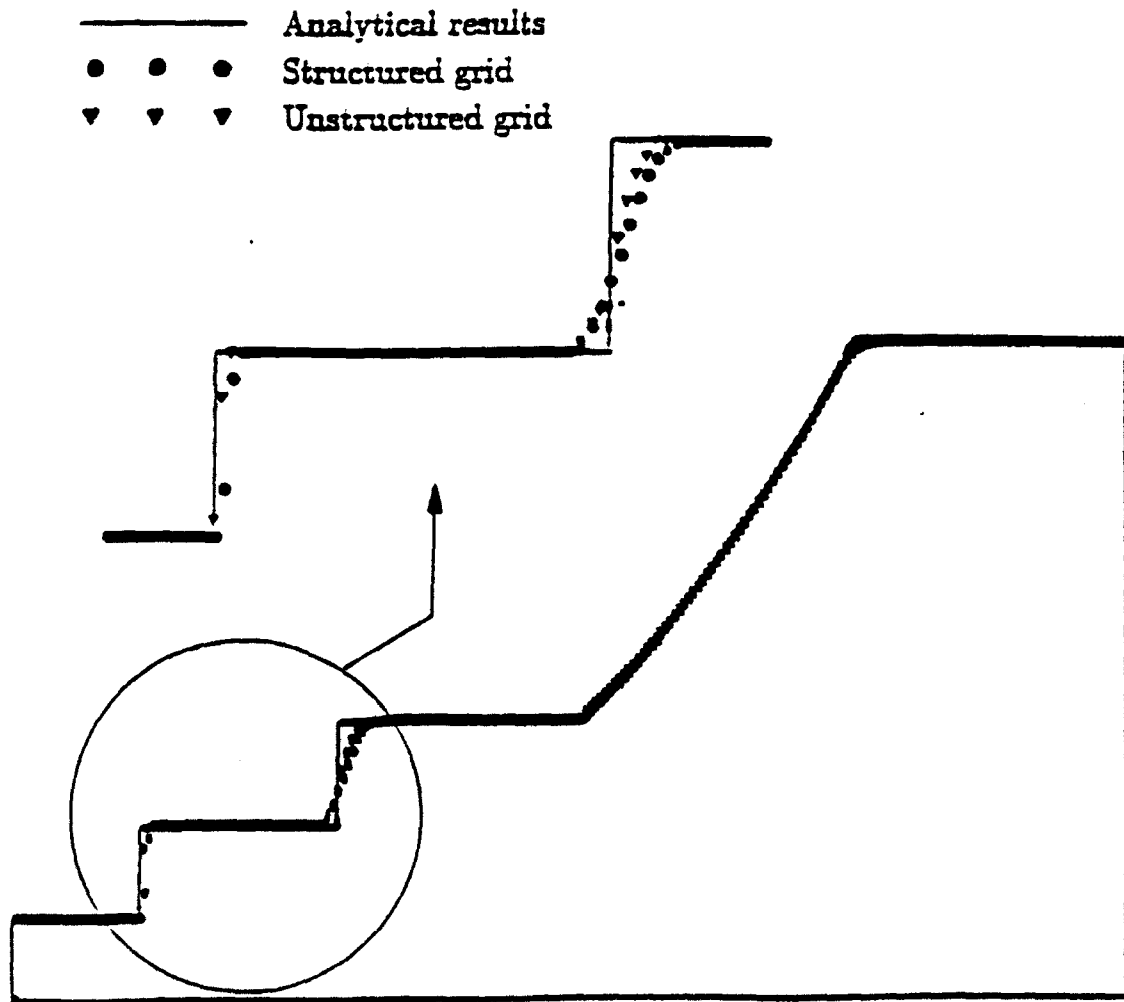


Figure 2.1.2 Density profile comparison between analytical results and results obtained by applying the second-order Godunov algorithm using structured or unstructured grids.

2.2 MULTIPHASE MULTICOMPONENT REACTIVE FLOW

Multiphase multicomponent reacting flows (MPMCRF) consist of material media (continua and particles) dispersed in a flow varying in space and time. Two basic approaches can be used to describe MPMCRFs, heterogeneous and homogeneous phase descriptions. For homogeneous mixtures one assumes that each mixture component occupies the same volume with other mixture components on an equal basis ($V_1=V_2=\dots=V_n=V$). This approach is justified for an interpenetrating mixture of gases or a dilute suspension of particles in a gas. In a heterogeneous description of a suspension, each component occupies only part of the global volume ($V_1+V_2+\dots+V_n=V$). Therefore in the mathematical description of the heterogeneous suspensions, in addition to the density of the i -th component ρ_i one needs to introduce the fractional volume of the components:

$$\phi_1 + \phi_2 + \dots + \phi_N = 1 \quad (\phi_i > 0), \quad (2.18)$$

which allows us to define the real density of each of the components as $\sigma_i = \frac{\rho_i}{\phi_i}$.

Consider a chemically reacting system containing an N -component gaseous phase and one solid particle phase. The conservation equations can be written as follows:³

Conservation of Mass

Global continuity for gaseous phase:

$$\frac{\partial \rho_g}{\partial t} + \frac{\partial}{\partial x_j} (\rho_g u_{(g)j}) = I_g. \quad (2.19)$$

Continuity of $N-1$ species or components of gaseous phase:

$$\frac{\partial Y^i}{\partial t} + \frac{\partial}{\partial x_j} [\rho_g Y^i (u_{(g)j} + V_j^i)] = \omega^i + I_g^i. \quad (2.20)$$

Continuity for solid particle phase:

$$-\frac{\partial \rho_p}{\partial t} + \frac{\partial}{\partial x_j} (\rho_p u_{(p)j}) = -I_p. \quad (2.21)$$

In the above equation of mass conservation, ρ_g is the partial gas density. The gas volume fraction is ϕ_g . The relation between partial gas density and material density σ_g is $\rho_g = \phi_g \sigma_g$. Similarly, we define the partial phase density ρ_p and material density σ_p . The relation between the two is then $\rho_p = \phi_p \sigma_p$. We assume volume conservation, which is

$$\phi_g + \phi_p = 1. \quad (2.22)$$

The species diffusion velocity V_i^j is calculated through Fick's law:

$$V_i^j = -\frac{D}{Y^j} \frac{\partial Y^j}{\partial x_i}, \quad (2.23)$$

where D is the diffusion coefficient. Finally, we assume mass conservation in all chemical reactions:

$$\sum_i^N w_i^j = 0 \quad \text{and} \quad I_p = -\sum_i^N I_g^i = -I_g. \quad (2.24)$$

Conservation of Momentum

Conservation of momentum for the gaseous phase:

$$\begin{aligned} & \frac{\partial(\rho_g u_{(g)})}{\partial t} + \frac{\partial}{\partial x_j} [\rho_g u_{(g)} u_{(g)} + \delta_{ij} \phi_g p_g] \\ &= \frac{\partial}{\partial x_j} \left[\left(\mu' - \frac{2}{3} \mu \right) \frac{\partial u_{(g)k}}{\partial x_k} \delta_{ij} + \mu \left(\frac{\partial u_{(g)j}}{\partial x_j} + \frac{\partial u_{(g)i}}{\partial x_i} \right) \right] \\ & - F_i^{(p)} + I_p u_{(p)i}. \end{aligned} \quad (2.25)$$

Conservation of momentum for the particle phase:

$$\frac{\partial(\rho_p u_{(p)})}{\partial t} + \frac{\partial}{\partial x_j} [\rho_p u_{(p)} u_{(p)} + \delta_{ij} \phi_p p_p] = \frac{\partial}{\partial x_j} (\tau_{(p)ij}) + F_i^{(p)} - I_p u_{(p)i}. \quad (2.26)$$

In the above momentum conservation equations, p_p and p_g are the pressure of the solid particle and gaseous phases respectively, $F_i^{(p)}$ represents the interaction force between the two phases, and $\tau_{(p)ij}$ is the stress tensor for the particle phase, to be determined by experimental or empirical correlations.

For the gaseous phase, the stress tensor can be written as

$$\tau_{(g)ij} = -p \delta_{ij} + \left(\mu' - \frac{2}{3} \mu \right) \frac{\partial u_k}{\partial x_k} \delta_{ij} + \mu \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right), \quad (2.27)$$

where μ is the dynamic viscosity and μ' is the second viscosity coefficient.

Conservation of Energy

The governing equation for conservation of energy for the gaseous phase is usually written

$$\begin{aligned}
& \frac{\partial [\rho_g (e_g + 0.5 u_{(g)i} u_{(g)i})]}{\partial t} + \frac{\partial}{\partial x_j} [\rho_g u_{(g)j} (e_g + 0.5 u_{(g)i} u_{(g)i}) + \phi_g p_g u_{(g)j}] \\
&= -\frac{\partial q_{(g)j}}{\partial x_j} + Q_g + \frac{\partial}{\partial x_j} \left(u_{(g)j} \left[\left(\mu' - \frac{2}{3} \mu \right) \frac{\partial u_{(g)k}}{\partial x_k} \delta_{ij} + \mu \left(\frac{\partial u_{(g)i}}{\partial x_i} \right) \right] \right) \\
&- F_{(p)i} u_{(p)i} + Q_p.
\end{aligned} \tag{2.28}$$

The equation for conservation of energy for the particle phase has the form

$$\begin{aligned}
& \frac{\partial}{\partial t} [\rho_p (C_p T_p + 0.5 u_{(p)i} u_{(p)i})] + \frac{\partial}{\partial x_j} [\rho_p u_{(p)j} (C_p T_p + 0.5 u_{(p)i} u_{(p)i}) + \phi_p u_{(p)j} p_p] \\
&= -\frac{\partial q_{(p)j}}{\partial x_j} + \frac{\partial}{\partial x_j} (u_{(p)j} \tau_{(p)j}) + F_{(p)i} u_{(p)i} - Q_p.
\end{aligned} \tag{2.29}$$

In the conservation of energy, $\frac{\partial q_{(g)j}}{\partial x_j}$ and $\frac{\partial q_{(p)j}}{\partial x_j}$ are the heat flux gradients in the j th direction in the gaseous and particle phases, respectively. Q_p is the energy source due to heterogeneous chemical reactions (between the gaseous and particle phases), plus heat transfer between the two phases. Here $Q_g = \sum_{i=1}^N (-\omega_i \Delta h_{f,i}^\circ)$ is the energy source due to homogeneous (gaseous) chemical reactions, which is defined in the chemical reaction model.

Conservation of Number of Particles

An equation for total conservation of particles is given by

$$\frac{\partial n_p}{\partial t} + \frac{\partial}{\partial x_j} (n_p u_{(p)j}) = 0. \tag{2.30}$$

Equation of State

The equation of state for all gases can be put into the generic form

$$e_g = f_g(p_g, \sigma_g, Y^1, \dots, Y^N), \tag{2.31}$$

where for an ideal gas the form is

$$e_s = \frac{P_g}{\sigma_g(\gamma_g - 1)} \quad (2.32)$$

and

$$p_g = \sigma_g R_u T_g \sum_{i=1}^N \frac{Y^i}{W^i} \quad (2.33)$$

An equation of state for the particle phase can be written in symbolic form as

$$p_p = f(\sigma_p, T_p), \quad (2.34)$$

where the exact form of Eq. (2.34) that is to be used in a numerical simulation depends on experimental data or results from physical approximations.

In the above equations, γ_g is the ratio of specific heats of the gaseous mixture and R_u is the universal gas constant.

Chemical Reaction Model

A phenomenological chemical reaction model for the gaseous phase (including M chemical reactions) has been formulated as

$$\omega^1 = W^1 \sum_{k=1}^M (v_k^{(1)} - v_k^{(1)}) B_k T^{\alpha_k} \exp\left(\frac{E_{\alpha k}}{R_u T_g}\right) \prod_{j=1}^N \left(\frac{X^j P_g}{R_u T_g}\right)^{\nu_{jk}} \quad (2.35)$$

Similarly, a phenomenological heterogeneous (for gas and particle phases) chemical reaction model can be written symbolically as

$$I_{(p)} = f(T_p, P_p, \dots), \quad (2.36)$$

and again the exact form of Eq. (2.36) will depend on experimental data or approximations from physical models.

The following nomenclature defines the symbols used in the above system of equations (2.19) - (2.36): B - chemical reaction collision frequency factor; C_s - specific heat for solid particle; e - internal energy; D - mass diffusion coefficient; $E_{\alpha k}$ - activation energy for the k th reaction; F_i - interphase force in i th direction; I - source function generated by chemical reaction; p_g - gas pressure; q_i - heat flux in the i th direction; R_u - universal gas constant; t - time; T - temperature; u_i - velocity in i th direction; V_i - species diffusion velocity in i th direction; W^i - molecular weight of i th component of gas; x_j - coordinate in j th direction; X^j - mole fraction of j th component of gas; Y^i - mass fraction of i th component of gas; α - temperature exponent of the k th reaction; γ - ratio of specific heat; λ - thermal conductivity of gas; μ - dynamic viscosity of gas; μ' - second

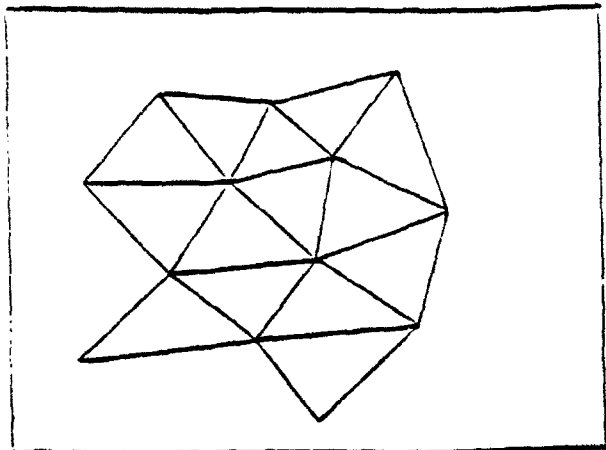
viscosity coefficient of gas; τ_{ij} - stress tensor; ω^i - mass rate of production of species i ; ρ - density; $\nu_{i,k}$ - stoichiometric coefficient for species i appearing as a reactant in the k th reaction; $\nu'_{i,k}$ - stoichiometric coefficient for species i appearing as a product in the k th reaction; ϕ - volume fraction; σ - material density. Subscripts are defined as follows: g - gas phase; p - particle phase; i,j,k - direction indexes; l - species index. Superscripts refer to species type.

The comprehensive mathematical model and system of equations given above for an MPMCRF simulation of advanced material synthesis processes is based on volume averaging, assuming that each phase or component can be described by continuous flow. Such averaging leads to a loss of information that can be recovered by appropriate closure relations. The closure relations such as interphase forces, chemical reaction models and the equations of state are usually developed from correlations involving experimental data or from simple physical or chemical models describing interphase or intraphase interactions. Such correlations are generally only valid within the range of known experimental data; the choice of appropriate closure models reflects the understanding of the underlying physical and chemical nature of the system to be simulated.

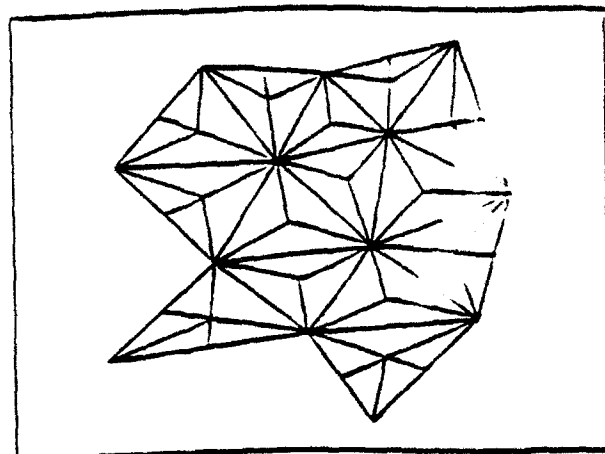
2.3 DIRECT DYNAMIC REFINEMENT METHOD FOR UNSTRUCTURED TRIANGULAR GRIDS

As stated, an unstructured grid is very well suited to implement boundary conditions on complex geometrical shapes and to refine the grid if necessary. This feature of the unstructured triangular grid is compatible with efficient use of memory resources. The adaptive grid enables the code to capture moving shocks and large-gradient flow features with high resolution. The memory resources available can be very efficiently distributed in the computational domain to accommodate the resolution needed to capture the main features of the solution's physical property. Dynamic refinement controls the resolution priorities. These priorities can be set according to the physical features that the user wishes to emphasize in the simulation. The user has control over the resolution of the physical features, without being restricted to the initial grid. The alternative to Direct Dynamic Refinement is the hierarchical dynamic refinement⁶ (H refinement) that keeps a history of the initial grid (mother grid) and the subdivision of each level (daughter grids). In the H refinement method, it is necessary to keep overhead information on the level of each triangle subdivision, and double indirect indexing is required to keep track of the H refinement process. As mentioned, H refinement relies heavily on the initial grid as it subdivides the mother grid, and returns to that grid after the passage of the shock.

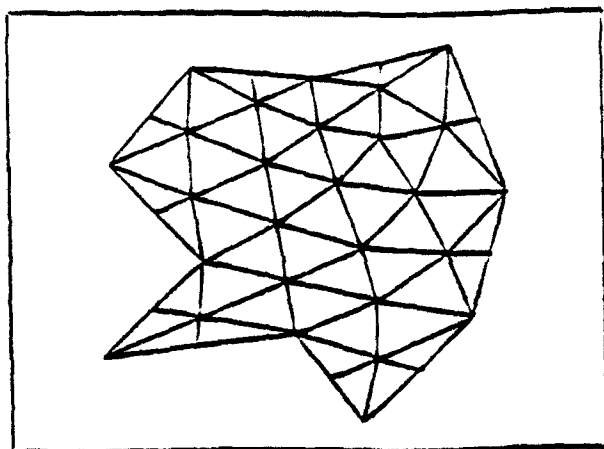
The Direct Dynamic Refinement (DDR) method for capturing the shock requires the refinement to be in the region ahead of the shock. This requirement minimizes the dissipation in the interpolation process when assigning values to the new triangles created in the refined region. Additionally, it requires that the coarsening of the grid be done after the passage of the shock. The interpolation and extrapolation in the refinement and coarsening of the grid is done in the region where the flow features are smooth.



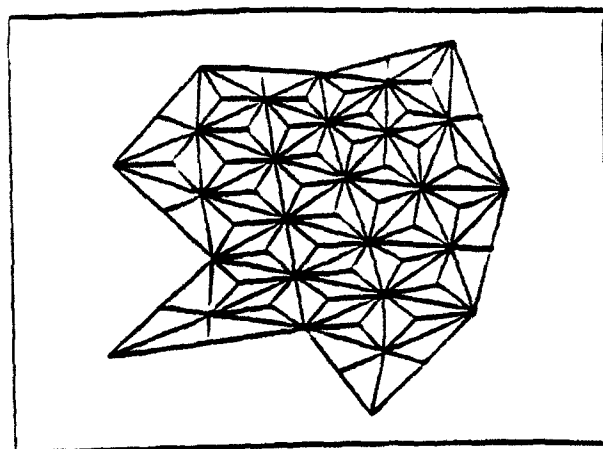
a. Original grid.



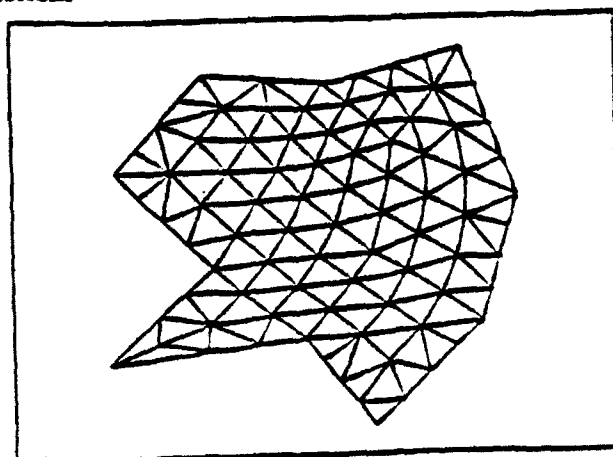
b. Grid after one refinement.



c. Grid after one refinement
and one reconnection.

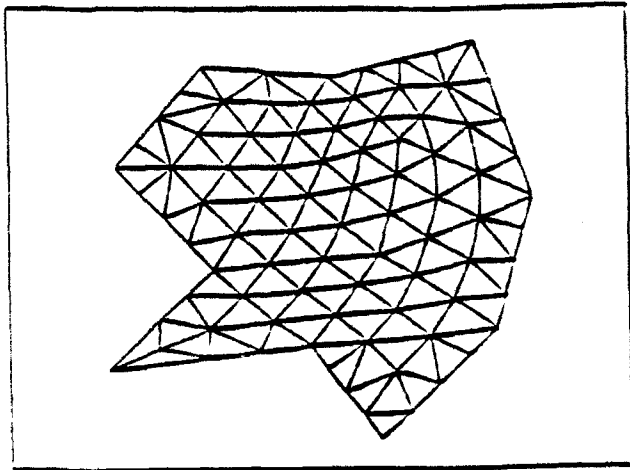


d. Second refinement.

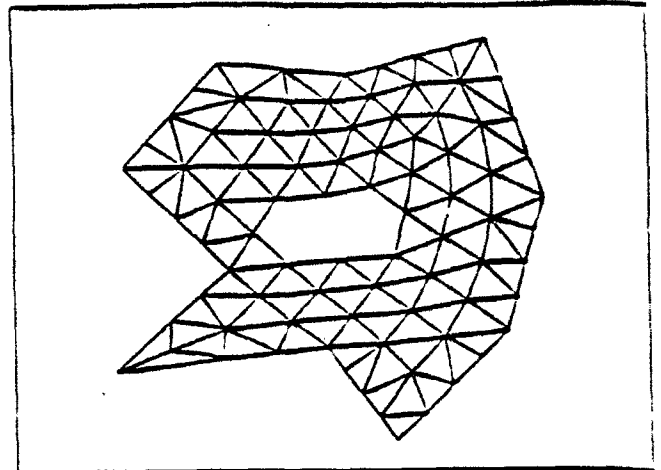


e. Second reconnection.

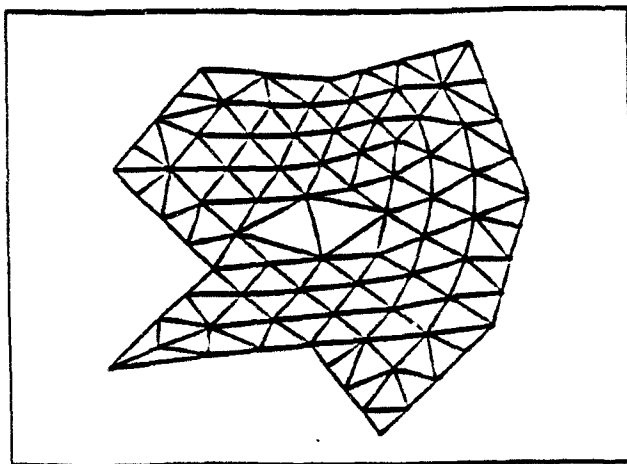
Figure 2.3.1 Illustration of the grid refinement process.



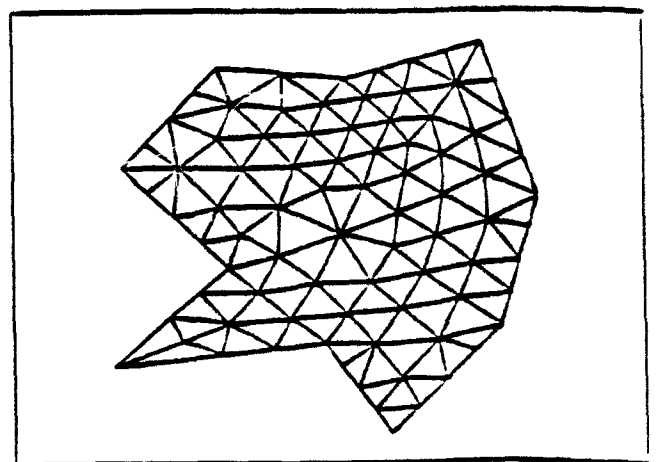
a. Original grid.



b. Point removal.



c. Constructing of new cells.



d. Grid after reconnection and relaxation.

Figure 2.3.2 Illustration of the grid coarsening process.

The physics of the problem is involved in the process that identifies the region of refinement and coarsening. Error criteria can be derived that will allow grid adaptation to stationary or moving pressure or density discontinuities, region of high vortical activity, etc. There should be an error indicator specially suited to capture and identify the region of importance for each of the physics features to be resolved.

The original FUGGS algorithm reported in Ref. 9 was modified to enable adaptivity of the grid in the course of the computation. In AUGUST, we have implemented an algorithm with multiple criteria for capturing a variety of features that might exist in the physics of the problem to be solved. To identify the location of a moving shock, we use the flux of total energy into triangles. The fluxes entering and leaving triangles are the most accurate physical variables computed by the Godunov algorithm for solving the Euler equations, and are used to update the physical variables for each timestep in each triangle. A shock wave means that there is a "step-function" change in the cell that is caused by an influx of energy, momentum or density. Stationary shocks can be identified by density gradients that are computed in the course of implementing the Second-Order Godunov algorithm.

In Fig. 2.3.1, we illustrate the basic process of refinement accomplished in the DDR. The original grid is shown in Fig. 2.3.1a. Figure 2.3.1b illustrates a one-step scheme refinement in which a new vertex is introduced into a triangular cell, forming three new cells. This is followed by reconnection, which modifies the grid as demonstrated in Fig. 2.3.1c. The process of refinement and reconnection can be continued until the necessary grid resolution is achieved, as illustrated in Figs. 2.3.1d and 2.3.1e. This direct approach to the grid refinement provides extreme flexibility in resolving local flow features. A similar simple method is applied to grid coarsening. In the first step of coarsening the marked vertices, all associated elements of the grid are simply removed, as shown in Fig. 2.3.2a. During the second step, this void in the grid is filled with new larger triangles (Fig. 2.3.2b) and then reconnected as shown in Fig. 2.3.2c. When a very large increase of the local grid density is required, these simple algorithms of grid addition and deletion can create triangles with an unacceptably large aspect ratio. To avoid this condition for very large grid densities (when the area of the triangles in the dense region is reduced to less than 2% of the initial area), we introduced local grid relaxation immediately after the grid deletion procedure.

AUGUST has proven to be a very robust and efficient algorithm capable of computing transient phenomena, and with the ability to sense the region of physical interest and resolve it by refining and coarsening the grid as needed.

2.4 STRUCTURED/UNSTRUCTURED COMPOSITE GRIDS

Structured rectangular grids allow the construction of numerical algorithms that perform an efficient and accurate integration of fluid conservation equations. The efficiency of these schemes results from the extremely low storage overhead needed for domain decomposition and the efficient and compact indexing that also defines domain

connectivity. These two factors allow code construction based on a structured domain decomposition that can be highly vectorized and parallelized. Integration in physical space on orthogonal and uniform grids produces the highest possible accuracy of the numerical algorithms. The disadvantage of structured rectangular grids is that they cannot be used for decomposition of computational domains with complex geometries.

The early developers of computational methods realized that, for many important applications of Computational Fluid Dynamics (CFD), it is unacceptable to describe curved boundaries of the computational domain using the stair-step approximation available with the rectangular domain decomposition technique. To overcome this difficulty, the techniques of boundary-fitted coordinates were developed. With these techniques, the computational domain is decomposed into quadrilaterals that can be fitted to the curved domain boundaries. The solution is then obtained in the physical space using the geometrical information defining the quadrilaterals, or in the computational coordinate system that is obtained by transformation of the original domain into a rectangular domain. The advantage of this technique is that it employs the same indexing method as the rectangular structured domain decomposition methods that also serve to define domain connectivity. The boundary-fitted coordinate approach leads to efficient codes, with approximately a 4:1 penalty in terms of memory requirement per cell as compared with rectangular domain decomposition. However, this approach is somewhat restricted in its domain decomposition capability, since distortion or large size variations of the quadrilaterals in one region of the domain leads to unwanted distortions or increased resolution in other parts of the domain. An example of this is the case of structured body fitted coordinates used for simulations of flows over a profile with sharp trailing edges. In this case, increased resolution in the vicinity of the trailing edge leads to increased resolution in the whole row of elements connected to the trailing edge elements.

The most effective methods of domain decomposition developed to overcome this disadvantage are those using unstructured triangular grids. These methods were developed to cope with very complex computational domains. The unstructured grid method, while efficient and powerful in domain decomposition, results in codes that must store large quantities of information defining the grid geometry and connectivity, and have large computational and storage overheads. As a rule, an unstructured grid code requires greater storage by a factor of 10, and will run about 20 times slower per cell per iteration than a structured rectangular code.

Unstructured grid methods are used to their best advantage when combined with grid adaptivity. This feature usually allows dynamic decomposition of the computational domain subregions, thus leading to an order-of-magnitude reduction in the number of cells for some problems, as compared to the unstructured grid lacking this adaptive capability. However, this advantage is highly dependent on the problem solved. Adaptive unstructured grids have an advantage over the unadaptive unstructured domain decomposition if the area of high-resolution domain decomposition is less than one tenth of the global area of the computational domain. This explains why the adaptive unstructured method may be extremely effective for solutions with multiple shock waves

in complex geometries, but becomes extremely inefficient when high resolution is needed in a substantial area of the computational domain.

Our approach to domain decomposition combines the structured and unstructured methods for achieving better efficiency and accuracy. Under this method, structured rectangular grids are used to cover most of the computational domain, and unstructured triangular grids are used only to patch between the rectangular grids (Fig. 2.4.1) or to conform to the curved boundaries of the computational domain (Fig. 2.4.2). In these figures, an unstructured triangular grid is used to decompose the regions of the computational domain that have a simple geometry.

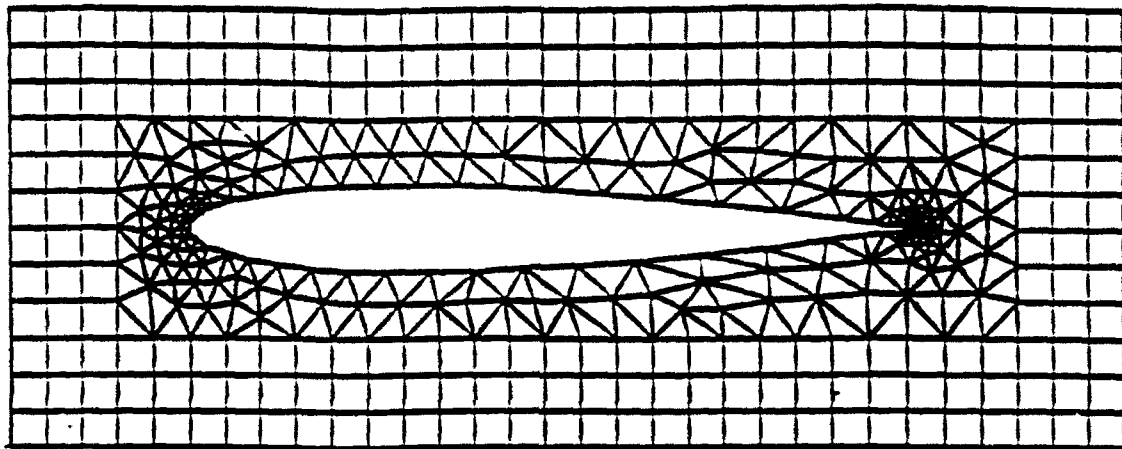


Figure 2.4.1 A possible candidate configuration for hybrid structured/unstructured domain decomposition.

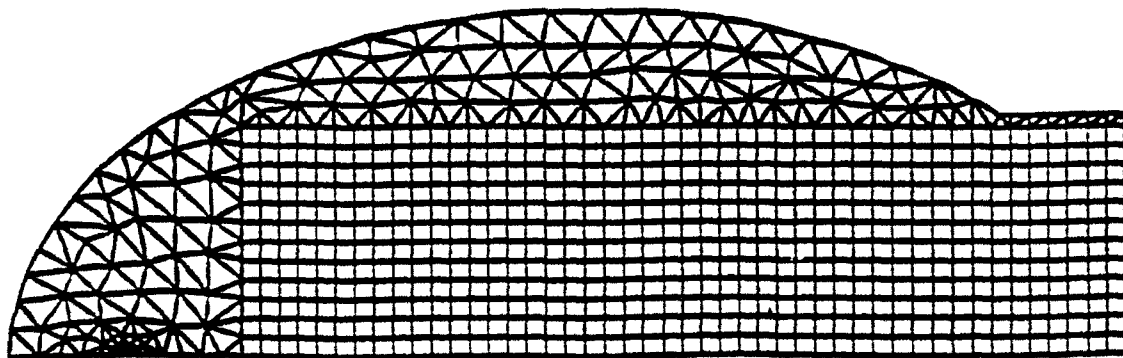


Figure 2.4.2 Hybrid structured/unstructured grid used to simulate ellipsoidal reflector, showing adaptation to curved boundaries.

2.5 THREE-DIMENSIONAL CAPABILITY

Once the 2D capability was fully developed, we initiated the development of a fully 3D CFD adaptive unstructured simulation capability. This part of our effort is not yet documented in published material.

The first step in solving a 3D CFD problem is to discretize the computational domain into tetrahedra. The grid generation is a recognized bottleneck in the time it takes to evaluate an aerodynamic configuration.¹⁵ One could even argue that it represents the most time-consuming portion of the evaluation process. There are a handful of codes that are capable of gridding any given domain into tetrahedra. In order to shorten the part to our objective of achieving a 3D adaptive solver capability, we decided to make use of an existing grid generator to provide the initial grid.

OCTREE,¹⁶ which was developed at Rensselaer Polytechnic Institute (RPI), is a Finite Octree 3D grid generator that provides the initial grid for our adaptive solver. The productivity of a 3D grid generator is a function of the complexity of the surfaces that define the domain of computation. Usually, this task is the most time-consuming and painful for the user. OCTREE does not have a CAD/CAM package to assist the user in defining the surfaces of the geometry to be gridded. Nevertheless, OCTREE is a very robust and reliable grid generator.

The OCTREE algorithm is based on the concept of dividing the computational domain into octants. In each step, the code defines three planes that halve the domain in each of the three dimensions, thus dividing the volume into eight octants. Those three planes intersect the surfaces of the geometry, defining vertices. All the vertices are collected and sorted into topological loops. If the vertices are not sufficient to define correct topological loops, the code will subdivide the corresponding octant into eight smaller octants until the topology is fully resolved. The user is allowed to specify the level of the local octree subdivision he wishes to resolve. Once the code subdivides the volume into the level of octree specified by the user or needed to resolve the local geometrical details, the code defines tetrahedra to fill the volume of the computation domain. The code provides the user with an option that improves the quality of the tetrahedra by smoothing and eliminating the very small ones.

As stated, OCTREE provides the initial grid for the 3D solver. The adaptivity of the mesh is controlled by specific physical features that the user defines based on the physics of the problem to be solved. The adaptivity of the mesh automatically traces the physical features in the simulation and refines and coarsens the mesh accordingly to the criteria and the resolution specified by the user.

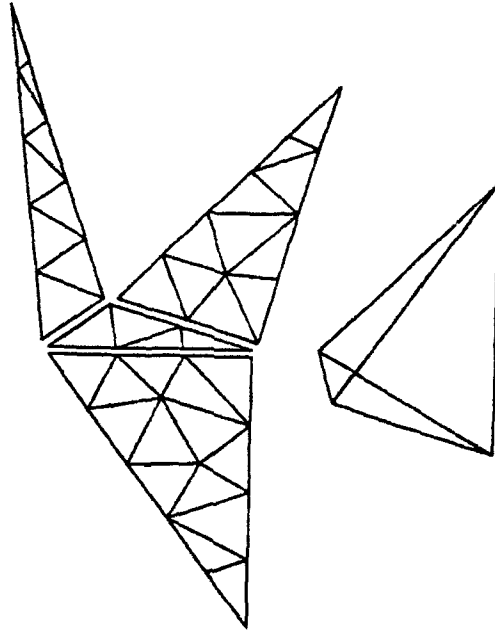


Figure 2.5.1 An elongated tetrahedron can be refined using smaller tetrahedra that are nearly regular.

The target tetrahedra are refined by first subdividing each of the four surfaces into smaller triangles that satisfy the resolution set by the user. There are no constraints on the way each face is subdivided. Each edge of the face is subdivided according to the local resolution needed, and the points along the edges are connected to construct the best triangles possible. The code adds points inside the face along with points on the edges to achieve an adequate triangulation of the faces. The triangles of the four faces of the target tetrahedra are used to define smaller tetrahedra that will fill the volume. If needed, the code will add points inside the volume of the target tetrahedron to achieve the best tetrahedra possible. The code has the ability to reconnect tetrahedra to improve quality. The reconnection is done by pulling out an edge, sorting all the tetrahedra connected to this edge, deleting these tetrahedra and filling the void with better shaped tetrahedra.

Figure 2.5.1 shows how the subdivision process can fill an irregular (elongated) tetrahedron with smaller tetrahedra that are nearly equilateral. (This is not the case with H refinement.) Figure 2.5.2 shows points used to create octree refinement to grid a problem involving surface-mine blast effects on the underside of a truck. Figure 2.5.3 is the corresponding tetrahedral grid. The calculated overpressures on the surface of the truck underbody for an eight-pound explosive are shown in Sec. 3.4.

The algorithm used to solve the 3D gasdynamic equations is an immediate extension of the 2D case described in Sec. 2.1. Thus, Eq. (2.6) is replaced by

$$\int_{\Omega} \nabla U_i^{\text{cell}} dV = \int_{\partial\Omega} U_i^{\text{face}} dS, \quad (2.6')$$

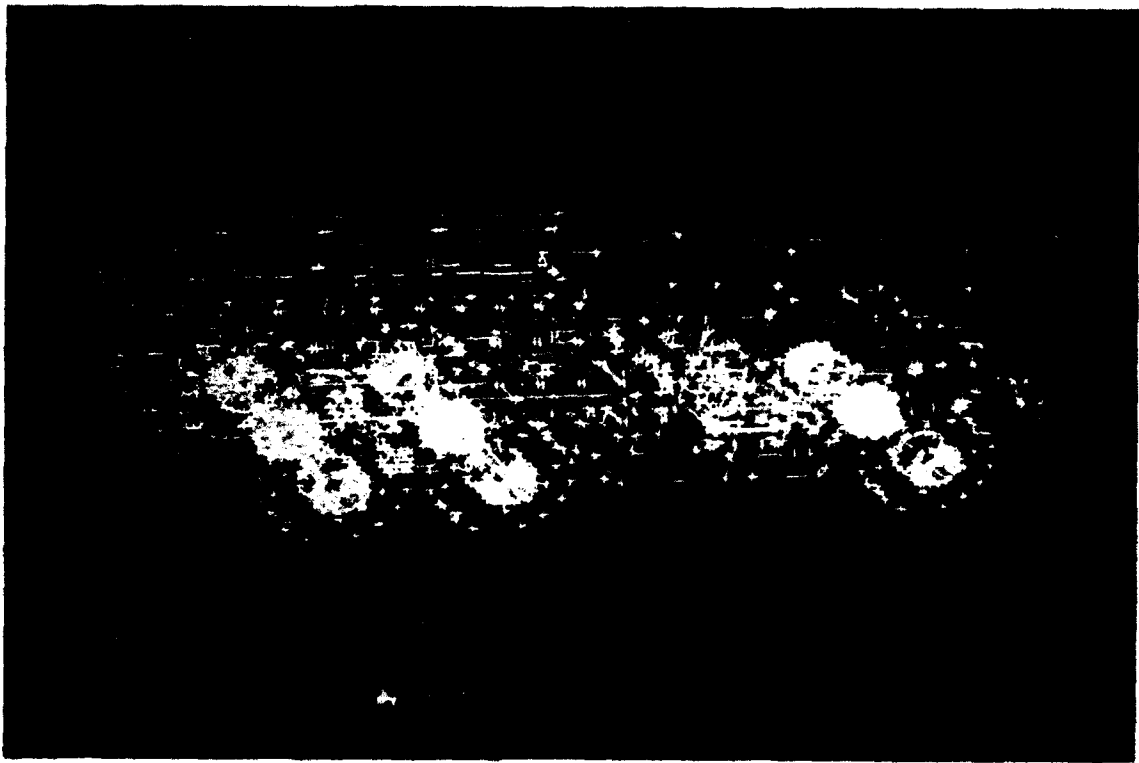


Figure 2.5.2 Points used to define structure of vehicle.

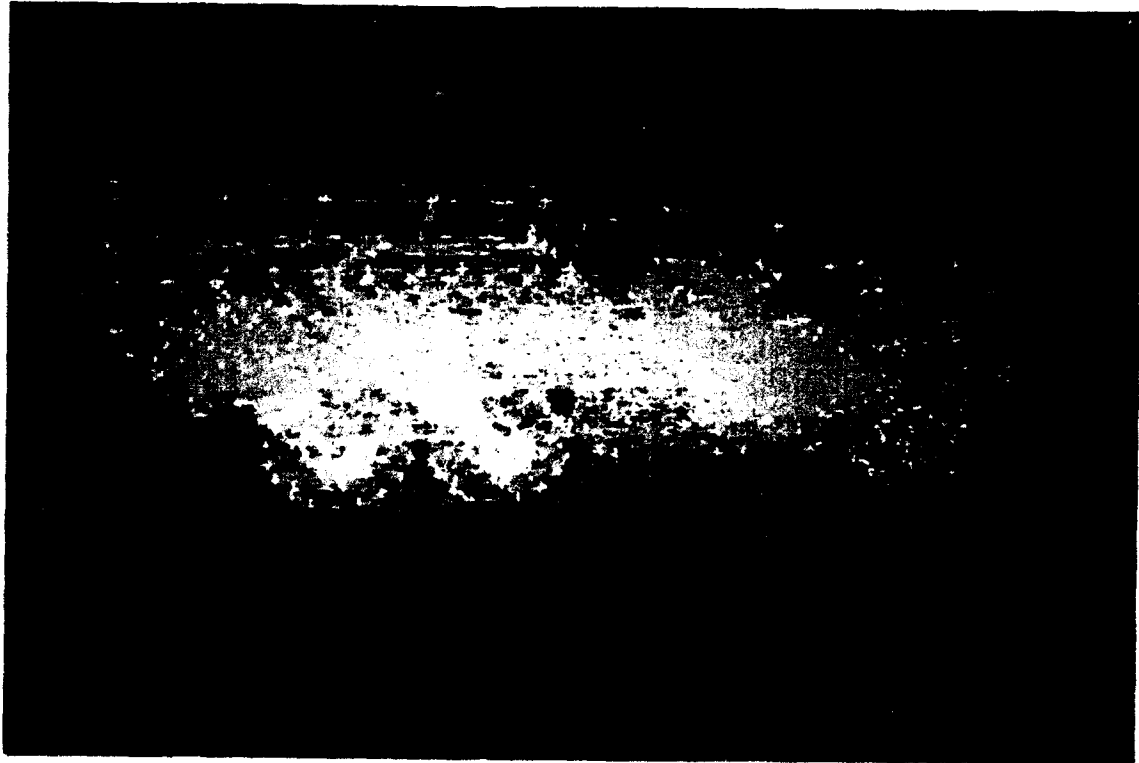


Figure 2.5.3 Tetrahedral grid generated by Finite Octree method.

where now Ω and $\partial\Omega$ are the volume and surface of a tetrahedron, and dV and dS are the corresponding differential elements. Its finite-difference approximation is

$$\nabla U_i^{\text{cell}} = \frac{1}{V} \sum_{j=1}^4 \tilde{U}_j^{\text{face}} \mathbf{n}_j \Delta S_j, \quad (2.7')$$

where the summation is over the four faces and \mathbf{n}_j is the normal to the j th face with surface area ΔS_j . In the equations corresponding to Eqs. (2.9) - (2.11), the range 1, 2, 3 is replaced by 1, 2, 3, 4. Equations (2.12) - (2.16) are formally unchanged, and Eq. (2.17) becomes

$$(M_x, M_n) = \begin{cases} \max(\lambda_i^+, 0) & \text{for the cell on the left of the edge} \\ \min(\lambda_i^-, 0) & \text{for the cell on the right of the edge.} \end{cases} \quad (2.17')$$

3. APPLICATIONS

The AUGUST code was extensively validated for a wide range of known CFD problems and has been shown to be a robust simulation tool. It has been utilized on a variety of problems which span flow regimes ranging from low subsonic Mach numbers to hypersonic Mach numbers (Table 3.1).

Appendix C contains a complete collection and description of the CFD problems addressed during the UUGM research. Additional details of the AUGUST code are contained in SAIC's progress report for the UUGM DARPA program, submitted in November 1991. Here we briefly describe the most noteworthy applications.

It is worth underscoring again that in the past it was necessary to use a sequence of codes as well as numerical parameter adjustment to bridge the gap in flow phenomena occurring in different flow regimes. An important point to be made here is that the AUGUST code allows robust, accurate and efficient solutions across these different regimes without the necessity of adjusting coefficients to enhance convergence accuracy or efficiency.

Table 3.1 AUGUST Applications

Problem	Activity
1. Calculation of potential flow about an ellipse.	Reported at the 4th International Symposium on Computational Fluid Dynamics, Davis, CA, Sept 1991.
2. Hypersonic flow past a flat plate.	Reported at AIAA Reno Meeting (AIAA-90-0699), 1990.

Problem	Activity
3. Shock on wedge with adaptive gridding.	Reported at the Free Lagrange Conference, Jackson Lake, WY, 1990.
4. Simulation of mine explosion under a vehicle.	Performed for U.S. Army Corps of Engineers, Ft. Belvoir, VA.
5. Simulation of pulsed detonation engine.	Published in J. of Propulsion and Power Nov/Dec 1991 Vol. 7 (6) pp. 857-865 and AIAA Meeting, Reno, NV 1992.
6. Shock focusing in air using structured/unstructured grids.	Presented at the ICAM Conference, Rutgers, NJ, June 1992.
7. Nonideal airburst calculations for multiphase media.	Performed for the Defense Nuclear Agency, Alexandria, VA.
8. Flow in the SARL wind tunnel.	Performed for Wright-Patterson AFB.
9. Simulation of a shock on a double wedge.	Presented at the Army workshop on Adaptive Methods for PDEs, RPI, March 1992.
10. Supersonic spray coating devices.	To be published.
11. Nanomaterial synthesis.	Published in Surf. Coating Tech. 49, 387-393 (1991).
12. Dusty flow over a cylinder.	To be published in AIAA Journal.
13. Image processing.	Presented at SPIE conference on Applications of Digital Image Processing, San Diego, July 1991.
14. Multiphase detonation.	Published in Combust. Sci. Tech. 89, 201-218 (1993).

3.1 POTENTIAL FLOW OVER AN ELLIPSE

One of the outstanding early CFD computational challenges (from the point of view that no satisfactory solution had been obtained) was associated with simulating subsonic (Mach 0.2 and less) flow over a symmetric elliptical airfoil using the Euler equations (Fig. 3.1.1). All previous attempts to compute the flow over such an ellipse resulted in spurious lift and drag values that were significantly larger than the classical

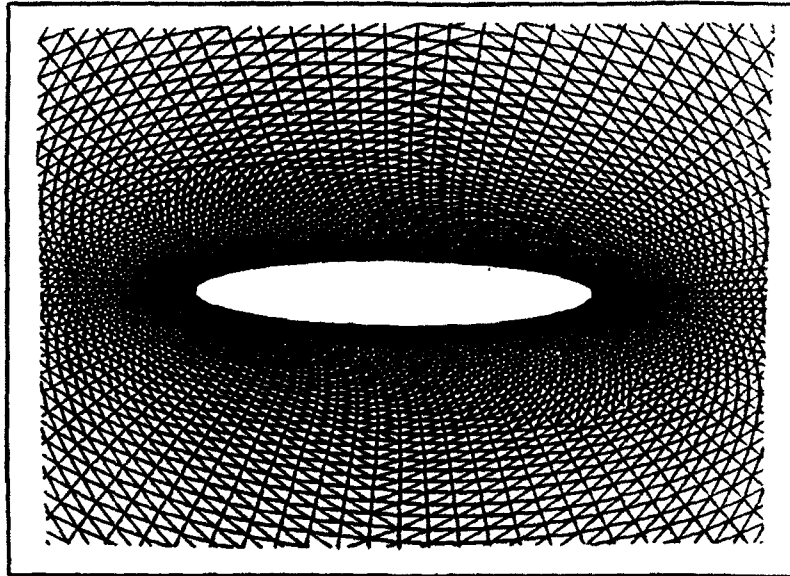


Figure 3.1.1 The grid used for simulating the flow over an ellipse.

potential flow solution. The potential flow result should have been closely approximated if there were no numerical viscosity present. This test case is important because, in transitioning from an Euler solver to a full Navier-Stokes solver, one needs confidence that the artificial (numerical) viscosity will not dominate the physical viscosity included in the Reynolds' stress terms. As shown in Appendix C-1, use of an earlier version of the AUGUST code, the Fast Unstructured Grid Godunov Solver (FUGGS) code provided solutions to this test case that were very close to the potential flow solution. Other attempts resulted in lift and drag values that were off by several orders of magnitude compared with the SAIC FUGGS results. The results described here were prepared for a poster presented to Dr. Arje Nachman, SAIC's UUGM AFOSR program monitor and Dr. James Crowley, SAIC's UUGM DARPA program manager.

3.2 HYPERSONIC FLOW PAST A FLAT PLATE

To demonstrate the versatility of the method for the entire range of flow regimes we have simulated a hypersonic flow test problem. One of the advantages of the Godunov methods is that over the whole range of calculations performed (low subsonic flow, supersonic flow, unsteady flow with strong shock, or hypersonic flow at Mach number $M=32$) it is unnecessary to change or adjust the numerical algorithm. In Ref. 17 the performance of first- and second-order Godunov methods was analyzed for hypersonic flow regimes. There, as a test problem, an analytical solution was used for a hypersonic flow around a flat plate of finite thickness. This solution was obtained based on the analogy between hypersonic flow over a flat plate of finite thickness and a strong planar explosion. Here we use an expression from Ref. 17 which defines the shape of the shock wave as a function of plate thickness d , γ is the adiabatic coefficient, and α is a nondimensional scale factor related to the energy released at the stagnation point.

$$Y_{\text{SHOCK}} = \left(\frac{1}{2} D_f \frac{dx^2}{2} \right)^{1/3}$$

where D_f is a coefficient of order unity,

$$a = k_1 (\gamma - 1)^{k_2 + k_3 \ln(\gamma - 1)}$$

with $k_1 = 0.36011$, $k_2 = 1.2537$, and $k_3 = -0.1847$.

As a direct comparison we solved the hypersonic flow problem for the same set of conditions as in Ref. 17:

$$U_\infty = 10011 \text{ m/sec}, p = 98.72 \text{ Pa}, \rho = 1.24 \times 10^{-3} \text{ kg/m}^3, \text{ and } \gamma = 1.2.$$

The grid used for this simulation is shown in Fig. 3.2.1a. This grid has ≈ 5500 vertices and its spatial resolution at the leading edge of the plate is of the same order as that of a 300×60 rectangular grid used in Ref. 5.

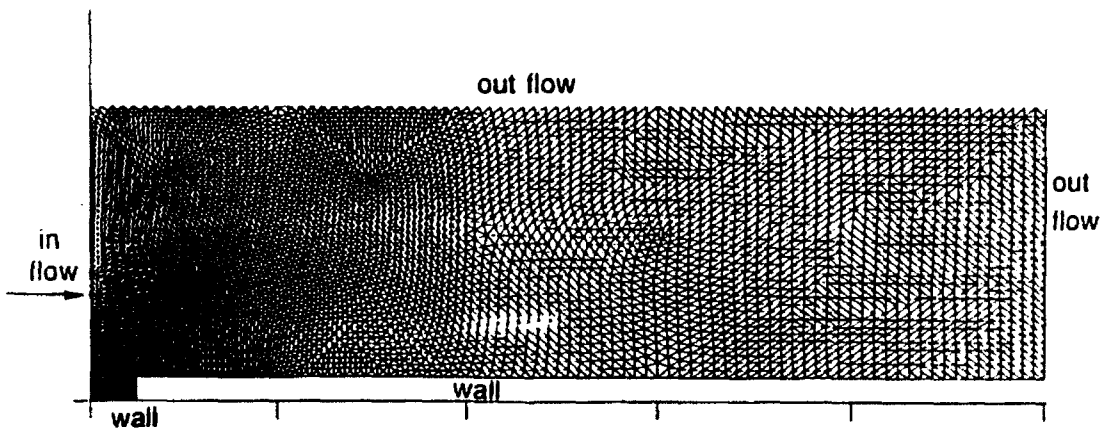


Figure 3.2.1a Grid for simulation of hypersonic flow over a flat plate.

Figure 3.2.1b shows results for this simulation in the form of pressure contours. Figure 3.2.1b also represents the location of the analytically calculated shock front by a discrete line (squares). The shock resolution and accuracy or its location are comparable to that obtained in Ref. 17 even though our triangular grid has less than one third as many nodes as the rectangular grid used in Ref. 17. This is because in constructing the triangular grid we had the flexibility to place the highest concentration of nodes in the area of the leading edge where the main properties of the flow are established.

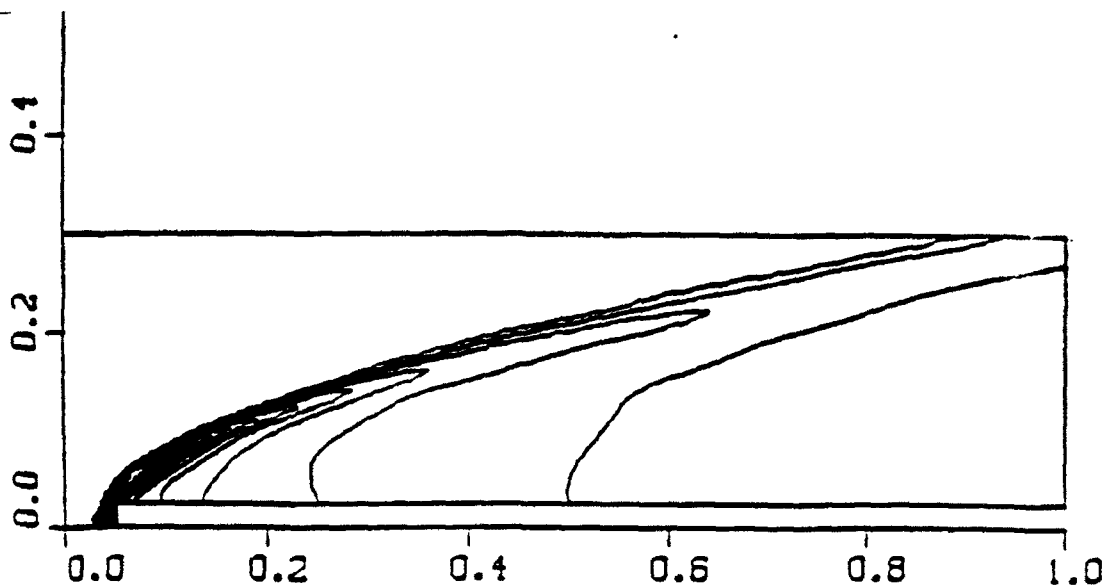


Figure 3.2.1b Second order solution for a flat plate, pressure contours. Mach = 32: 5509 grid vertices: $P_{\max} = 5.0 \times 10^4$ Pa, $P_{\min} = 98.7$ Pa.

3.3 SHOCK ON WEDGE WITH ADAPTIVE GRIDDING

An unstructured grid is very suitable for implementing boundary conditions on complex geometrical shapes and refining the grid if necessary. This feature of the unstructured triangular grid is compatible with efficient usage of memory resources. The adaptive grid enables the code to capture moving shocks and large-gradient flow features with high resolution. The memory resources available can be very efficiently distributed in the computational domain to accommodate the resolution needed to capture the main features of the physical property of the solution.

One strategy for doing this is called hierarchical dynamic refinement (H refinement). It keeps a history of the initial grid (other grid) and the subdivision of each level (daughter grid). H refinement subdivides the initial grid into two or four triangles in each level, and keeps track of the number of subdivision levels each triangle has undertaken. In the H refinement method, one has to keep overhead information on the level of each triangle subdivision, and needs double indirect indexing to keep track of the H refinement process. This slows down the computation by partially disabling the vectorization of the code. As mentioned, H refinement relies heavily on the initial grid as it subdivides the mother grid and returns back to it after the passage of the shock.

AUGUST and its predecessor FUGGS use a second-order Godunov solver on an unstructured grid. The refinement strategy incorporated in these codes is called Direct Dynamic Refinement. For shock capturing, Direct Dynamic Refinement basically requires

the refinement to be in the region ahead of the shock. This requirement minimizes the dissipation in the interpolation process when assigning values to the new triangles created in the refined region. Additionally, it requires that the coarsening of the grid should be done after the passage of the shock. In principle, the interpolation and extrapolation in the refinement and coarsening of the grid are done in the region where the flow features are smooth.

FUGGS was used with direct dynamic refinement to solve the transient behavior of the flow entering a channel with a wedge (prism) having an inclination of 27° . The flow enters the channel from the left with Mach number 8.7. A sequence of snapshots illustrates the density contours, and the grid for each timestep is given in Figs. 3.3.1a - 3.3.3a (contour plots) and 3.3.1b - 3.3.3b (grid). These figures clearly demonstrate the automatic adaptation of the grid to the moving shocks and the ability to capture the detailed physics of the simulation with very high resolution and minimal memory requirements. The initial grid can clearly be seen to the right of the shock ("ahead") in the early stage of the shock propagation from left to right. The coarsening algorithm is able to produce a reasonable mesh in the region trailing the shock as shown in the figures.

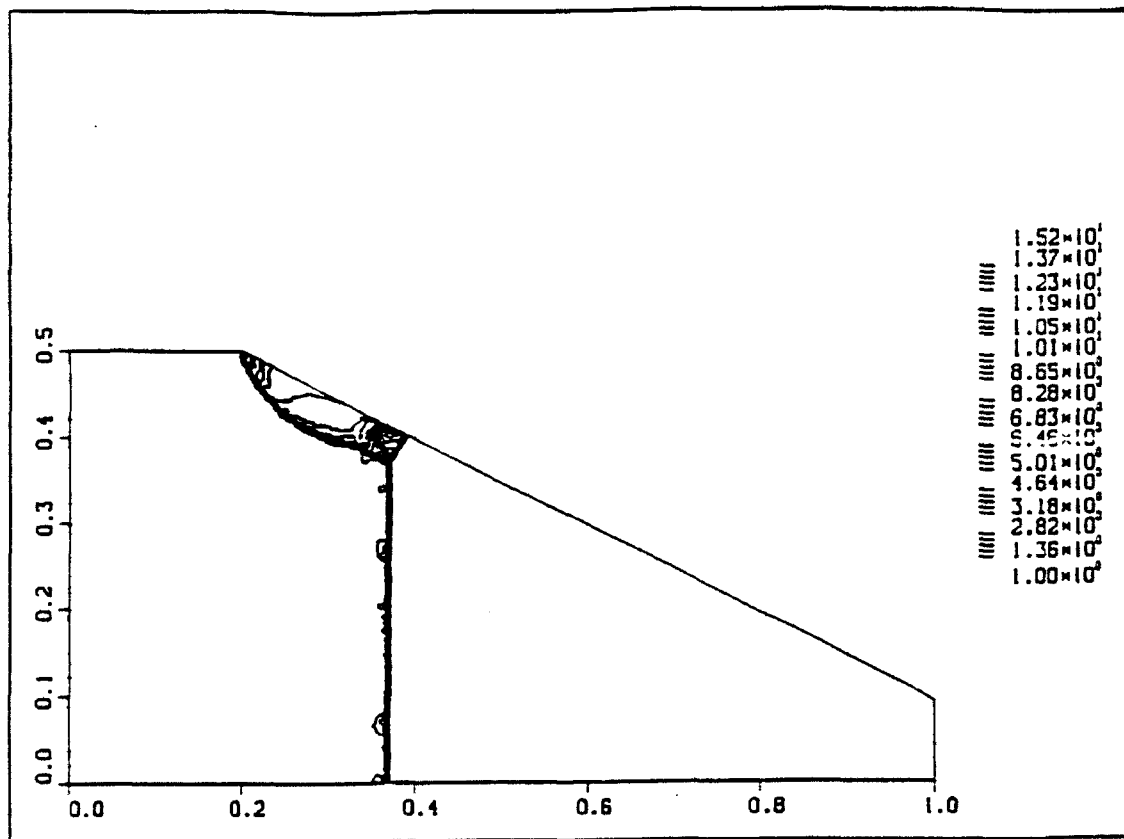


Figure 3.3.1a Density contours at early time for shock in planar channel
($M = 8.7$, wedge angle = 27°).

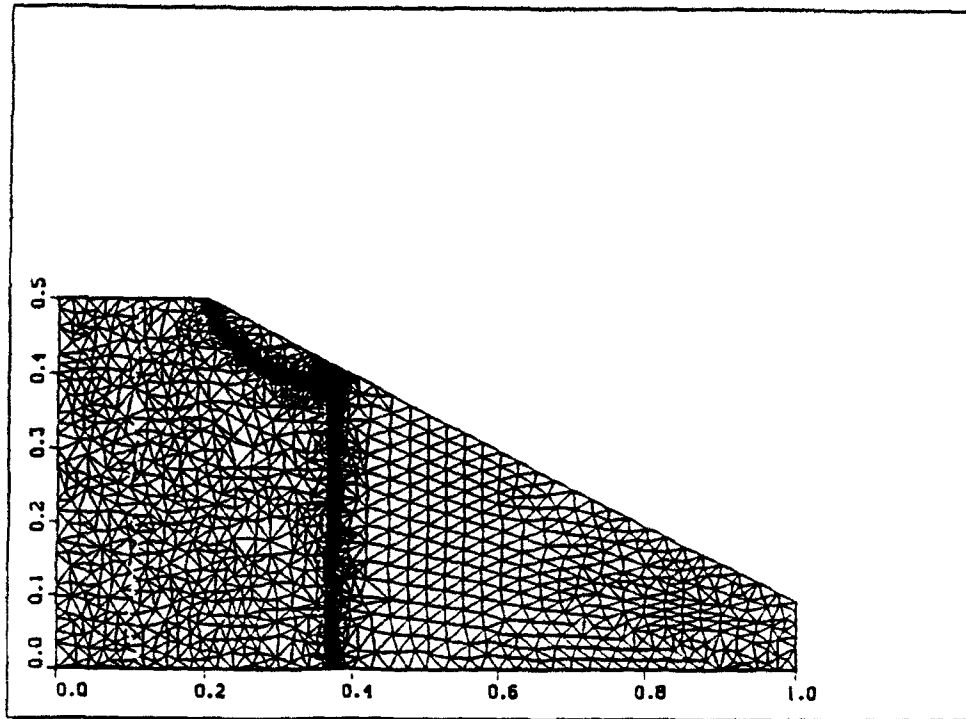


Figure 3.3.1b Grid at early time for shock in planar channel
($M = 8.7$, wedge angle = 27°).

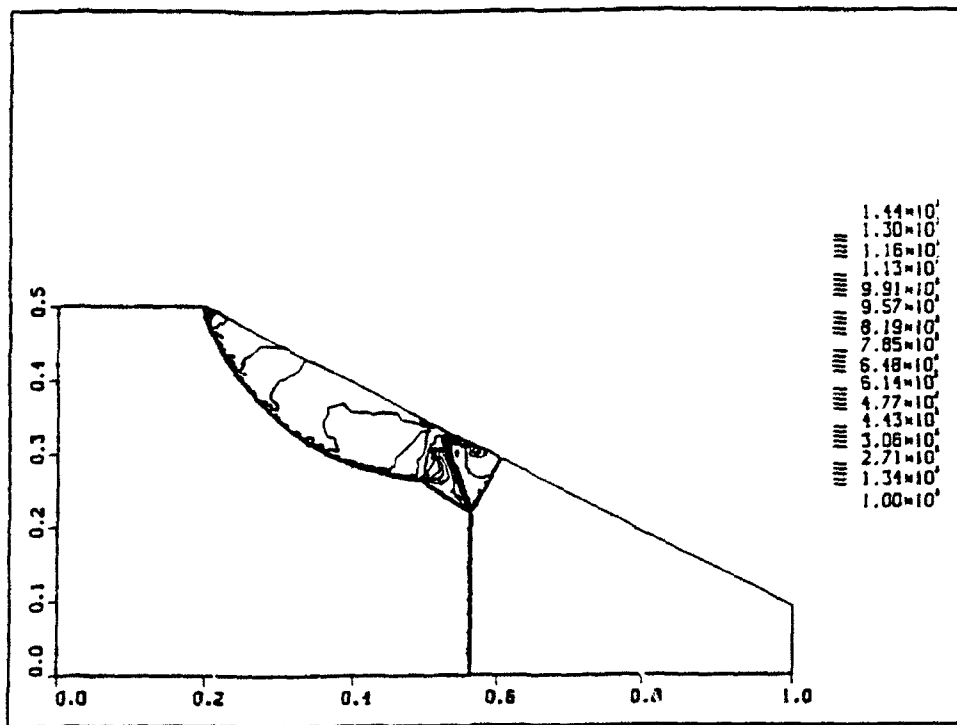


Figure 3.3.2a Density contours at intermediate time for shock in planar channel
($M = 8.7$, wedge angle = 27°).

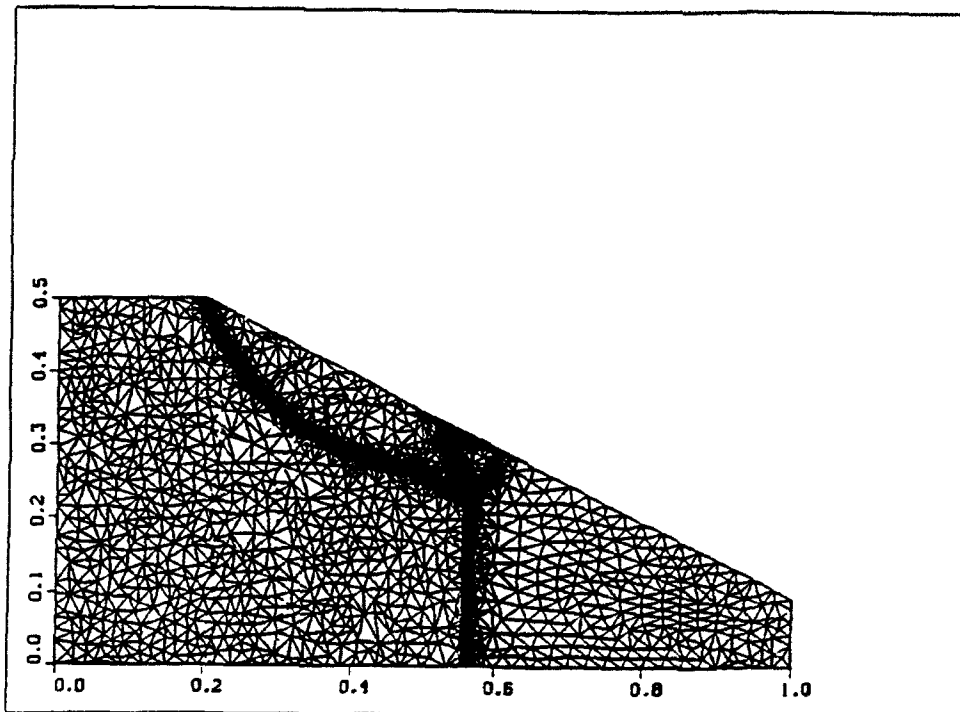


Figure 3.3.2b Grid at intermediate time for shock in planar channel
($M = 8.7$, wedge angle = 27°).

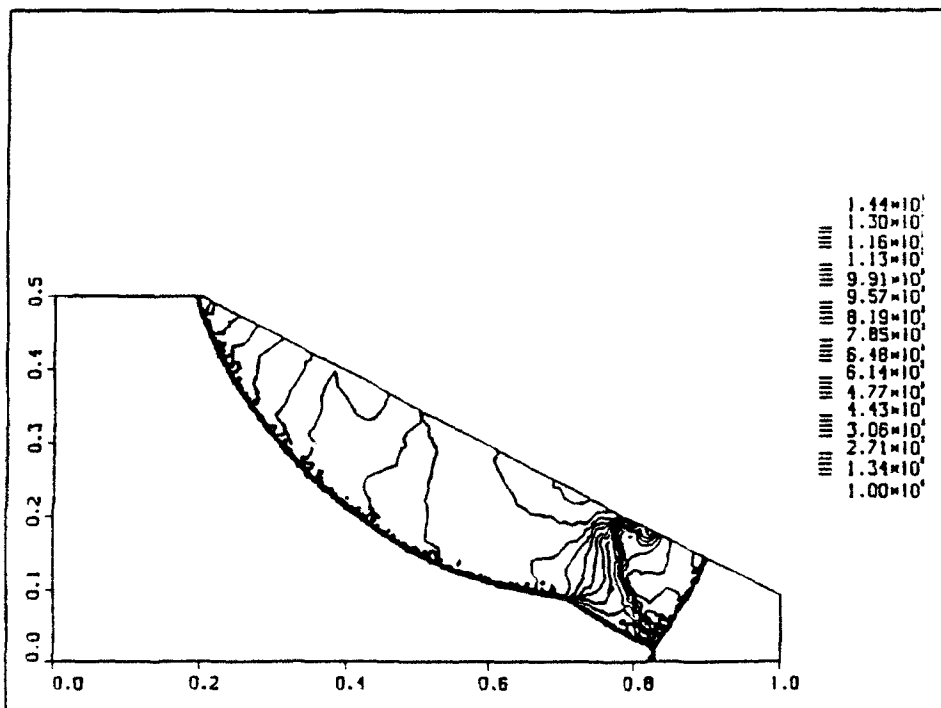


Figure 3.3.3a Density contours at late time for shock in planar channel
($M = 8.7$, wedge angle = 27°).

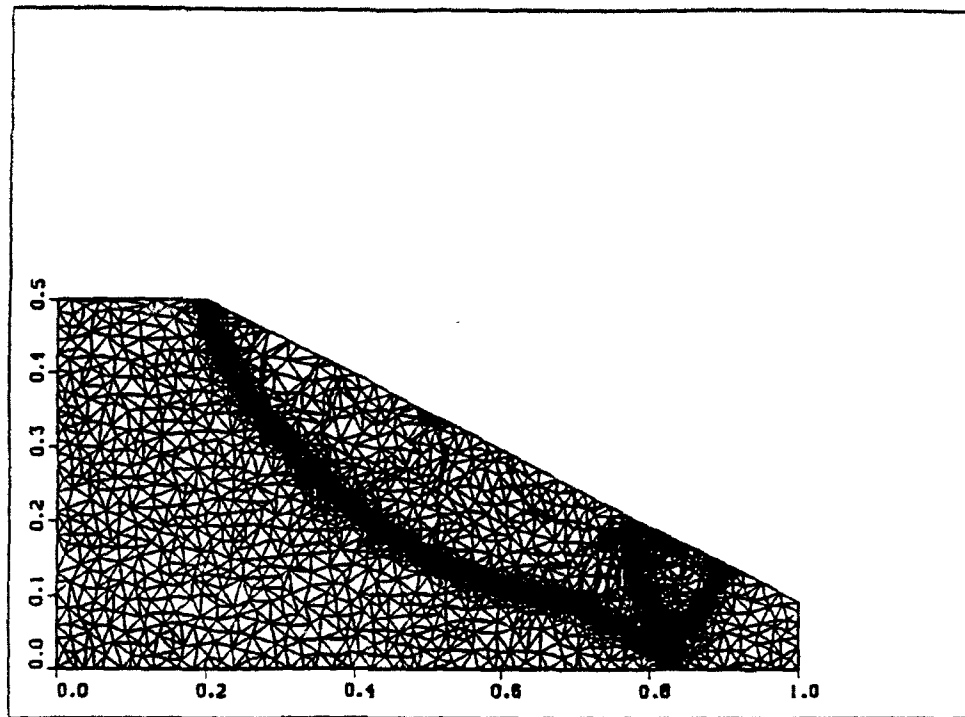


Figure 3.3.3b Grid at late time for shock in planar channel
($M = 8.7$, wedge angle = 27°).

3.4 MINE EXPLOSION UNDER VEHICLE

The main objective of this joint Marine-Army program was the development of vehicles hardened against antitank (AT) land mines. The basic vehicle is the M925 5-ton cargo truck. Numerical simulations were used to determine the dynamic loads produced by the AT mine detonation on the cargo bed and other structural elements of the truck.

The algorithms, techniques and codes developed under the UUGM program provided two key elements necessary for the numerical simulations for this project: a) flexibility in describing the very complex geometry of the truck; b) high resolution-calculation of the shocks and other discontinuities using an adaptive unstructured grid. A version of the AUGUST-2D code developed under the UUGM program is being used for the analysis of blast resistance of different truck geometries.

We have carried out four such calculations, using four, eight, eight, and 20 pounds of C-4 explosive. These employed fixed (nonadaptive) meshes with 30,000 (4-lb case), 21,000 (8-lb cases).

A one- or two-dimensional calculation was performed to produce the initial blast profiles laid down on the three-dimensional grid. Aside from the amount of explosive, the calculations differed in the following ways: all but the 4-lb blast were centered beneath

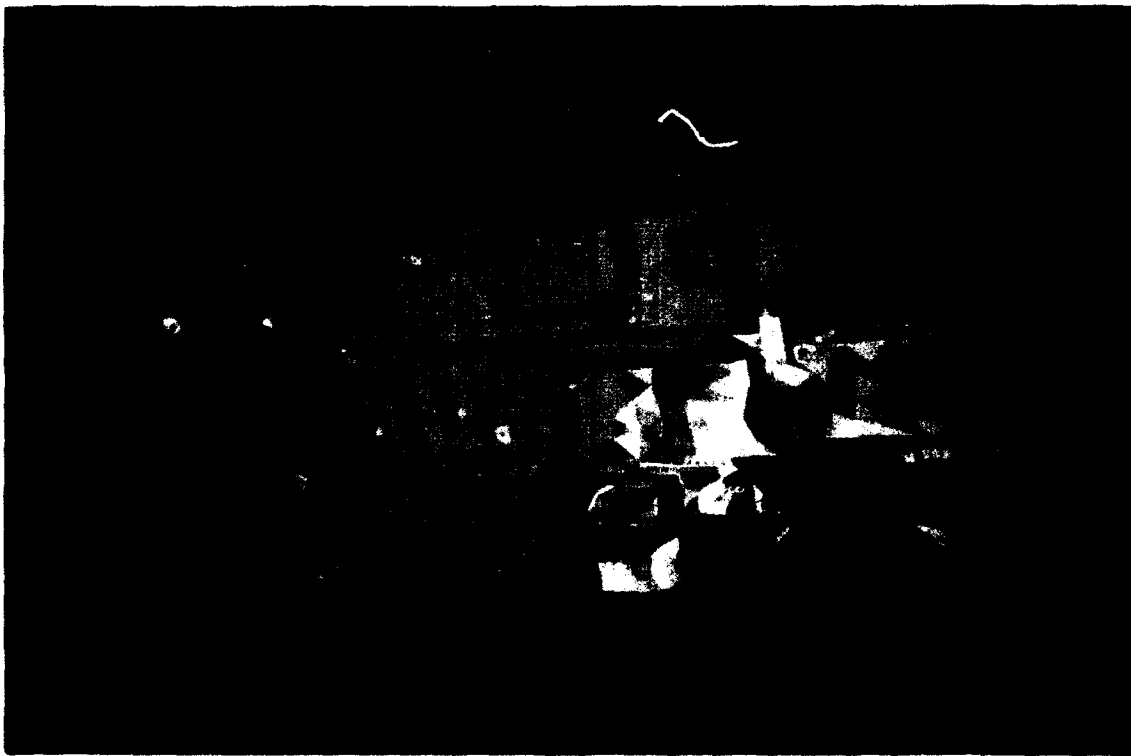
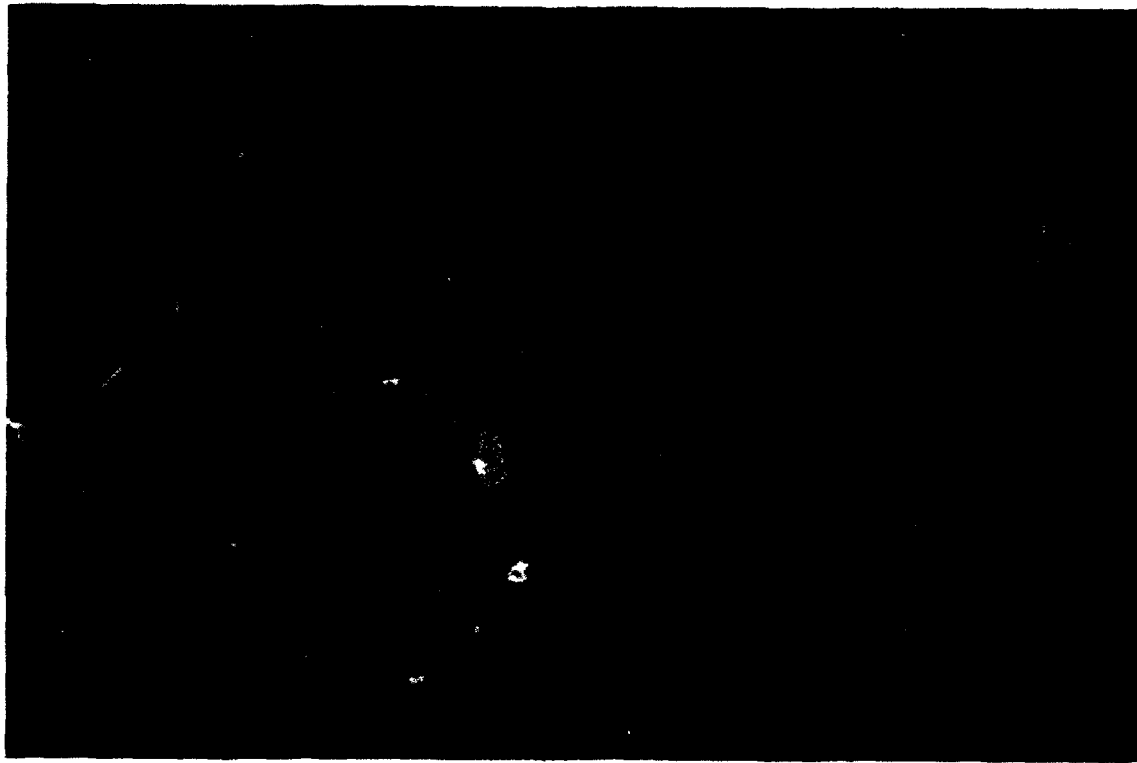


Figure 3.4.1 Two views of interaction between mine blast and M925 cargo truck:
pressure contours at $t = 0.574$ msec.

the left front wheel of the truck (the 4-lb blast was situated 70 cm further back); for the first 8-lb case a crater with diameter 60 cm and depth 30 cm was situated underneath the blast.

All but the second 8-lb case used an ideal-gas equation of state with $\lambda = 7/5$ for the air and detonation products. Twenty pressure "sensors" positioned on the mesh at points corresponding to the pressure gauges used in actual field tests were used to record the pressure and impulse histories there for comparison with the experimental data.

The calculations were run out to about 4.5 msec. The pressure stations closest to ground level and to the blast center exhibited peaks up to $\sim 10^3$ psi. In some cases multiple peaks were present, corresponding to reflected shocks.

An example of the domain decomposition of the computational grid for a typical mine-truck interaction problem is shown in Fig. 2.5.2. In Fig. 2.5.3 the unstructured triangular grid is used to describe a cross section of an M925 cargo truck. Use of unstructured grids allows detailed description of the truck geometry. Figure 3.4.2 shows results of the simulation in the form of pressure contours overlaid on the unstructured grid, viewed from two different directions half a millisecond after the detonation.

At Ft. Belvoir's request, SAIC also assessed the damage to a mine-clearing plow due to a single detonation of an AT mine at close range during the Desert Storm operation. At that time, Ft. Belvoir RDEC had responsibility for support of countermine activity in the Desert Storm operation.

To simulate the plow-mine blast interaction, SAIC used computational capabilities partially developed under the UUGM program. Use of unstructured triangular grids again enables detailed description of the plow geometry and use of Direct Dynamic grid Adaptation method allows detailed simulation of the complex pattern of the shock wave reflections.

In Fig. 3.4.2 the initial stage of the blast-plow cross section interaction is shown in the form of pressure contours overlaid on the dynamically adapting grids. In Fig. 3.4.3 a more advanced stage of the blast-structure interaction is shown in the same format as in Fig. 3.4.2. The adaptive grid allows high resolution of a complex blast interaction phenomena.

SAIC has also simulated the structural response of the plow to the dynamic load that is defined by the gas dynamic simulations described above. In Figs. 3.4.4a - d results are shown for the plow deformation in response to dynamic load. Recent experimental assessment of the plow damage showed that SAIC predictions correctly described blast damage to the plow.

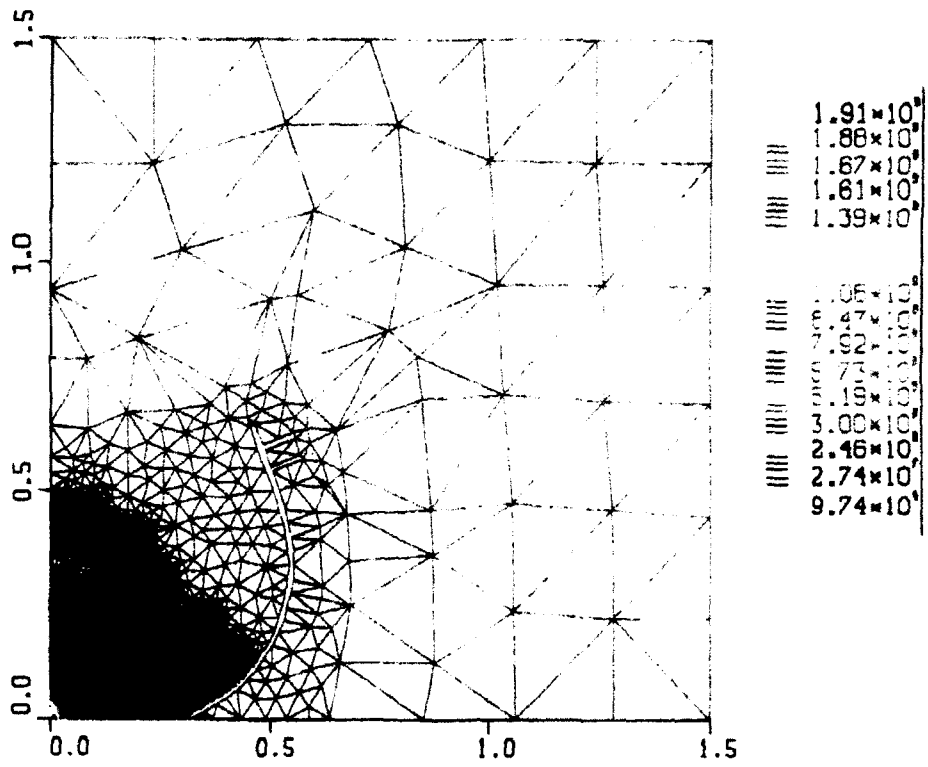


Figure 3.4.2 Blast - plow interaction: pressure contours in initial stage.

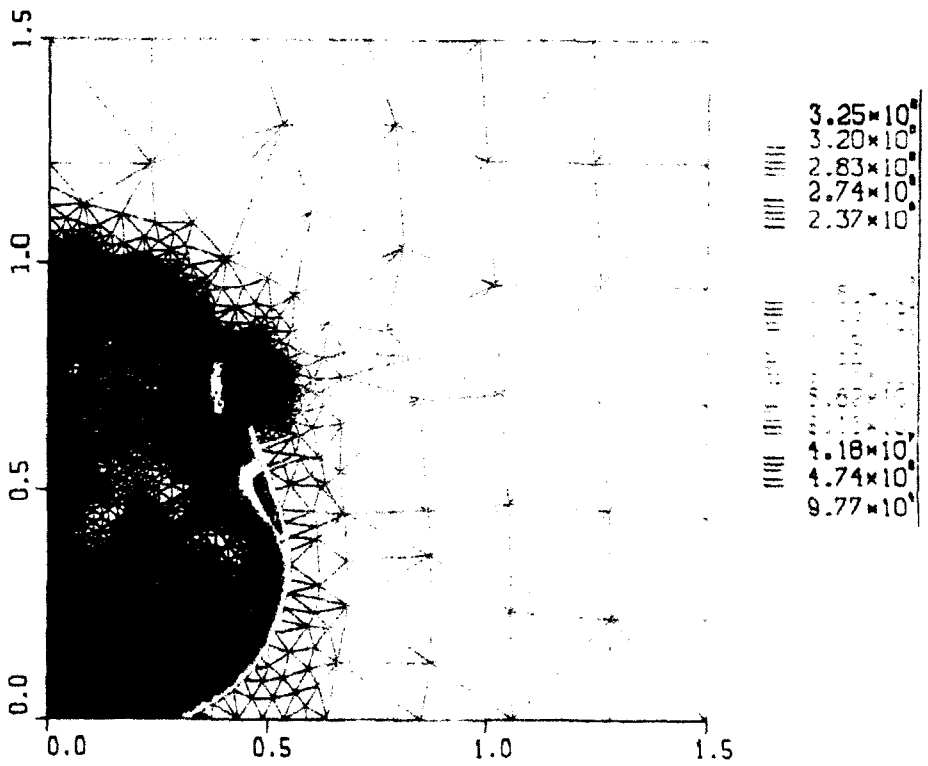
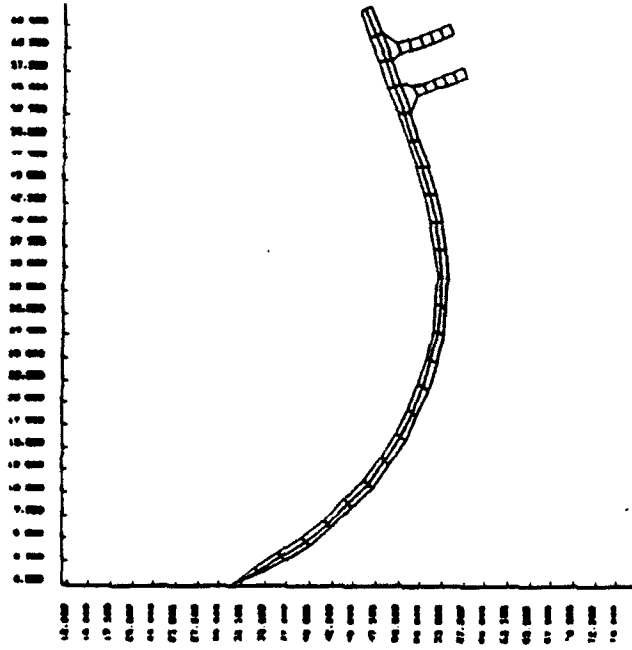
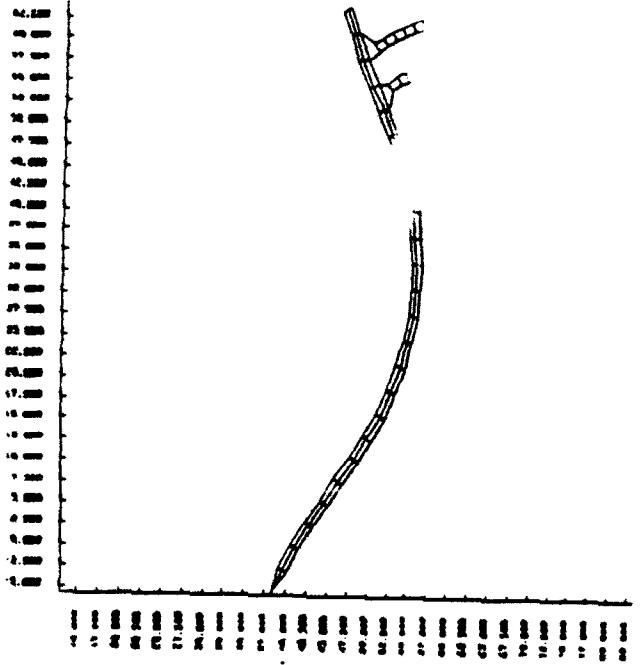


Figure 3.4.3 Blast - plow interaction: pressure contours in advanced stage

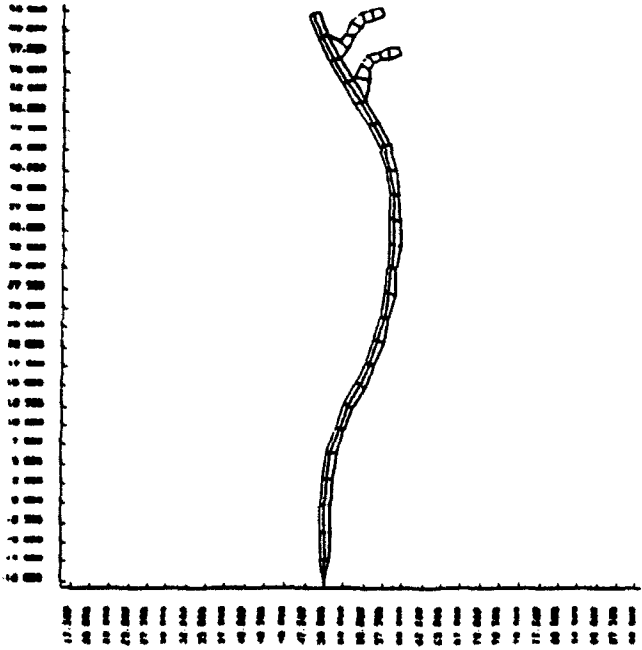
Flow explosion
def = 0.100e+01
line = 0.



Flow explosion
def = 0.100e+01
line = 0.200e+03



Flow explosion
def = 0.100e+01
line = 0.400e+03



Flow explosion
def = 0.100e+01
line = 0.600e+03

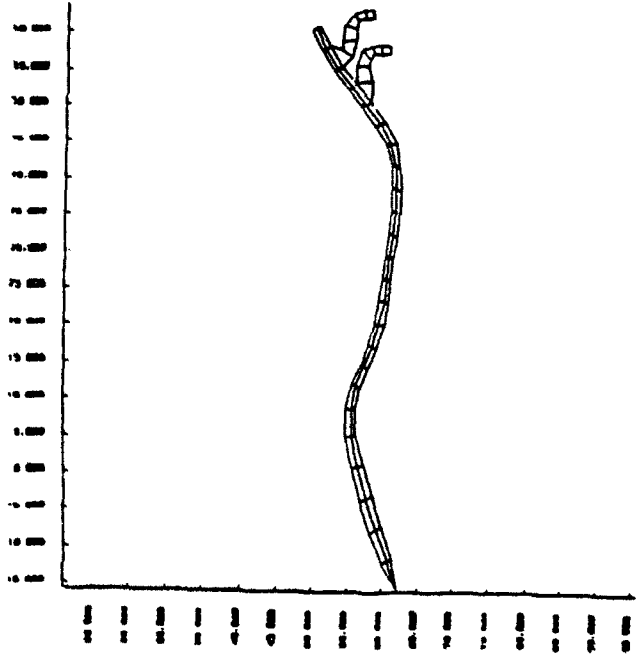
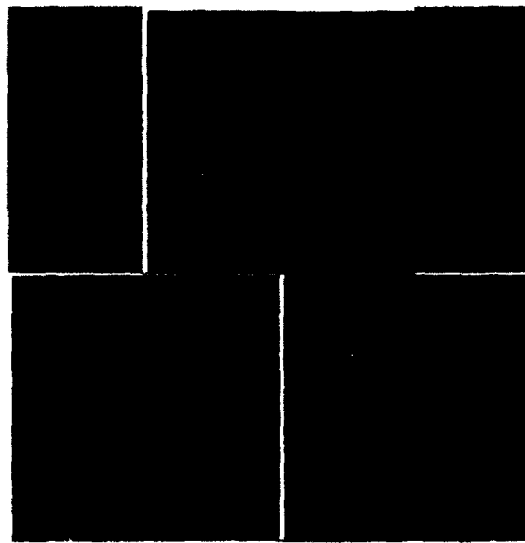
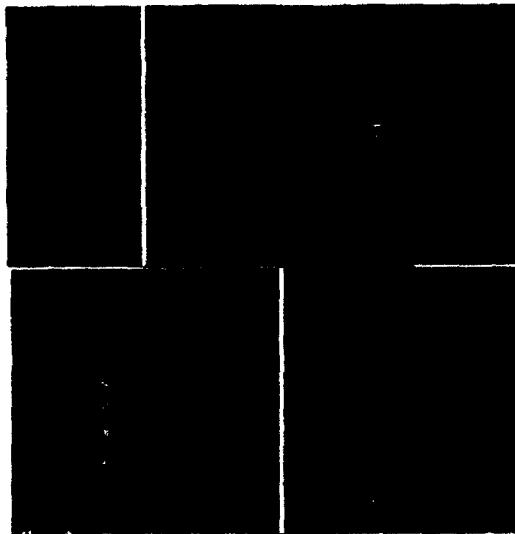


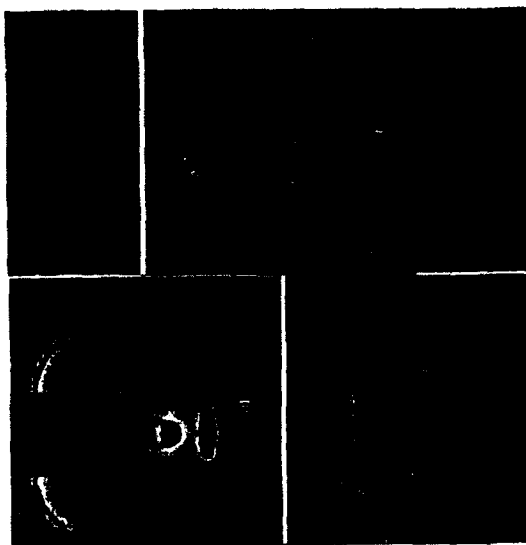
Figure 3.4.4 Structural response of the plow to blast load:
a) $t = 0$; b) $t = 200$ msec; c) $t = 400$ msec; d) $t = 600$ msec.



a. Predetonation stage



b. Detonation



c. Detonation product expansion

Figure 3.5.1 Pulsed detonation engine simulation: flow tracers.

3.5 PULSED DETONATION ENGINE

The main objective was the development of a revolutionary propulsion concept based on intermittent detonative combustion. Development of this concept will result in a new class of engines with performance surpassing those of small turbines at significantly reduced cost. SAIC's PDE research was noted in a recent article [Aviation Week, October 28, 1991, pages 68-89]. The PDE is currently considered as a candidate concept for numerous propulsion systems including the air-to-air missile, cruise missile, RPV engine, high altitude UAVs and others.

The codes developed under the UUGM program have enabled SAIC to conduct a detailed study of the PDE concept. The unstructured grids used in the simulations allowed us to describe the complex geometries of the detonation chamber and air inlets for a full missile configuration. Adaptive gridding allowed efficient and accurate simulation of the detonation and resulting shock waves interacting with the thrust-producing surfaces of the engine.

In Fig. 3.5.1 results are shown for the simulation of the PDE detonation cycle for a Mach 2 missile. Lagrangian flow tracers are used to track air and fuel trajectories in the engine. The figures demonstrate the sequence of stages in one PDE cycle. Shown in Figs. 3.5.1a-c are the fuel mixing stage, the detonation stage and the detonation products discharge stage, respectively. Detailed CFD analysis of various geometries and flow regimes allowed us to develop an understanding of the parametric dependence of the fundamental variables that determine the PDE performance.

3.6 SHOCK FOCUSING IN AIR

Research relating to focusing of shock and acoustic waves is of considerable practical interest for application to extracorporeal shock-wave lithotripsy (ESWL). A schematic of the cross section of such a reflector is shown in Fig. 3.6.1. Strong acoustic waves are generated in the left focal point of the ellipsoid by an instantaneous release of energy and are refocused at the right focal point. Ideally, focusing should be based on waves of acoustic intensity, since the nonlinear reflections of strong shock waves lead to significant distortions in wave propagation and impair simple geometrical focusing.

Figure 3.6.1 shows the computational domain and grid for the ellipsoidal reflector. Figure 3.6.2 shows the simulation results at time $t = 1.21 \times 10^{-6}$ sec. At this stage, the wave front that propagated to the left has undergone full reflection and the reflected wave propagates in the direction of the incident wave to the right. Figure 3.6.3 shows the pressure contours ($t = 8.41 \times 10^{-4}$ sec) when the maximum focused pressure is obtained in the system. The incident front has left the computational domain, and the maximum pressure occurs in a small volume in the vicinity of the right focal point. The maximum focused pressure has reached 1.37×10^5 Pa and is located 11 mm to the right of the focal point of the ellipsoid. In all the figures presented, the method of composite domain decomposition works extremely well, producing seamless solutions at the interfaces.

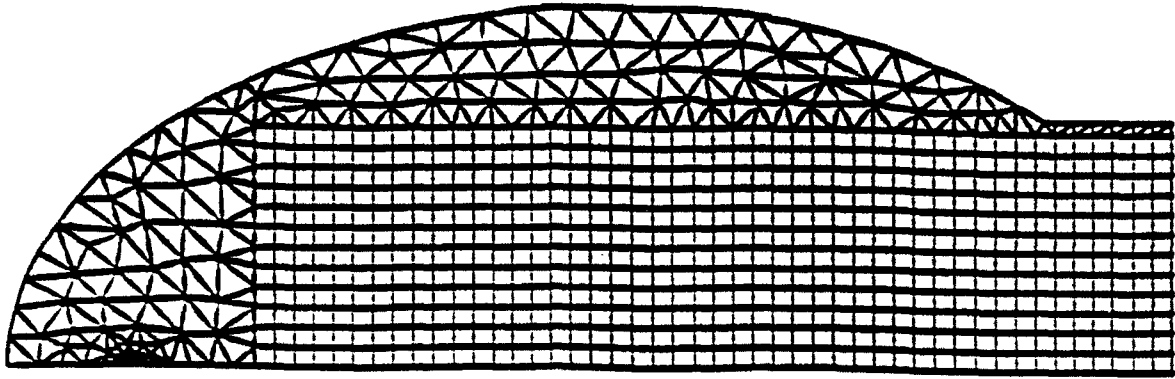


Figure 3.6.1a Hybrid structured/unstructured grid used for numerical simulation of ellipsoidal reflector.

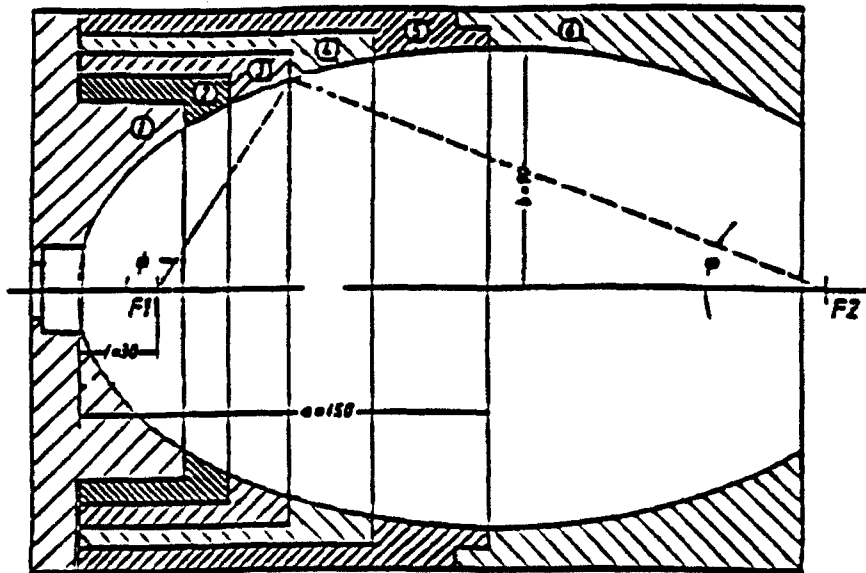


Figure 3.6.1b A schematic drawing of the center cross section of the ellipsoidal reflector.

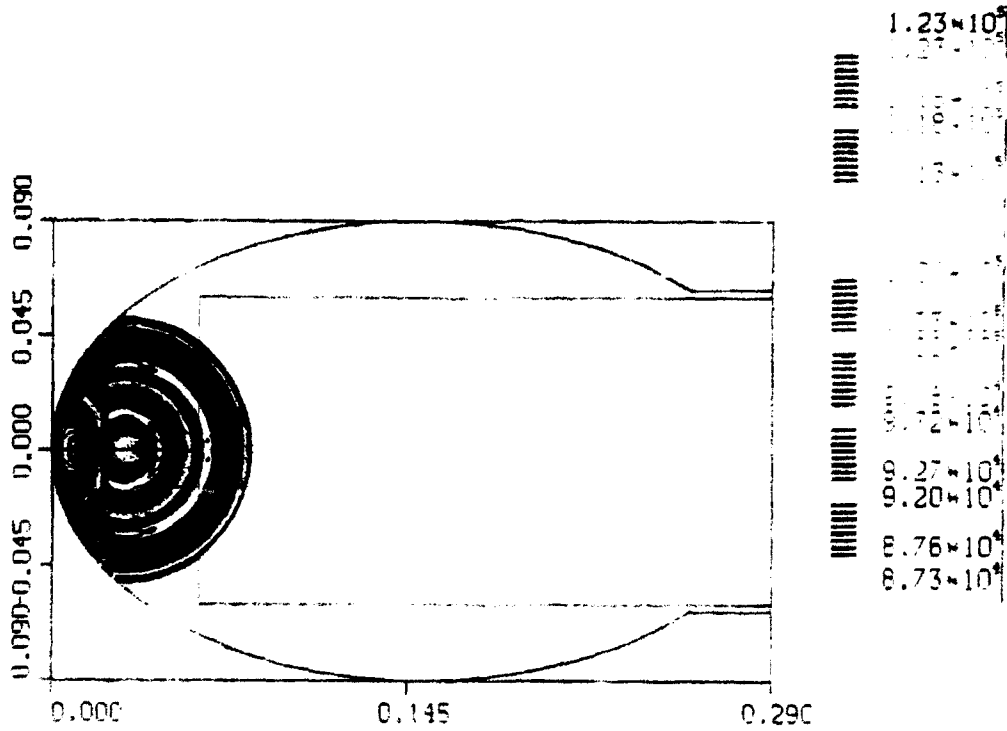


Figure 3.6.2 Pressure contours at time $t = 1.21 \times 10^{-6}$ sec showing the incident wave as reflected from the reflector wall.

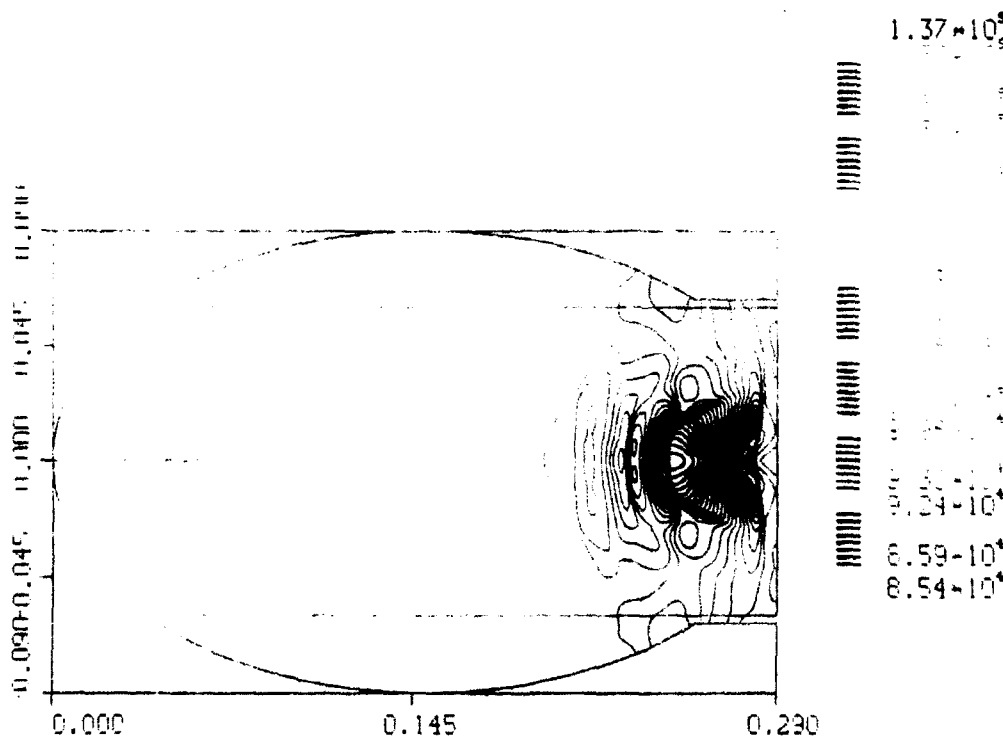


Figure 3.6.3 Pressure contours at time $t = 8.41 \times 10^{-4}$ sec showing the stage at which the maximum focused pressure is obtained.

3.7 NONIDEAL AIRBURST IN MULTIPHASE MEDIA

The main objective was to advance the understanding of the formation dynamics and microphysics of the multiphase flow of clouds developing as a result of a nuclear explosion. A main difficulty in analysis of nuclear cloud formation is the necessity to take into account physical phenomena that are interdependent and occur on vastly different scales. At about 30 seconds after a nuclear detonation, the cloud can be 4 km high and the shock wave will be at the distance of 10 km. The multiphase interactions that occur on a scale of 10-100 meters are very important and have to be accounted for.

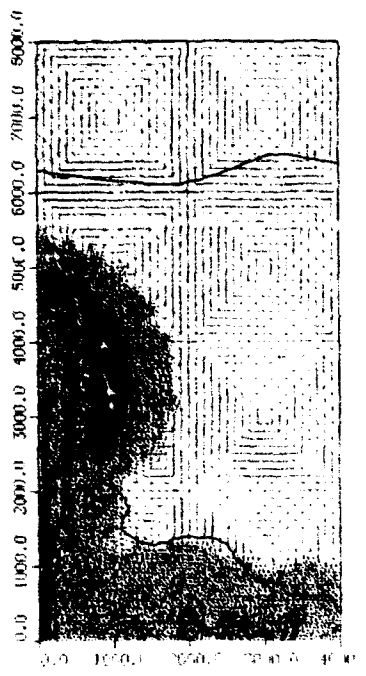
SAIC has developed a multiphase, multicomponent version of the AUGUST-2D code developed under the UUGM program. We use an explicit method for the solution of the multiphase flow described by equivalent Euler equations, and an implicit integration for simulation of the particle-fluid interactions. The grid adaptivity allows efficient and accurate simulation of this multiphase phenomenon. The grid adaptivity is used for adjusting the spatial scale of the domain decomposition to the scale required for accurate simulation of various physical interactions. Other code improvements such as introduction of the real-gas equation of state and Lagrangian particle tracing were employed to enable simulations and analysis of this complicated phenomena.

In Fig. 3.7.1a the computational domain and grid are shown for the nuclear cloud simulation. In this figure the temperature contours are overlaid on the unstructured grid. In Fig. 3.7.1b the particle density contours are shown for the same stage of the cloud evolution as in Fig. 3.7.1a. In Fig. 3.7.1c particle radius is shown for the same stage of the cloud evolution, and Fig. 3.7.1d shows locations of the Lagrangian tracers that mark evolution of the detonation products.

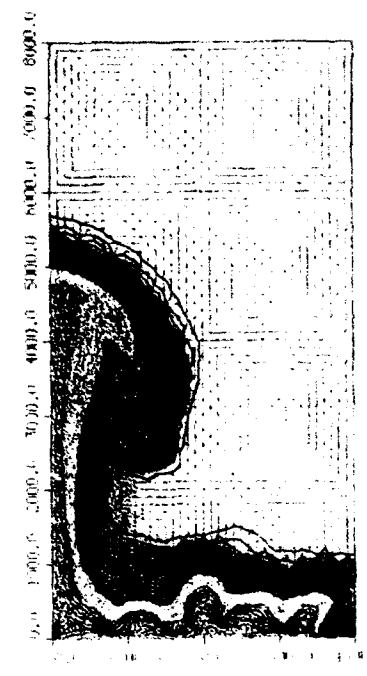
3.8 FLOW IN THE SARL WIND TUNNEL

One of the problems to which AUGUST 3D has been applied is that of modeling the SARL wind tunnel at Wright Laboratory. This example is a good test of the use of the Second-Order Godunov method to do nearly incompressible flow calculations. To illustrate the results, Fig. 3.8.1 shows the grid used for simulating the flow. The calculation was performed by specifying the inflow and outflow parameters and running the simulation to convergence. The run was performed at SAIC on the Stardent workstation and repeated on an Iris at FIMM. Figures 3.8.2 and 3.8.3 show the pressure levels in the tunnel. The results were visualized using AVS.

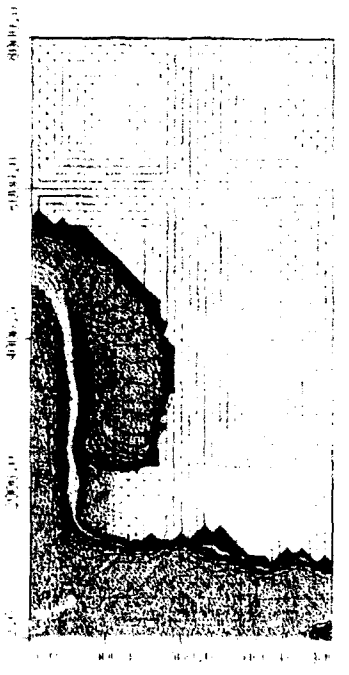
Figures 3.8.1 and 3.8.2 show two views of the pressure contours generated in a calculation of subsonic flow (Mach number 0.05). The results were confirmed by comparison with those obtained using a code with a structured grid, and by checking them against measurements.



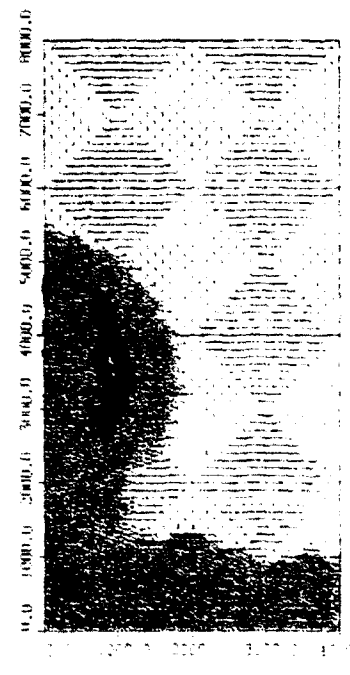
a. Temperature contours and grid.



b. Particle density.



c. Particle radius.



d. Lagrangian traces.

Figure 3.7.1 Formation of a radiative cloud. Multiphase simulation.

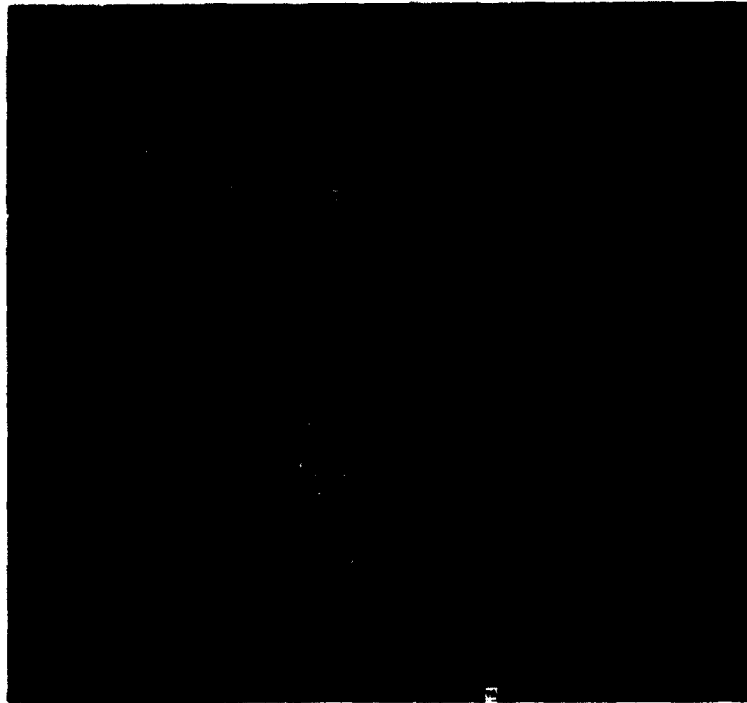


Figure 3.8.1 The unstructured grid used to simulate the SARL wind tunnel.



Figure 3.8.2 The pressure contours from the simulation of the SARL wind tunnel.

3.9 SHOCK ON DOUBLE WEDGE

A much more complicated problem, which has been extensively studied to benchmark and validate Euler solvers, is flow over a double wedge. This problem contains multiple fluid phenomena and is a stringent test for any solver. It includes shock formation, a Mach stem, rarefaction, a slip line, vortex generation and rollup, and is transient in nature. To validate our direct dynamic refinement method in AUGUST, we simulated a Mach 2.85 shock wave propagating in a channel and impinging on a symmetric 45° wedge, and also a Mach 8.7 shock impinging on a symmetric 27° wedge.

Both of these compared well with experimental results. Figure 3.9.1 shows an interferogram taken from Glaz et al.,¹⁷ showing the $M = 8.7$ shock interacting with the front surface of the 27° wedge. Our results are shown in Figs. 3.9.2 - 3.9.4. The first of these illustrates the grid and density when the shock is on top of the wedge. The shock is well resolved and the grid is well adapted in the vicinity of important features and coarsened in the region that the shock has passed through. The next two figures show the evolution of the flow and the grid after the wedge where comparison can be made with the experimental results. AUGUST produces no artificial features and recovers the phenomenology seen in the experiment.



Figure 3.9.1 Experimental interferogram of a shock hitting a 45° corner at $M_s = 2.85$.

In the figures showing the triangular grids, the area of a triangle in the dense region of the grid is roughly 100 times smaller than the area of a triangle in the initial grid. The figures show that the grid adaptivity is capable of capturing the flow gradients including shocks, contact discontinuities and slip lines. Formation of a triple point of the Mach reflection, slip line and strong vortex formation are seen in Fig. 3.9.2a. In fairness, most of the flow phenomena that is captured by AUGUST have also been captured by other CFD schemes.¹⁸ However, the accuracy estimated for the AUGUST numerical calculations in this example is on the order of 4%, equal to the accuracy of the experimental observations.

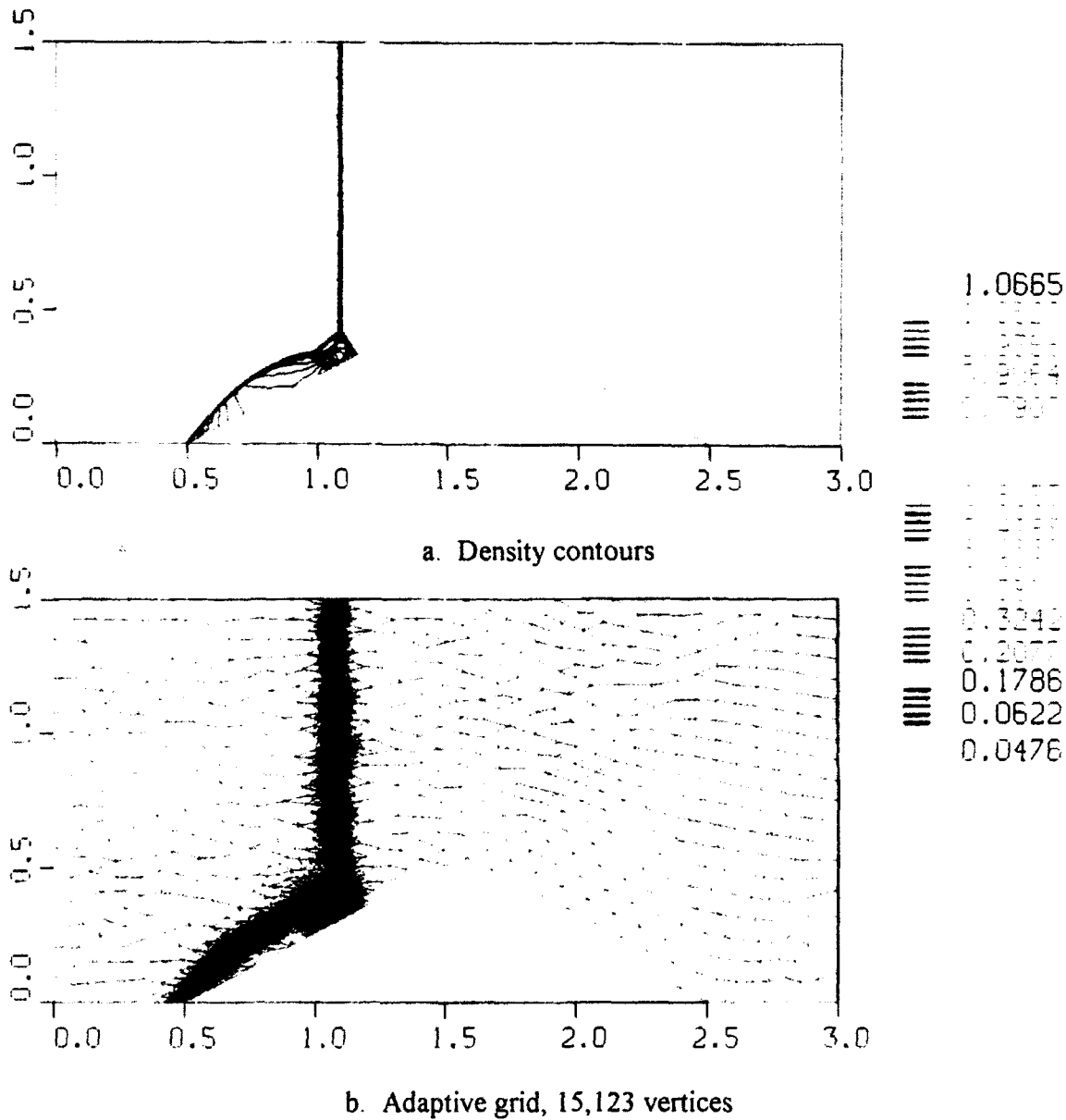
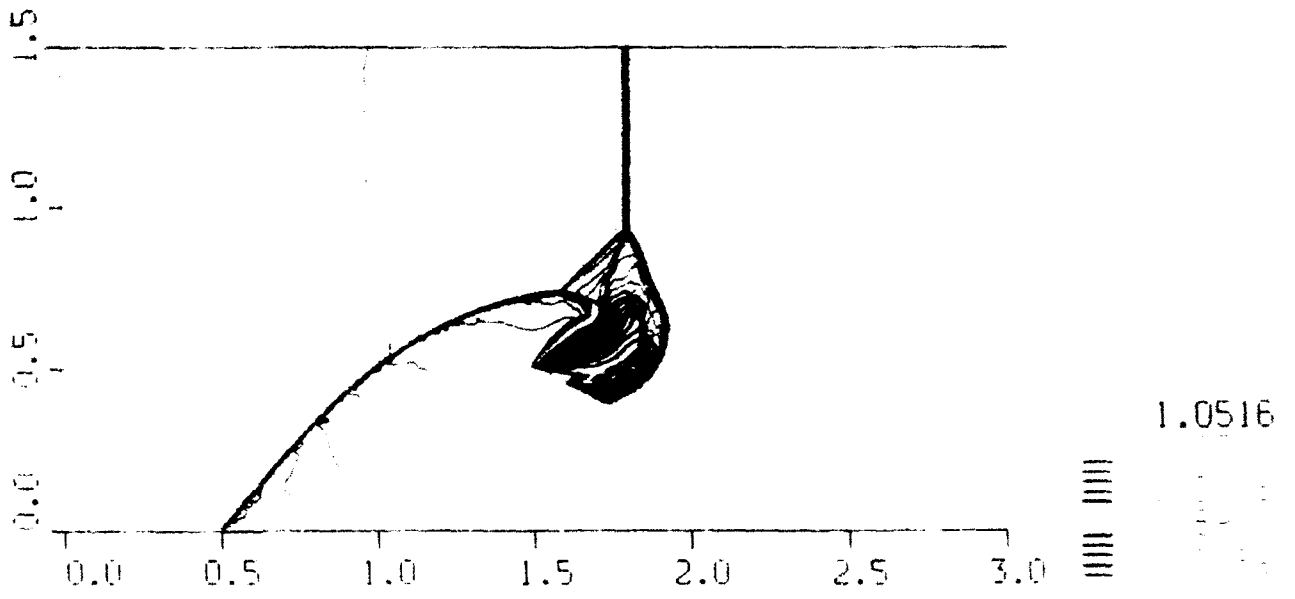
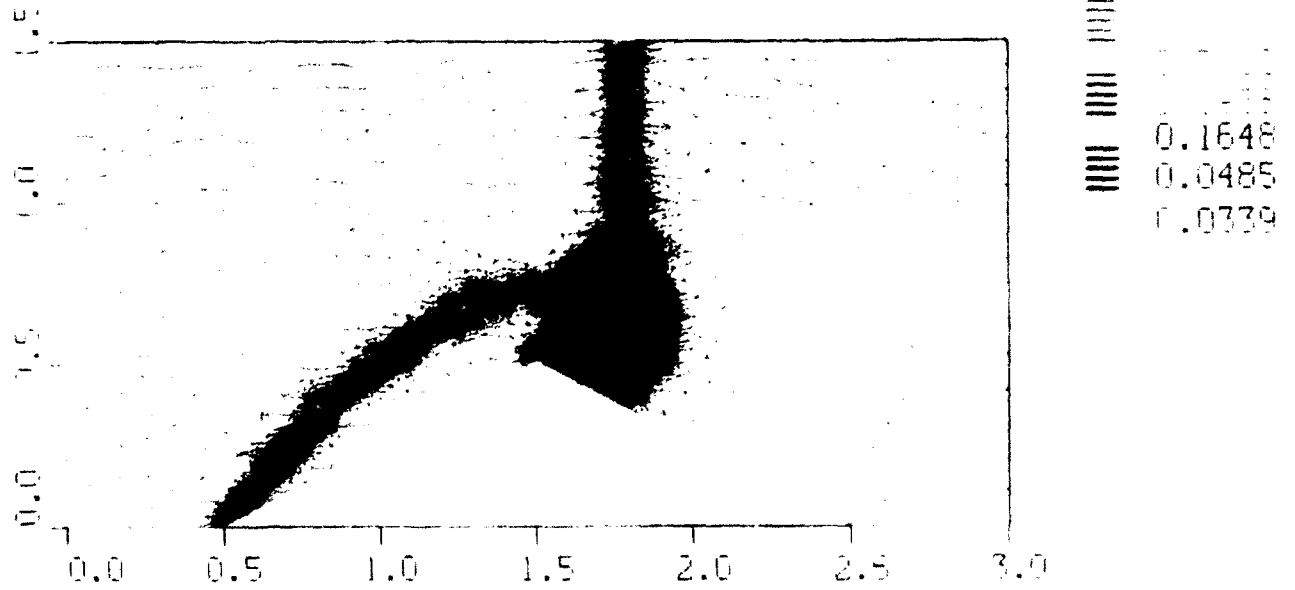


Figure 3.9.2 Interaction of a Mach 8.7 planar shock wave with a 27° double ramp: Mach reflection stage.

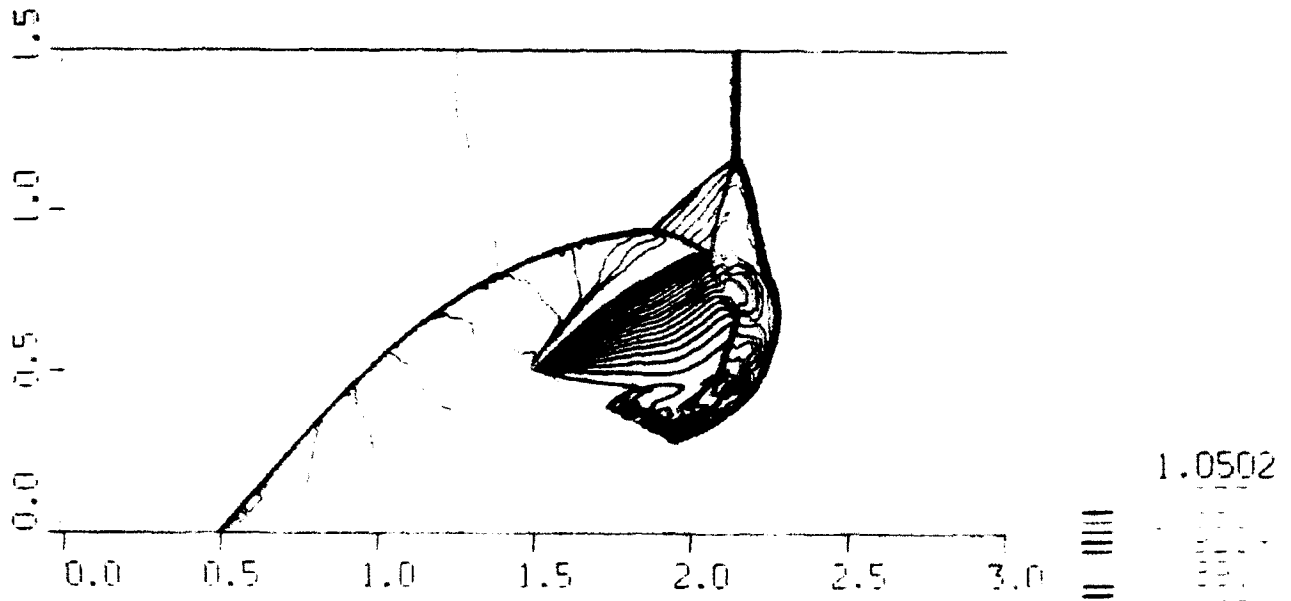


a. Density contours

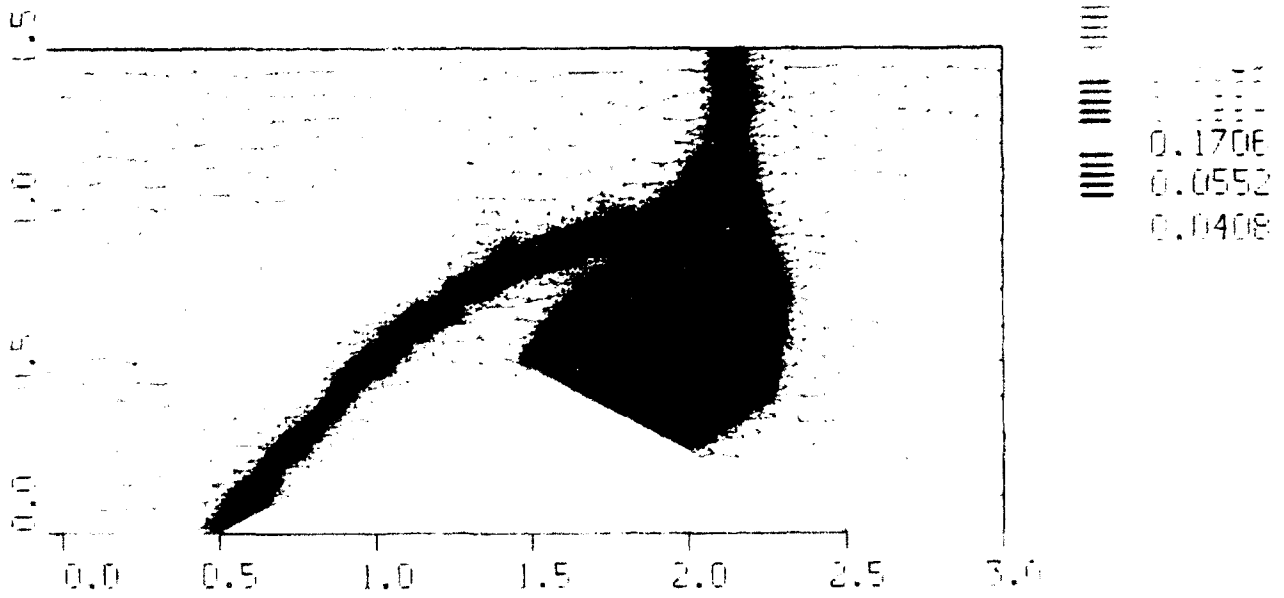


b. Adaptive grid, 27,132 vertices

Figure 3.9.3 Interaction of Mach 8.7 planar shock wave with a 27° double ramp: start of the diffraction stage.



a. Density contours



b. Adaptive grid, 46,462 vertices

Figure 3.9.4 Interaction of Mach 8.7 planar shock wave with a 27° double ramp: shock diffraction stage.

3.10 SUPERSONIC SPRAY COATING DEVICES

In this section we present the results of an application of the UUGM simulation technology to a sample problem involving spray-coating devices. Here we only treat the aerodynamic flow of particles in a high temperature gas which is moving supersonically; we consider a reasonably complex geometry including a simulated surface that is the substrate to be coated. The details of the surface interaction resulting in deposition are not treated in this example.

In Fig. 3.10.1 the computational domain and grid are shown for a model supersonic jet sprayer device that includes reactor nozzle, solid particle injector, and expansion nozzle. Also shown in Figure 3.10.1 is a perforated flat surface substrate placed in the flow field. The high-velocity high-temperature flow stream exiting the reactor nozzle accelerates the injected particles. The particles are heated during acceleration, melt, then expand with the flow in the nozzle, gain more speed, and finally impinge onto the surface. Details of the flow-surface interaction (here without boundary layers taken into account) will strongly affect the uniformity with which the surface will be "coated" by the particles carried by the flow.

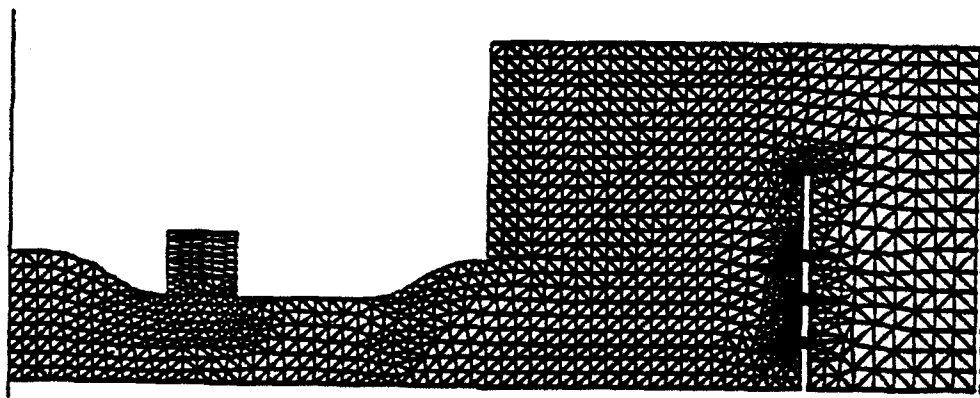


Figure 3.10.1 The figure shows the initial computational grid for the jet spray simulation demonstration. Shown are the nozzle, injection region and target surface depicted as a flat plate with perforations, oriented perpendicular to the mean spray flow. The boundary conditions used for the sample simulation were: $V_g = 1000$ m/sec, $\rho_g = 0.1$ kg/m³, $T_g = 3500$ K at the inlet of the reactor nozzle; $V_g = 1500$ m/sec, $\rho_g = 0.3$ kg/m³, $T_g = 1500$ K, $V_p = 1500$ m/sec, $T_p = 1500$ K, $N_p = 2000$ at the inlet of the reactor nozzle.

To trace the motion of the particles in the plasma spray device and the interaction pattern with the target surface we have injected Lagrangian "marker" particles (massless but moving with the local flow speed) in the particle injector flow stream. In Fig. 3.10.2 results are shown in the form of marker particle locations. To monitor the particle temperatures we have introduced particle coloring, where the color defines the local particle temperature. Thus one can evaluate the evolution of the particle temperature by observing the particle color transition. This coloring scheme can be used to show other parameters such as particle residence time or density. This represents a simple method of visualization that we have used successfully in past UUGM simulations.

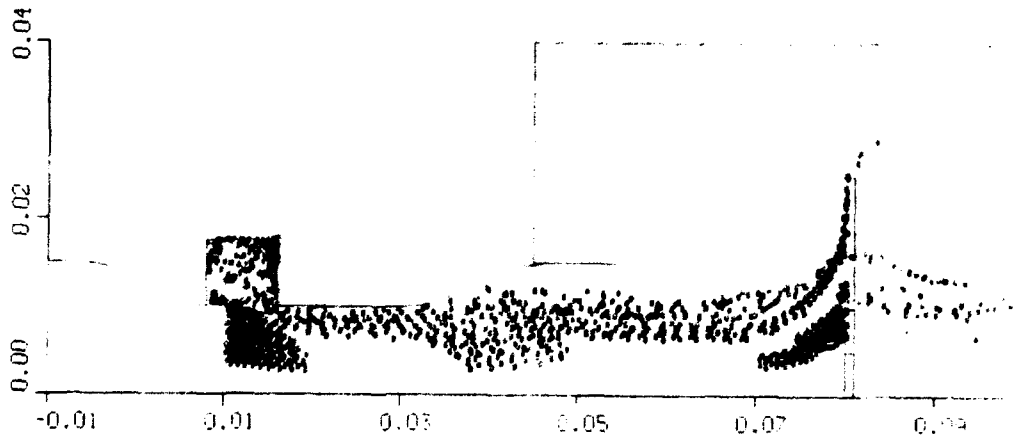


Figure 3.10.2 Lagrangian marker particles are shown in color representing the evolution of injected particle temperature as a function of particle position and time in the jet spray stream.

In Fig. 3.10.3 simulation results for the steady state are presented in the form of gas temperature contours for the jet spray system. Here it is possible to observe a very large temperature variation in the nozzle. The cold gas that is injected with the particles remains at the edge of the jet stream. At the same time the main jet cools through the expansion in the nozzle from 3500°K to 2000°K , and then undergoes a series of expansions and compressions in the system of shock waves created by overexpansion of the supersonic jet. Figure 3.10.3 also shows a nonuniform temperature distribution on the surface that is partially created by the gas flow through the perforated holes.

In Figs. 3.10.4 and 3.10.5 simulation results are shown for the density and pressure contours. Here we can observe the formation of several diamond-shaped shock structures as a result of supersonic flow over expansion. However, for the flow regimes in our simulation these shocks do not lead to a higher rate of mixing by injected cold gas with particles and the main hot gas stream. This can be noticed in the density contours, where one clearly observes that the high-density cold gas does not penetrate the main hot jet flow. By changing the condition (injection pressure, angle of entry, etc.) of the injected flow one can improve mixing, thus achieving higher particle temperatures and velocity.

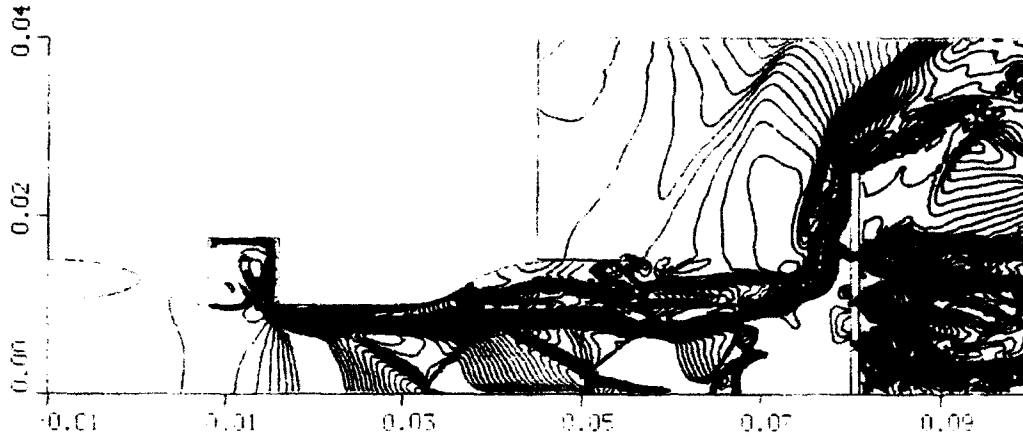


Figure 3.10.3 Gas temperature contours in the jet spray stream. The maximum temperature is 3500°K and the minimum is 600°K.

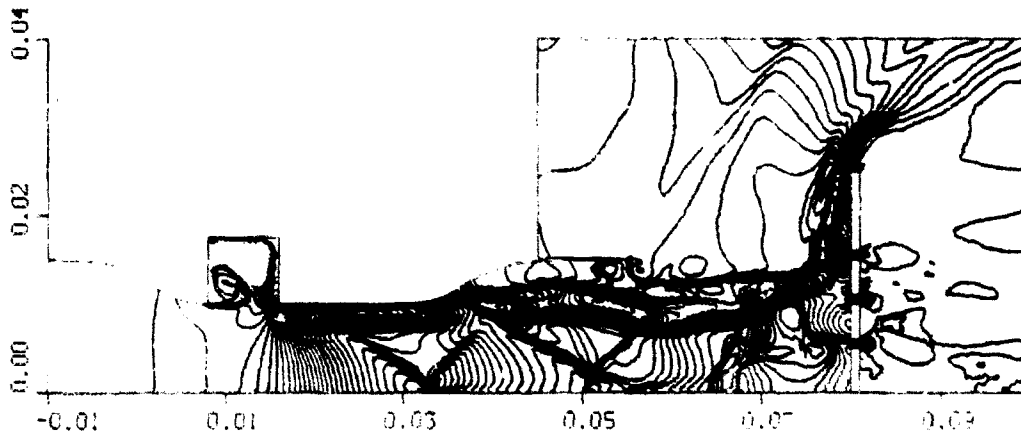


Figure 3.10.4 Gas density contours in the jet spray stream. The injected stream and the main flow mix poorly. The diamond patterns describe the shock wave pattern resulting from the flow's overexpansion.

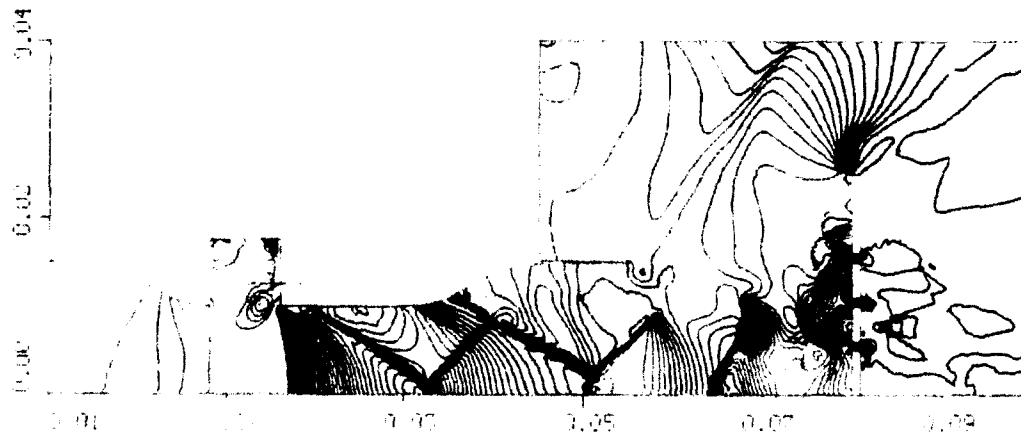


Figure 3.10.5 Pressure contours in the jet spray stream. The diamond patterns show that supersonic flow is maintained near the vicinity of the target surface.

3.11 DUSTY FLOW OVER A CYLINDER

A numerical study of two-phase compressible flow has been performed for the reflection and diffraction of a shock wave propagating over a semicircular cylinder in a dusty gas. The following model was used to derive the governing equations:

- (1) The gas is air and is assumed to be ideal;
- (2) The particles do not undergo a phase change because for the particles considered here (sand) the phase transition temperature is much higher than the temperatures typical for the simulated cases;
- (3) The particles are solid spheres of uniform diameter and have a constant material density;
- (4) The volume occupied by the particles is negligible;
- (5) The interaction between particles can be ignored;
- (6) The only force acting on the particles is drag and the only mechanism for heat transfer between the two phases is convection. The weight of the solid particles and their buoyant force are negligibly small compared to the drag force;
- (7) The particles have a constant specific heat and are assumed to have a uniform temperature distribution inside each particle.

Under the above assumptions, separate equations of continuity, momentum, and energy are written for each phase. The interaction effects between the two phases appear as source terms on the right-hand sides of the governing equations. The two phases are coupled by interactive drag force and heat transfer.

The objectives of the study were (a) to solve the two-phase compressible flow field and compare the simulation with available experimental results; (b) to observe and investigate the reflection and diffraction wave patterns when a shock wave propagates over a semicircular cylinder in a dusty gas, with particle radius and loading as parameters.

To test the accuracy of the two-dimensional computation, we first computed the pure gas flow case of shock wave reflection and diffraction over a semicircular cylinder. We then compared the simulation with experimental results. Shock wave reflection on a wedge has been extensively studied by many researchers (see e.g., review papers of Bendor and Dewey¹⁸ and Hornung.¹⁹ As one can see from Fig. 3.11.1, the results show excellent quantitative and qualitative agreement between the numerical simulation and experimental results.

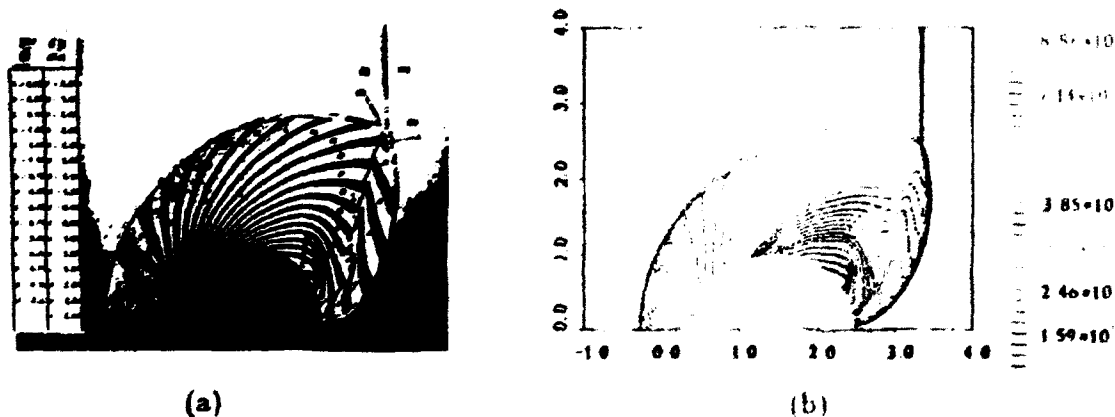


Figure 3.11.1 Comparison for $M_s = 2.8$ pure-gas flow: (a) interferogram from experiment; (b) density contours from present calculation.

In the two-phase simulation a planar shock with $M_s = 2.8$ propagates into an area of dusty gas and impinges on a semicircular cylinder. The interface between pure air and dusty air is located at $x = 0.0$ of the computational domain. The area of the dusty air contains a semicylinder with a radius of 1m. The size of the computational domain, initial parameters of the gas, parameters of the incoming shock, size of the semicylinder and its location in the computational domain, are the same as in the reflection and diffraction simulation in the pure gas case. The main objective of this set of simulations was to study the effects of particle size and particle loading on the parameters of the reflected and diffracted shock waves.

The first set of simulation results is shown for the case with dust parameters $r_p = 10\mu\text{m}$ and $\rho_p = 0.25 \text{ kg/m}^3$. The gas parameters and the parameters of the incoming shock wave were the same as in the pure gas case presented above. In Figs. 3.11.2a and 3.11.2b, the particle density and gas density contours are shown at the stage where significant diffraction has taken place and the shock front is approaching the trailing edge of the cylinder. To study the influence of particle loading on the dynamics of reflection and diffraction, we have simulated the case with a dust density of $\rho_p = 0.76 \text{ kg/m}^3$ and with $r_p = 10\mu\text{m}$. To examine the effect of particle size on the reflection-diffraction process, we simulated a case where the particle loading and gas flow conditions were the same as in the previous case with particle density $\rho_p = 0.76 \text{ kg/m}^3$, but the particle size was $r_p = 50\mu\text{m}$ (Fig. 3.11.3).

On the basis of these calculations we reached the following conclusions:

- (1) For a two-dimensional pure-gas flow, numerical results agree well with existing experimental data qualitatively and quantitatively, indicating that the gas phase is accurately simulated by the adaptive grid technique;

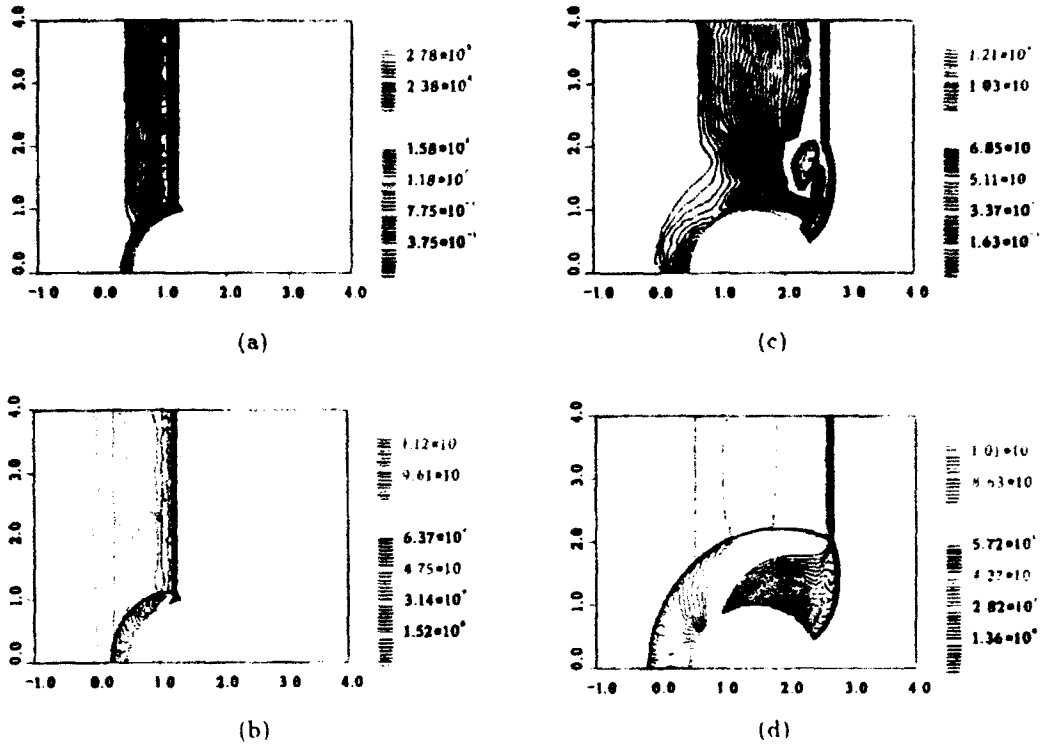


Figure 3.11.2 Density contours for the case $M_S = 2.8$, $\rho_p = 0.25 \text{ kg/m}^3$, $r_p = 10 \mu\text{m}$ at two different times: (a) particle density at t_1 , (b) gas density at t_1 ; (c) particle density at t_2 , (d) gas density at t_2 .

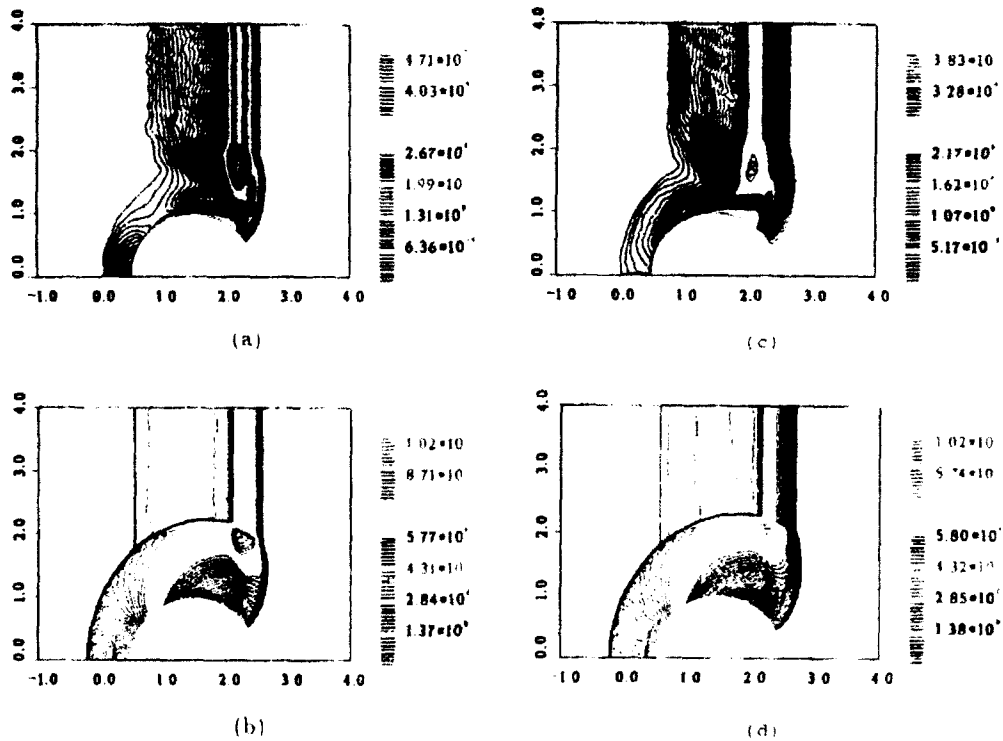


Figure 3.11.3 Density contours for the case $M_S = 2.8$, $\rho_p = 0.76 \text{ kg/m}^3$, for two different particle sizes: (a) particle density and (b) gas density for $r_p = 10 \mu\text{m}$; (c) particle density and (d) gas density for $r_p = 50 \mu\text{m}$.

(2) Particles in the gas can have a profound effect on the shock wave reflection and diffraction pattern, which is a function of particle size and loading. The less the particle loading, the less the influence of particle on the flow field;

(3) In the three simulation cases, particles accumulate behind the "back shoulder" of the semicircular cylinder due to the effect of particle inertia and the rarefaction wave;

(4) For different particle sizes at fixed particle loading, the larger particle will have a longer relaxation zone and less accumulation at the "back shoulder" and behind the incident shock. The gas density contours show a less distinguishable slip line in the small particle case than in the large particle case.

3.12 IMAGE PROCESSING

Very recently, there have been exploratory efforts in image processing based on nonlinear methods. If the purpose of an enhancement process is to highlight the edges of an image, then the technique used in the frequency domain is usually highpass filtering. An image can be blurred, however, by attenuating the high-frequency component of its Fourier transform. Since edges and other abrupt changes in the gray levels are associated with high-frequency components, image sharpening can be achieved in the frequency domain by a highpass filtering process, which attenuates the low-frequency without disturbing high-frequency information in the Fourier transform. The primary problem with this technique is that an ideal discontinuity has an infinite spectrum of frequencies associated with it. When filtering is applied, some frequencies are cut off, leading to a loss of edges in the image.

In computational fluid dynamics (CFD) similar problems exist in simulating flows with discontinuities. The problem of simulating flows with discontinuities is less forgiving, since an incorrect calculation usually leads to a complete distortion of the flow field. This has led CFD scientists to develop sophisticated algorithms that identify and preserve discontinuities while integrating the flow field in the computational domain. In the image domain, sharpening is usually done by differentiation. The most commonly used methods involve the use of either gradients or second derivatives of the pixel information. Central differencing is usually used to calculate the derivatives. CFD research has shown that this strategy will lead in many cases to smearing of the flow discontinuities (analog of the image edges in image enhancement).

A new and unique image sharpening method based on computational techniques developed for AUGUST has been developed. Preliminary experience shows that it can enhance image edges and deconvolve images with random noise. This indicates a potential application for image deconvolution from sparse and noisy data resulting from measurements of backscattered laser-speckle intensity.

The Second-Order Godunov Method used in AUGUST was developed from an understanding of the phenomenology of signal propagation in gasdynamical systems. The numerical algorithm implementing this method is not analytical and contains a set of steps that can be regarded as wave filters. These filters are designed to not smear the discontinuity (edge), suppress the spurious oscillations, and propagate the relevant signals through the system. The following algorithmic steps are performed to advance the solution for a single iteration in the Second-Order Godunov Method:

1. Local Extrapolation
2. Monotonicity Constraint
3. Characteristics Constraint
4. Riemann Problem Solution
5. Integration

Most of these steps have an analog in conventional image processing methods. Here we will give an explanation of the function of each algorithmic step of the Second-Order Godunov Method and where applicable, will point to its possible analog in conventional signal processing techniques.

Step 1 consists of extrapolation of the values in the computational grid (pixel) cell to the edges of the cell. Linear or nonlinear extrapolation can be used. This step is analogous to the standard edge-sharpening techniques used in image processing, with one important difference: the extrapolation is done not for the value itself but for its flux (change of value across cell boundary).

Step 2 includes a monotonicity constraint for the values at the cell edges. This is analogous to the nonlinear technique of locally monotonic regression only recently introduced for signal processing.

Step 3 subjects the values at the edges to the constraints derived from a solution of the one-dimensional characteristics. This step assures that the values at the edges have not been extrapolated from directions inconsistent with the characteristic solutions. This prevents extrapolation as well as smearing or overshoot of the discontinuities. For the image-processing application, this can be regarded as a form of automatic edge detection step where the shock waves are associated with the edges of an image.

Step 4 uses an exact solution of the system of the gasdynamic equations for calculation of the flux values based on the extrapolated values of the parameters at the left and right side of the edges. This step has no analogy in image processing. However, since the analytical solution includes discontinuities, an exact calculation of the flux at the edge location is allowed, even if this flux is calculated through a discontinuity.

Step 5 consists of finite-volume integration of the system of conservation laws. Here, the image is effectively treated as a flow field: the flux integration serves as a smoothing filter from the image perspective.

The effect of these steps is equivalent to the application of a unique filter stack with proven properties of discontinuity preservation and robustness.

The field of gray scale intensity of an image can be translated into a flow field. To every image pixel we assign to the corresponding cell of the computational domain values of the gasdynamical parameters proportional to the values of the gray scale. Our understanding of the basic gasdynamical processes plays a major role in completing the analogy. Appropriate mapping of the image gray scale intensity into a flow field creates conditions favorable for the formation or enhancement of field discontinuities. For example, a shock wave reflecting from a wall or a contact surface can increase in strength, or two colliding flow streams will produce a contact surface that will become stronger in time. If we have a numerical technique to resolve these discontinuities accurately, then with successive numerical integration of the flow field, the discontinuities will sharpen as the solution evolves in time. Then by inverse mapping of the flow field to the image gray scale field, we can reconstruct an enhanced image.

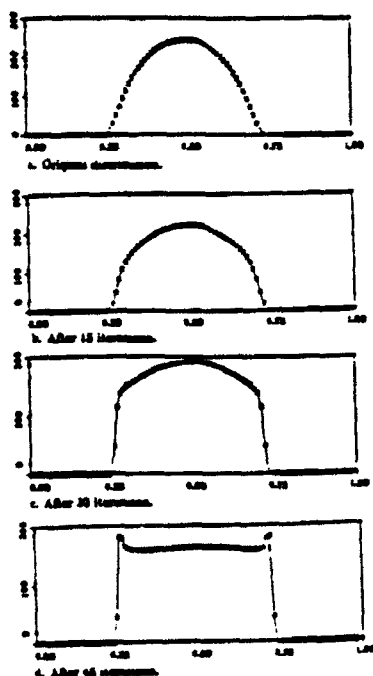


Figure 3.12.1 Edge enhancement for a sinusoidal distribution without noise.

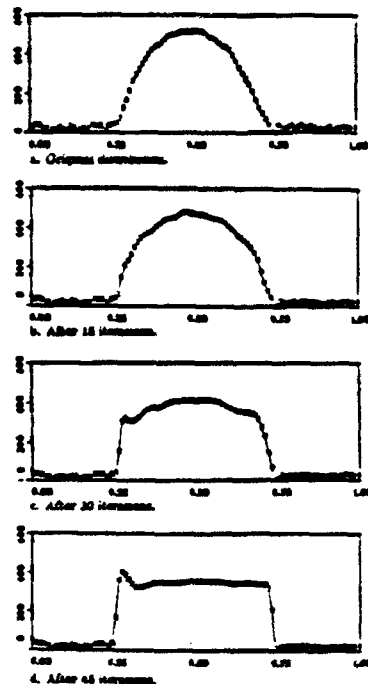


Figure 3.12.2 Edge enhancement for a sinusoidal distribution with 10% intensity random noise.

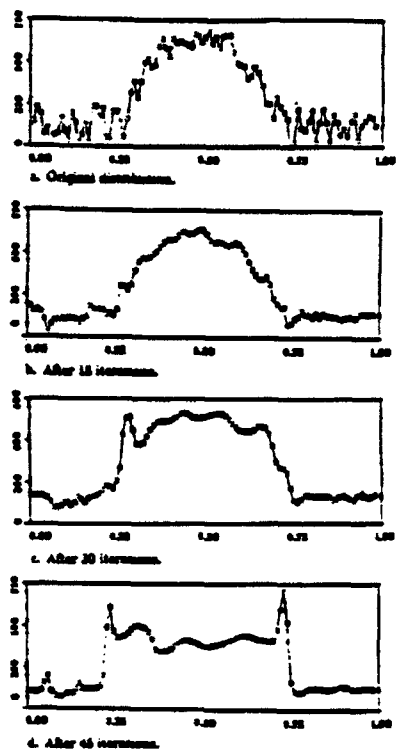


Fig. 3.12.3 Edge enhancement for a sinusoidal distribution with 50% intensity random noise.

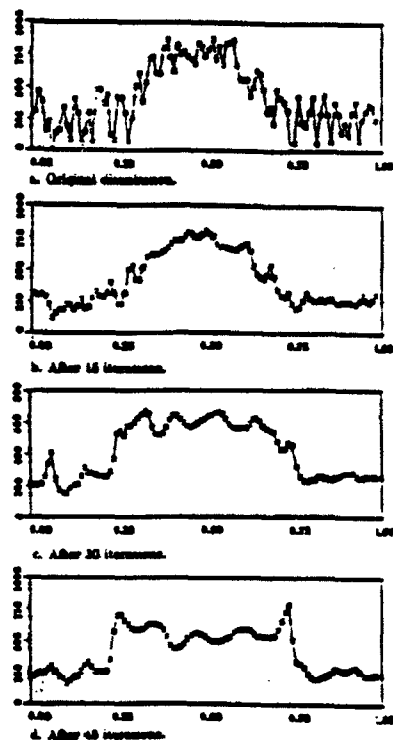


Fig. 3.12.4 Edge enhancement for a sinusoidal distribution with 100% intensity random noise.

Applications have been made to two-dimensional images derived from satellite reconnaissance and gamma-ray medical diagnostics (see Appendix C). Note that the images shown there are distorted by the xerographic process used to reproduce these illustrations, which also act as a nonlinear filter but is not tuned to these images.

Analogous extensions of nonlinear CFD techniques can be used for image compression.

3.13 DETONATION IN A MULTIPHASE MEDIUM

In this study the main subjects were the initiation, propagation, and structure of detonations occurring when combustible particles are intentionally or unintentionally dispersed into the air. Formation of this potentially explosive dust environment and the properties of its detonation are of significant practical interest in view of its destructive or creative effects. Previous experimental and theoretical studies of these phenomena addressed only homogeneous particle/oxidizer mixtures. However, intentional or accidental processes of the explosive dust dispersion always lead to inhomogeneous particle density distribution.

On the other hand, some industrial methods of explosive forming rely on detonation of explosive powder. This powder can be deposited as a thin layer over the surface area of the forming metal, with a residual concentration in the vicinity of the layer.

When the detonation wave is generated in a homogeneous mixture by "direct initiation," it starts with a strong blast wave from the initiating charge. As the blast wave decays, combustion of the reactive mixture behind its shock front starts to have a larger role in support of the shock wave motion. When the initial explosion energy exceeds some critical value, transition to steady state detonation occurs. In explosive dust mixtures with a nonuniform particle density, the initiation dynamics is significantly more complicated. The critical initiation energy sufficient for one of the explosive particle density regions is not necessarily adequate for other regions. We have demonstrated that the phenomenology of these interactions is distinctly different from the classical studies of multilayer detonations in gases. This is primarily because the energy content of adjacent layers in a typical multilayer experiment varies by a factor of two or four, whereas the energy content in explosive dust/air mixtures can vary by several orders of magnitude.

At present the physics of the energy release mechanisms in solid particles/air mixtures is not clearly understood. This can be attributed to the obvious difficulties of making a direct non-obtrusive measurement in the optically thick environment typical for this system. The chemical processes of single-particle combustion, which mainly occur in the gaseous phase, are significantly faster than the physical processes of particle gasification or disintegration. Thus, in the multiphase mixtures, the rate of energy release is mostly determined by physics of particle disintegration. It is very difficult to describe the details of particle disintegration in the complex environment prevalent behind the shock or detonation wave. Fortunately, in most cases of multiphase detonation, only the main features of the particle disintegration dynamics need to be captured to describe the phenomena.

In this work we considered solid particles consisting of explosive material. Two-dimensional simulations were done for the system of low particle density concentration clouds and ground layers formed by high concentrations of the RDX powder. We examined three cases of ground layer density distribution: a fourth power distribution with 12 mm above ground with a maximum density on the ground of 800 kg/m^3 ; a uniform 25-mm layer with a density of 100 kg/m^3 ; and a 12-mm uniform layer with a density of 250 kg/m^3 . In all these cases, the weight of the condensed phase per unit area was the same, which allowed examination of the effects of the particle density distribution on detonation wave parameters.

Figure 3.13.1 shows a setup for a typical two-dimensional simulation. Here the computational domain is $25\text{cm} \times 25\text{cm}$. The explosive powder density is distributed according to the 4th power law of the vertical distance, starting from the ground where the density is 800 kg/m^3 , and rising to 1.2cm, where the density is 0.75 kg/m^3 . From this point to 25cm height, the density is constant and equal to 0.75kg/m^3 . The density distribution is uniform in the x direction.

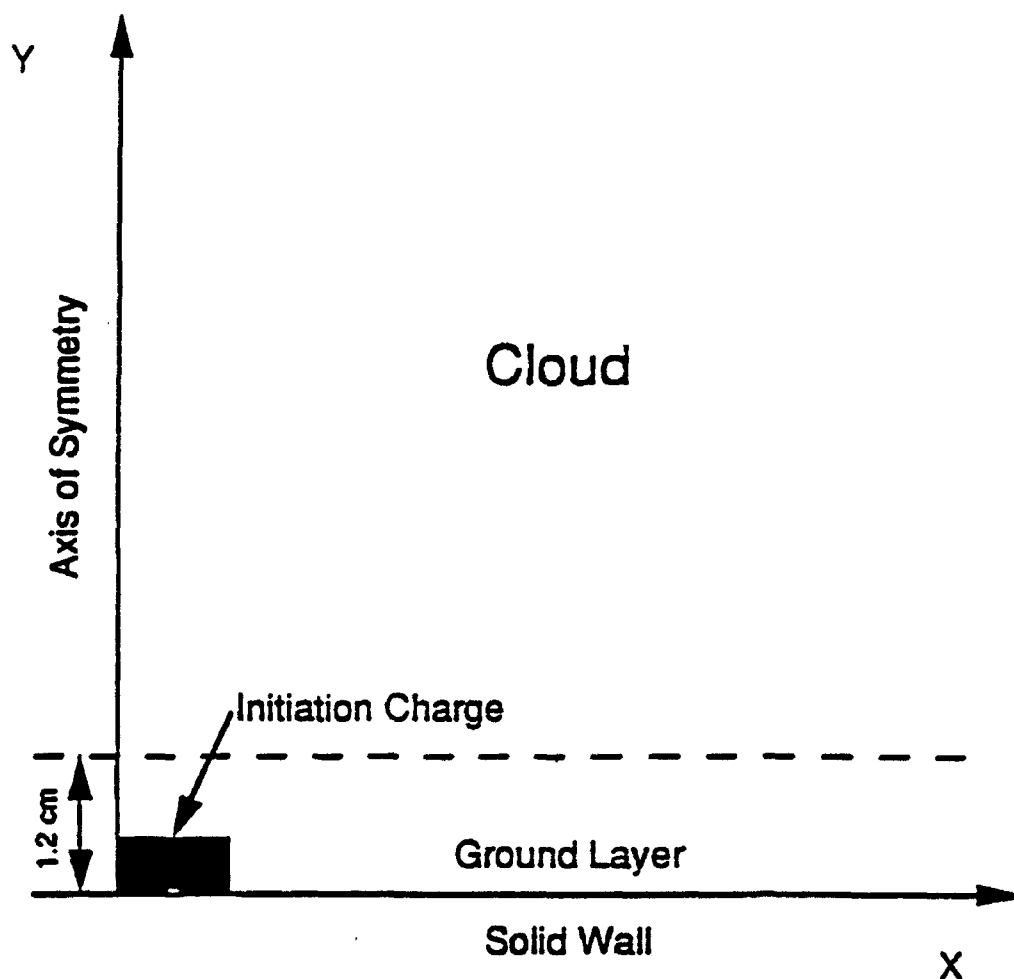


Figure 3.13.1 Computational domain and boundary conditions.

In all three cases, the detonation wave in the cloud in the computational domain was significantly overdriven and did not play an important role. We estimated that the self-sustained regime of the detonation wave in the cloud for the examined cloud concentrations can occur only at the distances of 2-3m above ground. At the same time, the particle density distribution in the layer determines the dynamics of the detonation wave as well as the pressure on the ground.

In all three two-dimensional simulations, we observed a very distinctive shape of the detonation wave front in the vicinity of the layer. In this area, the overdriven detonation in the cloud is preceding the detonation wave in the ground layer. This feature of the detonation front can be explained by the fact that the energy released in the ground layer detonation wave produces a faster propagating shock wave in the dilute cloud than in the ground layer which is heavily loaded with solid particles. However, these structures were not observed experimentally, and more studies are needed to examine their parameters.

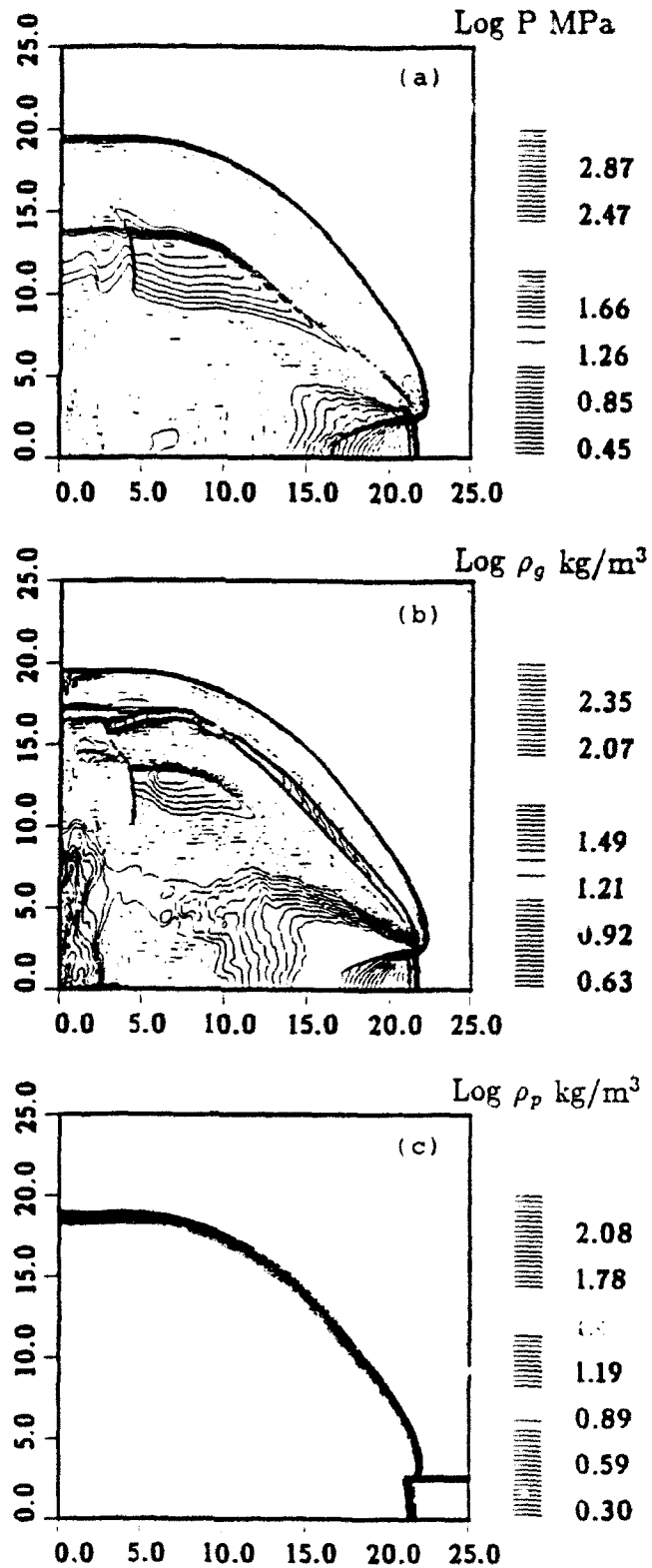


Figure 3.13.2 Explosive initially localized in 2.5-cm layer at constant density of 100 kg/m^3 . Density in the cloud is 0.75 kg/m^3 . (a), (b), and (c) are gas pressure, gas density, and particle density at $66 \text{ } \mu\text{sec}$, respectively.

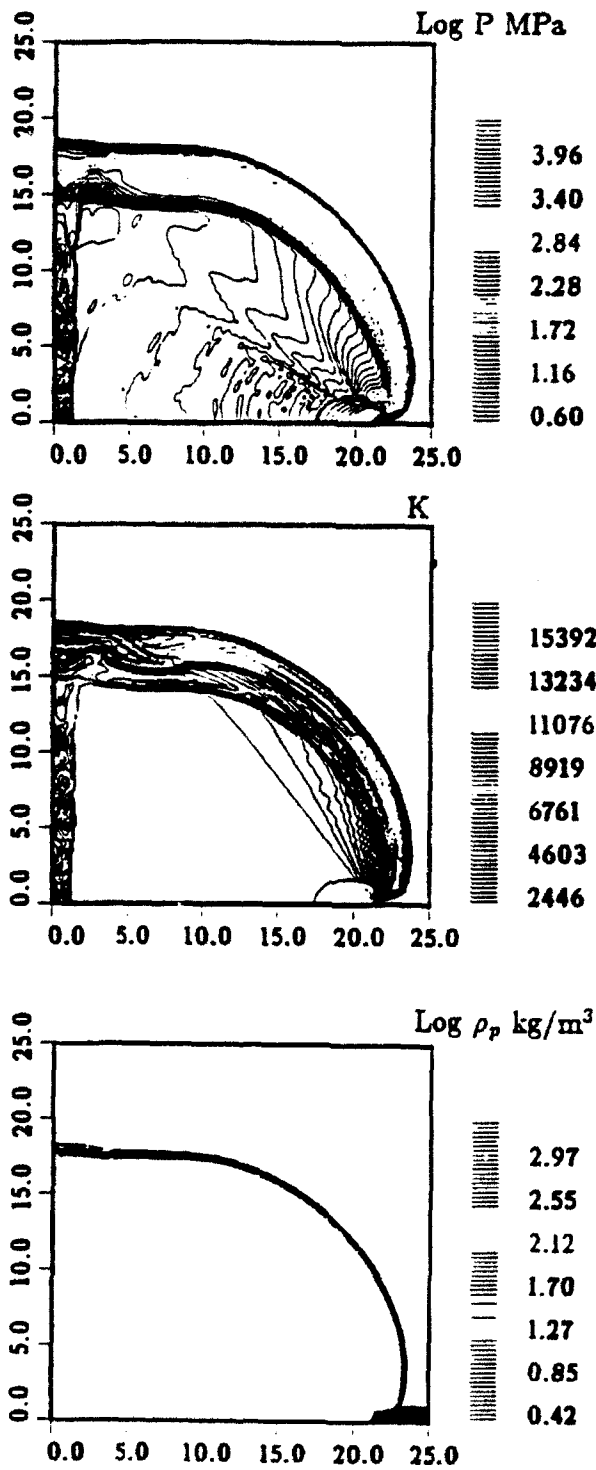


Figure 3.13.3 Particle density distributed in layer in accordance with the fourth power of height. Gas pressure, temperature, and particle density at 55 μ sec, respectively.

4. CONCLUSIONS

The AUGUST-2D and AUGUST-3D adaptive unstructured CFD simulation codes, developed under SAIC's UUGM (through a contract form ARPA's Applied and Computational Mathematics Program) program have been tested through the use of standard CFD benchmark test cases and have been applied to a wide range of academic problems for a variety of end-users. In most cases where these codes have been applied, significant improvements in accuracy, resolution, and ease of use have been noted. Use of the Second Order Godunov flow solver algorithm has provided a robust capability to treat low Mach number subsonic-to high Mach number hypersonic flow problems within one simulation code without the necessity of tuning the flow solver via adjustable parameters. In addition, the extension of the AUGUST family of codes to treat multiphase, multicomponent reactive flow phenomena provides the capability, for the first time, of simulating a wide variety of physically interesting and challenging problems that are rich in physics-chemical phenomena. The range of these problems includes: 1) full 3D flows about complex aircraft in all flight regimes (except rarefied flows), 2) shock-body interactions, 3) chemically reacting flows typical in combustion problems, and 4) detonation phenomena found in explosives, shock tubes, and specific applications to such devices as the pulsed detonation engine.

SAIC's UUGM program has resulted in over 20 publications in various stages of preparation, and numerous presentations at U.S. and international technical meetings, conferences, and workshops. The AUGUST family of simulation codes is presently being applied to several current materials development and synthesis areas of research. In particular, the ability of the AUGUST codes to capture the complex geometry of material synthesis reactor configurations, resolve the complex flow patterns, and treat the complex physics and chemistry of the synthesis process provides a simulation and modeling tool that is useful for design of such process reactors, analyse and evaluate experimental results, and (depending on successful benchmarking) provide a process control tool based on validated models. SAIC intends to exploit this capability in future programs.

SAIC's Applied Physics Operation, Hydrodynamic Modeling Division staff members performed the work under the DARPA UUGM program. Dr. Shmuel Eidelman and Dr. William Grossmann were co-program managers. Important contributions were made by Drs. Itzhak Lottati, Xiaolong Yang, Marty Fritts, Adam Drobot, Ahron Friedman, and Michael Kress. SAIC's UUGM team would like to acknowledge the support and interest of Dr. James Crowley (ARPA ACMP program manager), Drs. Lois Auslander and Helena Wisniewski (previously DARPA ACMP program managers), and Dr. Arje Nachman (AFOSR) who served as the ARPA agent for the UUGM program.

REFERENCES

1. T.J. Baker and A. Jameson, "A Novel Finite Element Method for the Calculation of Inviscid Flow Over a Complete Aircraft," Sixth International Symposium on Finite Element Methods in Flow Problems, Antibes, France (1986).
2. T.J. Baker, "Developments and Trends in Three-Dimensional Mesh Generations," Transonic Symposium held at NASA Langley Research Center, Virginia (1988).
3. R. Lohner, "Generation of Three-Dimensional Unstructured Grids by the Advanced-Front Method," AIAA 26th Aerospace Sciences Meeting, Reno, AIAA Paper 88-0515, January 1988.
4. R. Lohner and K. Morgan, "Improved Adaptive Refinement Strategies for Finite Element Aerodynamic Computations," AIAA 29th Aerospace Sciences Meeting, Reno, AIAA Paper 86-0499, January 1986.
5. A. Jameson, T.J. Baker, and N.P. Weatherill, "Calculation of Inviscid Transonic Flow Over a Complete Aircraft," AIAA 24th Aerospace Sciences Meeting, Reno, NV, AIAA Paper 86-0103, January 1986.
6. R. Lohner, "Adaptive Remeshing for Transient Problems, *Comp. Meth. Appl. Mech. Eng.* 75 195-214 (1989).
7. J. Peraire, M. Vahdati, K. Morgan, and O.C. Zienkiewicz, "Adaptive Remeshing for Compressible Flow Computations," *J. Comp. Phys.* 72, 449-466 (1987).
8. D. Mavriplis, "Accurate Multigrid Solution of the Euler Equations on Unstructured and Adaptive Meshes," AIAA 88-3707 (1988).
9. I. Lottati, S. Eidelman, and A. Drobot, "A Fast Unstructured Grid Second Order Godunov Solver (FUGGS)," 28th Aerospace Sciences Meeting, AIAA-90-0699, Reno, NV (1990).
10. I. Lottati, S. Eidelman, and A. Drobot, "Solution of Euler's Equations on Adaptive Grids Using a Fast Unstructured Grid Second Order Godunov Solver," Proceeding of the Free Lagrange Conference, Jackson Lake, WY, June 1990.
11. I. Lottati and S. Eidelman, "Second Order Godunov Solver on Adaptive Unstructured Grids," Proceeding of the 4th International Symposium on Computational Fluid Dynamics, Davis, CA, September 1991.
12. S. Eidelman, P. Collela, and R.P. Shreeve, "Application of the Godunov Method and Its Second Order Extension to Cascade Flow Modeling," *AIAA Journal* 22, 10 (1984).

13. B. van Leer, "Towards the Ultimate Conservative Difference Scheme, V.A. Second Order Sequel to Godunov's Method," *J. Comp. Phys.* 32, 101-136 (1979).
14. P. Collella and P. Woodward, "The Piecewise Parabolic Method (PPM) for Gas-dynamic Simulations," *J. Comp. Phys.* 54, 174-201 (1984).
15. J.F. Thompson, "Grid Generation Techniques in Computational Fluid Dynamics," *AIAA Tour.*, Vol. 22, No. 11, pp. 1505-1523, November, 1984.
16. M.S. Shepherd and M.K. Georger, "Automatic Three-Dimensional Mesh Generation by the Finite Octree Method," *Intern. J. Num. Meth. Eng.*, Vol. 32, pp. 709-749, (1991).
17. H.M. Glaz, P. Colella, L.I. Glass, and R.L. Deschambault, "A Detailed Numerical, Graphical and Experimental Study of Oblique Shock Wave Reflections," DNA-TK-86-365, 1986.
18. I.I. Glass and D.L. Zhang, "Interferometric Investigation of the Diffraction of Planar Shock Waves Over a Half-Diamond Cylinder in Air," UTIAS Report No. 322, March 1988.
19. H. Hornung, "Regular and Mach Reflection of Shock Waves," *Annual Review of Fluid Mechanics*, Vol. 18, pp. 33-58, 1986.

PUBLICATIONS UNDER THE UUGM PROJECT

1. S. Eidelman, W. Grossmann, and I. Lottati, "A Review of Propulsion Applications of the Pulsed Detonation Engine Concept," AIAA 89-2446, AIAA/ASME/SAE/ASEE 25th Joint Propulsion Conf., Monterey, CA, July 1989.
2. S. Eidelman, W. Grossmann, and I. Lottati, "Computational Analysis of Pulsed Detonation Engines and Applications," AIAA 90-0460, 28th Aerospace Sciences Meeting, Reno, NV, January 1990.
3. I. Lottati, S. Eidelman, and A. Drobot, "A Fast Unstructured Grid Second Order Godunov Solver (FUGGS)", AIAA 90-0699, 28th Aerospace Sciences Meeting, Reno, NV, January 1990.
4. S. Eidelman and I. Lottati, "Reflection of the Triple Point of the Mach Reflection in a Planar and Axisymmetric Converging Channels," 9th Mach Reflection Symposium, Freiburg, Germany, June 1990.
5. I. Lottati, S. Eidelman, and A. Drobot, "Solution of Euler's Equations on Adaptive Grids Using A Fast Unstructured Grid Second Order Godunov Solver (FUGGS)," in H.E. Trease, M.J. Fritts, and W.P. Crowley (Eds.), Proceedings of the Next Free-Lagrange Conference, Jackson Lake, WY, June 1990 [*Advances in the Free-Lagrange Method*, Springer-Verlag, New York (1992)].
6. S. Eidelman, W. Grossmann, and I. Lottati, "Air-Breathing Pulsed Detonation Engine Concept; A Numerical Study," AIAA 90-2420, AIAA/SAE/ASME/ASEE 26th Joint Propulsion Conf., Orlando, FL, July 1990.
7. E. Hyman, K. Tsang, I. Lottati, A. Drobot, B. Lane, R. Post, and H. Sawin, "Plasma Enhanced Chemical Vapor Deposition Modeling," *Surface and Coatings Tech.* 49, 387 (1991).
8. S. Eidelman, W. Grossmann, and A. Friedman, "Nonlinear Signal Processing Using Integration of Fluid Dynamics Equations," Applications of Digital Image Processing XIV, SPIE Vol. 1567, 1991.
9. S. Eidelman, W. Grossmann, and I. Lottati, "Review of Propulsion Applications and Numerical Simulations of the Pulsed Detonation Engine Concept," *J. Propulsion* 7, 857 (1991).
10. D.L. Book, S. Eidelman, I. Lottati, and X. Yang, "Numerical and Analytical Study of Transverse Supersonic Flow Over a Flat Cone," *Shock Waves* 1, 197, 1991.

11. S. Eidelman and X. Yang, "Detonation Wave Propagation in Variable Density Multi-phase Layers," AIAA 92-0346, 30th Aerospace Sciences Meeting, Reno, NV, January 1992.
12. S. Eidelman, I. Lottati, and W. Grossmann, "A Parametric Study of the Air-Breathing Pulsed Detonation Engine," AIAA 92-0392, 30th Aerospace Sciences Meeting & Exhibit, Reno, NV, January 1992.
13. I. Lottati and S. Eidelman, "A Second Order Godunov Scheme on a Spatial Adapted Triangular Grid," in U.S. Army Workshop on Adaptive Methods for Partial Differential Equations, Rensselaer, NJ, 1992.
14. I. Lottati and S. Eidelman, "Decomposition by Structured/unstructured Composite Grids for Efficient Integration in Domains with Complex Geometries," in *Adv. in Computer Methods for Partial Differential Equations VII*, R. Vichnevetsky, D. Knight, and G. Richter (Eds.), 1992.
15. X. Yang, S. Eidelman, and I. Lottati, "Two-Phase Compressible Flow Computation on Adaptive Unstructured Grid Using Upwind Schemes," in *Adv. in Computer Methods for Partial Differential Equations VII*, R. Vichnevetsky, D. Knight, and G. Richter (Eds.), 1992.
16. S. Eidelman and W. Grossmann, "Pulsed Detonation Engine Experimental and Theoretical Review," AIAA 92-3168, AIAA/SAE/ASME/ASEE 28th Joint Propulsion Conf. and Exhibit, Nashville, TN, July 1992.
17. S. Eidelman and A. Altshuler, "Synthesis of Nanoscale Materials Using Detonation of Solid Explosives," *1st Intern. Conf. on Nanostructured Materials*, Cancun, Mexico, September 1992.
18. S. Eidelman and X. Yang, "Detonation Wave Propagation in Combustible Mixtures with Variable Particle Density Distributions," *AIAA J.* **31**, 228, 1993.
19. S. Eidelman and X. Yang, "Detonation Wave Propagation in Combustible Particle/Air Mixture with Variable Particle Density Distributions," *Combust. Sci. and Tech.* **89**, 201, 1993.
20. X. Yang, S. Eidelman, and I. Lottati, "Computation of Shock Wave Reflection and Diffraction Over a Semicircular Cylinder in a Dusty Gas," AIAA 93-2940, 24th Fluid Dynamics Conf., Orlando, FL, July 1993.
21. I. Lottati and S. Eidelman, "Acoustic Wave Focusing in an Ellipsoidal Reflector for Extracorporeal Shock-wave Lithotripsy," AIAA 93-3089, 24th Fluid Dynamics Conf., Orlando, FL, July 1993.

APPENDIX A
CODE DESCRIPTION

APPENDIX A CODE DESCRIPTION

A.1 AUGUST (2D)

The subroutines in the AUGUST code are organized here as they appear in the listing in Appendix B. A brief description indicates the function performed by each subroutine.

TABLE A.1

LIST OF SUBROUTINES

1. MAIN	Governing program for AUGUST. Reads input files and sets the mode for the computation.
2. HYDRFL	Computes the fluxes at interfaces by applying the Godunov algorithm to solve the Riemann problem across the interface.
3. HYDRMN	Controls the computation. The integration of the fluxes and update of the physical variables, adaptation of the grid and writing to output files are performed in this subroutine.
4. GEOMTR	Calculates the geometrical quantities not provided by the input data file but needed for the computational algorithm. GEOMTR is only used once for starting a new simulation.
5. UPDATE	Reads the input file for a new simulation and calls GEOMTR to update the geometrical variables needed to perform the computation.

6. UPGRAD	Called if a restart run is performed. Will read the appropriate file written at the end of the previous run.
7. GRADNT	Computes the gradient of the physical variables to improve the prediction of those variables for the two sides of the interface. The gradients are subjected to the monotonicity condition that limits the projected values, thus preventing new maxima-minima from being caused artificially by interpolation (IOPORD = 2). Calls FCHART in order to compute projected values at the half timestep associated with the local characteristics of the flow.
8. GRDFLX	Computes the gradient of the pressure and Mach number in each cell. This information is used as an error indicator for the adaptation needed in a steady state solution.
9. FIRST	The equivalent of GRADNT if run in a first order mode (IOPORD = 1). Using FIRST assumes that the physical variables are constant in each cell. Takes care of the boundary conditions if the interface is a boundary.
10. FCHART	Computes the projected values at a half timestep for the two sides of the interface based on the local characteristics of the flow. Called by GRADNT, it modifies the projected values for the two sides of the interface and assigns them to the correct location in memory. Takes care of the boundary conditions if the interface is a boundary.

11. PRLCTN	Determines particle cell location in the initial phase of tracing a group of particles.
12. PRPTH	Advances the position of each particle, assuming that the particle has the flow velocity of the cell. PRPTH will find the cell location of the particle after it advances by the timestep of the computation.
13. VERCEN	Places an additional vertex at the center of a specified cell to refine the size of the cell by a factor of three.
14. DISECT	Places an additional vertex at the middle of a specified edge to refine the size of the two cells adjacent to the edge by a factor of two. This method of refinement is used only on the edges lying on the boundaries of the computational domain.
15. DYNPTN	Tests and flags the cells for specified refinement criteria. DYNPTN is called only if the parameter IOPADD = 1. Will start the refinement procedure by calling VERCEN and DISECT and will call DYYPTN for further refinement. This insures that the buffer zone ahead of the shock is resolved according to the specified area criteria (AREADD).
16. DYYPTN	Refines the cells flagged by DYNPTN by calling VERCEN and DISECT until the area of each flagged cell meets the area criteria specified by the parameter AREADD.

17. INTPTN	Refines the cells in the inlet region. Prepares the inlet region for the introduction of a shock wave. This initial refinement is essential to prevent additional refinement of the grid in the presence of a shock wave. It is called only if the parameters ICOND=0 and IOPTN= 2 (solution for transient phenomena).
18. DELPTN	Tests and flags the cells for the specified criteria for coarsening. DELPTN is called only if parameter IOPDEL = 1.
19. RELAXY	Relaxes the vertices of the cells that were created in the process of deleting a vertex.
20. VERDEL	Deletes a specified vertex.
21. RECNC	Tests two cells adjacent to the specified edge. Compares them to the two cells that can be created if this edge is flipped to pass between the other two vertices of the quadrilateral containing the original two cells. If the tests result in a better quality triangle, then RECNC will swap the edge.
22. EOS	Applies Gilmore equation of state to compute $\gamma = c_p/c_v$, giving the internal energy and density of the fluid in a cell. This option is controlled by the parameter IOPEOS = 1.
23. LIFTDR	A diagnostic to compute the lift, drag, and transfer momentum developed in the configuration. Takes into account all boundary edges that are specified as 5. It is controlled by the parameter IOPLFT = 1.

THE MAIN PROGRAM

All of the data input and initiation of a run (or a restart run) is performed in MAIN. The actual simulation is controlled by HYDRMN, which is called from MAIN. At the completion of a run, control is returned to MAIN and a successful termination prints the message STOP 777.

MAIN contains one name list (file no. 2) and requires an input file that contains the grid data description (file no. 16). The data organization for the grid file is described in Appendix A. There are five files that should be included: CINT00.H, CMSH00.H, CPHS10.H, CPHS20.H, CHYD00.H.

NAMELIST/DATA	ICOND	ICONP	ITRIGR	IOPTN
	XMCHIN	RIN	PIN	ALFA
	HRGG	IHRN	NTIME	MDUMP
	NDUMP	KDUMP	IOSPCL	IOPLFT
	IOPRCN	IOPORD	IOPBYN	IAXSYM
	IOPEOS	MPRTCL	IOPINT	IOPADD
	IOPDEL	AREADD	AREDEL	IWINDW
	ISTATC			

VARIABLE	PURPOSE
ICOND	= 0 READ INPUT GRID FOR A NEW SIMULATION = 1 READ THE GRID FROM PREVIOUS RUN

ICOND = 0:

MAIN will read the initial grid definition stored in file number 16. The current setting is to read the input file as provided by Smart, a two dimensional triangular grid generator that runs interactively on the Macintosh personal computer.

MAIN will call UPDATE, which will call GEOMTR. GEOMTR will compute essential geometrical parameters that are not provided by file 16. All geometrical information is dumped into output files (8 and 88) so that ICOND=0 is used only once at the beginning of a new simulation.

ICOND = 1:

MAIN will call UPGRAD, which will call one of the output files (8 or 88) written by the previous run. This will load the geometrical definition of the grid (either 8 or 88--- they are identical). Writing identical files provides a backup in the event that the job terminates for lack of time while in the process of writing to one of those output files.

VARIABLE	PURPOSE
ICONP	= 0 PRIMITIVE VARIABLES INITIALIZED = 1 VARIABLES READ FROM PREVIOUS RUN

ICONP = 0:

Initialize the primitive variables in computational domain with an initial value specified by the user. The two options set by the code are controlled by IOPTN.

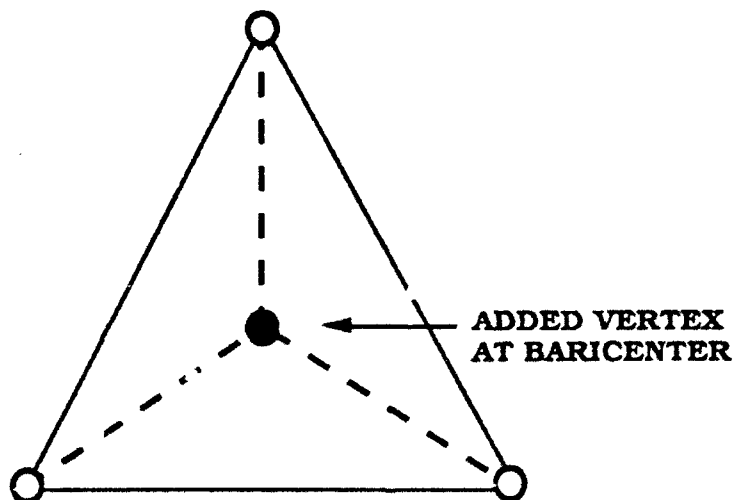
ICONP = 1:

The flow field condition reads in files 8 or 88 and provides a followup run set from the previous run.

VARIABLE	PURPOSE
ITRIGR	= 0 USING THE INPUT GRID AS THE INITIAL GRID = 1 THE INPUT GRID TRIPLED BY ADDING AN EXTRA VERTEX IN EACH TRIANGLE

ITRIGR = 1:

The original grid cells will be tripled by adding an extra vertex in the baricenter of each triangle. This option can be triggered at the beginning of a simulation only (ICOND = 0).



VARIABLE	PURPOSE
IOPTN	= 1 SOLUTION FOR STEADY STATE = 2 SOLUTION FOR TRANSIENT PHENOMENA

There are two choices available to set the initial condition of the problem.

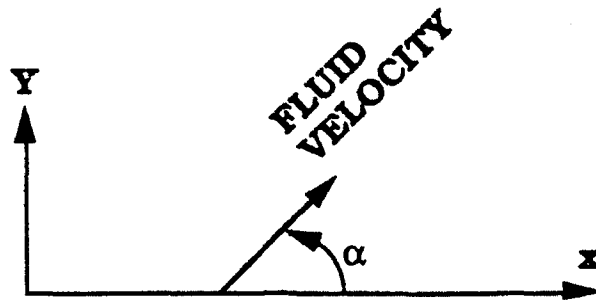
OPTN = 1.

Assign the conditions at the inlet to the computational domain. This is the fastest way to get a steady state solution for the conditions specified at the inlet. In this option, PIN (pressure), RIN (density) and XMCHIN (Mach number) are assigned to the pressure density and velocity (the speed of sound is computed in the code) and imposed at the inlet boundaries.

OPTN = 2.

Used if a shock wave is to be simulated moving from the inlet (edge boundary 8) to the outlet (edge boundary 7). For this setting, specify PIN (ambient pressure in the chamber), RIN (ambient density in the chamber) and XMCHIN (upstream Mach number). The code will use the normal shock wave relations for an adiabatic flow of a completely perfect fluid to compute the static-pressure ratio across the shock P_2/P_1 and the density ratio ρ_2/ρ_1 , and the ratio of the Mach number across the shock M_2/M_1 . These computed quantities are applied to set correctly the condition on the pressure density and velocity at the inlet boundary.

VARIABLE	PURPOSE
ALPHA	THE DIRECTION OF INFLOW IN DEGREES RELATIVE TO A RIGHT HAND COORDINATE SYSTEM. ALPHA = 0 MEANS FLOW FROM LEFT TO RIGHT.



The velocity computed by the code according to the input data provided by the user is split (projected) in the X and Y directions by using α .

VARIABLE	PURPOSE
HRGG	INITIAL γ IN THE EQUATION OF STATE. THE CODE RUNS USING THE IDEAL EQUATION OF STATE AS A BASELINE AND SHOULD BE MODIFIED IF SOMETHING ELSE IS DESIRED. IOPEOS=1 WILL TRIGGER THE USE OF GILMORE EQUATION OF STATE.

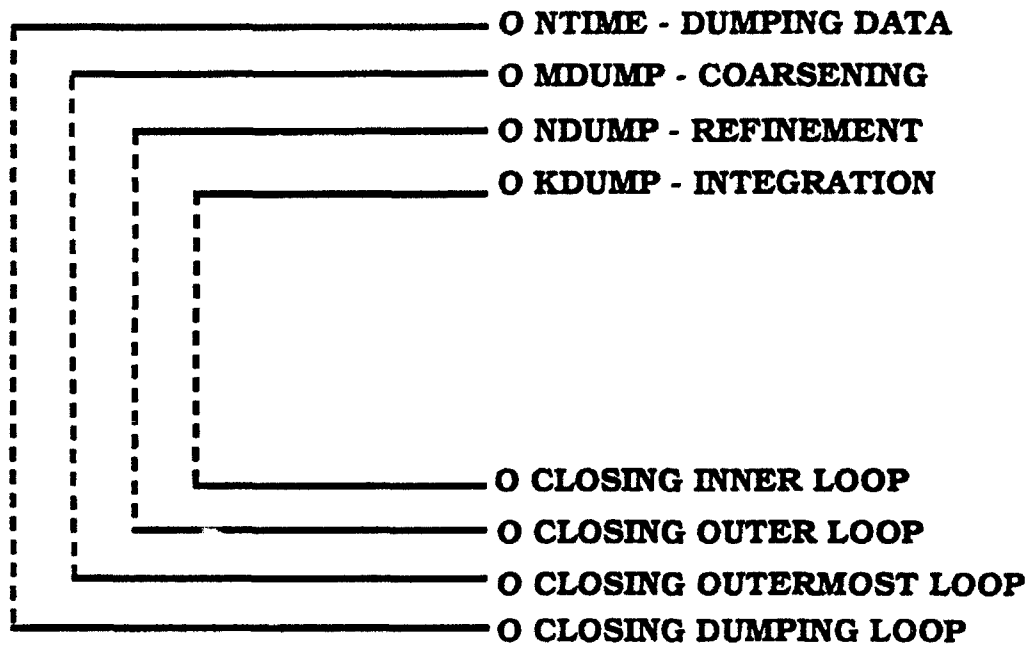
VARIABLE	PURPOSE
IHRN	NUMBER OF ITERATIONS IN THE RIEMANN SOLVER TO FIND THE DIAPHRAGM SOLUTION. (THREE TO FOUR SHOULD BE USED AND INCREASED ONLY FOR A VERY HIGH MACH NUMBER CASES.)

VARIABLE	PURPOSE
NTIME	NUMBER OF REPEATS FOR THE INTEGRATION/ REFINEMENT/ COARSENING SEQUENCE. AN OUTPUT DUMP IS DONE FOR EVERY SEQUENCE REPEAT.

VARIABLE	PURPOSE
MDUMP	NUMBER OF OUTERMOST LOOP ITERATIONS IN THE CALCULATION WHERE COARSENING OF THE GRID IS PERFORMED EVERY SEQUENCE REPEAT.

VARIABLE	PURPOSE
NDUMP	NUMBER OF OUTER LOOP ITERATIONS IN THE CALCULATION WHERE REFINING IS DONE FOR EVERY SEQUENCE REPEAT WITHOUT COARSENING.

VARIABLE	PURPOSE
KDUMP	NUMBER OF ITERATIONS PERFORMED WITH NO REFINEMENT OR COARSENING. THE INNER LOOP OF THE CALCULATION. IF KDUMP = 0, KDUMP WILL BE SET BY THE CODE AUTOMATICALLY ACCORDING TO THE SETTING OF THE VARIABLE AREADD.



VARIABLE	PURPOSE
IOSPCL	= 0 NOT USING REDEFINITION OF POINTS ON THE BOUNDARY = 1 USING REDEFINITION OF POINTS ON THE BOUNDARY

IOSPCL = 1:

Modifies the definition of points along the boundary according to a presetting in the code. The setting currently will redefine the points along the edge boundary 5 to exactly match NACA0012 airfoil shape. This is done to redefine points on a boundary that has an analytical definition of points, but where these points have been dislocated by a refining procedure.

VARIABLE	PURPOSE
IOPFLT	= 0 THE COMPUTATION OF LIFT DRAG AND MOMENT TURNED OFF = 1 THE COMPUTATION OF LIFT DRAG AND MOMENT TURNED ON

Set IOPFLT = 1 if integral quantities need to be computed. The current setting will calculate the lift, drag and moment on edge boundary 5.

VARIABLE	PURPOSE
IOPRCN	= 0 A GLOBAL SWAPPING (RECONNECTION) PROCEDURE IS OFF = 1 A GLOBAL SWAPPING (RECONNECTION) PROCEDURE IS ON

This swapping is done by calling subroutine RECNC. It is used only in a new simulation (ICOND = 0).

VARIABLE	PURPOSE
IOPORD	= 1 THE CODE WILL RUN FIRST ORDER GODUNOV METHOD = 2 THE CODE WILL RUN SECOND ORDER GODUNOV METHOD

IOPORD = 1 Subroutine FIRST is called.

IOPORD = 2 Subroutine GRADNT is called.

VARIABLE	PURPOSE
IOPYN	= 0 NO BUOYANCY EFFECTS ARE COMPUTED = 1 BUOYANCY EFFECTS IN THE X DIRECTION ARE COMPUTED = 2 BUOYANCY EFFECTS IN THE Y DIRECTION ARE COMPUTED

The buoyancy effect applies the gravity acceleration as $g = 9.81$.

VARIABLE	PURPOSE
LAXSYM	= 0 THE CODE WILL RUN IN A PURE TWO DIMENSIONAL MODEL = 1 THE CODE WILL RUN IN AN AXISYMMETRICAL MODE (X AS THE AXIS OF SYMMETRY) = 2 THE CODE WILL RUN IN AN AXISYMMETRICAL MODE (Y AS THE AXIS OF SYMMETRY)

VARIABLE	PURPOSE
IOPEOS	= 0 THE CODE WILL RUN WITH CONSTANT γ = 1 THE CODE WILL RUN WITH VARIABLE γ USING THE EQUATION OF STATE FOR AIR

IOPEOS = 0

The initial γ is not changed and is kept constant across the computational domain at all times (with value set by HRGG).

IOPEOS = 1

The γ of each cell will be modified according to local internal energy and density. Thus, if IOPEOS = 1, the actual pressure and density should be input (in the appropriate dimension). Otherwise (IOPEOS=0), a normalized pressure and density of unity can be used for simulation.

VARIABLE	PURPOSE
MPRTCL	= 0 NO PARTICLE TRACING = 1 THE CODE WILL TRACE PARTICLES

MPRTCL = 1.

The ability to trace particles will be turned on. Initially PRLCTN is called to identify the cell location of each particle. For each time step, PRPTHG will be called to update the cell location of each particle if it is relocated, assuming the particle moves at the same velocity as the fluid.

The initial location of the particles is defined in MAIN.

VARIABLE	PURPOSE
IOPINT	= 0 DOES NOT PREPARE A BUFFER ZONE. = 1 INITIALLY PREPARE A BUFFER ZONE AHEAD OF EDGE BOUNDARY 8

For simulating transient phenomena, the refining of the grid is done in the region ahead of the shock. In this way, we avoid interpolating in a region where large gradients reside. IOPINT = 1 will refine the region of the inlet flow to prepare a buffer zone (edge boundary 8). If refining is needed in another region, subroutine INTPTN should be modified accordingly.

VARIABLE	PURPOSE
IOPADD	= 0 THE REFINEMENT PROCEDURE IS TURNED OFF = 1 THE REFINEMENT PROCEDURE IS TURNED ON

VARIABLE	PURPOSE
IOPDEL	= 0 THE COARSENING PROCEDURE IS TURNED OFF = 1 THE COARSENING PROCEDURE IS TURNED ON

VARIABLE	PURPOSE
AREADD	SPECIFIES THE MINIMUM AREA VALUE THAT A TRIANGLE SHOULD HAVE AFTER REFINEMENT. SPECIFIED AS A FRACTION OF THE AVERAGE TRIANGLE AREA OF THE INITIAL GRID. THIS REFERENCE AREA IS KEPT CONSTANT THROUGH THE WHOLE SIMULATION.

VARIABLE	PURPOSE
AREDEL	SPECIFIES THE MAXIMUM VALUE THAT A TRIANGLE SHOULD HAVE AFTER COARSENING DEFINED AS A FRACTION OF THE REFERENCE AREA.

VARIABLE	PURPOSE
IWINDOW	= 0 NO RESTRICTION ON THE REGION FOR REFINING THE GRID = 1 SETTING A WINDOW FOR REFINING THE GRID

IWINDOW = 1

The user can specify a region in which the refinement process will take place. Otherwise, the refinement takes place everywhere in the computational domain.

VARIABLE	PURPOSE
ISTATC	= 0 THE ADAPTATION WILL BE DONE ON A MOVING WAVE = 1 THE ADAPTATION WILL BE DONE ON A STEADY STATE CONDITION

Because the criteria for refinement in the presence of a static shock are not suited to treating a moving shock, the code sets different error indicators for adapting the grid for the two cases.

ISTATIC=0

The energy and density net fluxes across each cell are tested for sensing the level of activity. This method is a very good error indicator for sensing transient phenomena as traveling shocks.

ISTATIC=1

The pressure and Mach gradients in each cell are tested for sensing steady state shocks.

The gradient of density is always tested as a third criteria for making sure that static shocks are not ignored in computing a transient flow.

HYDREL

Computes the fluxes across interfaces when the conditions for both sides are given. The fluxes are computed assuming a shock solution at a broken diaphragm simulated by the presence of the interface. The conditions existing on the two sides of the diaphragm will define the condition of the flow at the diaphragm location. These conditions are computed by solving the Riemann problem using the Godunov algorithm. The condition at the diaphragm defines the flux of energy, mass, and momentum passing across the interface. The Euler conservation law is applied to conserve energy, mass, and momentum crossing interfaces from one cell to the other.

Quantity	Side 1	Diaphragm (Interface)	Side 2
Density	ρ_1	ρ	ρ_2
Pressure	P_1	P	P_2
Velocity Perpendicular to Interface	u_1	u	u_2
Velocity Parallel to Interface	v_1	v	v_2

HYDRMN

Controls the code and the iteration loops. It calls HYDRFL to find the interface fluxes. These fluxes are integrated to update the physical variables in each cell. If adaptation of the grid is required, HYDRMN will set the criteria for controlling the adaptation of the grid. The refining (DYNPTN, DYYPTN) and coarsening (DELPTN) of the grid is invoked by HYDRMN. HYDRMN also controls the output by writing the necessary information on files for post processing data and for restarting the AUGUST code at a later time. It also manages print file diagnostics.

GEOMTR

Calculates geometrical variables that are not supplied by the input data and are needed to run the code. For example, it will compute:

- 1) Area of the cells;
- 2) Length of the edges;
- 3) Unit vector perpendicular to the edge. (For boundary edges, this unit vector is direct from the computational domain outward);
- 4) Unit vector directed from the baricenter of the left cell to the baricenter of the right cell. For boundary edges, the unit vector is perpendicular to the edge (from left cell outward).

The code will change the direction of the boundary edges so that all are arranged counterclockwise and the associated computational cell is always on the left side. GEOMTR is called once in the beginning of a new simulation.

UPDATE

Called in the beginning of a new simulation for setting geometrical variables not provided by the input data. (It calls GEOMTR.)

UPGRAD

Called if the run is a restart. UPGRAD will read the appropriate file (either 8 or 88) dumped by the previous run.

GRADNT

Compute the gradients of the physical variables in each cell. These computed gradients, along with the physical values at the baricenters, are applied using linear interpolation to predict the values on the interface.

The computed gradients are subjected to the monotonicity condition, ensuring that the projected values are bounded by the value of each quantity in the three adjacent cells, and to make sure that no new maxima or minima occurs. The projection of quantities to the interface improves the results from the code and provides second order accuracy in space.

GRADNT calls FCHART, which computes the projected values at the interfaces at the half timestep level according to the local characteristics of the flow in each cell bordering the interface cell. The assignment of values at the two sides of each interface is done at the end of FCHART. This same loop will also impose the boundary conditions for the interfaces at the boundaries of the computational domain.

GRDFLX

Computes the gradient of the Mach value and pressure gradient in each cell. These gradients are applied if the adaptation is done on a steady state converged solution. These variables, in addition to the computed density gradient, provide the criteria for adaptation if it is necessary to refine the grid for steady state problems.

FIRST

Assigns flow quantities to each side of an edge. These are based on the values at the baricenter of the triangles on either side of the edge. FIRST uses a first order approximation to find the values at the edge.

The user can specify FIRST or GRADNT by choosing 1 or 2 for the parameter IOPORD.

FCHART

Called by GRADNT to compute the values projected at the interfaces at the half timestep. These calculations are done by applying the local velocity characteristics in each cell. This projection in time improves the results and makes the code second order accurate in time.

PRLECTN

Identifies the initial cell location of each particle. Called once after specifying the starting location of each particle to be traced.

PRPTHG

Advances the particle position by the marching timestep. It finds the new cell location if a particle crosses an interface. The assumption is that the particles move at the fluid velocity.

VERCEN

Introduces a new vertex at the baricenter of the designated cell during the refinement process.

DISECT

Introduces a new vertex at the middle of a designated edge.

DYNPTN

Tests the cells according to the refining criteria and flags each cell which requires refinement. The flagged cells are refined in DYYPTN. The refinement is subjected to geometrical constraints on the cell shape to retain a high better quality refined grid.

The user can specify a window in the computational domain for refinement. The parameter to trigger this option is IWINDW = 1. For specifying the actual window, it may be necessary for the user to alter this subroutine and provide a definition of the geometrical area to be refined.

DYYPTN

Traces the cells that are flagged for refinement by DYNPTN. It subdivides them until each one of the refined cells meets the area refinement criteria of AREADD. Because each loop of refinement is restricted to a one-third reduction in cell area (calling VERCEN), DYYPTN will perform the necessary number of loops to meet the area reduction specified for refinement. AREADD is a fraction of the average area of the initial grid. This reference area is kept constant and fixes the minimum resolution in the simulation domain.

INTPTN

Performs the initial refinement of the grid before the initialization. The assumption is that a shock wave is introduced through the inlet boundary. Consequently, INTPTN will test for the inflow boundary interface and will refine the appropriate cells. (Note: It is not recommended that the code automatically refine the grid in the inlet region in the presence of a shock wave. If a shock wave is not introduced through the inlet, INTPTN should be modified to accommodate the change of the initial condition.)

DELPTN

Tests the cells according to coarsening criteria and flags them. Each triangle is tested to determine which vertex of the triangle is most appropriate for removal. This vertex is removed by calling VERDEL. DELPTN cannot delete nodes that have the status $JV(1,IV) = 3$. It is therefore recommended that nodes at sharp corners or nodes on important boundaries that are curved, be flagged as $JV(1,IV) = 3$.

RELAXY

Relaxes the cells that are created in the process of deleting a vertex. The relaxation procedure relocates the designated vertex to the mass center of the surrounding vertices.

LAPLAC

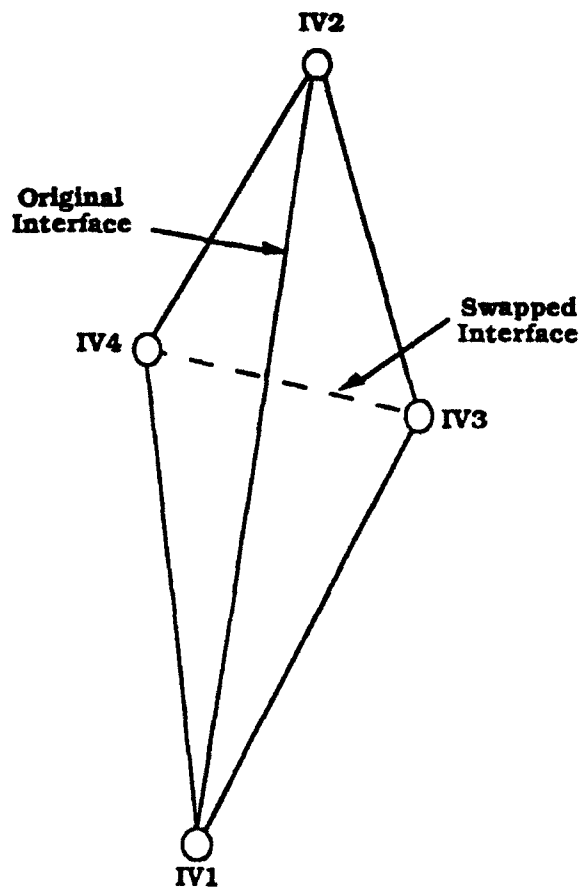
Computes the Laplacian of the pressure and density.

VERDEL

Deletes a designated vertex.

REORG

Tests the possibility of swapping the designated interface to create two triangles of better quality than the original two.



EOS

Computes γ using to the equation of state for air (Gilmore equation of state), given the density and internal energy of the air. The user may choose to apply the equation of state by setting IOPEOS = 1.

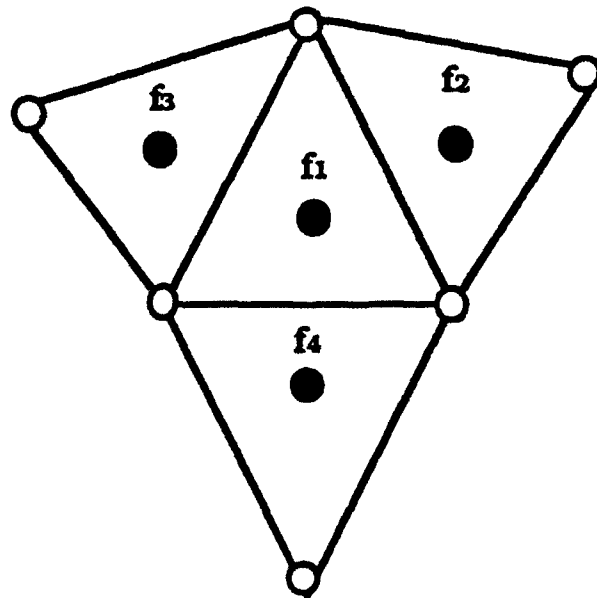
LIFTDR

Computes integral quantity diagnostics on any configuration. The integral quantities are lift, drag, and momentum and are found on boundary interfaces designated as 5.

GRADNT

Computes the gradient of a scalar variable at the center of a cell. It uses a least squares technique to interpolate the values at the center of four triangles (the cell and its three adjacent triangles) to fit (four equations with three unknowns).

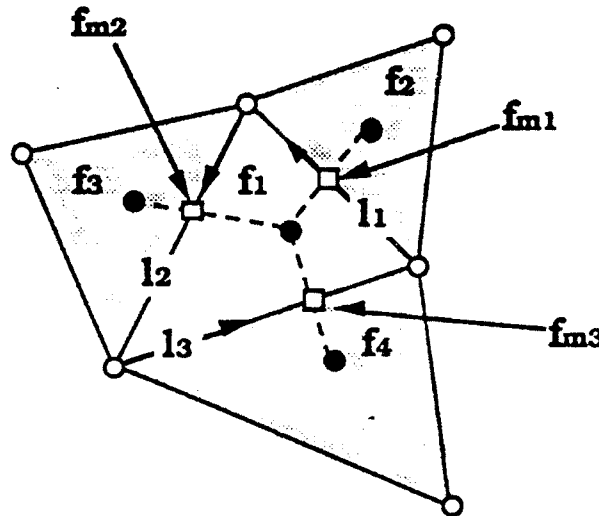
$$f = a_0 + a_1x + a_2y$$



Those gradients are subjected to a monotonicity limiter that ensures no new minima or maxima are produced artificially in the projected values at the interfaces.

The monotonicity algorithm involves the following steps

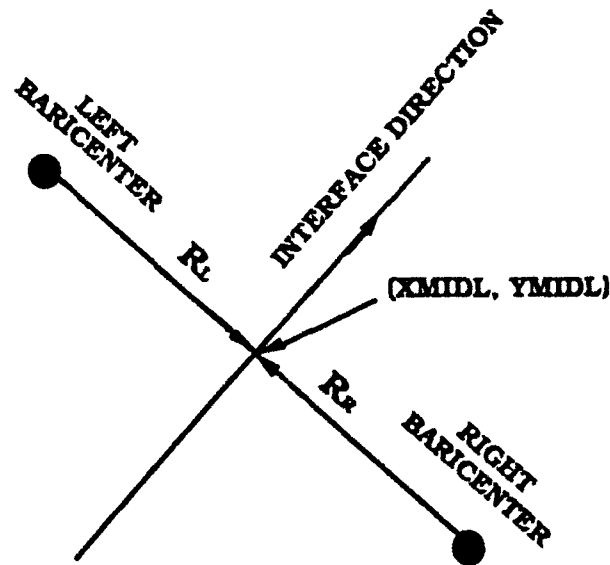
1)	find maximum and minimum of f_1, f_2, f_3, f_4 $f_{\max} = \text{Max} (f_1, f_2, f_3, f_4)$ $f_{\min} = \text{Min} (f_1, f_2, f_3, f_4)$
2)	compute $\Delta f_{\max} = f_{\max} - f_1$ $\Delta f_{\min} = f_{\min} - f_1$
3)	compute incremental projected values at the interfaces $f_{mjR} - f_R = \nabla f_R \cdot \bar{\eta}_{jR}$ $f_{mjL} - f_L = \nabla f_L \cdot \bar{\eta}_{jL}$



$$\Delta f_{mjR} = f_{mjR} - f_R = \nabla f_R \cdot \bar{\eta}_{jR}$$

$$\Delta f_{mjL} = f_{mjL} - f_L = \nabla f_L \cdot \bar{\eta}_{jL}$$

where j stands for every interface of the cell and f_{mj} is the interpolated value at the middle of the interface.



- 4) compute the limiter by calculating the minimum of indicator for each edge of the three edges of the cell.

$$\text{right to the interface RUVPR} = \frac{(1 + \text{sign } \Delta f_{m|R}) \Delta f_{\max} + (1 - \text{sign } \Delta f_{m|R}) \Delta f_{\min}}{2 \Delta f_{m|R}}$$

$$\text{left to the interface RUVPL} = \frac{(1 + \text{sign } \Delta f_{m|L}) \Delta f_{\max} + (1 - \text{sign } \Delta f_{m|L}) \Delta f_{\min}}{2 \Delta f_{m|L}}$$

This formulation ensures that

$$\text{if } \begin{cases} \Delta f_{m|j} > 0 & \text{RUV} = \frac{\Delta f_{\max}}{\Delta f_{m|j}} \\ \Delta f_{m|j} < 0 & \text{RUV} = \frac{\Delta f_{\min}}{\Delta f_{m|j}} \end{cases}$$

the outcome of RUV is always positive. If RUV > 1 then the projected value at the interfaces will introduce a new minimum or maximum relative to the values at the baricenters of the appropriate cells.

Select the minimum between the six values for RUV (two for every one of the three interfaces of the cell) not exceeding unity. The selected minimum

of RUVF is the required limiter. The gradient is multiplied by this limiter that is always less or equal to unity.

FCHART

Computes the projected values at the half time step level based on the local characteristics of the flow. This process extends the accuracy of the code to be second-order in time as well as in space.

The characteristic projection consists of several steps.

- 1) Calculate the velocity of sound in the two cells bordering the designated interface

$$CNLEFT = \sqrt{\gamma_L \cdot P_L / \rho_L} \quad \text{sound speed in left cell}$$

$$CNRIGHT = \sqrt{\gamma_R \cdot P_R / \rho_R} \quad \text{sound speed in right cell}$$

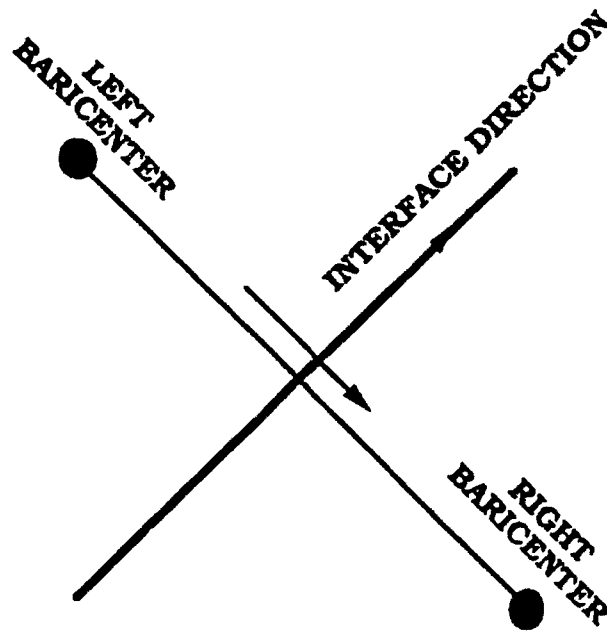
$$UVLEFT = \bar{U}_L \cdot \bar{t} \quad \text{velocity of fluid at the left cell projected in } \bar{t} \text{ direction}$$

$$UVRIGHT = \bar{U}_R \cdot \bar{t} \quad \text{velocity of fluid at the right cell projected in } \bar{t} \text{ direction}$$

where

$$\bar{t} = XXN \cdot \bar{i} + YYN \cdot \bar{j}$$

$$\bar{U} = U \cdot \bar{i} + V \cdot \bar{j}$$



- 2) To compute the interpolated left and right projected values at time $t^N + \Delta t/2$, we calculate the distances that the disturbances generated from the baricenter of the cells, traveling toward the interface:

$$ZZLEFT = (UVLEFT + CNLEFT) \cdot \Delta t/2$$

$$ZZRIGHT = - (UVRIGHT - CNRIGHT) \cdot \Delta t/2$$

If $ZZLEFT$ or $ZZRIGHT$ is negative they are reset to zero.

- 3) Calculate the distances that the flow will travel if it were to flow at the velocity of each of the local characteristics:

$$ZOLEFT = UVLEFT \cdot \Delta t/2$$

$$ZORIGHT = - UVRIGHT \cdot \Delta t/2$$

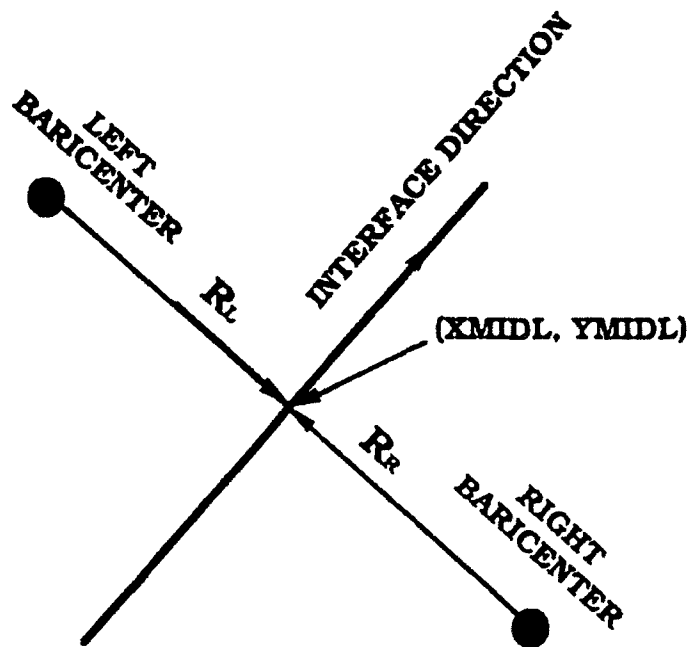
$$ZPLEFT = (UVLEFT + CNLEFT) \cdot \Delta t/2$$

$$ZPRIGT = - (UVRIGT + CNRIGT) \cdot \Delta t/2$$

$$ZMLEFT = (UVLEFT - CNLEFT) \cdot \Delta t/2$$

$$ZMRIGT = - (UVRIGT - CNRIGT) \cdot \Delta t/2.$$

- 4) Calculate the projected values of the nonconservative variables (density, velocity component (perpendicular and tangential to the interface), and pressure).



For the left cell:

$$\text{Density} \quad \text{HRRL} \quad = \quad \rho_L + \bar{V}\rho_L \cdot (\bar{r}_L - ZZLEFT \cdot \bar{t})$$

$$\text{Perpendicular Velocity} \quad \text{HUUL} \quad = \quad U_L + \bar{V}U_L \cdot (\bar{r}_L - ZZLEFT \cdot \bar{t})$$

$$\text{Tangential Velocity} \quad \text{HVVL} \quad = \quad V_L + \bar{V}V_L \cdot (\bar{r}_L - \text{ZZLEFT} \cdot \bar{t})$$

$$\text{Pressure} \quad \text{HPPL} \quad = \quad P_L + \bar{V}P_L \cdot (\bar{r}_L - \text{ZZLEFT} \cdot \bar{t})$$

$$\text{GMTLFT} \quad = \quad \rho_L \cdot \text{HRRL} \cdot \text{HPPL}$$

For the right cell:

$$\text{Density} \quad \text{HRRR} \quad = \quad \rho_R + \bar{V}\rho_R \cdot (\bar{r}_R - \text{ZZRIGHT} \cdot \bar{t})$$

$$\text{Perpendicular velocity} \quad \text{HUUR} \quad = \quad U_R + \bar{V}U_R \cdot (\bar{r}_R - \text{ZZRIGHT} \cdot \bar{t})$$

$$\text{Tangential velocity} \quad \text{HVVR} \quad = \quad V_R + \bar{V}V_R \cdot (\bar{r}_R - \text{ZZRIGHT} \cdot \bar{t})$$

$$\text{Pressure} \quad \text{HPPR} \quad = \quad P_R + \bar{V}P_R \cdot (\bar{r}_R - \text{ZZRIGHT} \cdot \bar{t})$$

$$\text{GMTRGT} \quad = \quad \rho_R \cdot \text{HRRR} \cdot \text{HPPR}$$

For the left cell, taking into account the following characteristics:

- For UVLEFT + CNLEFT:

$$\text{UUU} \quad = \quad \bar{V}U_L \cdot (\text{ZPLEFT} - \text{ZZLEFT}) \cdot \bar{t}$$

$$\text{PPP} \quad = \quad \bar{V}P_L \cdot (\text{ZPLEFT} - \text{ZZLEFT}) \cdot \bar{t}$$

$$\text{UPLFT} \quad = \quad -0.5 \cdot (\text{UUU} + \text{PPP} / \sqrt{\text{GMTLFT}}) / \sqrt{\text{GMTLFT}}$$

If UVLEFT + CNLEFT is negative, UPLFT is reset to zero.

- For UVLEFT - CNLEFT:

$$\text{UUU} \quad = \quad \bar{V}U_L \cdot (\text{ZMLEFT} - \text{ZZLEFT}) \cdot \bar{t}$$

$$\text{PPP} \quad = \quad \bar{V}P_L \cdot (\text{ZMLEFT} - \text{ZZLEFT}) \cdot \bar{t}$$

$$\text{UPLFT} \quad = \quad 0.5 \cdot (\text{UUU} - \text{PPP} / \sqrt{\text{GMTLFT}}) / \sqrt{\text{GMTLFT}}$$

If UVLEFT - CNLEFT is negative, UPLFT is reset to zero.

- For UVLEFT:

$$PPP = \bar{V}P_L \cdot (ZOLEFT - ZZLEFT) \cdot \bar{t}$$

$$RRRR = \rho_L + \bar{V}\rho_L \cdot (\bar{r}_L - ZOLEFT) \cdot \bar{t}$$

$$URLFT = PPP/GMTLFT + 1/HRRL - 1/RRRR$$

If UVLEFT is negative, URLEFT is reset to zero.

For the right cell, taking into account the following characteristics:

- For UVRIGT + CNRIGT:

$$UUU = \bar{V}U_R \cdot (ZZRIGT - ZPRIGT) \bar{t}$$

$$PPP = \bar{V}P_R \cdot (ZZRIGT - ZPRIGT) \bar{t}$$

$$UPRGT = -0.5 \cdot (UUU + PPP / \sqrt{GMTRGT}) / \sqrt{GMTRGT}$$

If UVRIGT + CNRIGT is positive, UMRGT is reset to zero.

- For UVRIGT - CNRIGT:

$$UUU = \bar{V}U_R \cdot (ZZRIGT - ZMRIGT) \cdot \bar{t}$$

$$PPP = \bar{V}P_R \cdot (ZZRIGT - ZMRIGT) \cdot \bar{t}$$

$$UMRGT = 0.5 \cdot (UUU - PPP / \sqrt{GMTRGT}) / \sqrt{GMTRGT}$$

If UVRIGT - CNRIGT is positive, UMRGT is reset to zero.

- For UVRIGT:

$$PPP = \bar{V}P_R \cdot (ZZRIGT - ZORIGT) \cdot \bar{t}$$

$$RRRR = \rho_R + \bar{V}\rho_R \cdot (\bar{r}_R + ZORIGT) \cdot \bar{t}$$

$$URRGT = PPP/GMTRGT + 1/HRRR - 1/RRRR$$

If UVRIGT - CNRIGT is positive, URRGT is reset to zero.

The projected values will be:

$$\begin{aligned}RRL &= 1/(1/HRRL - (UPLFT + UMLFT + URLFT)) \\UUL &= HUUL + (UPLFT - UMLFT) \sqrt{GMTLFT} \\VVL &= HVVL + (UPLFT - UMLFT) \sqrt{GMTLFT} \\PPL &= HPPL + (UPLFT + UMLFT) GMTLFT \\RRR &= 1/(1/HRRR - (UPRGT + UMRGT + URRGT)) \\UUR &= HUUR + (UPRGT - UMRGT) \sqrt{GMTRGT} \\VVR &= HVVR + (UPRGT - UMRGT) \sqrt{GMTRGT} \\PPR &= HPPR + (UPRGT + UMRGT) \cdot GMTRGT.\end{aligned}$$

Those values are the assigned condition for the two sides of the interface. If the interface is a boundary, the right condition is determined according to the type of boundary.

DYNPTN & DYYPTN

DYNPTN applies three distinct criteria to test cells to determine their need for refinement. They are as follows:

For unsteady dynamic simulation

- 1) total energy flux entering or leaving a cell
- 2) total density flux entering or leaving a cell
- 3) density gradient in each cell.

For steady state simulation

- 1) Pressure gradient in each cell
- 2) Mach number gradient in each cell
- 3) density gradient in each cell.

Cells that meet one of those three criteria are flagged, and are actually subdivided in DYYPTN until they meet the area criteria set for refinement (AREADD). The code will compute the maximum of each of the three criteria and set a 5% of the maximum or higher to the refinement criteria for the fluxes and 3% for the gradient. These criteria work extremely well for moving waves.

It should be noted that those error indicators and their levels are set according to the actual simulated condition. For different cases, other error indicators and level settings might be more appropriate than the above.

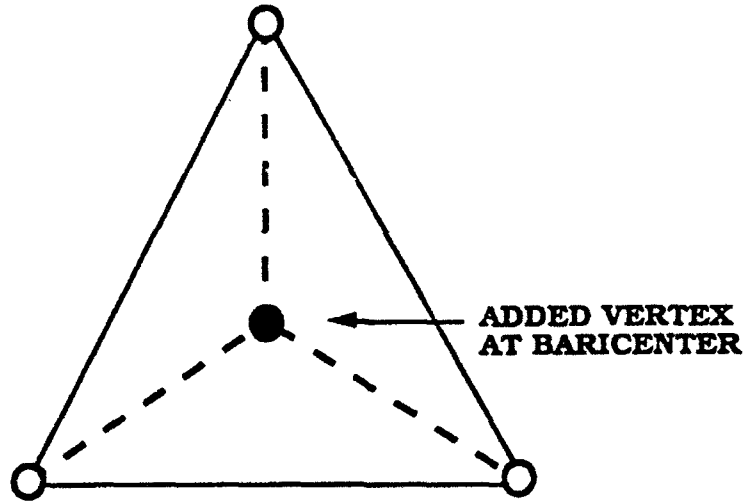
DELPTN

Tests the cells for coarsening criteria. The same criteria that refines the grid are applied to coarsen the grid but in a different setting. Each cell that has less than 5% of the fluxes and less than 3% of the gradient criteria is eligible for coarsening. The code will test the cell flagged for coarsening and will choose one of the three vertices of the cell for deletion by determining which of the three has the smallest aspect ratio. (The aspect ratio is defined as the ratio between the height emerging from the node and its corresponding base.) There are vertices that cannot be removed, such as corners or vertices that preserve the original shape of the boundaries ($JV(1,IV) = 3$).

After the vertex is deleted, a relaxing procedure is performed on the vertices surrounding the deleted vertex, as well as a swapping procedure to improve the quality of the triangles constructed in the deletion procedure.

VERCEN

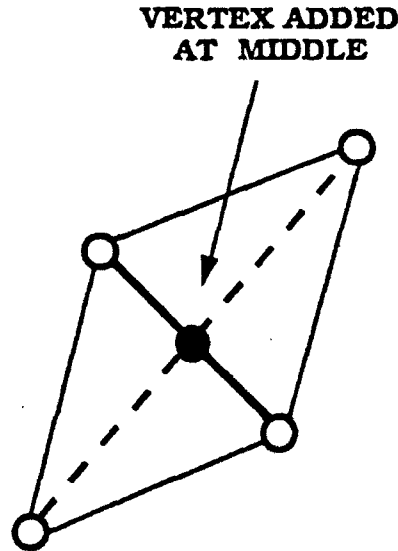
Adds an additional vertex at the baricenter of the designated cell.



VERCEN assigns one of the three new triangles the number of the original triangle and will add two more at the end of cells table. A new vertex plus three new interfaces are added at the end of the associated tables.

DISECT

Adds a new vertex at the middle of the designated edge.



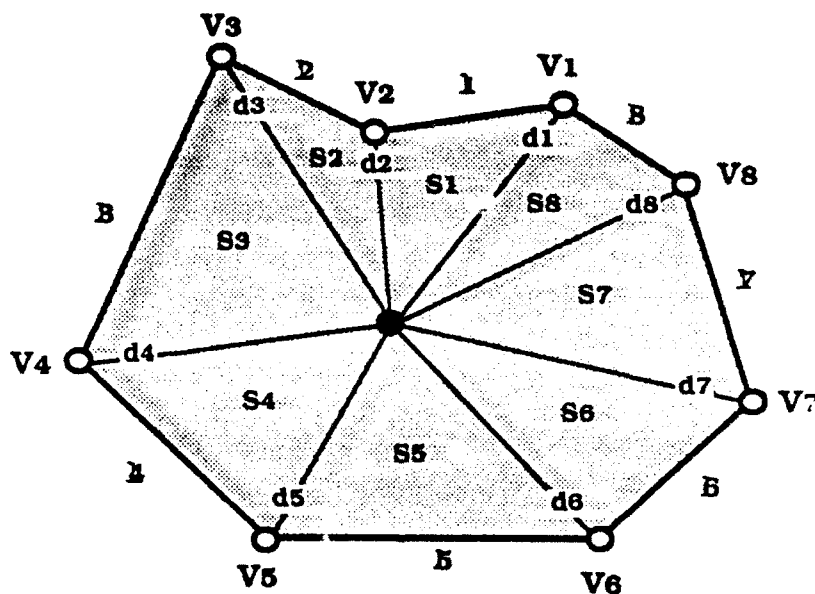
DISECT will add one new vertex, three new edges and two new triangles, all of which are added at the end of the corresponding tables (vertices, edges and cells).

VERDEL

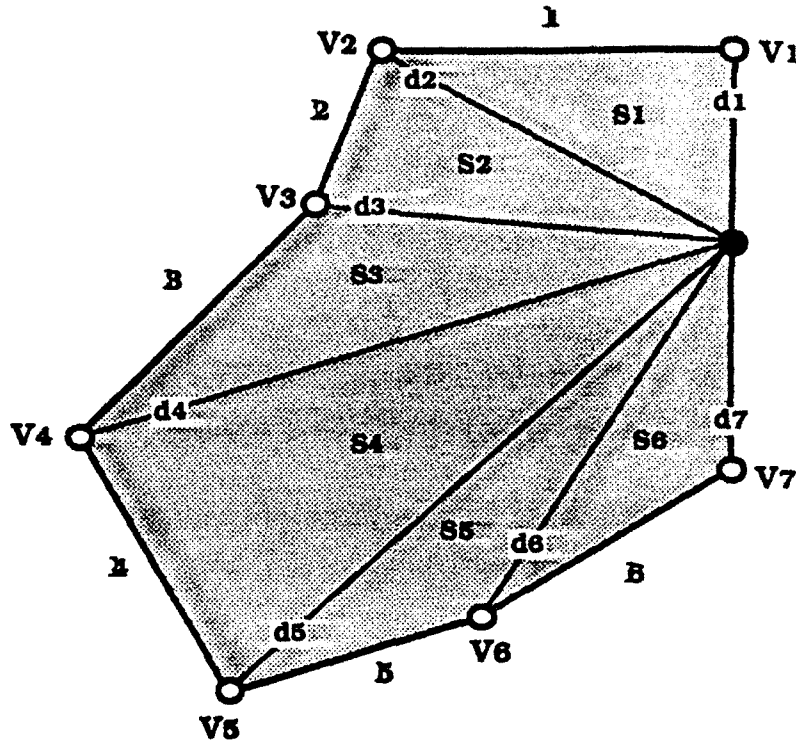
Forces deletion of a designated vertex. There are two types of vertices: deletion of a vertex in the interior of the computational domain and deletion of a vertex on the boundary. The steps of deleting a vertex are:

- 1) Identify the edges and cells surrounding the designated vertex in the computational domain

Interior Vertex to be Deleted



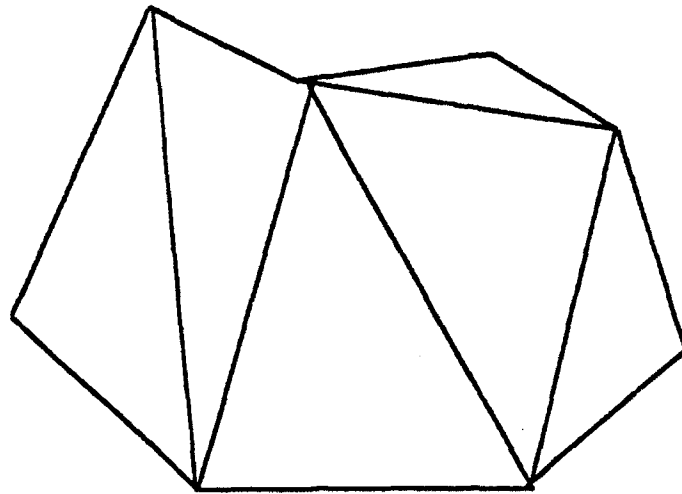
and on the boundary.



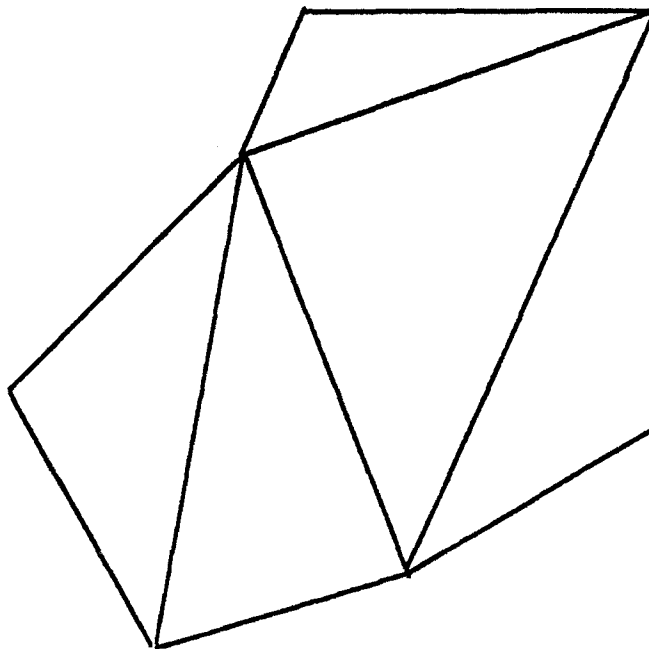
Deletion is more difficult and needs more computational resources than addition. The new vertices edges and cells being added are stacked at the bottom of the corresponding tables while undergoing deletion is always a member in the table. In order not to leave gaps in the table, a more complicated procedure was developed to replace the deleted member by the member at the bottom of the table.

- 2) Once the vertex, edges and cells joining the designated vertex are deleted we rezone the void (polygon) without adding new vertices. The adding of the new edges and cells are stacking at the end of the corresponding tables.

Interior Vertex



Boundary Vertex



3) A relaxation procedure is performed on the vertices of the polygon (void). This procedure improve the quality of the cells that fill the void.

- 4) A swap procedure is performed on the new edges that were added in the process of filling the void.

A.1.1 Pre-Processor for the Unstructured Grid

The input geometrical data for AUGUST should provide the following data:

- 1) Number of:
 - vertices (NV)
 - flagged vertices (NVM)
 - edges (NE)
 - cells (NS)
- 2) A table of vertices specifying:
 - number of vertex (IV)
 - x coordinate (XV(1,IV))
 - y coordinate (XV(2,IV)).
- 3) A table of flagged vertices that cannot be removed by the coarsening process:
 - number of vertex (IV)
 - status of vertex (JV(1, IV))

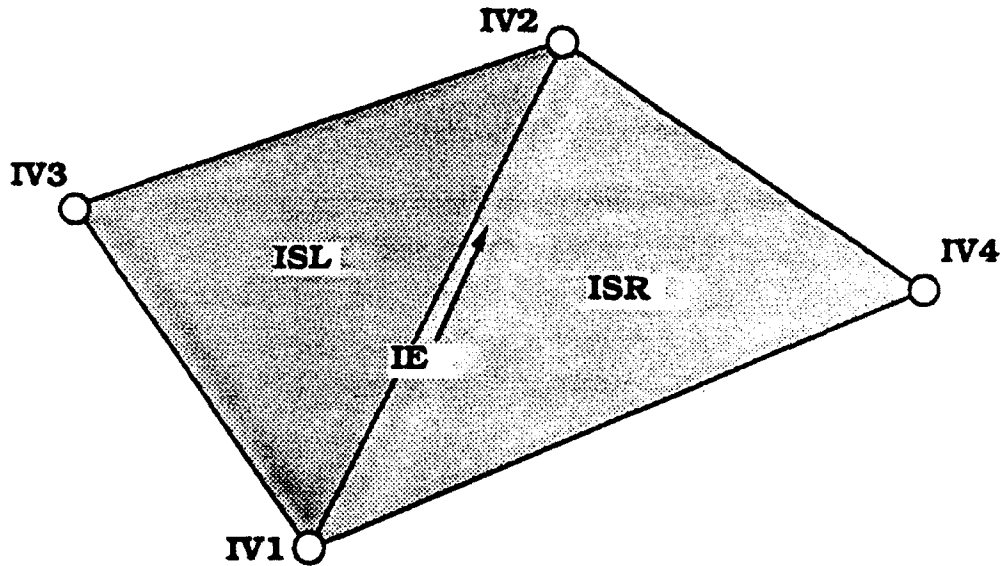
The only status of vertex that is currently implemented is the flagging node that does not allow removal:

JV(1,IV)=3

- 4) A table of edges specifying:
 - number of edges (IE)
 - vertex number indicating the beginning of the edge (JE(1,IE))
 - vertex number indicating the end of the edge (JE(2,IE))
 - cell number indicating the cell at the left of the edge (JE(3,IE))
 - cell number indicating the cell at the right of the edge (JE(4,IE))
 - number associated with the status of the edge (JE(5,IE))

If $JE(5,IE)=0$, the edge is an ordinary edge inside the computational domain.

If $JE(5,IE)\neq 0$, the edge lies on the boundary of the domain. The labeling number will indicate what type of boundary to be applied through this edge.



$IV1 = JE(1,IE)$ vertex indicating the beginning of the edge

$IV2 = JE(2,IE)$ vertex indicating the end of the edge

The direction of the edge is defined from IV1 to IV2.

ISL = JE(3,IE) left triangle
ISR = JE(4,IE) right triangle

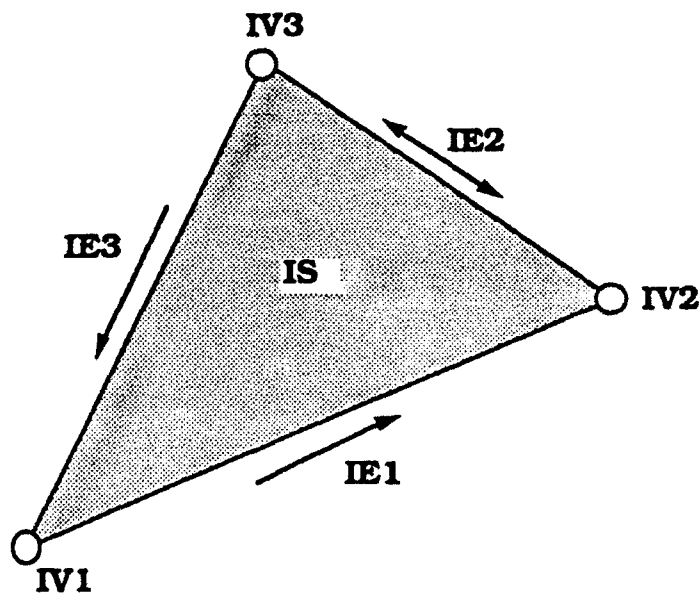
IJE5 = JE(5,IE) status of the edge

IJE5 = 5 simulating wall conditions
IJE5 = 6 simulating wall conditions
IJE5 = 7 simulating supersonic outlet conditions
IJE5 = 8 simulating supersonic inlet conditions

- 5) A table of cells specifying:
number of cells (IS)
number of first edge (JS(4,IS))
number of second edge (JS(5,IS))
number of third edge (JS(6,IS))

The sign of JS(4,IS), JS(5,IS), JS(6,IS) indicates whether the direction of the edge is counterclockwise (positive) or clockwise (negative).

The three associated vertices for the triangle JS(1,IS), JS(2,IS), JS(3,IS) are defined by the code in GEOMTR.



The three vertices of the cell are ordered in a counterclockwise arrangement.

IV1 = JS(1,IS) first vertex
IV2 = JS(2,IS) second vertex
IV3 = JS(3,IS) third vertex

IE1 = JS(4,IS) First edge of the triange directed from IV1 to IV2
 (IE1 is positive).

IE2 = JS(5,IS) Second edge directed originally from IV3 to IV2.
 (IE2 will be negative because its direction is
 clockwise)

IE3 = JS(6,IS) Third edge directed originally from IV3 to IV1 (IE3
 is positive)

A.1.2 Post-Processor for the Unstructured Grid

Postprocessing for visualization of the results on an unstructured grid is done in two different codes. The first code, DRAWBF, reads the data as dumped by AUGUST and performs the whole load of computation necessary to produce the information needed for the graphic.

The second code DRAWAF reads the data file written by DRAWBF and uses the DISSPLA software to produce the image on the screen. Breaking the postprocessing job into two separate codes enables the user to run the two codes on different machines.

DRAWBF

Reads an input data file produced by AUGUST and will read another input data file (drawbf.d) specifying the option that the user chooses to have processed.

The input data file drawbf.d specifies the window of the computational domain chosen by the user to be processed. This window is specified by XMIN, XMAX, DX and YMIN, YMAX, DY, where XMIN, XMAX, YMIN, YMAX, will specify the lower and upper limit of the region to be drawn. DX and DY will be parameters for DISSPLA to subdivide the axis into tick marks.

DISSPLA is constrained to seven colors. To extend the number of contour levels, the code can be set to draw a couple of levels in each color (7 x **NLEV** where **NLEV** is the number of levels for each color).

The user should specify the variable he wants to draw:

- IHYD = 1 is density,
- = 2 is velocity in the x direction
- = 3 is velocity in the y direction
- = 4 is pressure
- = 5 is gamma (γ)
- = 6 is Mach number
- = 7 is entropy
- = 8 is a vector plot of the velocity field
- = 9 is a plot of the location of particles

The last parameter that the user should specify is IREC. IREC specifies how many dumps are in the input file produced by AUGUST. If IREC=0, the user will get as many figures as the number of dumps produced by AUGUST. Otherwise, the user will get the figure corresponding to IREC specified in the input file.

Subroutine NEXTREC reads a whole dump from the input file (written by AUGUST). It will make sure that the allocation of memory is adequate according to the number of vertices, edges and triangles to be processed. If the memory allocation is not adequate, the code will stop with an explanatory message.

Subroutine LOADF loads the portion of data needed according to the specification of the window and according to the specified IHYD into the appropriate matrices in the code.

Subroutine PHYDR produces the data for the contour plots.

Subroutine VECTOR produces the data for the vector plot of the velocity field.

Subroutine TRACER produces the data for the location of particles.

DRAWAF

DRAWAF reads an input data file (drawbf.k) produced by DRAWBF and another input file (drawaf.d) that specifies the format chosen for display.

The parameters specified in drawaf.d are:

IFMESH=0

No grid is drawn.

IFMESH=1

Grid is drawn.

IOPTON=0

A single frame is drawn.

IOPTON=1

Two frames are drawn, one for the grid and one for displaying results. The frame for the grid is drawn even if IFMESH=0, but in this case the frame will stay empty.

IOPTON=2

Identical with IOPTN=1 except the level on the bar chart is written in engineering format (XE+Y). As in the former, it is written keeping a four decimal digit.

ICONFG=0

The basic dimension for the frames is specified as 6.0 x 3.0 inches (in the x and y axis, respectively). The code makes sure that the proportionality of the frame matches the physical window to be drawn, so that the figure will not be distorted. This is done by redefining the x or y dimension of the frame accordingly, but not to exceed the 6.0 x 3.0 on the screen (ICONFG=0 should be picked if IOPTON > 0 and a two-frame drawing is desired).

ICONFG>0

The same as ICONFG=0 except that the basic dimensions are defined now as 6.0 x 6.0 inches. This option should be specified if a one frame drawing is desired.

Caption 1
Caption 2

The user can specify a header for the drawing composed of two lines to be specified as Caption 1 and Caption 2 in the input file.

The standard drawing includes the number of vertices, edges and cells as well as the Mach number, lift, drag, moment, angle of attack (for drawing diagnostics for a wing profile). An indication of the nature of the results that appear on the drawing is also included, i.e., the physical variables drawn are identified by the parameter passing from DRAWBF.

It should be noted that the format of the output drawing is very easily redesigned to meet the needs of an individual user.

1. Read geometrical data defining the initial grid. The current format is set to read data file from Smart (two dimension grid generator).
2. Read geometrical data defining the grid read from a file dumped by a previous run of the code.
3. Initialize the physical variables according to IOPTN (either steady state or moving shock wave). If a different initial setting is needed, it should replace the current setting.
4. Read the physical variables from a file dumped by a previous run.

A.2 AUGUSTT (3D)

The subroutines in the AUGUSTT code are organized here as they appear in the listing in Appendix B. A brief description indicates the function performed by each subroutine.

TABLE A.2.1

LIST OF SUBROUTINES

The subroutines in the AUGUST code are organized here as they appear in the listing in Appendix B. A brief description indicates the function performed by each subroutine.

1. MAIN	Governing program for AUGUST. Reads input files and sets the mode for the computation.
2. HYDRFL	Computes the fluxes at interfaces by applying the Godunov algorithm to solve the Riemann problem across the interface.
3. HYDRMN	Controls the computation. The integration of the fluxes and update of the physical variables and writing to output files are performed in this subroutine.
4. GEOMTR	Calculates the geometrical quantities not provided by the input data file but needed for the computational algorithm. GEOMTR is only used once for starting a new simulation.
5. UPDATE	Reads the input file for a new simulation and calls GEOMTR to update the geometrical variables needed to perform the computation.
6. UPGRAD	Called if a restart run is performed. Will read the appropriate file written at the end of the previous run.

7. GRADNT	Computes the gradient of the physical variables to improve the prediction of those variables for the two sides of the interface. The gradients are subjected to the monotonicity condition that limits the projected values, thus preventing new maxima-minima to be caused artificially by interpolation (IOPORD = 2). Calls FCHART in order to compute projected values at the half timestep associated with the local characteristics of the flow.
8. FIRST	The equivalent of GRADNT if run in a first order mode (IOPORD = 1). Using FIRST assumes that the physical variables are constant in each cell. Takes care of the boundary conditions if the interface is a boundary.
9. FCHART	Computes the projected values at a half timestep for the two sides of the interface based on the local characteristics of the flow. Called by GRADNT, it modifies the projected values for the two sides of the interface and assigns them to the correct location in memory. Takes care of the boundary conditions if the interface is a boundary.

The MAIN Program

All of the data input and initiation of a run (or a restart run) is performed in MAIN. The actual simulation is controlled by HYDRMN, which is called from MAIN. At the completion of a run, control is returned to MAIN and a successful termination prints the message STOP 777.

MAIN contains one name list (file no. 2) and requires an input file that contains the grid data description (file no. 16). The data organization for the

grid file is described in Appendix A. The following files should be included: DMSH00.H, DPHS00.H, DHYD00.H.

NAMELIST/DATA	ICOND	ICONP	IOPTN
	XMCHIN	RIN	PIN
	ALFA	HRGG	IHRN
	NTIME	NDUMP	IOPORD

VARIABLE	PURPOSE
ICOND	= 0 READ INPUT GRID FOR A NEW SIMULATION = 1 READ THE GRID FROM PREVIOUS RUN

ICOND = 0:

MAIN will read the initial grid definition stored in file number 16. The current setting is to read the input file as provided by Smart, a two-dimensional triangular grid generator that runs interactively on a Macintosh personal computer.

MAIN will call UPDATE, which will call GEOMTR. GEOMTR will compute essential geometrical parameters that are not provided by file 16. All geometrical information is dumped into output files (8 and 88) so that ICOND=0 is used only once at the beginning of a new simulation.

ICOND = 1:

MAIN will call UPGRAD, which will call one of the output files (8 or 88) written by the previous run. This will load the geometrical definition of the grid (either 8 or 88---they are identical). Writing identical files provides a backup in the event that the job terminates for lack of time while in the process of writing to one of those output files.

VARIABLE	PURPOSE
ICONP	= 0 PRIMITIVE VARIABLES INITIALIZED = 1 VARIABLES READ FROM PREVIOUS RUN

ICONP = 0:

Initialize the primitive variables in computational domain with an initial value specified by the user. The two options set by the code are controlled by IOPTN.

ICONP = 1:

The flow field condition reads in files 8 or 88 and provides a follow-up run set from the previous run.

VARIABLE	PURPOSE
IOPTN	= 1 SOLUTION FOR STEADY STATE = 2 SOLUTION FOR TRANSIENT PHENOMENA

There are two choices available to set the initial condition of the problem.

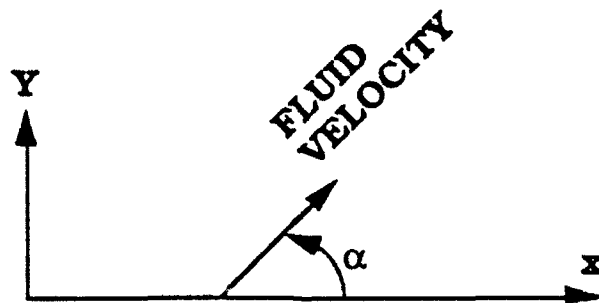
IOPTN = 1:

Assign the conditions at the inlet to the computational domain. This is the fastest way to get a steady-state solution for the conditions specified at the inlet. In this option, PIN (pressure), RIN (density) and XMCHIN (Mach number) are assigned to the pressure density and velocity (the speed of sound is computed in the code) and imposed at the inlet boundaries.

IOPTN = 2

Used if a shock wave is to be simulated moving from the inlet (edge boundary 8) to the outlet (edge boundary 7). For this setting, specify PIN (ambient pressure in the chamber), RIN (ambient density in the chamber) and XMCHIN (upstream Mach number). The code will use the normal shockwave relations for an adiabatic flow of a completely perfect fluid to compute the static-pressure ratio across the shock $P2/P1$ and the density ratio $r2/r1$, and the ratio of the Mach number across the shock $M2/M1$. These computed quantities are applied to set correctly the condition on the pressure density and velocity at the inlet boundary.

VARIABLE	PURPOSE
ALFA	THE DIRECTION OF INFLOW IN DEGREES RELATIVE TO A RIGHT-HAND COORDINATE SYSTEM. ALFA = 0 MEANS FLOW FROM LEFT TO RIGHT.



The velocity computed by the code according to the input data provided by the user is split (projected) in the X and Y directions by using α .

VARIABLE	PURPOSE
HRGG	INITIAL γ IN THE EQUATION OF STATE. THE CODE RUNS USING THE IDEAL EQUATION OF STATE AS A BASELINE AND SHOULD BE MODIFIED IF SOMETHING ELSE IS DESIRED. IOPEOS=1 WILL TRIGGER THE USE OF GILMORE EQUATION OF STATE.

VARIABLE	PURPOSE
IHRN	NUMBER OF ITERATIONS IN THE RIEMANN SOLVER TO FIND THE DIAPHRAGM SOLUTION. (THREE TO FOUR SHOULD BE USED AND THE NUMBER INCREASED ONLY FOR VERY HIGH MACH NUMBER CASES.)

VARIABLE	PURPOSE
NTIME	NUMBER OF REPEATS FOR THE INTEGRATION SEQUENCE. AN OUTPUT DUMP IS DONE FOR EVERY SEQUENCE REPEAT.

VARIABLE	PURPOSE
NDUMP	NUMBER OF OUTER LOOP ITERATIONS IN THE CALCULATION WHERE REFINING IS DONE FOR EVERY SEQUENCE REPEAT WITHOUT COARSENING.

VARIABLE	PURPOSE
IOPORD	= 1 THE CODE WILL RUN FIRST ORDER GODUNOV METHOD = 2 THE CODE WILL RUN SECOND ORDER GODUNOV METHOD

IOPORD = 1

Subroutine FIRST is called.

IOPORD = 2

Subroutine GRADNT is called.

HYDREL

Computes the fluxes across interfaces when the conditions for both sides are given. The fluxes are computed assuming a shock solution at a ruptured diaphragm simulated by the presence of the interface. The conditions existing on the two sides of the diaphragm will define the condition of the flow at the diaphragm location. These conditions are computed by solving the Riemann problem using the Godunov algorithm. The condition at the diaphragm defines the flux of energy, mass, and momentum passing across the interface. The Euler conservation law is applied to conserve energy, mass, and momentum crossing interfaces from one cell to the other.

Quantity	Side 1	Diaphragm (Interface)	Side 2
Density	ρ_1	ρ	ρ_2
Pressure	P_1	P	P_2
Velocity Perpendicular to Interface	u_1	u	u_2
Velocity Parallel to Interface	v_1	v	v_2
Velocity Parallel to Interface to Construct a Right-Hand Coordinate System (u, v, w.)	w_1	w	w_2

HYDRMN

Controls the code and the iteration loops. It calls HYDRFL to find the interface fluxes. These fluxes are integrated to update the physical variables in each cell. If adaptation of the grid is required, HYDRMN also controls the output by writing the necessary information on files for postprocessing data and for restarting the AUGUST code at a later time. It also manages print file diagnostics.

GEOMTR

Calculates geometrical variables that are not supplied by the input data and are needed to run the code. For example, it computes:

- 1) distances between baricenters of adjoining cells;
- 2) the location of the intersection between the line joining adjacent baricenter cells and the interface.

The code changes the direction of the boundary edges so that all are arranged counter clockwise and the associated computational cell is always on the left side. GEOMTR is called once in the beginning of a new simulation.

UPDATE

Called in the beginning of a new simulation for setting geometrical variables not provided by the input data. (It calls GEOMTR.)

UPGRAD

Called if the run is a restart. UPGRAD will read the appropriate file (either 8 or 88) dumped by the previous run.

GRADNT

Compute the gradients of the physical variables in each cell. These computed gradients, along with the physical values at the baricenters, are applied using linear interpolation to predict the values on the interface.

The computed gradients are subjected to the monotonicity condition, ensuring that the projected values are bounded by the value of each quantity in the three adjacent cells, and to make sure that no new maxima or minima occur. The projection of quantities to the interface improves the results from the code and provides second order accuracy in space.

GRADNT calls FCHART, which computes the projected values at the interfaces at the half timestep level according to the local characteristics of the flow in each cell bordering the interface cell. The assignment of values at the two sides of each interface is done at the end of FCHART. This same loop also imposes the boundary conditions for the interfaces at the boundaries of the computational domain.

FIRST

Assigns flow quantities to each side of an edge. These are based on the values at the baricenter of the triangles on either side of the edge. FIRST uses a first order approximation to find the values at the edge.

The user can specify FIRST or GRADNT by choosing 1 or 2 for the parameter IOPORD.

FCHART

Called by GRADNT to compute the values projected at the interfaces at the half timestep. These calculations are done by applying the local velocity characteristics in each cell. This projection in time improves the results and makes the code second order accurate in time.

GRADNT

Computes the gradient of a scalar variable at the center of a cell. The gradient theorem is applied for each cell.

$$\int_{\text{volume}} \nabla \cdot d\mathbf{v} = \int_{\text{four surfaces}} f \bar{\mathbf{n}} \, ds$$

Those gradients are subjected to a monotonicity limiter that ensures no new minima or maxima are produced artificially in the projected values at the interfaces.

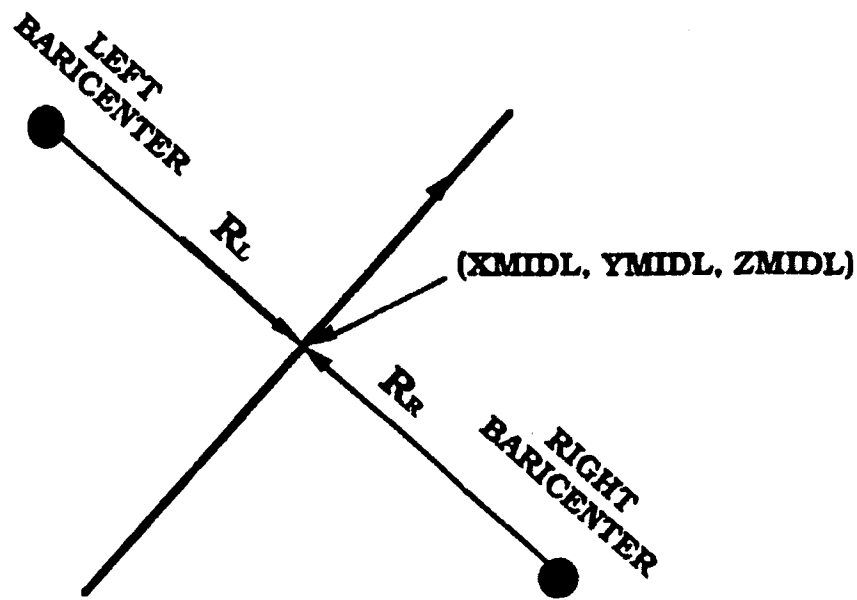
The monotonicity algorithm involves the following steps.

1)	find maximum and minimum of f_1, f_2, f_3, f_4, f_5 $f_{\max} = \text{Max}(f_1, f_2, f_3, f_4, f_5)$ $f_{\min} = \text{Min}(f_1, f_2, f_3, f_4, f_5)$
2)	compute $\Delta f_{\max} = f_{\max} - f_1$ $\Delta f_{\min} = f_{\min} - f_1$
3)	compute incremental projected values at the interfaces $f_{mjR} - f_R = \bar{\nabla} f_R \cdot \bar{\mathbf{r}}_{jR}$ $f_{mjL} - f_L = \bar{\nabla} f_L \cdot \bar{\mathbf{r}}_{jL}$

$$Df_{mjR} = f_{mjR} - f_R = \bar{\nabla} f_R \cdot \bar{\mathbf{r}}_{jR}$$

$$Df_{mjL} = f_{mjL} - f_L = \bar{\nabla} f_L \cdot \bar{\mathbf{r}}_{jL}$$

where j stands for every interface of the cell and f_{mj} is the interpolated value at the middle of the interface.



4) compute the limiter by calculating the minimum of indicator for each edge of the four surfaces of the cell.

$$\text{right to the interface RUVPR} = \frac{(1 + \text{sign } \Delta f_{mjR}) \Delta f_{\max} + (1 - \text{sign } \Delta f_{mjR}) \Delta f_{\min}}{2 \Delta f_{mjR}}$$

$$\text{left to the interface RUVPL} = \frac{(1 + \text{sign } \Delta f_{mjL}) \Delta f_{\max} + (1 - \text{sign } \Delta f_{mjL}) \Delta f_{\min}}{2 \Delta f_{mjL}}$$

This formulation ensures that:

$$\text{if } \begin{cases} \Delta f_{mj} > 0 & \text{RUV} = \frac{\Delta f_{\max}}{\Delta f_{mj}} \\ \Delta f_{mj} < 0 & \text{RUV} = \frac{\Delta f_{\min}}{\Delta f_{mj}} \end{cases}$$

the outcome of RUV is always positive. If $\text{RUV} > 1$ then the projected value at the interfaces will introduce a new minima or maxima as compared to the values at the baricenters of the appropriate cells.

Select the minimum between the six values for RUVP (two for every one of the three interfaces of the cell) not exceeding unity. The selected minimum of RUVP is the required limiter. The gradient is multiplied by this limiter that is always less or equal to unity.

RCHART

Computes the projected values at the half timestep level based on the local characteristics of the flow. This process extends the accuracy of the code to be second-order in time as well as in space.

The characteristics projection consists of several steps.

- 1) Calculate the velocity of sound in the two cells bordering the designated interface:

$$CNLEFT = \sqrt{\gamma_L \cdot P_L / \rho_L} \quad \text{sound speed in left cell}$$

$$CNRIGHT = \sqrt{\gamma_R \cdot P_R / \rho_R} \quad \text{sound speed in right cell}$$

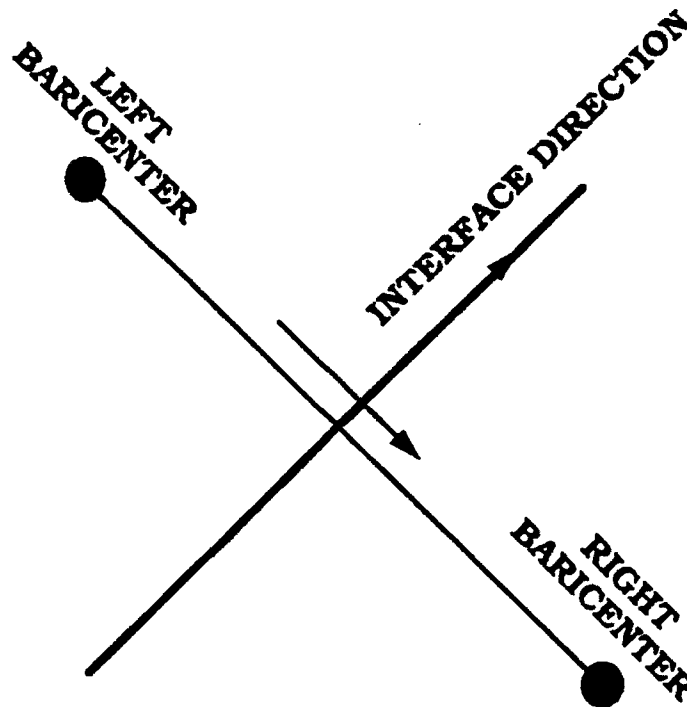
$$UVLEFT = \bar{U}_L \cdot \bar{t} \quad \text{velocity of fluid at the left cell projected in } \bar{t} \text{ direction}$$

$$UVRIGHT = \bar{U}_R \cdot \bar{t} \quad \text{velocity of fluid at the right cell projected in } \bar{t} \text{ direction}$$

where:

$$\bar{t} = \text{XXN} \cdot \bar{i} + \text{YYn} \cdot \bar{j} + \text{zzn} \cdot \bar{k}$$

$$\bar{U} = U \cdot \bar{i} + v \cdot \bar{j} + w \cdot \bar{k}$$



2) To compute the interpolated left and right projected values at time $t_N + \Delta t/2$, we calculate the distances that the disturbances generated from the baricenter of the cells, traveling toward the interface:

$$ZZLEFT = (UVLEFT + CNLEFT) \cdot \Delta t/2$$

$$ZZRIGHT = - (UVRIGHT - CNRIGHT) \cdot \Delta t/2$$

If $ZZLEFT$ or $ZZRIGHT$ are negative they are reset to zero.

3) Calculate the distances that the flow will travel if it were to flow at the velocity of each of the local characteristics:

$$ZOLEFT = UVLEFT \cdot \Delta t/2$$

$$ZORIGHT = - UVRIGHT \cdot \Delta t/2$$

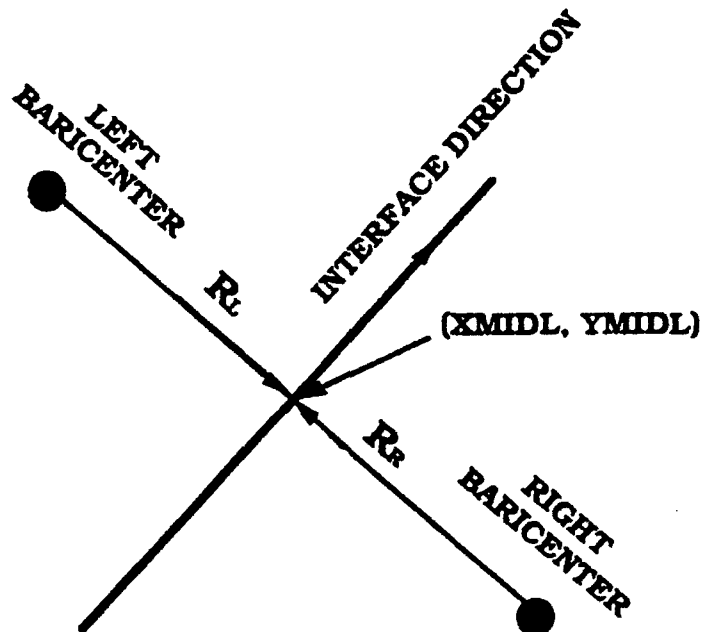
$$ZPLEFT = (UVLEFT + CNLEFT) \cdot \Delta t/2$$

$$ZPRIGHT = - (UVRIGHT + CNRIGHT) \cdot \Delta t/2$$

$$ZMLEFT = (UVLEFT - CNLEFT) \cdot \Delta t / 2$$

$$ZMRIGT = -(UVRIGT - CNRIGT) \cdot \Delta t / 2.$$

4) Calculate the projected values of the nonconservative variables (density, velocity component (perpendicular and tangential to the interface), and pressure).



For the left cell:

Density	HRRL	=	$\rho_L + \bar{V} \rho_L \cdot (\bar{r}_L - ZZLEFT \cdot \bar{t})$
---------	------	---	--

Perpendicular Velocity	HUUL	=	$U_L + \bar{V} U_L \cdot (\bar{r}_L - ZZLEFT \cdot \bar{t})$
------------------------	------	---	--

Tangential Velocity	HVVL	=	$V_L + \bar{V} V_L \cdot (\bar{r}_L - ZZLEFT \cdot \bar{t})$
---------------------	------	---	--

Pressure	HPPL	=	$P_L + \bar{V} P_L \cdot (\bar{r}_L - ZZLEFT \cdot \bar{t})$
----------	------	---	--

	GMTLFT	=	$\rho_L \cdot HRRL \cdot HPPL$
--	--------	---	--------------------------------

For the right cell:

Density	HRRR	=	$\rho_R + \bar{V}\rho_R \cdot (\bar{r}_R - ZZRIGHT \cdot \bar{t})$
Perpendicular velocity	HUUR	=	$U_R + \bar{V}U_R \cdot (\bar{r}_R - ZZRIGHT \cdot \bar{t})$
Tangential velocity	HVVR	=	$V_R + \bar{V}V_R \cdot (\bar{r}_R - ZZRIGHT \cdot \bar{t})$
Pressure	HPPR	=	$P_R + \bar{V}P_R \cdot (\bar{r}_R - ZZRIGHT \cdot \bar{t})$
	GMTRGT	=	$\rho_R \cdot HRRR \cdot HPPR$

For the left cell, taking into account the following characteristics:

- For (UVLEFT + CNLEFT):

$$UUU = \bar{V}U_L \cdot (ZPLEFT - ZZLEFT) \bar{t}$$

$$PPP = \bar{V}P_L \cdot (ZPLEFT - ZZLEFT) \bar{t}$$

$$UPLFT = -0.5 \cdot (UUU + PPP / \sqrt{GMTLFT}) / \sqrt{GMTLFT}$$

If UVLEFT + CNLEFT is negative, UPLFT is reset to zero.

- For UVLEFT - CNLEFT:

$$UUU = \bar{V}U_L \cdot (ZMLEFT - ZZLEFT) \cdot \bar{t}$$

$$PPP = \bar{V}P_L \cdot (ZMLEFT - ZZLEFT) \cdot \bar{t}$$

$$UMLFT = 0.5 \cdot (UUU - PPP / \sqrt{GMTLFT}) / \sqrt{GMTLFT}$$

If UVLEFT - CNLEFT is negative, UPLFT is reset to zero.

- For UVLEFT:

$$PPP = \bar{V}P_L \cdot (ZOLEFT - ZZLEFT) \cdot \bar{t}$$

$$RRRR = \rho_L + \bar{V}\rho_L \cdot (\bar{r}_L - ZOLEFT) \cdot \bar{t}$$

$$URLFT = PPP/GMTLFT + 1/HRRL - 1/RRRR$$

If UVLEFT is negative, URLEFT is reset to zero.

For the right cell, taking into account the following characteristics:

- For UVRIGT + CNRIGT:

$$UUU = \bar{V}U_R \cdot (ZZRIGT - ZPRIGT) \bar{t}$$

$$PPP = \bar{V}P_R \cdot (ZZRIGT - ZPRIGT) \bar{t}$$

$$UPRGT = -0.5 \cdot (UUU + PPP / \sqrt{GMTRGT}) / \sqrt{GMTRGT}$$

If UVRIGT + CNRIGT is positive, UMRGT is reset to zero.

- For UVRIGT - CNRIGT:

$$UUU = \bar{V}U_R \cdot (ZZRIGT - ZMRIGT) \cdot \bar{t}$$

$$PPP = \bar{V}P_R \cdot (ZZRIGT - ZMRIGT) \cdot \bar{t}$$

$$UMRGT = 0.5 \cdot (UUU - PPP / \sqrt{GMTRGT}) / \sqrt{GMTRGT}$$

If UVRIGT - CNRIGT is positive, UMRGT is reset to zero.

- For UVRIGT:

$$PPP = \bar{V}P_R \cdot (ZZRIGT - ZORIGT) \cdot \bar{t}$$

$$RRRR = \rho_R + \bar{V}\rho_R \cdot (\bar{r}_R + ZORIGT) \cdot \bar{t}$$

$$URRGT = PPP/GMTRGT + 1/HRRR - 1/RRRR$$

If UVRIGT - CNRIGT is positive, URRGT is reset to zero.

The projected values will be:

$$RRL = 1 / (1/HRRL - (UPLFT + UMLFT + URLFT))$$

$$UUL = HUUL + (UPLFT - UMLFT) \sqrt{GMTLFT}$$

$$\begin{aligned}
VVL &= HVVL + (UPLFT - UMLFT) \sqrt{GMTLFT} \\
PPL &= HPPL + (UPLFT + UMLFT) GMTLFT \\
RRR &= 1/(1/HRRR - (UPRGT + UMRGT + URRGT)) \\
UUR &= HUUR + (UPRGT - UMRGT) \sqrt{GMTRGT} \\
VVR &= HVVR + (UPRGT - UMRGT) \sqrt{GMTRGT} \\
PPR &= HPPR + (UPRGT + UMRGT) \cdot GMTRGT.
\end{aligned}$$

Those values are the assigned condition for the two sides of the interface. If the interface is a boundary, the right condition is determined according to the type of boundary.

A.2.1 Preprocessor for the Three-Dimensional Unstructured Grid

The input geometrical data for AUGUST should provide the following data:

- 1) Number of vertices (NV)
- 2) A table of vertices specifying:
 - number of vertex (IV)
 - x coordinate (XV(1,IV))
 - y coordinate (XV(2,IV))
 - z coordinate (XV(3,IV)).
- 3) Number of edges (NE)
- 4) A table of edges specifying
 - number of edges (IE)
 - vertex number indicating the beginning of the edge (JE(1,IE))
 - vertex number indicating the end of the edge (JE(2,IE))
 - IV1 = JE(1,IE) vertex indicating the beginning of the edge
 - IV2 = JE(2,IE) vertex indicating the end of the edge

The direction of the edge is defined from IV1 to IV2.

- 5) Number of sides (NS)
- 6) A table of sides (triangles) specifying:
 - number of sides (IS)
 - number of first vertice (JS(1,IS))
 - number of second vertices (JS(2,IS))
 - number of third vertices (JS(3,IS))
 - number of first edge (JS(4,IS))
 - number of second edge (JS(5,IS))
 - number of third edge (JS(6,IS))

The sign of JS(4,IS), JS(5,IS), JS(6,IS) indicates whether the direction of the edge is counter clockwise (positive) or clockwise (negative).

tetrahedra on left to the side (JS(7,IS))

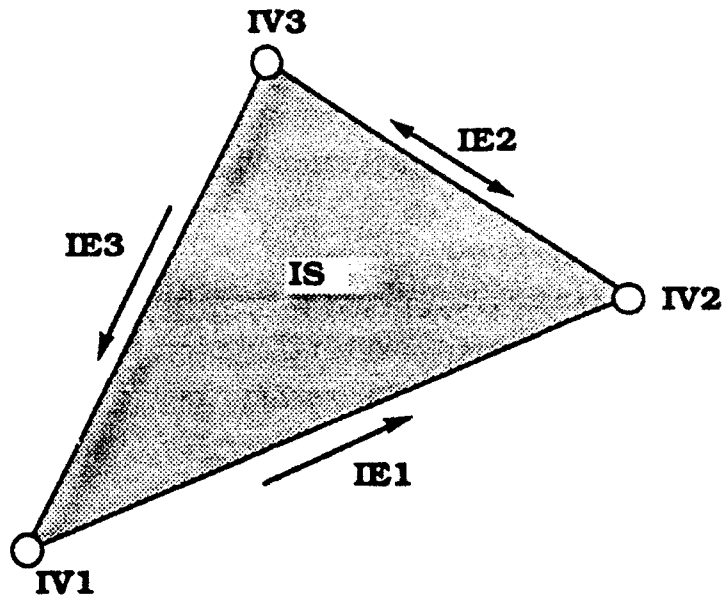
tetrahedra on right to the side (JS(8,IS))

Number associated with the status of the side (JS(9,IS)).

if JS(9,IS) = 0 the side is an ordinary side inside the computational domain.

if JS(9,IS) \neq 0 the side lies on the boundary of the domain.

The labeling number will indicate what type of boundary to applied through this side.



The three vertices of the side are ordered in a counter clockwise arrangement.

- | | |
|----------------|--|
| IV1 = JS(1,IS) | first vertex |
| IV2 = JS(2,IS) | second vertex |
| IV3 = JS(3,IS) | third vertex |
| IE1 = JS(4,IS) | First edge of the triangle directed from IV1 to IV2
(IE1 is positive). |
| IE2 = JS(5,IS) | Second edge directed originally from IV3 to IV2.
(IE2 will be negative because its direction is clockwise.) |
| IE3 = JS(6,IS) | Third edge directed originally from IV3 to IV1 (IE3
is positive). |
| IC1 = JS(7,IS) | tetrahedra on the left |
| IC2 = JS(8,IS) | tetrahedra on the right |

The normal to the side is directed from IC1 toward IC2. If the side is a boundary, the normal is always from the computational domain pointing outside (out of the fluid domain). The three vertices are ordered in a counter clockwise direction opposite to the direction of the normal to the side. For a boundary side, IC2 will be always zero.

IJS = JS(9,IS)	Status of the side
IJS9 = 6	Simulating wall conditions
IJS9 = 7	Simulating supersonic outlet conditions
IJS9 = 8	Simulating supersonic inlet conditions.

- 7) A table of sides specifying:
 x coordinate of baricenter of side (XS(1,IS))
 y coordinate of baricenter of side (XS(2,IS))
 z coordinate of baricenter of side (XS(3,IS))
 area of side (XS(4,IS))

- 8) A table of sides specifying:
 the three component of the vector normal to the side:

$$\vec{N} = XN(IS) \vec{i} + yN(IS) \vec{j} + ZN(IS) \vec{k}$$

the three component of the parallel vector tangential to the side:

$$\vec{P} = XP(IS) \vec{i} + YP(IS) \vec{j} + ZP(IS) \vec{k}$$

the three component of the parallel vector tangential to the side:

$$\vec{T} = XT(IS) \vec{i} + YT(IS) \vec{j} + ZT(IS) \vec{k}$$

where $\vec{P} \times \vec{T} = \vec{N}$ (the normal, perpendicular and parallel vectors form a local right-handed coordinate system).

- 9) number of cells (tetrahedrals) (NC)
- 10) A table of cells specifying:
 Number of cells (IC)
 Number of first vertex (JC(1,IC))
 Number of second vertex (JC(2,IC))
 Number of third vertex (JC(3,IC))
 Number of fourth vertex (JC(4,IC))
 Number of the first side (JC(5,IC))
 Number of the second side (JC(6,IC))

Number of the third side (JC(7,IC))

Number of the fourth side (JC(8,IC))

IV1 = JC(1,IC) first vertex

IV2 = JC(2,IC) second vertex

IV3 = JC(3,IC) third vertex

IV4 = JC(4,IC) fourth vertex

Seen from inside the tetrahedron, the first three vertices are counter clockwise around the large with the fourth vertex at the apex.

IS1 = JC(5,IC) first side

IS2 = JS(6,IC) second side

IS3 = JS(7,IC) third side

IS4 = JS(8,IC) fourth side

Face ISJ is opposite the IVJ vertex

11) A table of cells specifying:

x coordinate of the baricenter of cell (XC(1,IC))

y coordinate of the baricenter of cell (XC(2,IC))

z coordinate of the baricenter of cell (XC(3,IC))

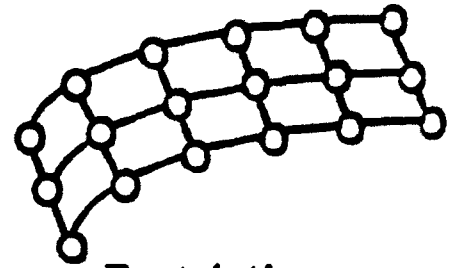
Volume of the cell (XC(4,IC))

A.2.2 Face(Triangle) information

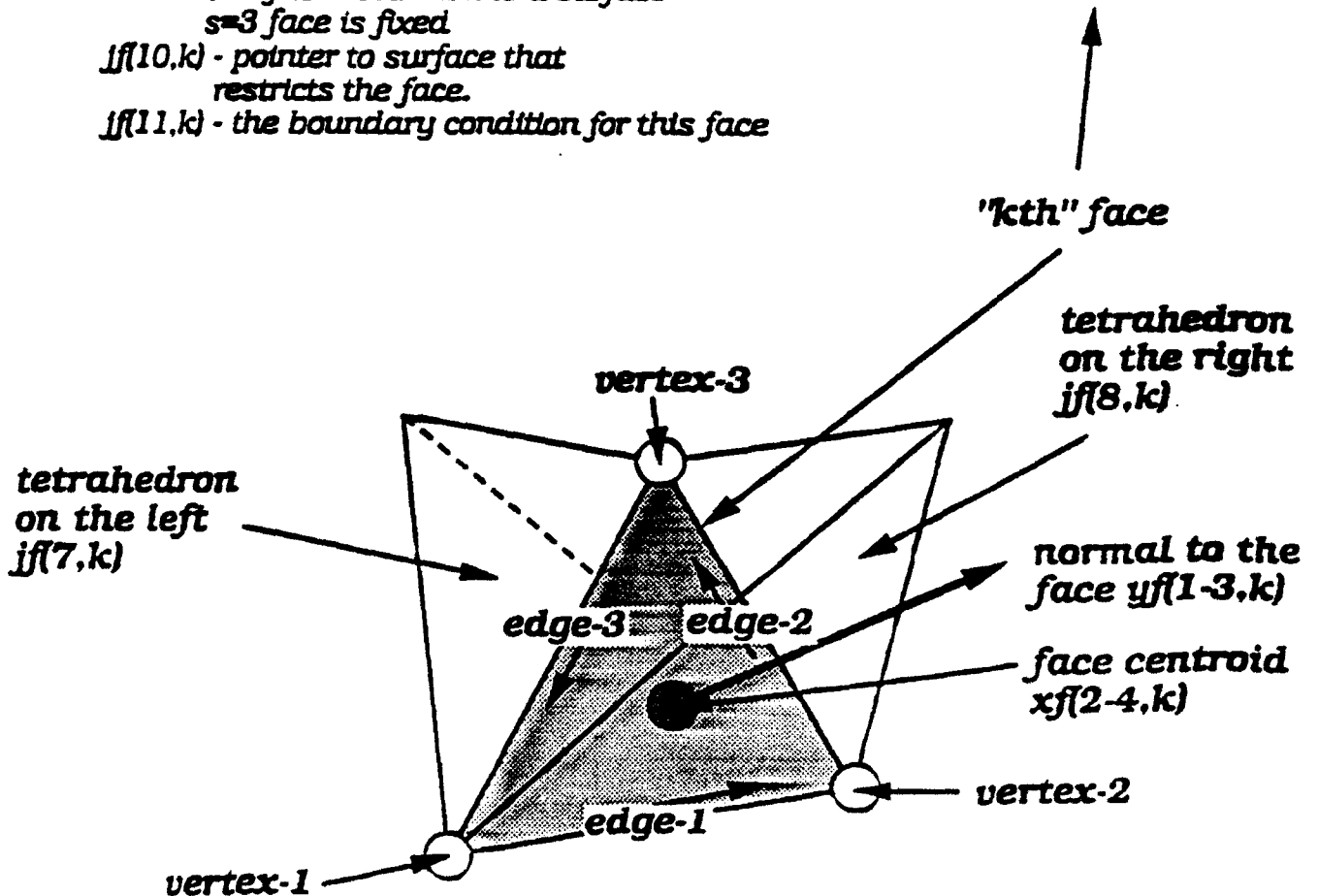
$xf(1,k)$ - area of the k th face
 $xf(2,k)$ - x position of face centroid
 $xf(3,k)$ - y position of face centroid
 $xf(4,k)$ - z position of face centroid

$yf(1,k)$ - x component of normal to face
 $yf(2,k)$ - y component of normal to face
 $yf(3,k)$ - z component of normal to face

$jf(1,k)$ - the index of the first vertex
 $jf(2,k)$ - the index of the second vertex
 $jf(3,k)$ - the index of the third vertex
 $jf(4,k)$ - the signed index of the first edge
 $jf(5,k)$ - the signed index of the second edge
 $jf(6,k)$ - the signed index of the third edge
 $jf(7,k)$ - the index of the tetrahedron
 to the left of face
 $jf(8,k)$ - the index of the tetrahedron
 to the right of face
 $jf(9,k)$ - status of the k th face
 $s=0$ face unrestricted
 $s=1$ not used
 $s=2$ face restricted to a surface
 $s=3$ face is fixed
 $jf(10,k)$ - pointer to surface that
 restricts the face.
 $jf(11,k)$ - the boundary condition for this face



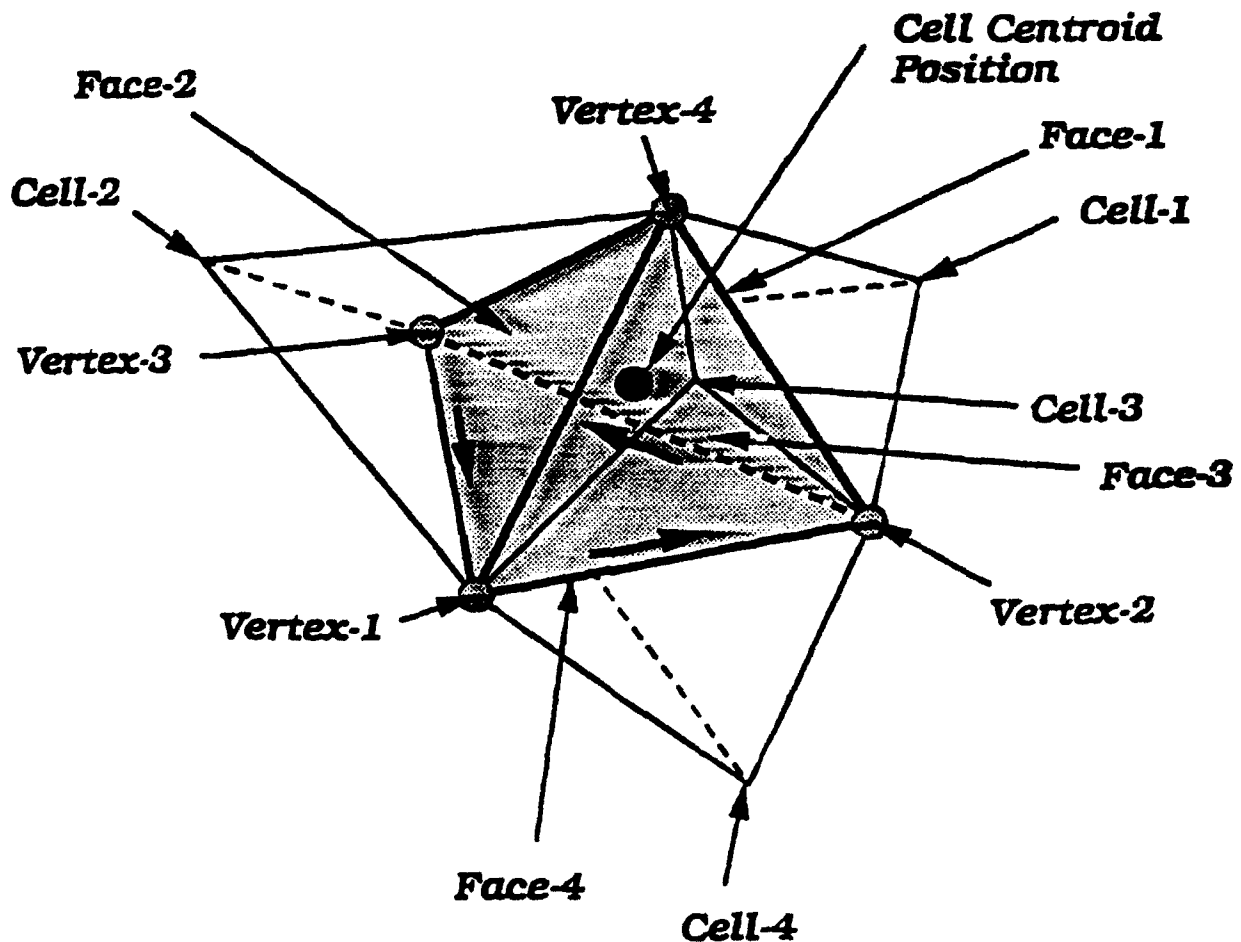
Restricting
 line/surface
 element $jf(10,k)$
 when $jf(9,k) > 0$



Cell(Tetrahedral) information

xc(1.k) - x position of cell centroid
xc(2.k) - y position of cell centroid
xc(3.k) - z position of cell centroid
xc(4.k) - volume of cell

jc(1.k) - the index of the first base vertex
jc(2.k) - the index of the second base vertex
jc(3.k) - the index of the third base vertex
jc(4.k) - the index of fourth vertex opposite base
jc(5.k) - the index of face opposite first vertex
jc(6.k) - the index of face opposite first vertex
jc(7.k) - the index of face opposite second vertex
jc(8.k) - the index of face opposite third vertex
jc(9.k) - the index of face opposite fourth vertex
jc(10.k) - the index of cell opposite first vertex
jc(11.k) - the index of cell opposite second vertex
jc(12.k) - the index of cell opposite third vertex
jc(13.k) - the index of cell opposite fourth vertex



APPENDIX B

LISTINGS

Thu Jul 1 14:17:00 1993

threed.f

Module List - order of occurrence

page i

#	routine	page
1	main	1
2	HYDRFL	13
3	RYDRFL	19
4	KYDRFL	22
5	HYDRMN	26
6	GEOMTR	33
7	UPGRAD	38
8	GRADNT	39
9	FIRST	51
10	FCHART	53
11	EOS1	59
12	MATRLA	62
13	PSM	64
14	BILD	64
15	MATRLX	64
16	VOLMTETC	66

Thu Jul 1 14:17:00 1993

threed.f

Module List - alphabetical order

#	routine	page
1	BILD	64
2	EOS1	59
3	FCHART	53
4	FIRST	51
5	GEOMTR	33
6	GRADNT	39
7	HYDRFL	13
8	HYDRMN	26
9	KYDRFL	22
10	MATRLA	62
11	MATRLX	64
12	PSM	64
13	RYDRFL	19
14	UPGRAD	38
15	VOLMTETC	66
16	main	1

```

1      1      PROGRAM AUGUSTT
2      2      C
3      3      C-----
4      4      C
5      5      C      The AUGUSTT Code
6      6      C
7      7      C      - Adaptive
8      8      C      - Unstructured
9      9      C      - Godunov
10     10     C      - Upwind
11     11     C      - Second order
12     12     C      - Triangular
13     13     C      - Three dimension
14     14     C
15     15     C      The geometry structure comes from BERMUDA
16     16     C      The solver is based on FUGGS
17     17     C
18     18     C
19     19     C      Version: 1.00      22 July, 1991
20     20     C
21     21     C      Authors:  Itzhak Lottati (703)749-8648
22     22     C      Shmuel Eidelman (703)448-6491
23     23     C      Adam Drobot (703)734-5840
24     24     C
25     25     C      Science Applications International Corporation
26     26     C      Applied Physics Operation
27     27     C      1710 Goodridge Drive
28     28     C      McLean, Virginia 22102
29     29     C
30     30     C-----
31     31     C
32     32     C
33     33     C-----I
34     34     C      I
35     35     C      BERMUDA IS A MULTIDIMENSIONAL CODE WHICH IS BASED ON THE I
36     36     C      USE OF TRIANGULAR GRIDS AS THE FUNDAMENTAL MESH I
37     37     C      FOR FIELD LIKE QUANTITIES. THE CODE REQUIRES I
38     38     C      THAT ALL QUANTITIES ARE BASED AT THE BARICENTER I
39     39     C      OF SIDES/TRIANGLES. I
40     40     C      I
41     41     C      THE QUIP IS THAT THOSE WHO WORK ON BERMUDA I
42     42     C      TRIANGLES ARE NEVER HEARD FROM AGAIN. I
43     43     C      I
44     44     C      THE BASIC MODULES IN BERMUDA INCLUDE: I
45     45     C      I
46     46     C      A HYDRODYNAMICS CODE I
47     47     C      .BASED ON A FIRST ORDER GODUNOV I
48     48     C      METHOD OR A SECOND ORDER GODUNOV I
49     49     C      WITH MESH ADAPTATION. I
50     50     C      I
51     51     C-----I
52     52     C
53     53     C      GRID SETUP TABLES AND THEIR MEANING:
54     54     C
55     55     C      +-----+
56     56     C      +
57     57     C      + LIST OF VERTICES +
58     58     C      + +
59     59     C      + IV - VERTEX INDEX +
60     60     C      + XV(1,IV) - X POSITION OF VERTEX +
61     61     C      + XV(2,IV) - Y POSITION OF VERTEX +
62     62     C      + XV(3,IV) - Z POSITION OF VERTEX +
63     63     C      + +
64     64     C      +-----+
65     65     C
66     66     C      +-----+
67     67     C      +
68     68     C      + LIST OF EDGES +
69     69     C      + +
70     70     C      + IE - EDGE INDEX +
71     71     C      + JE(1,IE) - INDEX OF LOWER EDGE VERTEX +
72     72     C      + JE(2,IE) - INDEX OF UPPER EDGE VERTEX +
73     73     C      + JE(3,IE) - INDEX OF LEFT SIDE +

```


74	74	C	+	JE(4,IE)	- INDEX OF RIGHT SIDE	+	74	
75	75	C	+	XE(1,IE)	- LENGTH OF EDGE	+	75	
76	76	C	+	XE(2,IE)	- DISTANCE BETWEEN ADJOINING SIDE	+	76	
77	77	C	+		POINTS.	+	77	
78	78	C	+			+	78	
79	79	C	+	+++++				79
80	80	C	+	+++++				80
81	81	C	+	+++++				81
82	82	C	+	+++++				82
83	83	C	+	LIST OF SIDES		+	83	
84	84	C	+			+	84	
85	85	C	+	IS	- SIDE INDEX	+	85	
86	86	C	+	JS(1,IS)	- INDEX OF FIRST VERTEX	+	86	
87	87	C	+	JS(2,IS)	- INDEX OF SECOND VERTEX	+	87	
88	88	C	+	JS(3,IS)	- INDEX OF THIRD VERTEX	+	88	
89	89	C	+			+	89	
90	90	C	+	THE VERTICES RUN AROUND THE SIDE IN ORDER		+	90	
91	91	C	+	COUNTER-CLOCKWISE FASHION		+	91	
92	92	C	+			+	92	
93	93	C	+	JS(4,IS)	- INDEX OF THE FIRST EDGE	+	93	
94	94	C	+	JS(5,IS)	- INDEX OF THE SECOND EDGE	+	94	
95	95	C	+	JS(6,IS)	- INDEX OF THE THIRD EDGE	+	95	
96	96	C	+			+	96	
97	97	C	+	THE EDGES ARE ARRANGED IN COUNTER-CLOCKWISE		+	97	
98	98	C	+	FASHION. EDGE ONE RUNS FROM VERTEX-ONE TO		+	98	
99	99	C	+	VERTEX-TWO ETC.. THE SIGN OF JS(4-6,IS) INDICATES		+	99	
100	100	C	+	IF EDGE DATA IS STORED THE SAME WAY. IF IT IS		+	100	
101	101	C	+	JS>0 AND IT IS REVERSED JS<0		+	101	
102	102	C	+	JS(7,IS)	- INDEX OF CELL ON LEFT	+	102	
103	103	C	+	JS(8,IS)	- INDEX OF CELL ON RIGHT	+	103	
104	104	C	+			+	104	
105	105	C	+	XS(1,IS)	- X POSITION OF CENTROID OF TRIANGLE	+	105	
106	106	C	+	XS(2,IS)	- Y POSITION OF CENTROID OF TRIANGLE	+	106	
107	107	C	+	XS(3,IS)	- Z POSITION OF CENTROID OF TRIANGLE	+	107	
108	108	C	+	XS(4,IS)	- AREA OF TRIANGLE	+	108	
109	109	C	+	XS(5,IS)	- DISTANCE BETWEEN ADJOINING CELLS	+	109	
110	110	C	+		POINTS CROSSING TRIANGLE IS	+	110	
111	111	C	+			+	111	
112	112	C	+			+	112	
113	113	C	+	+++++				113
114	114	C	+	+++++				114
115	115	C	+	+++++				115
116	116	C	+	LIST OF CELLS		+	116	
117	117	C	+			+	117	
118	118	C	+			+	118	
119	119	C	+	IC	- CELL INDEX	+	119	
120	120	C	+	JC(1,IC)	- INDEX OF FIRST VERTEX	+	120	
121	121	C	+	JC(2,IC)	- INDEX OF SECOND VERTEX	+	121	
122	122	C	+	JC(3,IC)	- INDEX OF THIRD VERTEX	+	122	
123	123	C	+	JC(4,IC)	- INDEX OF FOURTH VERTEX	+	123	
124	124	C	+			+	124	
125	125	C	+	THE CONVENTION FOR VERTICES IS THAT 1-3		+	125	
126	126	C	+	ARE ARRANGED COUNTER-CLOCKWISE ABOUT THE		+	126	
127	127	C	+	BASE AND THAT 4 IS AT THE APEX.		+	127	
128	128	C	+			+	128	
129	129	C	+	JC(5,IC)	- INDEX OF FIRST SIDE	+	129	
130	130	C	+	JC(6,IC)	- INDEX OF SECOND SIDE	+	130	
131	131	C	+	JC(7,IC)	- INDEX OF THIRD SIDE	+	131	
132	132	C	+	JC(8,IC)	- INDEX OF FOURTH SIDE	+	132	
133	133	C	+			+	133	
134	134	C	+	THE CONVENTION FOR SIDES IS THAT SIDE ONE COVERS		+	134	
135	135	C	+	THE SPACE BETWEEN VERTEX-ONE, VERTEX-TWO, AND THE		+	135	
136	136	C	+	VERTEX AT THE APEX ETC.. SIDE FOUR IS THE BASE		+	136	
137	137	C	+			+	137	
138	138	C	+	XC(1,IC)	- X POSITION OF CELL POINT	+	138	
139	139	C	+	XC(2,IC)	- Y POSITION OF CELL POINT	+	139	
140	140	C	+	XC(3,IC)	- Z POSITION OF CELL POINT	+	140	
141	141	C	+	XC(4,IC)	- CELL VOLUME.	+	141	
142	142	C	+			+	142	
143	143	C	+			+	143	
144	144	C	+	+++++				144
145	145	C	+	+++++				145
146	146	C	+	+++++				146
147	147	C	+	+++++				147

```

148 148 C
149 149 C --- DEFINITION FOR ALL HYDRODYNAMIC QUANTITIES -----
150 150 C
151 151 C-----I
152 152 C I
153 153 C USE OF PARAMETERS: I
154 154 C I
155 155 C MHQ - MAXIMUM NUMBER OF HYDRO QUANTITIES. I
156 156 C I
157 157 C I
158 158 C-----I
159 159 C
160 160 C-----
161 161 C
162 162 C include 'dmsh00.h'
163 163 C include 'dhyd0.h'
164 164 C include 'dphsm0.h'
165 165 C include 'dntr10.h'
166 166 C
167 167 REAL XX(600),PP(600),HR(600),
168 168 UU(600),GG(600),AA(600),EE(600)
169 169 DOUBLE PRECISION VOL1,VOL2,VOL3,VOL4,VOLL,XXI,YYI,ZZI
170 170 DOUBLE PRECISION DEFVOL
171 171 OPEN(2,FILE='data.dd',FORM='FORMATTED')
172 172 OPEN(4,FILE='thermo.d',FORM='FORMATTED')
173 173 OPEN(8,FILE='threed2.5',FORM='UNFORMATTED')
174 174 OPEN(88,FILE='threed82',FORM='UNFORMATTED')
175 175 OPEN(9,FILE='threed3',FORM='UNFORMATTED')
176 176 OPEN(10,FILE='threed4',FORM='FORMATTED')
177 177 OPEN(15,FILE='AVSfmhail.inp',FORM='FORMATTED')
178 178 OPEN(14,FILE='AVSsmhail.inp',FORM='FORMATTED')
179 179 OPEN(16,FILE='OUTPUT.MSH',FORM='FORMATTED')
180 180 OPEN(26,FILE='EXPLSV.RND',FORM='FORMATTED')
181 181 OPEN(17,FILE='ve0640.stv',FORM='FORMATTED')
182 182 OPEN(18,FILE='f0640.stv',FORM='FORMATTED')
183 183 OPEN(19,FILE='pr640.stv',FORM='FORMATTED')
184 184 OPEN(11,FILE='truck.input.8b',STATUS='OLD')
185 185 C
186 186 C-----
187 187 C
188 188 C NAMELIST /DATA/ ICOND,ICONP,IOPTN,XMCHIN,RIN,PIN,ALFA,HRGG,IHRN,
189 189 C NTIME,NDUMP,IOPORD
190 190 C
191 191 C-----I
192 192 C I
193 193 C I
194 194 C --- MEANING OF NAMELIST VARIABLES: I
195 195 C I
196 196 C ICOND = 0 READ INPUT GRID FOR A NEW RUN I
197 197 C = 1 READ THE GRID FROM PREVIOUS RUN I
198 198 C ICONP = 0 PRIMITIVE VARIABLES SET TO ZERO I
199 199 C = 1 VARIABLES READ FROM PREVIOUS RUN I
200 200 C IOPTN = 1 SOLUTION FOR STEADY STATE, I
201 201 C = 2 SOLUTION FOR TRANSIENT PHENOMENA I
202 202 C I
203 203 C XMCHIN = FOR TRANSIENT SHOCK CALCULATIONS(IOPTN=2)THIS VARIABLE I
204 204 C IS USED TO SPECIFY THE UPSTREAM MACH NUMBER I
205 205 C I
206 206 C RIN = THE AMBIENT DENSITY IN THE CHAMBER I
207 207 C I
208 208 C PIN = THE AMBIENT PRESSURE IN THE CHAMBER I
209 209 C I
210 210 C APPLYING NORMAL SHOCK WAVES RELATIONS FOR AN ADIABATIC I
211 211 C FLOW RELATION STATIC-PRESSURE RATIO ACROSS THE SHOCK I
212 212 C AS WELL AS THE DENSITY RATIO AND MACH NUMBER RATIO I
213 213 C ARE COMPUTED TO SET CORRECTLY THE CONDITION AT THE I
214 214 C INLET EDGES( EDGE BOUNDARY 8 ) OF THE COMPUTATIONAL I
215 215 C DOMAIN I
216 216 C I
217 217 C FOR STEADY STATE SHOCK CALCULATIONS(IOPTN=1)THIS IS THE I
218 218 C INFLOW MACH NUMBER. ALL DOMAIN VELOCITIES ARE THEN I
219 219 C INITIALIZED WITH THIS VALUE. I
220 220 C I
221 221 C RIN = THE AMBIENT DENSITY AT INFINITY I

```

```

222 222 C
223 223 C          PIN = THE AMBIENT PRESSURE AT INFINITY
224 224 C
225 225 C          ALL COMPUTATIONAL DOMAIN ARE THEN INITIALIZED WITH
226 226 C          THOSE VALUES.
227 227 C
228 228 C  ALFA = THE DIRECTION OF INFLOW IN DEGREES RELATIVE TO A RIGHT
229 229 C          HAND COORDINATE SYSTEM. ALFA=0 MEANS FLOW FROM LEFT TO
230 230 C          RIGHT. ALFA=90 MEANS FROM BOTTOM TO TOP. ALFA=-90 OR 270
231 231 C          MEANS FLOW FROM TOP TO BOTTOM ETC.
232 232 C  HRGG = INITIAL GAMMA IN THE EQUATION OF STATE
233 233 C          THE CODE RUNS USING THE AIR EQUATION AS A BASELINE AND
234 234 C          SHOULD BE MODIFIED IF SOMETHING ELSE IS DESIRED.
235 235 C  IHRN = NUMBER OF ITERATIONS IN THE RIEMANN SOLVER TO FIND THE
236 236 C          DIAPHRAGM SOLUTION.(3 to 4 SHOULD BE USED AND INCREASED
237 237 C          ONLY FOR HIGH MACH NUMBER CASES).
238 238 C
239 239 C  NTIME = NUMBER OF REPEATS FOR THE INTEGRATION SEQUENCE.
240 240 C          AN OUTPUT DUMP IS DONE EVERY SEQUENCE REPEAT.
241 241 C  NDUMP = NUMBER OF ITERATIONS IN THE INNER LOOP
242 242 C
243 243 C          + _____ o NTIME - DUMPING DATA
244 244 C          |
245 245 C          |           + _____ o NDUMP - INTEGRATION
246 246 C          |           |
247 247 C          |           |           + _____ o ..... INNER LOOP
248 248 C          |           |           |
249 249 C          |           |           |           + _____ o ..... DUMPING LOOP
250 250 C          + _____
251 251 C
252 252 C  IOPORD = 1 THE CODE WILL RUN FIRST ORDER GODUNOV METHOD
253 253 C          = 2 THE CODE WILL RUN SECOND ORDER GODUNOV METHOD
254 254 C
-----
255 255 C
256 256 C
257 257 C  ICOND = 0
258 258 C  ICONP = 0
259 259 C  IOPTN = 1
260 260 C  IEOS = 1
261 261 C
262 262 C  XMCHIN = 2.5
263 263 C  RIN = 1.25
264 264 C  PIN = 101350.
265 265 C  RGAS = 8314.3
266 266 C  GPERCC = .001
267 267 C
268 268 C  ALFA = 0.
269 269 C  HRGG = 1.4
270 270 C  IHRN = 4
271 271 C  NTIME = 12
272 272 C  NDUMP = 200
273 273 C  IOPORD = 2
274 274 C
275 275 C --- READ THE INPUT DATA -----
276 276 C
277 277 C  READ (2,DATA)
278 278 C
279 279 C --- PRINTOUT THE RUN PARAMETERS -----
280 280 C
281 281 C  PRINT 101,  ICOND,ICONP,IOPTN,XMCHIN,RIN,PIN,ALFA,HRGG,IHRN,
282 282 C          NTIME,NDUMP,IOPORD
283 283 C
284 284 C --- SET RUN CONDITIONS AND PRINTOUT TO CONSOLE -----
285 285 C
286 286 C  READING GRID DATA FROM EDGE.ZON
287 287 C
288 288 C  THIRD = 1. / 3.
289 289 C  IF( ICOND . EQ . 0 ) THEN
290 290 C
291 291 C  CALL UPDATE
292 292 C
293 293 C  ELSE
294 294 C
295 295 C  CALL UPGRAD

```

```

296 296 C
297 297     END IF
298 298     CALL MATRLA
299 299     CALL MATRLX
300 300 C
301 301 C --- INITIALIZATION OF THE PROBLEM
302 302 C
303 303     HRSM = 1.E-8
304 304     HRGP = HRGG + 1.
305 305     HRGM = HRGG - 1.
306 306     CF = HRGP / ( 2. * HRGG )
307 307     TT = 0.
308 308 C
309 309     PIRAD = ATAN( 1. ) / 45.
310 310     ALPHA = ALFA * PIRAD
311 311     PRINT *,ALFA,PIRAD,ALPHA
312 312 C
313 313     COSS = COS( ALPHA )
314 314     SINN = SIN( ALPHA )
315 315     TANN = TAN( ALPHA )
316 316 C
317 317 C-----
318 318 C
319 319 C --- SET THE INITIAL VALUE FOR PRIMITIVE VARIABLES -----
320 320 C
321 321 C(2)>>>>
322 322     TLIMIT = .9
323 323     ITER = 6
324 324     IF( ICOND .EQ. 0 ) THEN
325 325         UVIN = XMCHIN * SORT( HRGG * PIN / RIN )
326 326         UIN = UVIN * COSS
327 327         VIN = UVIN * SINN
328 328         WIN = 0.
329 329 C
330 330     DO 150 IC = 1 , NC
331 331         HYDV( IC , 1 ) = RIN
332 332         HYDV( IC , 2 ) = 0.
333 333         HYDV( IC , 3 ) = 0.
334 334         HYDV( IC , 4 ) = 0.
335 335         HYDV( IC , 5 ) = PIN
336 336         HYDV( IC , 6 ) = 1.E-6
337 337         HYDV( IC , 7 ) = 1.4
338 338         HYDV( IC , 8 ) = PIN / ( HYDV( IC , 7 ) - 1. )
339 339 C
340 340 150 CONTINUE
341 341     RADIUS = .0001
342 342     EXPLSV = 8.
343 343     DO IC = 1 , NC
344 344         XXI = XC( 1 , IC )
345 345         YYI = XC( 2 , IC )
346 346         ZZI = XC( 3 , IC )
347 347         RSS = SORT( XXI * XXI + YYI * YYI + ZZI * ZZI )
348 348         IF( RSS .LT. RADIUS ) THEN
349 349             print*,xxi,yyi,zzi,radius
350 350             HYDV( IC , 1 ) = EXPLSV * .4536 * .75 / 3.141569 /
351 351                 ( RADIUS * RADIUS * RADIUS )
352 352             HYDV( IC , 6 ) = 1.
353 353             HYDV( IC , 8 ) = HYDV( IC , 1 ) * 1080. * 4.185 *
354 354                 1000. * 1.01 / .7
355 355         NITER = 0
356 356         DST = HYDV( IC , 1 ) * GPERCC
357 357         VOL = WMX * ( 1. - DST / FSX ) / DST / XGX
358 358         EME0 = HYDV( IC , 8 ) / HYDV( IC , 1 ) * WMX / RGAS
359 359 C
360 360         IYY = ( EME0 - EMEOX( 3 ) ) / RANGEX + 1
361 361         IYY = MAXO( 1 , MINO( IYY , 47 ) )
362 362 C
363 363         K = IYY + 2
364 364         IYY = IYY
365 365         + INT( AMAX1( EME0 - EMEOX( K ) , 0. ) / DYX( K ) )
366 366         - INT( AMAX1( EMEOX( K + 1 ) - EME0 , 0. ) / DYX( K ) )
367 367         IYY = MAXO( 1 , MINO( IYY , 47 ) )
368 368 C
369 369         KI = IYY + 2

```

```

370 370      K2 = K1 + 1
371 371      RT = ( EMEOX - EMEOX( K1 ) ) / ( EMEOX( K2 ) - EMEOX( K1 ) )
372 372      T = TX( K1 ) + 100. * RT
373 373      CVM = CVMX( K1 ) + RT * ( CVMX( K2 ) - CVMX( K1 ) )
374 374      ERS = 0.
375 375      C
376 376 10    CONTINUE
377 377      P = RGAS * T / VOL / GPERCC
378 378      RGAMM1 = CVM
379 379      C
380 380      X = COVX / VOL / ( ( T + THETAX ) ** ALFAX )
381 381      Z = X * EXP( BETAX * X )
382 382      X = 1. + BETAX * X
383 383      RT = ALFAX * T / ( T + THETAX )
384 384      ERS = ERS + RT * Z * T
385 385      C
386 386      IF ( ITER .EQ. NITER ) GO TO 20
387 387      C
388 388      CVM = CVM * XGX + SCVX
389 389      *      + RT * Z * ( 2. - RT / ALFAX - RT * X )
390 390      T = T - AMIN1( ERS / CVM , TLIMIT * T )
391 391      C
392 392      NITER = NITER + 1
393 393      C
394 394      RT = 0.01 * T
395 395      K1 = RT
396 396      K1 = MINO ( K1, 49 )
397 397      K1 = MAXO ( K1, 3 )
398 398      K2 = K1 + 1
399 399      RT = RT - K1
400 400      CVM = CVMX(K1) + RT * ( CVMX( K2 ) - CVMX( K1 ) )
401 401      ERS = EMEOX( K1 ) + RT * ( EMEOX( K2 ) - EMEOX( K1 ) )
402 402      ERS = ERS - EMEO
403 403      C
404 404      GO TO 10
405 405      C
406 406 20    CONTINUE
407 407      P = P * ( 1. + Z )
408 408      RGAMM1 = ( RGAMM1 +
409 409      *      RT * Z * ( 2. - RT / ALFAX - RT * X ) ) / ( 1. + Z )
410 410      X = X * Z / ( 1. + Z )
411 411      RGAMM1 = RGAMM1 / ( ( 1. - RT * X ) ** 2 + X * RGAMM1 )
412 412      ERS = ERS / EMEO
413 413      HYDV( IC , 7 ) = 1. / RGAMM1 + 1.
414 414      HYDV( IC , 5 ) = P
415 415      END IF
416 416      END DO
417 417      C
418 418      XCOUNT = 0
419 419      DO IC = 1 , NC
420 420      RCOUNT = HYDV( IC , 8 ) + .5 * HYDV( IC , 1 ) *
421 421      .      ( HYDV( IC , 2 ) * HYDV( IC , 2 ) +
422 422      .      HYDV( IC , 3 ) * HYDV( IC , 3 ) +
423 423      .      HYDV( IC , 4 ) * HYDV( IC , 4 ) )
424 424      XCOUNT = XCOUNT + XC( 4 , IC ) * RCOUNT
425 425      END DO
426 426      PRINT * , XCOUNT
427 427      IIJJ=1
428 428      IF(IIJJ.EQ.0) GO TO 1122
429 429      C      remove the followed IF statement for regular run
430 430      C      IF( IOPTN .EQ. 2 ) THEN
431 431      C      IF( IOPTN .EQ. 1 ) THEN
432 432      C
433 433      NX = 360
434 434      DO 190 IX = 1 , NX
435 435      XX(IX) = (IX-.5)*.002
436 436 190    CONTINUE
437 437      READ (11,1001) (PP(IX),IX=1,NX)
438 438      READ (11,1001) (UU(IX),IX=1,NX)
439 439      READ (11,1001) (HR(IX),IX=1,NX)
440 440      READ (11,1001) (AA(IX),IX=1,NX)
441 441      READ (11,1001) (GG(IX),IX=1,NX)
442 442      READ (11,1001) (EE(IX),IX=1,NX)
443 443 1001  FORMAT(6E12.5)

```

```

444 444 C
445 445 ICOUNT = 0
446 446 DO 260 IC = 1 , NC
447 447 C XXI = XC( 1 , IC ) + .2667
448 448 XXI = XC( 1 , IC ) + .1143
449 449 C YYI = XC( 2 , IC ) - 1.96596
450 450 C ZZI = XC( 3 , IC ) - 1.25
451 451 ZZI = XC( 3 , IC ) - 1.905
452 452 RSS = SQRT( XXI * XXI + YYI * YYI + ZZI * ZZI )
453 453 C YYS = SQRT( XXI * XXI + YYI * YYI )
454 454 DO 270 IX = 1 , NX-1
455 455 XDD1 = XX( IX )
456 456 XDD2 = XX( IX+1 )
457 457 IF( RSS . GT . XDD1 . AND . RSS . LT . XDD2 ) THEN
458 458 XKSI = ( RSS - XDD1 ) / ( XDD2 - XDD1 )
459 459 ICOUNT = ICOUNT + 1
460 460 C
461 461 HYDV(IC,1) = HR(IX) * (1.-XKSI) +
462 462 HR(IX+1) * XKSI
463 463 C
464 464 HYDUVW = UU(IX) * (1.-XKSI) +
465 465 UU(IX+1) * XKSI
466 466 HYDV(IC,4) = ZZI / RSS * HYDUVW
467 467 HYDVUV = YYS / RSS * HYDUVW
468 468 C
469 469 THETA = ATAN2( YYI , XXI )
470 470 HYDV(IC,2) = HYDVUV * COS( THETA )
471 471 HYDV(IC,3) = HYDVUV * SIN( THETA )
472 472 C
473 473 HYDV(IC,5) = PP(IX) * (1.-XKSI) +
474 474 PP(IX+1) * XKSI
475 475 C
476 476 HYDV(IC,5) = 1.08*HYDV(IC,5)
477 477 HYDV(IC,7) = GG(IX) * (1.-XKSI) +
478 478 GG(IX+1) * XKSI
479 479 HYDV(IC,6) = AA(IX) * (1.-XKSI) +
480 480 AA(IX+1) * XKSI
481 481 HYDV(IC,8) = EE(IX) * (1.-XKSI) +
482 482 EE(IX+1) * XKSI
483 483 C
484 484 GOTO 301
485 485 ENDF
486 486 270 CONTINUE
487 487 301 CONTINUE
488 488 NITER = 6
489 489 IF(NITER.EQ.0) THEN
490 490 IF( HYDV( IC , 6 ) . LT . .2 ) THEN
491 491 DST = HYDV( IC , 1 ) * GPERCC
492 492 VOL = WMA * ( 1. - DST / FSA ) / DST / XGA
493 493 TT = HYDV( IC , 5 ) * VOL * GPERCC / RGAS
494 494 C
495 495 T = TT
496 496 RT = 0.01 * T
497 497 K1 = RT
498 498 K1 = MINO ( K1 , 49 )
499 499 K1 = MAXO ( K1 , 3 )
500 500 K2 = K1 + 1
501 501 RT = RT - K1
502 502 ENERGY = EMOA( K1 ) + RT * ( EMOA( K2 ) - EMOA( K1 ) )
503 503 ENERGY = ENERGY * RGAS / WMA
504 504 C
505 505 DO ITER = 1 , NITER
506 506 X = COVA / VOL / ( T + THETA ) ** ALFAA
507 507 C
508 508 BETAZX = BETAA * X
509 509 RT = X * EXP( BETAZX )
510 510 RTINV = 1. / ( 1. + RT )
511 511 C - ERS IS THE FUNCTION, RT IS THE DERIVATIVE
512 512 ERS = T - TT * RTINV
513 513 RT = 1. - TT * RTINV * RTINV * ALFAA * RT * ( 1. + BETAZX ) /
514 514 ( T + THETA )
515 515 ERS = ERS / RT
516 516 T = T - ERS
517 517 END DO

```

```

518 518 C
519 519 RT = 0.01 * T
520 520 K1 = RT
521 521 K1 = MINO ( K1, 49 )
522 522 K1 = MAXO ( K1, 3 )
523 523 K2 = K1 + 1
524 524 RT = RT - K1
525 525 ENERGY = EMOEA( K1 ) + RT * ( EMOEA( K2 ) - EMOEA( K1 ) )
526 526 C
527 527 X = COVA / VOL / ( ( T + THETAA ) ** ALFAA )
528 528 EX = EXP( BETAA * X )
529 529 Z = X * EX
530 530 RT = ALFAA * T / ( T + THETAA )
531 531 ENERGY = ENERGY + RT * Z * T
532 532 HYDV( IC , 8 ) = ENERGY * RGAS / WMA
533 533 EME0 = HYDV( IC , 8 ) / HYDV( IC , 1 ) * WMA / RGAS
534 534 C
535 535 IYY = ( EME0 - EMOEA( 3 ) ) / RANGEA + 1
536 536 IYY = MAXO( 1, MINO( IYY, 47 ) )
537 537 C
538 538 K = IYY + 2
539 539 IYY = IYY
540 540 . + INT( AMAX1( EME0 - EMOEA( K ), 0. ) / DYA( K ) )
541 541 . - INT( AMAX1( EMOEA( K + 1 ) - EME0, 0. ) / DYA( K ) )
542 542 IYY = MAXO( 1, MINO( IYY, 47 ) )
543 543 C
544 544 K1 = IYY + 2
545 545 K2 = K1 + 1
546 546 RT = ( EME0 - EMOEA( K1 ) ) / ( EMOEA( K2 ) - EMOEA( K1 ) )
547 547 T = TA( K1 ) + 100. * RT
548 548 CVM = CVMA( K1 ) + RT * ( CVMA( K2 ) - CVMA( K1 ) )
549 549 ERS = 0.
550 550 C
551 551 P = RGAS * T / VOL / GPERCC
552 552 RGAMM1 = CVM
553 553 HYDV( IC , 7 ) = 1. / RGAMM1 + 1.
554 554 HYDV( IC , 5 ) = P
555 555 C
556 556 ELSE
557 557 C
558 558 DST = HYDV( IC , 1 ) * GPERCC
559 559 VOL = WMX * ( 1. - DST / FSX ) / DST / XGX
560 560 TT = HYDV( IC , 5 ) * VOL * GPERCC / RGAS
561 561 C
562 562 T = TT
563 563 RT = 0.01 * T
564 564 K1 = RT
565 565 K1 = MINO ( K1, 49 )
566 566 K1 = MAXO ( K1, 3 )
567 567 K2 = K1 + 1
568 568 RT = RT - K1
569 569 ENERGY = EMEOX( K1 ) + RT * ( EMEOX( K2 ) - EMEOX( K1 ) )
570 570 ENERGY = ENERGY * RGAS / WMX
571 571 C
572 572 DO ITER = 1, NITER
573 573 X = COVX / VOL / ( T + THETAX ) ** ALFAX
574 574 C
575 575 BETAZX = BETAX * X
576 576 RT = X * EXP( BETAZX )
577 577 RTINV = 1. / ( 1. + RT )
578 578 C - ERS IS THE FUNCTION, RT IS THE DERIVATIVE
579 579 ERS = T - TT * RTINV
580 580 RT = 1. - TT * RTINV * RTINV * ALFAX * RT * ( 1. + BETAZX ) /
581 581 ( T + THETAX )
582 582 ERS = ERS / RT
583 583 T = T - ERS
584 584 END DO
585 585 C
586 586 RT = 0.01 * T
587 587 K1 = RT
588 588 K1 = MINO ( K1, 49 )
589 589 K1 = MAXO ( K1, 3 )
590 590 K2 = K1 + 1
591 591 RT = RT - K1

```

```

592 592 ENERGY = EMEOX( K1 ) + RT * ( EMEOX( K2 ) - EMEOX( K1 ) ) 592
593 593 C 593
594 594 X = COVX / VOL / ( ( T + THETAX ) ** ALFAX ) 594
595 595 EX = EXP( BETAX * X ) 595
596 596 Z = X * EX 596
597 597 RT = ALFAX * T / ( T + THETAX ) 597
598 598 ENERGY = ENERGY + RT * Z * T 598
599 599 HYDV( IC , 8 ) = ENERGY * RGAS / WMX 599
600 600 VOL = WMX * ( 1. - DST / FSX ) / DST / XGX 600
601 601 EMEO = HYDV( IC , 8 ) / HYDV( IC , 1 ) * WMX / RGAS 601
602 602 C 602
603 603 IYY = ( EMEO - EMEOX( 3 ) ) / RANGEX + 1 603
604 604 IYY = MAXO( 1 , MINO( IYY , 47 ) ) 604
605 605 C 605
606 606 K = IYY + 2 606
607 607 IYY = IYY 607
608 608 . + INT( AMAX1( EMEO - EMEOX( K ) , 0. ) / DYX( K ) ) 608
609 609 . - INT( AMAX1( EMEOX( K + 1 ) - EMEO , 0. ) / DYX( K ) ) 609
610 610 IYY = MAXO( 1 , MINO( IYY , 47 ) ) 610
611 611 C 611
612 612 K1 = IYY + 2 612
613 613 K2 = K1 + 1 613
614 614 RT = ( EMEO - EMEOX( K1 ) ) / ( EMEOX( K2 ) - EMEOX( K1 ) ) 614
615 615 T = TX( K1 ) + 100. * RT 615
616 616 CVM = CVMX( K1 ) + RT * ( CVMX( K2 ) - CVMX( K1 ) ) 616
617 617 ERS = 0. 617
618 618 C 618
619 619 401 CONTINUE 619
620 620 P = RGAS * T / VOL / GPERCC 620
621 621 RGAMM1 = CVM 621
622 622 C 622
623 623 X = COVX / VOL / ( ( T + THETAX ) ** ALFAX ) 623
624 624 Z = X * EXP( BETAX * X ) 624
625 625 X = 1. + BETAX * X 625
626 626 RT = ALFAX * T / ( T + THETAX ) 626
627 627 ERS = ERS + RT * Z * T 627
628 628 C 628
629 629 IF ( ITER .EQ. NITER ) GO TO 201 629
630 630 C 630
631 631 CVM = CVM * XGX + SCVX 631
632 632 * + RT * Z * ( 2. - RT / ALFAX - RT * X ) 632
633 633 T = T - AMIN1( ERS / CVM , TLIMIT * T ) 633
634 634 C 634
635 635 NITER = NITER + 1 635
636 636 C 636
637 637 RT = 0.01 * T 637
638 638 K1 = RT 638
639 639 K1 = MINO( K1 , 49 ) 639
640 640 K1 = MAXO( K1 , 3 ) 640
641 641 K2 = K1 + 1 641
642 642 RT = RT - K1 642
643 643 CVM = CVMX(K1) + RT * ( CVMX( K2 ) - CVMX( K1 ) ) 643
644 644 ERS = EMEOX( K1 ) + RT * ( EMEOX( K2 ) - EMEOX( K1 ) ) 644
645 645 ERS = ERS - EMEO 645
646 646 C 646
647 647 GO TO 401 647
648 648 C 648
649 649 201 CONTINUE 649
650 650 P = P * ( 1. + Z ) 650
651 651 RGAMM1 = ( RGAMM1 + 651
652 652 * RT * Z * ( 2. - RT / ALFAX - RT * X ) ) / ( 1. + Z ) 652
653 653 X = X * Z / ( 1. + Z ) 653
654 654 RGAMM1 = RGAMM1 / ( ( 1. - RT * X ) ** 2 + X * RGAMM1 ) 654
655 655 ERS = ERS / EMEO 655
656 656 HYDV( IC , 7 ) = 1. / RGAMM1 + 1. 656
657 657 HYDV( IC , 5 ) = P 657
658 658 END IF 658
659 659 END IF 659
660 660 260 CONTINUE 660
661 661 C 661
662 662 C(2)---- 662
663 663 ELSE 663
664 664 C 664
665 665 XMSQR = XMCHIN * XMCHIN 665

```



```

666 666 PINL = PIN 666
667 667 RINL = RIN 667
668 668 RINRTO = ( HRGG + 1. ) * XMSQR / 668
669 669 ( ( HRGG - 1. ) * XMSQR + 2. ) 669
670 670 PINRTO = ( 2. * HRGG * XMSQR - ( HRGG - 1. ) ) / 670
671 671 ( HRGG + 1. ) 671
672 672 PIN = PINRTO * PINL 672
673 673 RIN = RINRTO * RINL 673
674 674 YMCHIN = SQRT( ( ( HRGG - 1. ) * XMSQR + 2. ) / 674
675 675 ( 2. * HRGG * XMSQR - ( HRGG - 1. ) ) ) 675
676 676 PRINT*,HRGG,RIN,PIN,YMCHIN 676
677 677 PRINT*,HRGG,RINL,PINL,XMCHIN 677
678 678 UVIN = XMCHIN * SQRT( HRGG * PINL / RINL ) - 678
679 679 YMCHIN * SQRT( HRGG * PIN / RIN ) 679
680 680 UIN = UVIN * COSS 680
681 681 VIN = UVIN * SINN 681
682 682 WIN = 0. 682
683 683 C 683
684 684 DO 155 IC = 1 , NC 684
685 685 HYDV( IC , 1 ) = RINL 685
686 686 HYDV( IC , 2 ) = UIN 686
687 687 HYDV( IC , 3 ) = VIN 687
688 688 HYDV( IC , 4 ) = WIN 688
689 689 HYDV( IC , 5 ) = PINL 689
690 690 C 690
691 691 155 CONTINUE 691
692 692 ENDIF 692
693 693 C remove the followed END IF for regular run 693
694 694 C ENDIF 694
695 695 C ENDIF 695
696 696 C(2)<<<< 696
697 697 C 697
698 698 1122 CONTINUE 698
699 699 IF( ICOND . EQ . 0 ) THEN 699
700 700 NPRTCL = 25 700
701 701 XPRTCL(1,1) = .443 701
702 702 XPRTCL(2,1) = 1.0414 702
703 703 XPRTCL(3,1) = 1.4224 703
704 704 XPRTCL(1,2) = -.002 704
705 705 XPRTCL(2,2) = .3556 705
706 706 XPRTCL(3,2) = 0.5842 706
707 707 XPRTCL(1,3) = -.275 707
708 708 XPRTCL(2,3) = -.3058 708
709 709 XPRTCL(3,3) = -1.4224 709
710 710 XPRTCL(1,4) = 2.032 710
711 711 XPRTCL(2,4) = -.3048 711
712 712 XPRTCL(3,4) = -4.572 712
713 713 XPRTCL(1,5) = .3048 713
714 714 XPRTCL(2,5) = .1016 714
715 715 XPRTCL(3,5) = .3048 715
716 716 XPRTCL(1,6) = .4572 716
717 717 XPRTCL(2,6) = .1016 717
718 718 XPRTCL(3,6) = .4572 718
719 719 XPRTCL(1,7) = .6096 719
720 720 XPRTCL(2,7) = .1016 720
721 721 XPRTCL(3,7) = .3048 721
722 722 XPRTCL(1,8) = .4572 722
723 723 XPRTCL(2,8) = .1016 723
724 724 XPRTCL(3,8) = .1524 724
725 725 XPRTCL(1,9) = 1.3462 725
726 726 XPRTCL(2,9) = .1016 726
727 727 XPRTCL(3,9) = .3048 727
728 728 XPRTCL(1,10) = 1.4986 728
729 729 XPRTCL(2,10) = .1016 729
730 730 XPRTCL(3,10) = .4572 730
731 731 XPRTCL(1,11) = 1.651 731
732 732 XPRTCL(2,11) = .1016 732
733 733 XPRTCL(3,11) = .3048 733
734 734 XPRTCL(1,12) = 1.4986 734
735 735 XPRTCL(2,12) = .1016 735
736 736 XPRTCL(3,12) = .1524 736
737 737 XPRTCL(1,13) = .6096 737
738 738 XPRTCL(2,13) = .7740 738
739 739 XPRTCL(3,13) = 1.0668 739

```

```

740 740 XPRTCL(1,14) = .6096 740
741 741 XPRTCL(2,14) = .8138 741
742 742 XPRTCL(3,14) = .5334 742
743 743 XPRTCL(1,15) = 1.4224 743
744 744 XPRTCL(2,15) = .7740 744
745 745 XPRTCL(3,15) = 1.0668 745
746 746 XPRTCL(1,16) = 1.4224 746
747 747 XPRTCL(2,16) = .8128 747
748 748 XPRTCL(3,16) = .5334 748
749 749 XPRTCL(1,17) = -.3058 749
750 750 XPRTCL(2,17) = 1.3208 750
751 751 XPRTCL(3,17) = -.4318 751
752 752 XPRTCL(1,18) = .2032 752
753 753 XPRTCL(2,18) = .7590 753
754 754 XPRTCL(3,18) = 1.1898 754
755 755 XPRTCL(1,19) = .254 755
756 756 XPRTCL(2,19) = .1772 756
757 757 XPRTCL(3,19) = 1.1948 757
758 758 XPRTCL(1,20) = .9144 758
759 759 XPRTCL(2,20) = .4064 759
760 760 XPRTCL(3,20) = .9652 760
761 761 XPRTCL(1,21) = .2032 761
762 762 XPRTCL(2,21) = .7680 762
763 763 XPRTCL(3,21) = 1.1888 763
764 764 XPRTCL(1,22) = .1532 764
765 765 XPRTCL(2,22) = .7670 765
766 766 XPRTCL(3,22) = 1.1888 766
767 767 XPRTCL(1,23) = .1532 767
768 768 XPRTCL(2,23) = .7665 768
769 769 XPRTCL(3,23) = 1.1878 769
770 770 XPRTCL(1,24) = .1532 770
771 771 XPRTCL(2,24) = .7765 771
772 772 XPRTCL(3,24) = 1.1898 772
773 773 XPRTCL(1,25) = .1532 773
774 774 XPRTCL(2,25) = .7655 774
775 775 XPRTCL(3,25) = 1.1898 775
776 776 DO IK = 1 , NPRTCL 776
777 777 RMINH = 100000000. 777
778 778 DO IC = 1 , NC 778
779 779 I1=JC(1,IC) 779
780 780 I2=JC(2,IC) 780
781 781 I3=JC(3,IC) 781
782 782 I4=JC(4,IC) 782
783 783 XXI = XPRTCL( 1 , IK ) 783
784 784 YYI = XPRTCL( 2 , IK ) 784
785 785 ZZI = XPRTCL( 3 , IK ) 785
786 786 CALL VOLMTETC ( I1, I2, I3, XXI, YYI, ZZI , VOL1 ) 786
787 787 CALL VOLMTETC ( I1, I2, I4, XXI, YYI, ZZI , VOL2 ) 787
788 788 CALL VOLMTETC ( I1, I3, I4, XXI, YYI, ZZI , VOL3 ) 788
789 789 CALL VOLMTETC ( I2, I3, I4, XXI, YYI, ZZI , VOL4 ) 789
790 790 XXI = XV( 1 , I4 ) 790
791 791 YYI = XV( 2 , I4 ) 791
792 792 ZZI = XV( 3 , I4 ) 792
793 793 CALL VOLMTETC ( I1, I2, I3, XXI, YYI, ZZI , VOLL ) 793
794 794 DEFVOL=DABS(VOL1)+DABS(VOL2)+DABS(VOL3)+DABS(VOL4) 794
795 795 -DABS(VOLL) 795
796 796 IF( DABS(DEFVOL/VOLL) . LT . .001 ) THEN 796
797 797 IJKPRT(IK) = IC 797
798 798 PRINT*,ik,vol1,vol2,vol3,vol4 798
799 799 PRINT*,ic,voll,defvol,defvol/voll 799
800 800 PRINT*,(XV(kk,jc(1,ic)),kk=1,3) 800
801 801 PRINT*,(XV(kk,jc(2,ic)),kk=1,3) 801
802 802 PRINT*,(XV(kk,jc(3,ic)),kk=1,3) 802
803 803 PRINT*,(XV(kk,jc(4,ic)),kk=1,3) 803
804 804 PRINT*,(JS(9,jc(kk,ic)),kk=5,8) 804
805 805 END IF 805
806 806 END DO 806
807 807 END DO 807
808 808 DO IK = 1 , NPRTCL 808
809 809 IC = IJKPRT(IK) 809
810 810 ISS = JC(5,IC) 810
811 811 DO IKK = 5 , 8 811
812 812 IS = JC(IKK,IC) 812
813 813 IBC = JS(9,IS) 813

```

```

814 814         IF( IBC . EQ . 6 ) THEN      814
815 815             ISS = IS                 815
816 816             END IF                   816
817 817         END DO                       817
818 818             IJKPRT(IK) = ISS          818
819 819         END DO                       819
820 820         END IF                       820
821 821     C                                821
822 822         PRINT * , ICOND, ICONP        822
823 823     C                                823
824 824         IF( ICONP . EQ . 1 ) THEN    824
825 825             READ(8) RIN,PIN,RINL,PINL,UVIN,UIN,VIN,WIN,TT 825
826 826             PRINT * , RIN,PIN,RINL,PINL,UVIN,UIN,VIN,WIN,TT 826
827 827     C             READ (8) NPRTCL    827
828 828     C             IF(NPRTCL.GT.0)    828
829 829     C             . READ (8) (IJKPRT(IK),IK=1,NPRTCL) 829
830 830             DO II = 1 , 5            830
831 831             READ(8) ((HYDV(IC,IK),IK=1,8),IC=1,NC) 831
832 832             END DO                   832
833 833     C                                833
834 834         END IF                       834
835 835     C                                835
836 836         ZCOUNT = 0                   836
837 837         DO 380 IC = 1 , NC            837
838 838             RCOUNT = HYDV( IC , 8 ) + .5 * HYDV( IC , 1 ) * 838
839 839             .             ( HYDV( IC , 2 ) * HYDV( IC , 2 ) + 839
840 840             .             HYDV( IC , 3 ) * HYDV( IC , 3 ) + 840
841 841             .             HYDV( IC , 4 ) * HYDV( IC , 4 ) ) 841
842 842             ZCOUNT = ZCOUNT + XC( 4 , IC ) * RCOUNT 842
843 843     380 CONTINUE                     843
844 844             YCOUNT = ZCOUNT - XCOUNT 844
845 845             PRINT * , ZCOUNT, YCOUNT 845
846 846             CALL HYDRMN              846
847 847     C                                847
848 848     C --- EXIT POINT FROM PROGRAM ----- 848
849 849     C                                849
850 850     C                                850
851 851             STOP 777                  851
852 852     C                                852
853 853     C                                853
854 854     C --- FORMATS -----            854
855 855     C                                855
856 856     101 FORMAT(1H , 'ICOND=',12,5X,'ICONP=',12,5X,'IOPTN=',12,/,1X, 856
857 857     .             'XMCHIN=',F13.6,5X,'RIN=',F13.6,5X,'PIN=',F13.6,/,1X, 857
858 858     .             'ALFA=',F13.6,5X,'HRGG=',F13.6,5X,'IHRN=',12,5X,/,1X, 858
859 859     .             'NTIME=',12,5X,'NOUMP=',15,5X,'IOPORD=',12) 859
860 860                                         860
861 861         END                           861
862 862     C                                862

```

```

863     1      SUBROUTINE HYDRFL                                863
864     2      C                                                864
865     3      C-----[                                         865
866     4      C                                                866
867     5      C      HYDRFL IS A 2 DIMENSIONAL RIEMANN SOLVER THAT INTEGRATES I 867
868     6      C      FLUXES ACROSS NORMAL INTERFACES TO UPDATE VERTICES I 868
869     7      C      VARIABLES .                                     I 869
870     8      C                                                I 870
871     9      C-----[                                         871
872    10      C                                                872
873    11      C      include 'dmsh00.h'                            873
874    12      C      include 'dhyd0.h'                            874
875    13      C      include 'dphsm0.h'                          875
876    14      C      include 'dmtri0.h'                            876
877    15      C                                                877
878    16      REAL DELP(128),WSOP(128),WSOM(128),WSOO(128),      878
879    17      .      RSTAR(128),CSTAR(128),PMAX(128),PMIN(128) 879
880    18      REAL RRIGHT(128),URIGHT(128),VRIGHT(128),PRIGHT(128) 880
881    19      REAL RLEFTT(128),ULEFTT(128),VLEFTT(128),PLEFTT(128) 881
882    20      REAL ENRGYI(128),ANRGYI(128)                        882
883    21      C                                                883
884    22      C --- BEGIN LOOP OVER ALL EDGES IN THE DOMAIN ----- 884
885    23      C                                                885
886    24      DO 280 IH = 1 , 6                                    886
887    25      DO 280 IC = 1 , NC                                  887
888    26      HYDFLX( IC , IH ) = 0.                             888
889    27      280 CONTINUE                                       889
890    28      C                                                890
891    29      NS1 = 1                                              891
892    30      NS2 = NOFVES( 1 )                                    892
893    31      DO 110 INS = 1 , NVEES                              893
894    32      C                                                894
895    33      C --- FETCH HYDRO QUANTITIES -----                895
896    34      C                                                896
897    35      DO 120 IS = NS1 , NS2                                897
898    36      KS = IS - NS1 + 1                                  898
899    37      C                                                899
900    38      RRR( KS ) = RR( IS )                                900
901    39      UUR( KS ) = UR( IS )                                901
902    40      VVR( KS ) = VR( IS )                                902
903    41      MWR( KS ) = WR( IS )                                903
904    42      PPR( KS ) = PR( IS )                                904
905    43      AAR( KS ) = AR( IS )                                905
906    44      EER( KS ) = ER( IS )                                906
907    45      GGR( KS ) = GR( IS )                                907
908    46      C                                                908
909    47      RRL( KS ) = RL( IS )                                909
910    48      UUL( KS ) = UL( IS )                                910
911    49      VVL( KS ) = VL( IS )                                911
912    50      WWL( KS ) = WL( IS )                                912
913    51      PPL( KS ) = PL( IS )                                913
914    52      AAL( KS ) = AL( IS )                                914
915    53      EEL( KS ) = EL( IS )                                915
916    54      GGL( KS ) = GL( IS )                                916
917    55      C                                                917
918    56      120 CONTINUE                                       918
919    57      C                                                919
920    58      DO 130 KS = 1 , NOFVES( INS )                       920
921    59      C                                                921
922    60      C --- THIS SECTION OF CODE SOLVES FOR "PSTAR" AND "USTAR" IN 922
923    61      C      THE RIEMANN PROBLEM USING NEWTON'S METHOD.    923
924    62      C                                                924
925    63      WLEFT( KS ) = SQRT( GGL( KS ) * PPL( KS ) * RRL( KS ) ) 925
926    64      WRIGT( KS ) = SQRT( GGR( KS ) * PPR( KS ) * RRR( KS ) ) 926
927    65      WLESQ( KS ) = WLEFT( KS ) * WLEFT( KS )            927
928    66      WRISQ( KS ) = WRIGT( KS ) * WRIGT( KS )            928
929    67      C                                                929
930    68      PMIN( KS ) = AMIN1( PPL( KS ) , PPR( KS ) )        930
931    69      PSML( KS ) = HRSM * PMIN( KS )                      931
932    70      C                                                932
933    71      C --- FORM THE STARTING GUESS FOR THE SOLUTION ----- 933
934    72      C                                                934
935    73      PSTAR( KS ) = ( WLEFT( KS ) * PPR( KS ) +          935
936    74      .      WRIGT( KS ) * PPL( KS ) -                    936

```

```

937 75 . WLEFT( KS ) * WRIGT( KS ) * 937
938 76 . ( UUR( KS ) - UUL( KS ) ) / 938
939 77 . ( WLEFT( KS ) + WRIGT( KS ) ) 939
940 78 PSTAR( KS ) = AMAX1( PSTAR( KS ) , PSML( KS ) ) 940
941 79 130 CONTINUE 941
942 80 C 942
943 81 DO 140 I = 1 , IHRN 943
944 82 C 944
945 83 C --- BEGIN THE NEWTON ITERATION ----- 945
946 84 C 946
947 85 DO 150 KS = 1 , NOFVES( INS ) 947
948 86 CFFL = ( GGL( KS ) + 1. ) / ( 2. * GGL( KS ) ) 948
949 87 WLEFS( KS ) = ( 1. + CFFL * ( PSTAR( KS ) / 949
950 88 PPL( KS ) - 1. ) ) * WLESQ( KS ) 950
951 89 WLEFT( KS ) = SQRT( WLEFS( KS ) ) 951
952 90 ZLEFT( KS ) = 2. * WLEFT( KS ) * WLEFS( KS ) / 952
953 91 ( WLESQ( KS ) + WLEFS( KS ) ) 953
954 92 USTL( KS ) = UUL( KS ) - 954
955 93 ( PSTAR( KS ) - PPL( KS ) ) / WLEFT( KS ) 955
956 94 150 CONTINUE 956
957 95 C 957
958 96 DO 152 KS = 1 , NOFVES( INS ) 958
959 97 CFFR = ( GGR( KS ) + 1. ) / ( 2. * GGR( KS ) ) 959
960 98 WRIFS( KS ) = ( 1. + CFFR * ( PSTAR( KS ) / 960
961 99 PPR( KS ) - 1. ) ) * WRISQ( KS ) 961
962 100 WRIGT( KS ) = SQRT( WRIFS( KS ) ) 962
963 101 ZRIGT( KS ) = 2. * WRIGT( KS ) * WRIFS( KS ) / 963
964 102 ( WRISQ( KS ) + WRIFS( KS ) ) 964
965 103 USTR( KS ) = UUR( KS ) + 965
966 104 ( PSTAR( KS ) - PPR( KS ) ) / WRIGT( KS ) 966
967 105 152 CONTINUE 967
968 106 C 968
969 107 DO 160 KS = 1 , NOFVES( INS ) 969
970 108 DPST( KS ) = ZLEFT( KS ) * ZRIGT( KS ) * 970
971 109 ( USTR( KS ) - USTL( KS ) ) / 971
972 110 ( ZLEFT( KS ) + ZRIGT( KS ) ) 972
973 111 PSTAR( KS ) = PSTAR( KS ) - DPST( KS ) 973
974 112 PSTAR( KS ) = AMAX1( PSTAR( KS ) , PSML( KS ) ) 974
975 113 160 CONTINUE 975
976 114 C 976
977 115 140 CONTINUE 977
978 116 C 978
979 117 C --- FORM FINAL SOLUTIONS ----- 979
980 118 C 980
981 119 DO 170 KS = 1 , NOFVES( INS ) 981
982 120 CFFL = ( GGL( KS ) + 1. ) / ( 2. * GGL( KS ) ) 982
983 121 WLEFT( KS ) = SQRT( WLESQ( KS ) * ( 1. + 983
984 122 CFFL * ( PSTAR( KS ) / PPL( KS ) - 1. ) ) ) 984
985 123 170 CONTINUE 985
986 124 C 986
987 125 DO 172 KS = 1 , NOFVES( INS ) 987
988 126 CFFR = ( GGR( KS ) + 1. ) / ( 2. * GGR( KS ) ) 988
989 127 WRIGT( KS ) = SQRT( WRISQ( KS ) * ( 1. + 989
990 128 CFFR * ( PSTAR( KS ) / PPR( KS ) - 1. ) ) ) 990
991 129 172 CONTINUE 991
992 130 C 992
993 131 DO 180 KS = 1 , NOFVES( INS ) 993
994 132 USTAR( KS ) = ( PPL( KS ) - PPR( KS ) + 994
995 133 WLEFT( KS ) * UUL( KS ) + 995
996 134 WRIGT( KS ) * UUR( KS ) ) / 996
997 135 ( WLEFT( KS ) + WRIGT( KS ) ) 997
998 136 180 CONTINUE 998
999 137 C 999
1000 138 DO 190 KS = 1 , NOFVES( INS ) 1000
1001 139 C 1001
1002 140 C --- BEGIN PROCEDURE TO OBTAIN FLUXES FROM REIMANN FORMALISM -- 1002
1003 141 C 1003
1004 142 IF( USTAR( KS ) . LE . 0.0 ) THEN 1004
1005 143 C 1005
1006 144 RO( KS ) = RRR( KS ) 1006
1007 145 PO( KS ) = PPR( KS ) 1007
1008 146 UO( KS ) = UUR( KS ) 1008
1009 147 CO( KS ) = SQRT( HRGG * PPR( KS ) / RRR( KS ) ) 1009
1010 148 WO( KS ) = WRIGT( KS ) 1010

```

```

1011 149          GO( KS ) = GGR( KS )
1012 150          ISN( KS ) = 1
1013 151          C
1014 152          ENRGYI( KS ) = EER( KS )
1015 153          ANRGYI( KS ) = AAR( KS )
1016 154          VGDNV( KS ) = VVR( KS )
1017 155          WGDNV( KS ) = WWR( KS )
1018 156          C
1019 157          ELSE
1020 158          C
1021 159          RO( KS ) = RRL( KS )
1022 160          PO( KS ) = PPL( KS )
1023 161          UO( KS ) = UUL( KS )
1024 162          CO( KS ) = SQRT( HRGG * PPL( KS ) / RRL( KS ) )
1025 163          WO( KS ) = WLEFT( KS )
1026 164          GO( KS ) = GGL( KS )
1027 165          ISN( KS ) = - 1
1028 166          C
1029 167          ENRGYI( KS ) = EEL( KS )
1030 168          ANRGYI( KS ) = AAL( KS )
1031 169          VGDNV( KS ) = VVL( KS )
1032 170          WGDNV( KS ) = WVL( KS )
1033 171          END IF
1034 172          190 CONTINUE
1035 173          C
1036 174          DO 200 KS = 1 , NOFVES( INS )
1037 175             DELP( KS ) = PSTAR( KS ) - PO( KS )
1038 176             WSOP( KS ) = ISN( KS ) * UO( KS ) + WO( KS ) / RO( KS )
1039 177             WSOM( KS ) = ISN( KS ) * UO( KS ) + CO( KS )
1040 178          200 CONTINUE
1041 179          C
1042 180          DO 210 KS = 1 , NOFVES( INS )
1043 181             IF( DELP( KS ) . GT . 0. ) THEN
1044 182                WS00( KS ) = WSOP( KS )
1045 183             ELSE
1046 184                WS00( KS ) = WSOM( KS )
1047 185             END IF
1048 186          210 CONTINUE
1049 187          C
1050 188          DO 220 KS = 1 , NOFVES( INS )
1051 189          C
1052 190          C --- USE OUTER STATE SOLUTION -----
1053 191          C
1054 192             PGDNV( KS ) = PO( KS )
1055 193             UGDNV( KS ) = UO( KS )
1056 194             CGDNV( KS ) = CO( KS )
1057 195             RGDNV( KS ) = RO( KS )
1058 196          220 CONTINUE
1059 197          C
1060 198          C --- COMPUTE STARRED VALUES -----
1061 199          C
1062 200          DO 230 KS = 1 , NOFVES( INS )
1063 201             RSTAR( KS ) = 1. / ( 1. / RO( KS ) - DELP( KS ) /
1064 202                ( WO( KS ) * WO( KS ) ) )
1065 203             CSTAR( KS ) = SQRT( GO( KS ) * PSTAR( KS ) / RSTAR( KS ) )
1066 204             WSOM( KS ) = ISN( KS ) * USTAR( KS ) + CSTAR( KS )
1067 205          230 CONTINUE
1068 206          C
1069 207          DO 240 KS = 1 , NOFVES( INS )
1070 208             IF( DELP( KS ) . GT . 0. ) THEN
1071 209                SPIN( KS ) = WSOP( KS )
1072 210             ELSE
1073 211                SPIN( KS ) = WSOM( KS )
1074 212             END IF
1075 213          240 CONTINUE
1076 214          C
1077 215          DO 250 KS = 1 , NOFVES( INS )
1078 216          C
1079 217             IF( WS00( KS ) . GE . 0. ) THEN
1080 218                IF( SPIN( KS ) . GE . 0. ) THEN
1081 219          C
1082 220          C --- USE THE STARRED STATE RESULTS -----
1083 221          C
1084 222             RGDNV( KS ) = RSTAR( KS )

```

```

1085 223 UGDNV( KS ) = USTAR( KS )
1086 224 CGDNV( KS ) = CSTAR( KS )
1087 225 PGDNV( KS ) = PSTAR( KS )
1088 226 ELSE
1089 227 C
1090 228 C --- EVALUATE THE INSIDE RAREFACTION WAVE -----
1091 229 C
1092 230 CGDNV( KS ) = ( CSTAR( KS ) * 2. -
1093 231 . ISN( KS ) * USTAR( KS ) * ( GO( KS ) - 1. ) )
1094 232 . / ( GO( KS ) + 1. )
1095 233 UGDNV( KS ) = - ISN( KS ) * CGDNV( KS )
1096 234 RGDNV( KS ) = ( CGDNV( KS ) / CO( KS ) ) **
1097 235 . ( 2. / ( GO( KS ) - 1. ) ) * RO( KS )
1098 236 PGDNV( KS ) = CGDNV( KS ) * CGDNV( KS ) * RGDNV( KS ) /
1099 237 . GO( KS )
1100 238 C
1101 239 END IF
1102 240 C
1103 241 END IF
1104 242 250 CONTINUE
1105 243 C
1106 244 DO 260 KS = 1 , NOFVES( INS )
1107 245 IS = KS + NS1 - 1
1108 246 C
1109 247 ICL = JS( 7 , IS )
1110 248 ICR = JS( 8 , IS )
1111 249 C
1112 250 CTT = SORT( GO( KS ) * PGDNV( KS ) / RGDNV( KS ) )
1113 251 XSS = XS( 5 , IS )
1114 252 XYZ = 1. / XSS
1115 253 C
1116 254 IATRB = JS( 9 , IS )
1117 255 IF( IATRB .EQ. 0 ) THEN
1118 256 C
1119 257 XXN = ( XC( 1 , ICR ) - XC( 1 , ICL ) ) * XYZ
1120 258 YYN = ( XC( 2 , ICR ) - XC( 2 , ICL ) ) * XYZ
1121 259 ZZN = ( XC( 3 , ICR ) - XC( 3 , ICL ) ) * XYZ
1122 260 C
1123 261 VEL =
1124 262 . ( UGDNV( KS ) * XN( IS ) +
1125 263 . VGDNV( KS ) * XP( IS ) +
1126 264 . WGDNV( KS ) * XT( IS ) ) * XXN +
1127 265 . ( UGDNV( KS ) * YN( IS ) +
1128 266 . VGDNV( KS ) * YP( IS ) +
1129 267 . WGDNV( KS ) * YT( IS ) ) * YYN +
1130 268 . ( UGDNV( KS ) * ZN( IS ) +
1131 269 . VGDNV( KS ) * ZP( IS ) +
1132 270 . WGDNV( KS ) * ZT( IS ) ) * ZZN
1133 271 C
1134 272 DTU = XSS / ( CTT + ABS( VEL ) )
1135 273 DTT = AMINI( DTU , DTT )
1136 274 C
1137 275 ELSE
1138 276 C
1139 277 XXN = ( XYZMDL( 1 , IS ) - XC( 1 , ICL ) ) * XYZ
1140 278 YYN = ( XYZMDL( 2 , IS ) - XC( 2 , ICL ) ) * XYZ
1141 279 ZZN = ( XYZMDL( 3 , IS ) - XC( 3 , ICL ) ) * XYZ
1142 280 C
1143 281 VEL =
1144 282 . ( UGDNV( KS ) * XN( IS ) +
1145 283 . VGDNV( KS ) * XP( IS ) +
1146 284 . WGDNV( KS ) * XT( IS ) ) * XXN +
1147 285 . ( UGDNV( KS ) * YN( IS ) +
1148 286 . VGDNV( KS ) * YP( IS ) +
1149 287 . WGDNV( KS ) * YT( IS ) ) * YYN +
1150 288 . ( UGDNV( KS ) * ZN( IS ) +
1151 289 . VGDNV( KS ) * ZP( IS ) +
1152 290 . WGDNV( KS ) * ZT( IS ) ) * ZZN
1153 291 C
1154 292 DTU = XSS / ( CTT + ABS( VEL ) )
1155 293 DTT = AMINI( DTU , DTT )
1156 294 C
1157 295 END IF
1158 296 260 CONTINUE

```

```

1159 297 C
1160 298 DO 270 KS = 1 , NOFVES( INS )
1161 299 IS = KS + NS1 - 1
1162 300 C
1163 301 C ... FLUX FOR DENSITY .....
1164 302 C
1165 303 RO( KS ) = RGDNV( KS ) * UGDNV( KS )
1166 304 C
1167 305 C ... FLUX FOR MOMENTUM DENSITY .....
1168 306 C
1169 307 UO( KS ) = PGDNV( KS ) * XN( IS ) +
1170 308 . RO( KS ) * ( UGDNV( KS ) * XN( IS ) +
1171 309 . VGDNV( KS ) * XP( IS ) +
1172 310 . WGDNV( KS ) * XT( IS ) )
1173 311 CO( KS ) = PGDNV( KS ) * YN( IS ) +
1174 312 . RO( KS ) * ( UGDNV( KS ) * YN( IS ) +
1175 313 . VGDNV( KS ) * YP( IS ) +
1176 314 . WGDNV( KS ) * YT( IS ) )
1177 315 WO( KS ) = PGDNV( KS ) * ZN( IS ) +
1178 316 . RO( KS ) * ( UGDNV( KS ) * ZN( IS ) +
1179 317 . VGDNV( KS ) * ZP( IS ) +
1180 318 . WGDNV( KS ) * ZT( IS ) )
1181 319 C
1182 320 C ... FLUX FOR ENERGY DENSITY .....
1183 321 C
1184 322 PO( KS ) = UGDNV( KS ) * ( ENRGYI( KS ) +
1185 323 . .5 * RGDNV( KS ) * ( UGDNV( KS ) * UGDNV( KS ) +
1186 324 . VGDNV( KS ) * VGDNV( KS ) +
1187 325 . WGDNV( KS ) * WGDNV( KS ) ) )
1188 326 C
1189 327 C ... FLUX FOR COMBUSTION INTERFACE TRACKING .....
1190 328 C
1191 329 AO( KS ) = UGDNV( KS ) * RGDNV( KS ) * ANRGYI( KS )
1192 330 C
1193 331 270 CONTINUE
1194 332 C
1195 333 DO 290 IS = NS1 , NS2
1196 334 KS = IS - NS1 + 1
1197 335 C
1198 336 ICL = JS( 7 , IS )
1199 337 ICR = JS( 8 , IS )
1200 338 C
1201 339 IATRB = JS( 9 , IS )
1202 340 IF( IATRB .EQ. 0 ) THEN
1203 341 C
1204 342 C ... FLUX FOR DENSITY .....
1205 343 C
1206 344 DLENG = XS( 4 , IS ) * RO( KS )
1207 345 HYDFLX( ICL , 1 ) = HYDFLX( ICL , 1 ) + DLENG
1208 346 HYDFLX( ICR , 1 ) = HYDFLX( ICR , 1 ) - DLENG
1209 347 C
1210 348 C ... FLUX FOR MOMENTUM DENSITY ( U DIRECTION ) .....
1211 349 C
1212 350 DLENG = XS( 4 , IS ) * UO( KS )
1213 351 HYDFLX( ICL , 2 ) = HYDFLX( ICL , 2 ) + DLENG
1214 352 HYDFLX( ICR , 2 ) = HYDFLX( ICR , 2 ) - DLENG
1215 353 C
1216 354 C ... FLUX FOR MOMENTUM DENSITY ( V DIRECTION ) .....
1217 355 C
1218 356 DLENG = XS( 4 , IS ) * CO( KS )
1219 357 HYDFLX( ICL , 3 ) = HYDFLX( ICL , 3 ) + DLENG
1220 358 HYDFLX( ICR , 3 ) = HYDFLX( ICR , 3 ) - DLENG
1221 359 C
1222 360 C ... FLUX FOR MOMENTUM DENSITY ( W DIRECTION ) .....
1223 361 C
1224 362 DLENG = XS( 4 , IS ) * WO( KS )
1225 363 HYDFLX( ICL , 4 ) = HYDFLX( ICL , 4 ) + DLENG
1226 364 HYDFLX( ICR , 4 ) = HYDFLX( ICR , 4 ) - DLENG
1227 365 C
1228 366 C ... FLUX FOR ENERGY DENSITY .....
1229 367 C
1230 368 DLENG = XS( 4 , IS ) * PO( KS )
1231 369 HYDFLX( ICL , 5 ) = HYDFLX( ICL , 5 ) + DLENG
1232 370 HYDFLX( ICR , 5 ) = HYDFLX( ICR , 5 ) - DLENG

```


1233	371	C		1233
1234	372	C	... FLUX FOR COMBUSTION INTERFACE TRACKING.....	1234
1235	373	C		1235
1236	374		DLENG = XS(4 , IS) * AO(KS)	1236
1237	375		HYDFLX(ICL , 6) = HYDFLX(ICL , 6) + DLENG	1237
1238	376		HYDFLX(ICR , 6) = HYDFLX(ICR , 6) - DLENG	1238
1239	377	C		1239
1240	378		ELSE	1240
1241	379	C		1241
1242	380	C	... FLUX FOR DENSITY	1242
1243	381	C		1243
1244	382		DLENG = XS(4 , IS) * RO(KS)	1244
1245	383		HYDFLX(ICL , 1) = HYDFLX(ICL , 1) + DLENG	1245
1246	384	C		1246
1247	385	C	... FLUX FOR MOMENTUM DENSITY (U DIRECTION)	1247
1248	386	C		1248
1249	387		DLENG = XS(4 , IS) * UO(KS)	1249
1250	388		HYDFLX(ICL , 2) = HYDFLX(ICL , 2) + DLENG	1250
1251	389	C		1251
1252	390	C	... FLUX FOR MOMENTUM DENSITY (V DIRECTION)	1252
1253	391	C		1253
1254	392		DLENG = XS(4 , IS) * CO(KS)	1254
1255	393		HYDFLX(ICL , 3) = HYDFLX(ICL , 3) + DLENG	1255
1256	394	C		1256
1257	395	C	... FLUX FOR MOMENTUM DENSITY (W DIRECTION)	1257
1258	396	C		1258
1259	397		DLENG = XS(4 , IS) * WO(KS)	1259
1260	398		HYDFLX(ICL , 4) = HYDFLX(ICL , 4) + DLENG	1260
1261	399	C		1261
1262	400	C	... FLUX FOR ENERGY DENSITY	1262
1263	401	C		1263
1264	402		DLENG = XS(4 , IS) * PO(KS)	1264
1265	403		HYDFLX(ICL , 5) = HYDFLX(ICL , 5) + DLENG	1265
1266	404	C		1266
1267	405	C	... FLUX FOR COMBUSTION INTERFACE TRACKING.....	1267
1268	406	C		1268
1269	407		DLENG = XS(4 , IS) * AO(KS)	1269
1270	408		HYDFLX(ICL , 6) = HYDFLX(ICL , 6) + DLENG	1270
1271	409	C		1271
1272	410		END IF	1272
1273	411	290	CONTINUE	1273
1274	412	C		1274
1275	413		NS1 = NS2 + 1	1275
1276	414		NS2 = NS2 + NOFVES(INS + 1)	1276
1277	415	110	CONTINUE	1277
1278	416	C		1278
1279	417		RETURN	1279
1280	418		END	1280
1281	419	C		1281

```

1282 1      SUBROUTINE RYDRFL
1283 2      C
1284 3      C-----I
1285 4      C
1286 5      C      RYDRFL IS A 2 DIMENSIONAL RIEMANN SOLVER THAT INTEGRATES
1287 6      C      FLUXES ACROSS NORMAL INTERFACES TO UPDATE VERTICES
1288 7      C      VARIABLES .
1289 8      C
1290 9      C-----I
1291 10     C
1292 11     include 'dmsh00.h'
1293 12     include 'dhyd0.h'
1294 13     include 'dphsm0.h'
1295 14     include 'dmtr10.h'
1296 15     C
1297 16     REAL DELP(128),WSOP(128),WSOM(128),WSOO(128),
1298 17     .      RSTAR(128),CSTAR(128),PMAX(128),PMIN(128)
1299 18     REAL RRIGHT(128),URIGHT(128),VRIGHT(128),PRIGHT(128)
1300 19     REAL RLEFT(128),ULEFT(128),VLEFT(128),PLEFT(128)
1301 20     REAL ENRGYI(128),ANRGYI(128)
1302 21     C
1303 22     C --- BEGIN LOOP OVER ALL EDGES IN THE DOMAIN -----
1304 23     C
1305 24     NS1 = 1
1306 25     NS2 = NOFVES( 1 )
1307 26     DO 110 INS = 1 , NVEES
1308 27     C
1309 28     C --- FETCH HYDRO QUANTITIES -----
1310 29     C
1311 30     DO 120 IS = NS1 , NS2
1312 31     .      KS = IS - NS1 + 1
1313 32     C
1314 33     .      ICL = JS( 7 , IS )
1315 34     .      IBC = JS( 9 , IS )
1316 35     C
1317 36     .      RRL( KS ) = HYDV( ICL , 1 )
1318 37     .      UUL( KS ) = HYDV( ICL , 2 ) * XN( IS ) +
1319 38     .      .      HYDV( ICL , 3 ) * YN( IS ) +
1320 39     .      .      HYDV( ICL , 4 ) * ZN( IS )
1321 40     .      VVL( KS ) = HYDV( ICL , 2 ) * XP( IS ) +
1322 41     .      .      HYDV( ICL , 3 ) * YP( IS ) +
1323 42     .      .      HYDV( ICL , 4 ) * ZP( IS )
1324 43     .      WWL( KS ) = HYDV( ICL , 2 ) * XT( IS ) +
1325 44     .      .      HYDV( ICL , 3 ) * YT( IS ) +
1326 45     .      .      HYDV( ICL , 4 ) * ZT( IS )
1327 46     .      PPL( KS ) = HYDV( ICL , 5 )
1328 47     .      AAL( KS ) = HYDV( ICL , 6 )
1329 48     .      EEL( KS ) = HYDV( ICL , 8 )
1330 49     .      GGL( KS ) = HYDV( ICL , 7 )
1331 50     C
1332 51     .      RRR( KS ) = RRL( KS )
1333 52     .      IF( IBC . EQ . 0 ) THEN
1334 53     .      UUR( KS ) = UUL( KS )
1335 54     .      ELSE
1336 55     .      UUR( KS ) = - UUL( KS )
1337 56     .      END IF
1338 57     .      VVR( KS ) = VVL( KS )
1339 58     .      WWR( KS ) = WWL( KS )
1340 59     .      PPR( KS ) = PPL( KS )
1341 60     .      AAR( KS ) = AAL( KS )
1342 61     .      EER( KS ) = EEL( KS )
1343 62     .      GGR( KS ) = GGL( KS )
1344 63     C
1345 64     120 CONTINUE
1346 65     C
1347 66     DO 130 KS = 1 , NOFVES( INS )
1348 67     C
1349 68     C --- THIS SECTION OF CODE SOLVES FOR "PSTAR" AND "USTAR" IN
1350 69     C      THE RIEMANN PROBLEM USING NEWTON'S METHOD.
1351 70     C
1352 71     .      WLEFT( KS ) = SQRT( GGL( KS ) * PPL( KS ) * RRL( KS ) )
1353 72     .      WRIGT( KS ) = SQRT( GGR( KS ) * PPR( KS ) * RRR( KS ) )
1354 73     .      WLESQ( KS ) = WLEFT( KS ) * WLEFT( KS )
1355 74     .      WRISQ( KS ) = WRIGT( KS ) * WRIGT( KS )

```

```

1356 75 C
1357 76 PMIN( KS ) = AMIN1( PPL( KS ) , PPR( KS ) )
1358 77 PSML( KS ) = HRSM * PMIN( KS )
1359 78 C
1360 79 C --- FORM THE STARTING GUESS FOR THE SOLUTION -----
1361 80 C
1362 81 PSTAR( KS ) = ( WLEFT( KS ) * PPR( KS ) +
1363 82 WRIGT( KS ) * PPL( KS ) -
1364 83 WLEFT( KS ) * WRIGT( KS ) *
1365 84 ( UUR( KS ) - UUL( KS ) ) /
1366 85 ( WLEFT( KS ) + WRIGT( KS ) )
1367 86 PSTAR( KS ) = AMAX1( PSTAR( KS ) , PSML( KS ) )
1368 87 130 CONTINUE
1369 88 C
1370 89 DO 140 I = 1 , IHRN
1371 90 C
1372 91 C --- BEGIN THE NEWTON ITERATION -----
1373 92 C
1374 93 DO 150 KS = 1 , NOFVES( INS )
1375 94 CFFL = ( GGL( KS ) + 1. ) / ( 2. * GGL( KS ) )
1376 95 WLEFS( KS ) = ( 1. + CFFL * ( PSTAR( KS ) /
1377 96 PPL( KS ) - 1. ) ) * WLESQ( KS )
1378 97 WLEFT( KS ) = SQRT( WLEFS( KS ) )
1379 98 ZLEFT( KS ) = 2. * WLEFT( KS ) * WLEFS( KS ) /
1380 99 ( WLESQ( KS ) + WLEFS( KS ) )
1381 100 USTL( KS ) = UUL( KS ) -
1382 101 ( PSTAR( KS ) - PPL( KS ) ) / WLEFT( KS )
1383 102 150 CONTINUE
1384 103 C
1385 104 DO 152 KS = 1 , NOFVES( INS )
1386 105 CFFR = ( GGR( KS ) + 1. ) / ( 2. * GGR( KS ) )
1387 106 WRIFS( KS ) = ( 1. + CFFR * ( PSTAR( KS ) /
1388 107 PPR( KS ) - 1. ) ) * WRISQ( KS )
1389 108 WRIGT( KS ) = SQRT( WRIFS( KS ) )
1390 109 ZRIGT( KS ) = 2. * WRIGT( KS ) * WRIFS( KS ) /
1391 110 ( WRISQ( KS ) + WRIFS( KS ) )
1392 111 USTR( KS ) = UUR( KS ) +
1393 112 ( PSTAR( KS ) - PPR( KS ) ) / WRIGT( KS )
1394 113 152 CONTINUE
1395 114 C
1396 115 DO 160 KS = 1 , NOFVES( INS )
1397 116 DPST( KS ) = ZLEFT( KS ) * ZRIGT( KS ) *
1398 117 ( USTR( KS ) - USTL( KS ) ) /
1399 118 ( ZLEFT( KS ) + ZRIGT( KS ) )
1400 119 PSTAR( KS ) = PSTAR( KS ) - DPST( KS )
1401 120 PSTAR( KS ) = AMAX1( PSTAR( KS ) , PSML( KS ) )
1402 121 160 CONTINUE
1403 122 C
1404 123 140 CONTINUE
1405 124 C
1406 125 C --- FORM FINAL SOLUTIONS -----
1407 126 C
1408 127 DO 170 KS = 1 , NOFVES( INS )
1409 128 CFFL = ( GGL( KS ) + 1. ) / ( 2. * GGL( KS ) )
1410 129 WLEFT( KS ) = SQRT( WLESQ( KS ) * ( 1. +
1411 130 CFFL * ( PSTAR( KS ) / PPL( KS ) - 1. ) ) )
1412 131 170 CONTINUE
1413 132 C
1414 133 DO 172 KS = 1 , NOFVES( INS )
1415 134 CFFR = ( GGR( KS ) + 1. ) / ( 2. * GGR( KS ) )
1416 135 WRIGT( KS ) = SQRT( WRISQ( KS ) * ( 1. +
1417 136 CFFR * ( PSTAR( KS ) / PPR( KS ) - 1. ) ) )
1418 137 172 CONTINUE
1419 138 C
1420 139 DO 180 KS = 1 , NOFVES( INS )
1421 140 USTAR( KS ) = ( PPL( KS ) - PPR( KS ) +
1422 141 WLEFT( KS ) * UUL( KS ) +
1423 142 WRIGT( KS ) * UUR( KS ) ) /
1424 143 ( WLEFT( KS ) + WRIGT( KS ) )
1425 144 180 CONTINUE
1426 145 C
1427 146 DO 190 KS = 1 , NOFVES( INS )
1428 147 C
1429 148 C --- BEGIN PROCEDUR. TO OBTAIN FLUXES FROM REIMANN FORMALISM --

```

```

1430 149 C
1431 150 IF( USTAR( KS ) . LE . 0.0 ) THEN
1432 151 C
1433 152 RO( KS ) = RRR( KS )
1434 153 PO( KS ) = PPR( KS )
1435 154 UO( KS ) = UUR( KS )
1436 155 CO( KS ) = SQRT( HRGG * PPR( KS ) / RRR( KS ) )
1437 156 WO( KS ) = WRIGT( KS )
1438 157 GO( KS ) = GGR( KS )
1439 158 ISN( KS ) = 1
1440 159 C
1441 160 ENRGYI( KS ) = EER( KS )
1442 161 ANRGYI( KS ) = AAR( KS )
1443 162 VGDNV( KS ) = VVR( KS )
1444 163 WGDNV( KS ) = WWR( KS )
1445 164 C
1446 165 ELSE
1447 166 C
1448 167 RO( KS ) = RRL( KS )
1449 168 PO( KS ) = PPL( KS )
1450 169 UO( KS ) = JUL( KS )
1451 170 CO( KS ) = SQRT( HRGG * PPL( KS ) / RRL( KS ) )
1452 171 WO( KS ) = WLEFT( KS )
1453 172 GO( KS ) = GGL( KS )
1454 173 ISN( KS ) = - 1
1455 174 C
1456 175 ENRGYI( KS ) = EEL( KS )
1457 176 ANRGYI( KS ) = AAL( KS )
1458 177 VGDNV( KS ) = VVL( KS )
1459 178 WGDNV( KS ) = WWL( KS )
1460 179 END IF
1461 180 190 CONTINUE
1462 181 C
1463 182 DO 200 KS = 1 , NOFVES( INS )
1464 183 DELP( KS ) = PSTAR( KS ) - PO( KS )
1465 184 WSOP( KS ) = ISN( KS ) * UO( KS ) + WO( KS ) / RO( KS )
1466 185 WSOM( KS ) = ISN( KS ) * UO( KS ) + CO( KS )
1467 186 200 CONTINUE
1468 187 C
1469 188 DO 210 KS = 1 , NOFVES( INS )
1470 189 IF( DELP( KS ) . GT . 0. ) THEN
1471 190 WSOO( KS ) = WSOP( KS )
1472 191 ELSE
1473 192 WSOO( KS ) = WSOM( KS )
1474 193 END IF
1475 194 210 CONTINUE
1476 195 C
1477 196 DO 220 KS = 1 , NOFVES( INS )
1478 197 C
1479 198 C --- USE OUTER STATE SOLUTION -----
1480 199 C
1481 200 PGDNV( KS ) = PO( KS )
1482 201 UGDNV( KS ) = UO( KS )
1483 202 CGDNV( KS ) = CO( KS )
1484 203 RGDNV( KS ) = RO( KS )
1485 204 220 CONTINUE
1486 205 C
1487 206 C --- COMPUTE STARRED VALUES -----
1488 207 C
1489 208 DO 230 KS = 1 , NOFVES( INS )
1490 209 RSTAR( KS ) = 1. / ( 1. / RO( KS ) - DELP( KS ) /
1491 210 ( WO( KS ) * MO( KS ) ) )
1492 211 CSTAR( KS ) = SQRT( GO( KS ) * PSTAR( KS ) / RSTAR( KS ) )
1493 212 WSOM( KS ) = ISN( KS ) * USTAR( KS ) + CSTAR( KS )
1494 213 230 CONTINUE
1495 214 C
1496 215 DO 240 KS = 1 , NOFVES( INS )
1497 216 IF( DELP( KS ) . GT . 0. ) THEN
1498 217 SPIN( KS ) = WSOP( KS )
1499 218 ELSE
1500 219 SPIN( KS ) = WSOM( KS )
1501 220 END IF
1502 221 240 CONTINUE
1503 222 C

```

```

1504 223      DO 250 KS = 1 , NOFVES( INS )
1505 224      C
1506 225      IF( WSOO( KS ) . GE . 0. ) THEN
1507 226          IF( SPIN( KS ) . GE . 0. ) THEN
1508 227      C
1509 228      C --- USE THE STARRED STATE RESULTS -----
1510 229      C
1511 230          RGDNV( KS ) = RSTAR( KS )
1512 231          UGDNV( KS ) = USTAR( KS )
1513 232          CGDNV( KS ) = CSTAR( KS )
1514 233          PGDNV( KS ) = PSTAR( KS )
1515 234      ELSE
1516 235      C
1517 236      C --- EVALUATE THE INSIDE RAREFACTION WAVE -----
1518 237      C
1519 238          CGDNV( KS ) = ( CSTAR( KS ) * 2. -
1520 239              .      ISN( KS ) * USTAR( KS ) * ( GO( KS ) - 1. ) )
1521 240              .      / ( GO( KS ) + 1. )
1522 241          UGDNV( KS ) = - ISN( KS ) * CGDNV( KS )
1523 242          RGDNV( KS ) = ( CGDNV( KS ) / CO( KS ) ) **
1524 243              .      ( 2. / ( GO( KS ) - 1. ) ) * RO( KS )
1525 244          PGDNV( KS ) = CGDNV( KS ) * CGDNV( KS ) * RGDNV( KS ) /
1526 245              .      GO( KS )
1527 246      C
1528 247      END IF
1529 248      C
1530 249      END IF
1531 250      250 CONTINUE
1532 251      C
1533 252      DO 260 KS = 1 , NOFVES( INS )
1534 253          IS = KS + NS1 - 1
1535 254          RR( IS ) = RGDNV( KS )
1536 255          PR( IS ) = PGDNV( KS )
1537 256      260 CONTINUE
1538 257      C
1539 258          NS1 = NS2 + 1
1540 259          NS2 = NS2 + NOFVES( INS + 1 )
1541 260      110 CONTINUE
1542 261      C
1543 262      RETURN
1544 263      END
1545 264      C

```

```

1546 1      SUBROUTINE KYDRFL
1547 2      C
1548 3      C-----I
1549 4      C-----I
1550 5      C      KYDRFL IS A 2 DIMENSIONAL RIEMANN SOLVER THAT INTEGRATES I
1551 6      C      FLUXES ACROSS NORMAL INTERFACES TO UPDATE VERTICES I
1552 7      C      VARIABLES . I
1553 8      C-----I
1554 9      C-----I
1555 10     C
1556 11     include 'dmsh00.h'
1557 12     include 'dhyd0.h'
1558 13     include 'dphsm0.h'
1559 14     include 'dmtri0.h'
1560 15     C
1561 16     REAL DELP(128),WSOP(128),WSOM(128),WSOO(128),
1562 17     .      RSTAR(128),CSTAR(128),PMAX(128),PMIN(128)
1563 18     REAL RRIGHT(128),URIGHT(128),VRIGHT(128),PRIGHT(128)
1564 19     REAL RLEFTT(128),ULEFTT(128),VLEFTT(128),PLEFTT(128)
1565 20     INTEGER NOFVEP(128)
1566 21     C
1567 22     C --- FETCH HYDRO QUANTITIES -----
1568 23     C
1569 24     DO 120 KS = 1 , NPRTCL
1570 25         IS = IJKPRT( KS )
1571 26         ICL = JS( 7 , IS )
1572 27         IBC = JS( 9 , IS )
1573 28     C
1574 29         RRL( KS ) = HYDV( ICL , 1 )

```

```

1575 30      UUL( KS ) = HYDV( ICL , 2 ) * XN( IS ) +      1575
1576 31      .      HYDV( ICL , 3 ) * YN( IS ) +      1576
1577 32      .      HYDV( ICL , 4 ) * ZN( IS )      1577
1578 33      VVL( KS ) = HYDV( ICL , 2 ) * XP( IS ) +      1578
1579 34      .      HYDV( ICL , 3 ) * YP( IS ) +      1579
1580 35      .      HYDV( ICL , 4 ) * ZP( IS )      1580
1581 36      WHL( KS ) = HYDV( ICL , 2 ) * XT( IS ) +      1581
1582 37      .      HYDV( ICL , 3 ) * YT( IS ) +      1582
1583 38      .      HYDV( ICL , 4 ) * ZT( IS )      1583
1584 39      PPL( KS ) = HYDV( ICL , 5 )      1584
1585 40      AAL( KS ) = HYDV( ICL , 6 )      1585
1586 41      EEL( KS ) = HYDV( ICL , 8 )      1586
1587 42      GGL( KS ) = HYDV( ICL , 7 )      1587
1588 43      C      1588
1589 44      RRR( KS ) = RRL( KS )      1589
1590 45      IF( IBC .EQ. 0 ) THEN      1590
1591 46      UUR( KS ) = UUL( KS )      1591
1592 47      ELSE      1592
1593 48      UUR( KS ) = - UUL( KS )      1593
1594 49      END IF      1594
1595 50      VVR( KS ) = VVL( KS )      1595
1596 51      WWR( KS ) = WML( KS )      1596
1597 52      PPR( KS ) = PPL( KS )      1597
1598 53      AAR( KS ) = AAL( KS )      1598
1599 54      EER( KS ) = EEL( KS )      1599
1600 55      GGR( KS ) = GGL( KS )      1600
1601 56      C      1601
1602 57      120 CONTINUE      1602
1603 58      C      1603
1604 59      DO 130 KS = 1 , NPRTCL      1604
1605 60      C      1605
1606 61      C --- THIS SECTION OF CODE SOLVES FOR "PSTAR" AND "USTAR" IN      1606
1607 62      C THE RIEMANN PROBLEM USING NEWTON'S METHOD.      1607
1608 63      C      1608
1609 64      WLEFT( KS ) = SQRT( GGL( KS ) * PPL( KS ) * RRL( KS ) )      1609
1610 65      WRIGT( KS ) = SQRT( GGR( KS ) * PPR( KS ) * RRR( KS ) )      1610
1611 66      WLESQ( KS ) = WLEFT( KS ) * WLEFT( KS )      1611
1612 67      WRISQ( KS ) = WRIGT( KS ) * WRIGT( KS )      1612
1613 68      C      1613
1614 69      PMIN( KS ) = AMIN1( PPL( KS ) , PPR( KS ) )      1614
1615 70      PSML( KS ) = HRSM * PMIN( KS )      1615
1616 71      C      1616
1617 72      C --- FORM THE STARTING GUESS FOR THE SOLUTION' -----      1617
1618 73      C      1618
1619 74      PSTAR( KS ) = ( WLEFT( KS ) * PPR( KS ) +      1619
1620 75      .      WRIGT( KS ) * PPL( KS ) -      1620
1621 76      .      WLEFT( KS ) * WRIGT( KS ) *      1621
1622 77      .      ( UUR( KS ) - UUL( KS ) ) ) /      1622
1623 78      .      ( WLEFT( KS ) + WRIGT( KS ) )      1623
1624 79      PSTAR( KS ) = AMAX1( PSTAR( KS ) , PSML( KS ) )      1624
1625 80      130 CONTINUE      1625
1626 81      C      1626
1627 82      DO 140 I = 1 , IHRN      1627
1628 83      C      1628
1629 84      C --- BEGIN THE NEWTON ITERATION -----      1629
1630 85      C      1630
1631 86      DO 150 KS = 1 , NPRTCL      1631
1632 87      CFFL = ( GGL( KS ) + 1. ) / ( 2. * GGL( KS ) )      1632
1633 88      WLEFS( KS ) = ( 1. + CFFL * ( PSTAR( KS ) /      1633
1634 89      .      PPL( KS ) - 1. ) ) * WLESQ( KS )      1634
1635 90      WLEFT( KS ) = SQRT( WLEFS( KS ) )      1635
1636 91      ZLEFT( KS ) = 2. * WLEFT( KS ) * WLEFS( KS ) /      1636
1637 92      .      ( WLESQ( KS ) + WLEFS( KS ) )      1637
1638 93      USTL( KS ) = UUL( KS ) -      1638
1639 94      .      ( PSTAR( KS ) - PPL( KS ) ) / WLEFT( KS )      1639
1640 95      150 CONTINUE      1640
1641 96      C      1641
1642 97      DO 152 KS = 1 , NPRTCL      1642
1643 98      CFFR = ( GGR( KS ) + 1. ) / ( 2. * GGR( KS ) )      1643
1644 99      WRIFS( KS ) = ( 1. + CFFR * ( PSTAR( KS ) /      1644
1645 100      .      PPR( KS ) - 1. ) ) * WRISQ( KS )      1645
1646 101      WRIGT( KS ) = SQRT( WRIFS( KS ) )      1646
1647 102      ZRIGT( KS ) = 2. * WRIGT( KS ) * WRIFS( KS ) /      1647
1648 103      .      ( WRISQ( KS ) + WRIFS( KS ) )      1648

```

```

1649 104          USTR( KS ) = UUR( KS ) +
1650 105          ( PSTAR( KS ) - PPR( KS ) ) / WRIGT( KS )
1651 106 152 CONTINUE
1652 107 C
1653 108      DO 160 KS = 1 , NPRTCL
1654 109          DPST( KS ) = ZLEFT( KS ) * ZRIGHT( KS ) *
1655 110          ( USTR( KS ) - USTL( KS ) ) /
1656 111          ( ZLEFT( KS ) + ZRIGHT( KS ) )
1657 112          PSTAR( KS ) = PSTAR( KS ) - DPST( KS )
1658 113          PSTAR( KS ) = AMAX1( PSTAR( KS ) , PSML( KS ) )
1659 114 160 CONTINUE
1660 115 C
1661 116 140 CONTINUE
1662 117 C
1663 118 C --- FORM FINAL SOLUTIONS -----
1664 119 C
1665 120      DO 170 KS = 1 , NPRTCL
1666 121          CFFL = ( GGL( KS ) + 1. ) / ( 2. * GGL( KS ) )
1667 122          WLEFT( KS ) = SQRT( WLESQ( KS ) * ( 1. +
1668 123          CFFL * ( PSTAR( KS ) / PPL( KS ) - 1. ) ) )
1669 124 170 CONTINUE
1670 125 C
1671 126      DO 172 KS = 1 , NPRTCL
1672 127          CFFR = ( GGR( KS ) + 1. ) / ( 2. * GGR( KS ) )
1673 128          WRIGT( KS ) = SQRT( WRISQ( KS ) * ( 1. +
1674 129          CFFR * ( PSTAR( KS ) / PPR( KS ) - 1. ) ) )
1675 130 172 CONTINUE
1676 131 C
1677 132      DO 180 KS = 1 , NPRTCL
1678 133          USTAR( KS ) = ( PPL( KS ) - PPR( KS ) +
1679 134          WLEFT( KS ) * UUL( KS ) +
1680 135          WRIGT( KS ) * UUR( KS ) ) /
1681 136          ( WLEFT( KS ) + WRIGT( KS ) )
1682 137 180 CONTINUE
1683 138 C
1684 139      DO 190 KS = 1 , NPRTCL
1685 140 C
1686 141 C --- BEGIN PROCEDURE TO OBTAIN FLUXES FROM REIMANN FORMALISM --
1687 142 C
1688 143          IF( USTAR( KS ) . LE . 0.0 ) THEN
1689 144 C
1690 145              RO( KS ) = RRR( KS )
1691 146              PO( KS ) = PPR( KS )
1692 147              UO( KS ) = UUR( KS )
1693 148              CO( KS ) = SQRT( HRGG * PPR( KS ) / RRR( KS ) )
1694 149              WO( KS ) = WRIGT( KS )
1695 150              GO( KS ) = GGR( KS )
1696 151              ISN( KS ) = 1
1697 152 C
1698 153              VGDNV( KS ) = VVR( KS )
1699 154              WGDNV( KS ) = WWR( KS )
1700 155 C
1701 156          ELSE
1702 157 C
1703 158              RO( KS ) = RRL( KS )
1704 159              PO( KS ) = PPL( KS )
1705 160              UO( KS ) = UUL( KS )
1706 161              CO( KS ) = SQRT( HRGG * PPL( KS ) / RRL( KS ) )
1707 162              WO( KS ) = WLEFT( KS )
1708 163              GO( KS ) = GGL( KS )
1709 164              ISN( KS ) = - 1
1710 165 C
1711 166              VGDNV( KS ) = VVL( KS )
1712 167              WGDNV( KS ) = WWL( KS )
1713 168          END IF
1714 169 190 CONTINUE
1715 170 C
1716 171      DO 200 KS = 1 , NPRTCL
1717 172          DELP( KS ) = PSTAR( KS ) - PO( KS )
1718 173          WSOP( KS ) = ISN( KS ) * UO( KS ) + WO( KS ) / RO( KS )
1719 174          WSOM( KS ) = ISN( KS ) * UO( KS ) + CO( KS )
1720 175 200 CONTINUE
1721 176 C
1722 177      DO 210 KS = 1 , NPRTCL

```

```

1723 178      IF( DELP( KS ) . GT . 0. ) THEN
1724 179      WSOO( KS ) = WSOP( KS )
1725 180      ELSE
1726 181      WSOO( KS ) = WSOM( KS )
1727 182      END IF
1728 183 210 CONTINUE
1729 184 C
1730 185      DO 220 KS = 1 , NPRTCL
1731 186 C
1732 187 C --- USE OUTER STATE SOLUTION -----
1733 188 C
1734 189      PGDNV( KS ) = PO( KS )
1735 190      UGDNV( KS ) = UO( KS )
1736 191      CGDNV( KS ) = CO( KS )
1737 192      RGDNV( KS ) = RO( KS )
1738 193 220 CONTINUE
1739 194 C
1740 195 C --- COMPUTE STARRED VALUES -----
1741 196 C
1742 197      DO 230 KS = 1 , NPRTCL
1743 198      RSTAR( KS ) = 1. / ( 1. / RO( KS ) - DELP( KS ) /
1744 199      ( WO( KS ) * WO( KS ) ) )
1745 200      CSTAR( KS ) = SQRT( GO( KS ) * PSTAR( KS ) / RSTAR( KS ) )
1746 201      WSOM( KS ) = ISN( KS ) * USTAR( KS ) + CSTAR( KS )
1747 202 230 CONTINUE
1748 203 C
1749 204      DO 240 KS = 1 , NPRTCL
1750 205      IF( DELP( KS ) . GT . 0. ) THEN
1751 206      SPIN( KS ) = WSOP( KS )
1752 207      ELSE
1753 208      SPIN( KS ) = WSOM( KS )
1754 209      END IF
1755 210 240 CONTINUE
1756 211 C
1757 212      DO 250 KS = 1 , NPRTCL
1758 213 C
1759 214      IF( WSOO( KS ) . GE . 0. ) THEN
1760 215      IF( SPIN( KS ) . GE . 0. ) THEN
1761 216 C
1762 217 C --- USE THE STARRED STATE RESULTS -----
1763 218 C
1764 219      RGDNV( KS ) = RSTAR( KS )
1765 220      UGDNV( KS ) = USTAR( KS )
1766 221      CGDNV( KS ) = CSTAR( KS )
1767 222      PGDNV( KS ) = PSTAR( KS )
1768 223      ELSE
1769 224 C
1770 225 C --- EVALUATE THE INSIDE RAREFACTION WAVE -----
1771 226 C
1772 227      CGDNV( KS ) = ( CSTAR( KS ) * 2. -
1773 228      ISN( KS ) * USTAR( KS ) * ( GO( KS ) - 1. ) )
1774 229      / ( GO( KS ) + 1. )
1775 230      UGDNV( KS ) = - ISN( KS ) * CGDNV( KS )
1776 231      RGDNV( KS ) = ( CGDNV( KS ) / CO( KS ) ) **
1777 232      ( 2. / ( GO( KS ) - 1. ) ) * RO( KS )
1778 233      PGDNV( KS ) = CGDNV( KS ) * CGDNV( KS ) * RGDNV( KS ) /
1779 234      GO( KS )
1780 235 C
1781 236      END IF
1782 237 C
1783 238      END IF
1784 239 250 CONTINUE
1785 240 C
1786 241      DO 260 KS = 1 , NPRTCL
1787 242      RR( KS ) = RGDNV( KS )
1788 243      PR( KS ) = PGDNV( KS )
1789 244 260 CONTINUE
1790 245 C
1791 246      RETURN
1792 247      END
1793 248 C

```



```

1794 1 SUBROUTINE HYDRMN 1794
1795 2 C 1795
1796 3 C -----I 1796
1797 4 C I 1797
1798 5 C HYDRMN IS A 2 DIMENSIONAL RIEMANN SOLVER THAT CALCULATES I 1798
1799 6 C FLUXES ACROSS NORMAL INTERFACES. I 1799
1800 7 C IT IS CONFIGURED TO WORK IN EITHER TWO OR THREE I 1800
1801 8 C DIMENSIONAL SITUATIONS. THE HYDRODYNAMIC QUANTITIES I 1801
1802 9 C CAN BE SIDE OR VERTEX CENTERED FOR 2-D AND CELL OR I 1802
1803 10 C VERTEX CENTERED FOR 3-D. THE SPECIFIC USE IS BASED I 1803
1804 11 C ON THE CONTENTS OF "OPHYD". I 1804
1805 12 C I 1805
1806 13 C I 1806
1807 14 C -----I 1807
1808 15 C 1808
1809 16 C THE USE OF THE HYDRO VARIABLES IS AS FOLLOWS: 1809
1810 17 C 1810
1811 18 C ++++++ 1811
1812 19 C + 1812
1813 20 C + HYDV(IV,IH) CONTAINS VERTEX CENTERED HYDRO- + 1813
1814 21 C + DYNAMIC QUANTITIES. IT IS USED WITH ALL CASES. + 1814
1815 22 C + 1815
1816 23 C + HYDE(IH,IE) CONTAINS EDGE CENTERED HYDRO- + 1816
1817 24 C + DYNAMIC FLUX QUANTITIES WITH ORIENTATION + 1817
1818 25 C + DETERMINED BY THE "SIDE" "VERTEX" OR "CELL " + 1818
1819 26 C + OPTIONS . IT IS USED FOR THE CASES WHERE + 1819
1820 27 C + "OPHYD" = "SIDE 2D" 2-D SIDE CENTERED + 1820
1821 28 C + = "VERTEX2D" 2-D VERTEX CENTERED + 1821
1822 29 C + = "VERTEX3D" 3-D VERTEX CENTERED + 1822
1823 30 C + 1823
1824 31 C + HYDC(IC,IH) CONTAINS CELL CENTERED HYDRO- + 1824
1825 32 C + DYNAMIC QUANTITIES. IT IS USED FOR THE CASE + 1825
1826 33 C + "OPHYD" = "CELL 3D" 3-D CELL CENTERED + 1826
1827 34 C + 1827
1828 35 C + 1828
1829 36 C + IV - VERTEX INDEX + 1829
1830 37 C + IS - SIDE INDEX + 1830
1831 38 C + IE - EDGE INDEX + 1831
1832 39 C + IC - CELL INDEX + 1832
1833 40 C + IH - HYDRO INDEX + 1833
1834 41 C + 1834
1835 42 C + 1 = RO DENSITY IN ***** + 1835
1836 43 C + 2 = UX X VELOCITY ***** + 1836
1837 44 C + 3 = UY Y VELOCITY ***** + 1837
1838 45 C + 4 = UZ ***** + 1838
1839 46 C + 5 = PO PRESSURE IN ***** + 1839
1840 47 C + 6 = EN ENERGY IN ***** + 1840
1841 48 C + 1841
1842 49 C ++++++ 1842
1843 50 C 1843
1844 51 C include 'dmsh00.h' 1844
1845 52 C include 'dhyd0.h' 1845
1846 53 C include 'dphsm0.h' 1846
1847 54 C include 'dmtr10.h' 1847
1848 55 C 1848
1849 56 C REAL RRN(128),URN(128),VRN(128),WRN(128),EPN(128), 1849
1850 57 C ARN(128),XS2S(128),XSAR(128) 1850
1851 58 C REAL HYDVR(128),HYDVU(128),HYDVV(128),HYD VW(128), 1851
1852 59 C HYDVP(128) 1852
1853 60 C INTEGER NDUMMY1,NDUMMY2,NDUMMY3 1853
1854 61 C INTEGER IDUMMY(4),VDATA(2),FDATA(2) 1854
1855 62 C CHARACTER*31 VLABEL 1855
1856 63 C CHARACTER*32 FLABEL 1856
1857 64 C CHARACTER*6 CTRL,CTET 1857
1858 65 C INTEGER ISURF(400000) 1858
1859 66 C 1859
1860 67 C REAL CD1(4),CD2(4) 1860
1861 68 C 1861
1862 69 C NDUMMY1=1 1862
1863 70 C NDUMMY2=4 1863
1864 71 C NDUMMY3=0 1864
1865 72 C IDUMMY(1) = 0 1865
1866 73 C IDUMMY(2) = 0 1866
1867 74 C IDUMMY(3) = 0 1867

```

```

1868 75 IDUMMY(4) = 0
1869 76 VDATA(1) = 1
1870 77 VDATA(2) = 1
1871 78 FDATA(1) = 1
1872 79 FDATA(2) = 4
1873 80 VLABEL=' pressure, new / m**2'
1874 81 FLABEL=' tets faces,'
1875 82 CTRI=' tri '
1876 83 CTET=' tet '
1877 84 C TLIMIT=TT
1878 85 TLIMIT=30.
1879 86 C
1880 87 IJKNUM = 0
1881 88 IF( ICONP . EQ . 1 ) THEN
1882 89 REWIND 10
1883 90 REWIND 26
1884 91 READ (26,*) IJKNUM
1885 92 DO KKJ = 1 , IJKNUM
1886 93 READ (26,*) RO,(RRN(IK),IK=1,NPRTCL)
1887 94 WRITE (10,*) RO,(RRN(IK),IK=1,NPRTCL)
1888 95 END DO
1889 96 END IF
1890 97 DO 120 JT = 1 , NTIME
1891 98 IF(JT.GT.5) IEOS=0
1892 99 C
1893 100 DO KK = 1 , 5
1894 101 DO IV = 1 , NV
1895 102 HNUM( IV , KK ) = 0.
1896 103 END DO
1897 104 END DO
1898 105 DO 140 ITT = 1 , NDUMP
1899 106 C
1900 107 C --- SELECT ORDER OF INTEGRATION -----
1901 108 C
1902 109 IF(IOPORD.EQ.1)THEN
1903 110 CALL FIRST
1904 111 ELSEIF(IOPORD.EQ.2)THEN
1905 112 CALL GRADNT
1906 113 ENDIF
1907 114 C
1908 115 DTT = 1.E24
1909 116 C
1910 117 CALL HYDRFL
1911 118 C
1912 119 DTT = DTT * .4
1913 120 TT = TT + DTT
1914 121 PRINT *,JT,ITT,DTT,TT,NS
1915 122 C
1916 123 NC1 = 1
1917 124 NC2 = NOFVEC( 1 )
1918 125 DO 110 INC = 1 , NVEEC
1919 126 C
1920 127 DO 150 IC = NC1 , NC2
1921 128 KC = IC - NC1 + 1
1922 129 RRR( KC ) = HYDV( IC , 1 )
1923 130 UUR( KC ) = HYDV( IC , 2 )
1924 131 VVR( KC ) = HYDV( IC , 3 )
1925 132 WWR( KC ) = HYDV( IC , 4 )
1926 133 PPR( KC ) = HYDV( IC , 5 )
1927 134 AAR( KC ) = HYDV( IC , 6 )
1928 135 C
1929 136 RRL( KC ) = HYDFLX( IC , 1 )
1930 137 UUL( KC ) = HYDFLX( IC , 2 )
1931 138 VVL( KC ) = HYDFLX( IC , 3 )
1932 139 WWL( KC ) = HYDFLX( IC , 4 )
1933 140 PPL( KC ) = HYDFLX( IC , 5 )
1934 141 AAL( KC ) = HYDFLX( IC , 6 )
1935 142 C
1936 143 XS2S( KC ) = XC( 2 , IC )
1937 144 XSAR( KC ) = SVOLM( IC )
1938 145 150 CONTINUE
1939 146 C
1940 147 DO 170 KC = 1 , NOFVEC( INC )
1941 148 IC = KC + NC1 - 1

```

```

1942 149      RRN( KC ) = RRR( KC )
1943 150      URN( KC ) = RRR( KC ) * UUR( KC )
1944 151      VRN( KC ) = RRR( KC ) * VVR( KC )
1945 152      WRN( KC ) = RRR( KC ) * WWR( KC )
1946 153      EPN( KC ) = HYDV( IC , 8 ) + .5 * RRR( KC ) *
1947 154      .      ( UUR( KC ) * UUR( KC ) +
1948 155      .      VVR( KC ) * VVR( KC ) +
1949 156      .      WWR( KC ) * WWR( KC ) )
1950 157      ARN( KC ) = RRR( KC ) * AAR( KC )
1951 158      170 CONTINUE
1952 159      C --- COMPUTE THE SOURCE TERM FOR AXI SYMMETRY FLOW PROBLEM
1953 160      C --- IF THE FLOW IS NOT AXI SYMMETRY , COMMENT LOOP 160
1954 161      C
1955 162      DO 190 KC = 1 , NOFVEC( INC )
1956 163      IC = KC + NC1 - 1
1957 164      DTA = DTT * XSAR( KC )
1958 165      C
1959 166      RRLL = DTA * RRL( KC )
1960 167      UULL = DTA * UUL( KC )
1961 168      VVLL = DTA * VVL( KC )
1962 169      WWLL = DTA * WWL( KC )
1963 170      RRN( KC ) = RRN( KC ) - RRLL
1964 171      C
1965 172      URN( KC ) = URN( KC ) - UULL
1966 173      C
1967 174      VRN( KC ) = VRN( KC ) - VVLL
1968 175      C
1969 176      WRN( KC ) = WRN( KC ) - WWLL
1970 177      C
1971 178      PPLL = DTA * PPL( KC )
1972 179      EPN( KC ) = EPN( KC ) - PPLL
1973 180      C
1974 181      AALL = DTA * AAL( KC )
1975 182      ARN( KC ) = ARN( KC ) - AALL
1976 183      C
1977 184      190 CONTINUE
1978 185      C
1979 186      DO 195 IC = NC1 , NC2
1980 187      KC = IC - NC1 + 1
1981 188      HDUM = 1. / RRN( KC )
1982 189      HYDV( IC , 1 ) = RRN( KC )
1983 190      HYDV( IC , 2 ) = URN( KC ) * HDUM
1984 191      HYDV( IC , 3 ) = VRN( KC ) * HDUM
1985 192      HYDV( IC , 4 ) = WRN( KC ) * HDUM
1986 193      HYDV( IC , 6 ) = ARN( KC ) * HDUM
1987 194      195 CONTINUE
1988 195      C
1989 196      DO 200 IC = NC1 , NC2
1990 197      KC = IC - NC1 + 1
1991 198      HYDV( IC , 8 ) = ( EPN( KC ) - .5 * HYDV( IC , 1 ) *
1992 199      .      ( HYDV( IC , 2 ) * HYDV( IC , 2 ) +
1993 200      .      HYDV( IC , 3 ) * HYDV( IC , 3 ) +
1994 201      .      HYDV( IC , 4 ) * HYDV( IC , 4 ) ) )
1995 202      200 CONTINUE
1996 203      C
1997 204      IF( IEOS .EQ. 1 ) THEN
1998 205      TLIMIT = .9
1999 206      ITER = 6
2000 207      DO IC = NC1 , NC2
2001 208      KC = IC - NC1 + 1
2002 209      C
2003 210      NITER = 0
2004 211      IF( HYDV( IC , 6 ) .LE. .2 ) THEN
2005 212      C
2006 213      DST = HYDV( IC , 1 ) * GPERCC
2007 214      VOL = WMA * ( 1. - DST / FSA ) / DST / XGA
2008 215      EME0 = HYDV( IC , 8 ) / HYDV( IC , 1 ) * WMA / RGAS
2009 216      C
2010 217      IYY = ( EME0 - EMEOA( 3 ) ) / RANGEA + 1
2011 218      IYY = MAXO( 1 , MINO( IYY , 47 ) )
2012 219      C
2013 220      K = IYY + 2
2014 221      IYY = IYY
2015 222      . + INT( AMAX1( EME0 - EMEOA( K ) , 0. ) / OYA( K ) )

```

```

2016 223 . - INT( AMAX1( EMEOA( K + 1 ) - EMEO , 0. ) / DYA( K ) )
2017 224 IYY = MAXO( 1, MINO( IYY , 47 ) )
2018 225 C
2019 226 K1 = IYY + 2
2020 227 K2 = K1 + 1
2021 228 RT = ( EMEO - EMEOA( K1 ) ) / ( EMEOA( K2 ) - EMEOA( K1 ) )
2022 229 T = TA( K1 ) + 100. * RT
2023 230 CVM = CVM( K1 ) + RT * ( CVM( K2 ) - CVM( K1 ) )
2024 231 ERS = 0.
2025 232 C
2026 233 P = RGAS * T / VOL / GPERCC
2027 234 RGAMM1 = CVM
2028 235 HYDV( IC , 7 ) = 1. / RGAMM1 + 1.
2029 236 HYDV( IC , 5 ) = P
2030 237 C
2031 238 ELSE
2032 239 C
2033 240 DST = HYDV( IC , 1 ) * GPERCC
2034 241 VOL = WMX * ( 1. - DST / FSX ) / DST / XGX
2035 242 EMEO = HYDV( IC , 8 ) / HYDV( IC , 1 ) * WMX / RGAS
2036 243 C
2037 244 IYY = ( EMEO - EMEOX( 3 ) ) / RANGEX + 1
2038 245 IYY = MAXO( 1, MINO( IYY , 47 ) )
2039 246 C
2040 247 K = IYY + 2
2041 248 IYY = IYY
2042 249 . + INT( AMAX1( EMEO - EMEOX( K ) , 0. ) / DYX( K ) )
2043 250 . - INT( AMAX1( EMEOX( K + 1 ) - EMEO , 0. ) / DYX( K ) )
2044 251 IYY = MAXO( 1, MINO( IYY , 47 ) )
2045 252 C
2046 253 K1 = IYY + 2
2047 254 K2 = K1 + 1
2048 255 RT = ( EMEO - EMEOX( K1 ) ) / ( EMEOX( K2 ) - EMEOX( K1 ) )
2049 256 T = TX( K1 ) + 100. * RT
2050 257 CVM = CVM( K1 ) + RT * ( CVM( K2 ) - CVM( K1 ) )
2051 258 ERS = 0.
2052 259 C
2053 260 10 CONTINUE
2054 261 P = RGAS * T / VOL / GPERCC
2055 262 RGAMM1 = CVM
2056 263 C
2057 264 X = COVX / VOL / ( ( T + THETAX ) ** ALFAX )
2058 265 Z = X * EXP( BETAX * X )
2059 266 X = 1. + BETAX * X
2060 267 RT = ALFAX * T / ( T + THETAX )
2061 268 ERS = ERS + RT * Z * T
2062 269 C
2063 270 IF ( ITER .EQ. NITER ) GO TO 20
2064 271 C
2065 272 CVM = CVM * XGX + SCVX
2066 273 * + RT * Z * ( 2. - RT / ALFAX - RT * X )
2067 274 T = T - AMINI( ERS / CVM , TLIMIT * T )
2068 275 C
2069 276 NITER = NITER + 1
2070 277 C
2071 278 RT = 0.01 * T
2072 279 K1 = RT
2073 280 K1 = MINO ( K1, 49 )
2074 281 K1 = MAXO ( K1, 3 )
2075 282 K2 = K1 + 1
2076 283 RT = RT - K1
2077 284 CVM = CVM( K1 ) + RT * ( CVM( K2 ) - CVM( K1 ) )
2078 285 ERS = EMEOX( K1 ) + RT * ( EMEOX( K2 ) - EMEOX( K1 ) )
2079 286 ERS = ERS - EMEO
2080 287 C
2081 288 GO TO 10
2082 289 C
2083 290 20 CONTINUE
2084 291 P = P * ( 1. + Z )
2085 292 RGAMM1 = ( RGAMM1 +
2086 293 * RT * Z * ( 2. - RT / ALFAX - RT * X ) ) / ( 1. + Z )
2087 294 X = X * Z / ( 1. + Z )
2088 295 RGAMM1 = RGAMM1 / ( ( 1. - RT * X ) ** 2 + X * RGAMM1 )
2089 296 ERS = ERS / EMEO

```

```

2090 297 HYDV( IC , 7 ) = 1. / RGAMM1 + 1.
2091 298 HYDV( IC , 5 ) = P
2092 299 END IF
2093 300 END DO
2094 301 C
2095 302 ELSE
2096 303 C
2097 304 DO IC = NC1 , NC2
2098 305 HYDV( IC , 5 ) = HYDV( IC , 8 ) * ( HYDV( IC , 7 ) - 1. )
2099 306 END DO
2100 307 END IF
2101 308 C
2102 309 NC1 = NC2 + 1
2103 310 NC2 = NC2 + NOFVEC( INC + 1 )
2104 311 110 CONTINUE
2105 312 C
2106 313 IF( NPRTCL . NE . 0 ) CALL KYDRFL
2107 314 IJKNUM = IJKNUM + 1
2108 315 WRITE(10,*) TT,(PR(KKJJ),KKJJ=1,NPRTCL)
2109 316 140 CONTINUE
2110 317 C
2111 318 PMAX = -10000000.
2112 319 DO 415 IC = 1 , NC
2113 320 IV1 = JC( 1 , IC )
2114 321 IV2 = JC( 2 , IC )
2115 322 IV3 = JC( 3 , IC )
2116 323 IV4 = JC( 4 , IC )
2117 324 HNUMM = HYDV( IC , 5 )
2118 325 HNUMN = XC( 4 , IC )
2119 326 HNUM( IV1 , 5 ) = HNUM( IV1 , 5 ) + HNUMM * HNUMN
2120 327 HNUM( IV1 , 1 ) = HNUM( IV1 , 1 ) + HNUMM
2121 328 HNUM( IV2 , 5 ) = HNUM( IV2 , 5 ) + HNUMM * HNUMN
2122 329 HNUM( IV2 , 1 ) = HNUM( IV2 , 1 ) + HNUMM
2123 330 HNUM( IV3 , 5 ) = HNUM( IV3 , 5 ) + HNUMM * HNUMN
2124 331 HNUM( IV3 , 1 ) = HNUM( IV3 , 1 ) + HNUMM
2125 332 HNUM( IV4 , 5 ) = HNUM( IV4 , 5 ) + HNUMM * HNUMN
2126 333 HNUM( IV4 , 1 ) = HNUM( IV4 , 1 ) + HNUMM
2127 334 415 CONTINUE
2128 335 DO IV = 1 , NV
2129 336 HNUM( IV , 5 ) = HNUM( IV , 5 ) / HNUM( IV , 1 )
2130 337 END DO
2131 338 DO IV = 1 , NV
2132 339 IF( HNUM( IV , 5 ) .GT. PMAX ) PMAX = HNUM( IV , 5 )
2133 340 END DO
2134 341 PRINT * , PMAX
2135 342 C
2136 343 ISNS = 0
2137 344 DO 300 IS = 1 , NS
2138 345 IF(JS(9,IS).EQ.6.AND.XS(2,IS).LT.1.9649) THEN
2139 346 ISNS=ISNS+1
2140 347 ISURF(ISNS)=IS
2141 348 END IF
2142 349 300 CONTINUE
2143 350 print*,ISNS
2144 351 C
2145 352 C STEVE FORMAT
2146 353 C
2147 354 DO 312 IV = 1 , NV
2148 355 WRITE(17,1001) IV,(XV(KK,IV),KK=1,3)
2149 356 1001 FORMAT('n,',IS,',',2(F10.5,','),F10.5)
2150 357 312 CONTINUE
2151 358 C
2152 359 DO 322 IS = 1 , ISNS
2153 360 IK=ISURF(IS)
2154 361 WRITE(18,1002) IS,(JS(KK,IK),KK=1,3),JS(3,IK)
2155 362 1002 FORMAT('en,',4(I10.','),I10)
2156 363 322 CONTINUE
2157 364 C
2158 365 WRITE(19,1005) TT
2159 366 1005 FORMAT('time.',E13.5)
2160 367 ITWO = 1
2161 368 IZERO = 0
2162 369 DO 342 IS = 1 , ISNS
2163 370 IK=ISURF(IS)

```

2164	371	WRITE(19,1003) IS,ITWO,IZERO,HNUM(JS(1,IK),5),HNUM(JS(2,IK),5),	2164
2165	372	HNUM(JS(3,IK),5),HNUM(JS(3,IK),5)	2165
2166	373	1003 FORMAT('sfe.',2(I5,' '), 'pres.',I5,' ',3(E12.5,' '),E12.5)	2166
2167	374	342 CONTINUE	2167
2168	375	C	2168
2169	376	WRITE(14,10101) 3*ISNS,ISNS,NDUMMY1,NDUMMY3,NDUMMY3	2169
2170	377	10101 FORMAT(5I8)	2170
2171	378	10102 FORMAT(I8,3E20.7)	2171
2172	379	10103 FORMAT(2I8,A6,3I8)	2172
2173	380	10104 FORMAT(I8,E20.7)	2173
2174	381	CALL RYDRFL	2174
2175	382	KKVV = 0	2175
2176	383	DO 310 IV = 1, ISNS	2176
2177	384	IK=ISURF(IV)	2177
2178	385	IV1 = JS(1,IK)	2178
2179	386	IV2 = JS(2,IK)	2179
2180	387	IV3 = JS(3,IK)	2180
2181	388	XXV = XV(1,IV1)	2181
2182	389	YYV = XV(2,IV1)	2182
2183	390	ZZV = XV(3,IV1)	2183
2184	391	XNN = -XN(IK)	2184
2185	392	YNN = -YN(IK)	2185
2186	393	ZNN = -ZN(IK)	2186
2187	394	XXX = XXV + XNN * .001	2187
2188	395	YYY = YYV + YNN * .001	2188
2189	396	ZZZ = ZZV + ZNN * .001	2189
2190	397	KKVV = KKVV + 1	2190
2191	398	WRITE(14,10102) KKVV,XXX,YYY,ZZZ	2191
2192	399	XXV = XV(1,IV2)	2192
2193	400	YYV = XV(2,IV2)	2193
2194	401	ZZV = XV(3,IV2)	2194
2195	402	XXX = XXV + XNN * .001	2195
2196	403	YYY = YYV + YNN * .001	2196
2197	404	ZZZ = ZZV + ZNN * .001	2197
2198	405	KKVV = KKVV + 1	2198
2199	406	WRITE(14,10102) KKVV,XXX,YYY,ZZZ	2199
2200	407	XXV = XV(1,IV3)	2200
2201	408	YYV = XV(2,IV3)	2201
2202	409	ZZV = XV(3,IV3)	2202
2203	410	XXX = XXV + XNN * .001	2203
2204	411	YYY = YYV + YNN * .001	2204
2205	412	ZZZ = ZZV + ZNN * .001	2205
2206	413	KKVV = KKVV + 1	2206
2207	414	WRITE(14,10102) KKVV,XXX,YYY,ZZZ	2207
2208	415	310 CONTINUE	2208
2209	416	KKVV = 0	2209
2210	417	DO 320 IS = 1, ISNS	2210
2211	418	IK=ISURF(IS)	2211
2212	419	WRITE(14,10103) IS,IS,CTRI,KKVV+1,KKVV+2,KKVV+3	2212
2213	420	KKVV = KKVV + 3	2213
2214	421	320 CONTINUE	2214
2215	422	WRITE(14,10101) VDATA	2215
2216	423	WRITE(14,*) VLABEL	2216
2217	424	KKVV = 0	2217
2218	425	DO 430 IV = 1, ISNS	2218
2219	426	IK=ISURF(IV)	2219
2220	427	PRR = PR(IK)	2220
2221	428	WRITE(14,10104) KKVV+1,PRR	2221
2222	429	WRITE(14,10104) KKVV+2,PRR	2222
2223	430	WRITE(14,10104) KKVV+3,PRR	2223
2224	431	KKVV = KKVV + 3	2224
2225	432	430 CONTINUE	2225
2226	433	ISNS = 0	2226
2227	434	DO IS = 1, NS	2227
2228	435	IF(JS(9,IS).EQ.6) THEN	2228
2229	436	XXS = XS(1,IS)	2229
2230	437	YYS = XS(2,IS)	2230
2231	438	ZZS = XS(3,IS)	2231
2232	439	ISNS=ISNS+1	2232
2233	440	ISURF(ISNS)=IS	2233
2234	441	END IF	2234
2235	442	END DO	2235
2236	443	print*, ISNS	2236
2237	444	WRITE(15,10101) 3*ISNS,ISNS,NDUMMY1,NDUMMY3,NDUMMY3	2237

```

2238 445      KKVV = 0
2239 446      DO 410 IV = 1 , ISNS
2240 447          IK=ISURF(IV)
2241 448          IV1 = JS(1,IK)
2242 449          IV2 = JS(2,IK)
2243 450          IV3 = JS(3,IK)
2244 451          XNN = -XN(IV)
2245 452          YNN = -YN(IV)
2246 453          ZNN = -ZN(IV)
2247 454          XXV = XV(1,IV1)
2248 455          YV = XV(2,IV1)
2249 456          ZV = XV(3,IV1)
2250 457          XXX = XXV + XNN * .001
2251 458          YYY = YV + YNN * .001
2252 459          ZZZ = ZV + ZNN * .001
2253 460          KKVV = KKVV + 1
2254 461          WRITE(15,10102) KKVV,XXX,YYY,ZZZ
2255 462          XXV = XV(1,IV2)
2256 463          YV = XV(2,IV2)
2257 464          ZV = XV(3,IV2)
2258 465          XXX = XXV + XNN * .001
2259 466          YYY = YV + YNN * .001
2260 467          ZZZ = ZV + ZNN * .001
2261 468          KKVV = KKVV + 1
2262 469          WRITE(15,10102) KKVV,XXX,YYY,ZZZ
2263 470          XXV = XV(1,IV3)
2264 471          YV = XV(2,IV3)
2265 472          ZV = XV(3,IV3)
2266 473          XXX = XXV + XNN * .001
2267 474          YYY = YV + YNN * .001
2268 475          ZZZ = ZV + ZNN * .001
2269 476          KKVV = KKVV + 1
2270 477          WRITE(15,10102) KKVV,XXX,YYY,ZZZ
2271 478 410      CONTINUE
2272 479          KKVV = 0
2273 480          DO 420 IS = 1 , ISNS
2274 481              IK=ISURF(IS)
2275 482              WRITE(15,10103) IS,IS,CTRI,KKVV+1,KKVV+2,KKVV+3
2276 483              KKVV = KKVV + 3
2277 484 420      CONTINUE
2278 485          WRITE(15,10101) VDATA
2279 486          WRITE(15,*) VLABEL
2280 487          KKVV = 0
2281 488          DO 330 IV = 1 , ISNS
2282 489              IK=ISURF(IV)
2283 490              PRR = PR(IV)
2284 491              WRITE(15,10104) KKVV+1,PRR
2285 492              WRITE(15,10104) KKVV+2,PRR
2286 493              WRITE(15,10104) KKVV+3,PRR
2287 494              KKVV = KKVV + 3
2288 495 330      CONTINUE
2289 496          C
2290 497          C
2291 498          C-----
2292 499          C
2293 500          C          I-----I
2294 501          C          I  OUTPUT FILE FOR RESTARTS  I
2295 502          C          I-----I
2296 503          C
2297 504          IF( JT . EQ . 1 ) THEN
2298 505              REWIND 8
2299 506              WRITE(9) NV,NE,NS,NC,NTIME
2300 507              WRITE(9) ((XV(IK,IV),IK=1,3),IV=1,NV)
2301 508              WRITE(9) ((JE(KK,IE),KK=1,2),IE=1,NE)
2302 509              WRITE(9) ((JS(KK,IS),KK=1,9),(XS(KI,IS),KI=1,5),
2303 510                  .      XN(IS),YN(IS),ZN(IS),XP(IS),YP(IS),ZP(IS),
2304 511                  .      XT(IS),YT(IS),ZT(IS),IS=1,NS)
2305 512              WRITE(9) ((XYZMDL(KI,IS),KI=1,4),IS=1,NS)
2306 513              WRITE(9) ((JC(KK,IC),KK=1,8),(XC(KI,IC),KI=1,4),IC=1,NC)
2307 514              WRITE(9) ((RGRAD(IC,KI),UGRAD(IC,KI),VGRAD(IC,KI),
2308 515                  .      WGRAD(IC,KI),PGRAD(IC,KI),KI=1,3),IC=1,NC)
2309 516              WRITE(9) SAREVG,
2310 517                  .      NVECE,NREME,NVFCV,NREMV,NVECS,NREMS,NVECC,NREMC
2311 518              WRITE(9) RIN,PIN,RINL,PINL,UVIN,UVIN,WIN,WIN,TT

```

```

2312 519      WRITE(9) NPRTCL
2313 520      IF(NPRTCL.GT.0)
2314 521      . WRITE(9) (IJKPRT(IK),IK=1,NPRTCL)
2315 522      END IF
2316 523      WRITE(9) ((HYDV(IC,IK),IK=1,8),IC=1,NC)
2317 524      C
2318 525      REWIND 88
2319 526      WRITE(88) NV,NE,NS,NC,NTIME
2320 527      WRITE(88) ((XV(IK,IV),IK=1,3),IV=1,NV)
2321 528      WRITE(88) ((JE(KK,IE),KK=1,2),IE=1,NE)
2322 529      WRITE(88) ((JS(KK,IS),KK=1,9),(XS(KI,IS),KI=1,5),
2323 530      .      XN(IS),YN(IS),ZN(IS),XP(IS),YP(IS),ZP(IS),
2324 531      .      XT(IS),YT(IS),ZT(IS),IS=1,NS)
2325 532      WRITE(88) ((XYZMDL(KI,IS),KI=1,4),IS=1,NS)
2326 533      WRITE(88) ((JC(KK,IC),KK=1,8),(XC(KI,IC),KI=1,4),IC=1,NC)
2327 534      WRITE(88) ((RGRAD(IC,KI),UGRAD(IC,KI),VGRAD(IC,KI),
2328 535      .      WGRAD(IC,KI),PGRAD(IC,KI),KI=1,3),IC=1,NC)
2329 536      WRITE(88) SAREVG,
2330 537      . NVECE,NREME,NVECV,NREMV,NVECS,NREMS,NVECC,NREMC
2331 538      WRITE(88) RIN,PIN,RINL,PINL,UVIN,UIN,VIN,WIN,TT
2332 539      WRITE(88) NPRTCL
2333 540      IF(NPRTCL.GT.0)
2334 541      . WRITE(88) (IJKPRT(IK),IK=1,NPRTCL)
2335 542      WRITE(88) ((HYDV(IC,IK),IK=1,8),IC=1,NC)
2336 543      C
2337 544      120 CONTINUE
2338 545      REWIND 10
2339 546      REWIND 26
2340 547      WRITE(26,*) IJKNUM
2341 548      DO KKJ = 1, IJKNUM
2342 549      READ (10,*) RO,(RRN(IK),IK=1,NPRTCL)
2343 550      WRITE (26,*) RO,(RRN(IK),IK=1,NPRTCL)
2344 551      END DO
2345 552      C
2346 553      RETURN
2347 554      END
2348 555      C

```

```

2349 1      SUBROUTINE GEOMTR
2350 2      C
2351 3      C-----I
2352 4      C I
2353 5      C      GEOMTR COMPUTE THE DUAL MESH AFTER INITIALIZATION THE GRID I
2354 6      C I
2355 7      C-----I
2356 8      C
2357 9      include 'dmsh00.h'
2358 10     include 'dhycm0.h'
2359 11     include 'dphsm0.h'
2360 12     include 'dmtr10.h'
2361 13      C
2362 14      C --- DEFINING BOUNDARY EDGES AND COMPUTING BAR CENTER OF TRIANGLES
2363 15      C
2364 16      PRINT * , NE,NS
2365 17      DO 110 IC = 1 , NC
2366 18      SVOLM( IC ) = 1. / XC( 4 , IC )
2367 19      110 CONTINUE
2368 20      C
2369 21      DO 120 IS = 1 , NS
2370 22      C
2371 23      ICL = JS( 7 , IS )
2372 24      ICR = JS( 8 , IS )
2373 25      C
2374 26      IV1 = JS( 1 , IS )
2375 27      IV2 = JS( 2 , IS )
2376 28      IV3 = JS( 3 , IS )
2377 29      C
2378 30      X1 = XV( 1 , IV1 )
2379 31      Y1 = XV( 2 , IV1 )
2380 32      Z1 = XV( 3 , IV1 )
2381 33      C
2382 34      A = XN( IS )

```



```

2383 35      B = YN( IS )
2384 36      C = ZN( IS )
2385 37      C
2386 38      D = - ( A * X1 + B * Y1 + C * Z1 )
2387 39      C
2388 40      XCL = XC( 1 , ICL )
2389 41      YCL = XC( 2 , ICL )
2390 42      ZCL = XC( 3 , ICL )
2391 43      C
2392 44      DD = A * XCL + B * YCL + C * ZCL + D
2393 45      C
2394 46      IATRB = JS( 9 , IS )
2395 47      IF( IATRB .EQ. 0 ) THEN
2396 48      C
2397 49      XCR = XC( 1 , ICR )
2398 50      YCR = XC( 2 , ICR )
2399 51      ZCR = XC( 3 , ICR )
2400 52      C
2401 53      XX = XCR - XCL
2402 54      YY = YCR - YCL
2403 55      ZZ = ZCR - ZCL
2404 56      C
2405 57      DDD = A * XX + B * YY + C * ZZ
2406 58      C
2407 59      XYZ = - DD / DDD
2408 60      XYZMDL( 4 , IS ) = XYZ
2409 61      XYZMDL( 1 , IS ) = XCL + XYZ * XX
2410 62      XYZMDL( 2 , IS ) = YCL + XYZ * YY
2411 63      XYZMDL( 3 , IS ) = ZCL + XYZ * ZZ
2412 64      XS( 5 , IS ) = SQRT( XX * XX + YY * YY + ZZ * ZZ )
2413 65      C
2414 66      ELSE
2415 67      C
2416 68      XYZ = - DD
2417 69      XYZMDL( 1 , IS ) = XCL + XYZ * A
2418 70      XYZMDL( 2 , IS ) = YCL + XYZ * B
2419 71      XYZMDL( 3 , IS ) = ZCL + XYZ * C
2420 72      XS( 5 , IS ) = ABS( XYZ )
2421 73      XYZMDL( 4 , IS ) = 1.
2422 74      C
2423 75      END IF
2424 76      C
2425 77      120 CONTINUE
2426 78      C
2427 79      RETURN
2428 80      END
2429 81      C
2430 82      SUBROUTINE UPDATE
2431 83      C
2432 84      C-----I
2433 85      C-----I
2434 86      C      UPDATE COMPUTE THE DUAL MESH AFTER INITIALIZATION THE GRID I
2435 87      C-----I
2436 88      C-----I
2437 89      C
2438 90      include 'dmsh00.h'
2439 91      include 'dhyd0.h'
2440 92      include 'dphsm0.h'
2441 93      include 'dntrio.h'
2442 94      C
2443 95      C
2444 96      C      READ IN VERTEX INFORMATION
2445 97      C
2446 98      READ (16,*) NV,NE,NC,NS
2447 99      DO 1110 IK = 1 , NV
2448 100      READ (16,*) IJ,XV(1,IK),XV(2,IK),XV(3,IK)
2449 101      XXX =XV(1,IK) + 34.5
2450 102      YYY =XV(2,IK) - 65.75
2451 103      ZZZ =XV(3,IK) + 11.5
2452 104      XV(1,IK)=XXX*.0254
2453 105      XV(2,IK)=YYY*.0254
2454 106      XV(3,IK)=ZZZ*.0254
2455 107      1110 CONTINUE
2456 108      PRINT * , NV

```

```

2457 109 C 2457
2458 110 C READ IN EDGE INFORMATION ( EDGES OF TRIANGLES). 2458
2459 111 C 2459
2460 112 DO 1120 IK = 1 , NE 2460
2461 113 READ (16,*) IJ,JE(1,IK),JE(2,IK) 2461
2462 114 1120 CONTINUE 2462
2463 115 PRINT * , NE 2463
2464 116 C 2464
2465 117 C READ IN CELL (TETRAHIDRAL) INFORMATION. 2465
2466 118 C 2466
2467 119 C CELL INFORMATION, FOR EACH CELL FOUR VERTICES 2467
2468 120 C 2468
2469 121 DO 1130 IK = 1 , NC 2469
2470 122 READ (16,*) IJ,JC(1,IK),JC(2,IK),JC(3,IK),JC(4,IK) 2470
2471 123 1130 CONTINUE 2471
2472 124 C 2472
2473 125 DO 1200 IK = 1 , NC 2473
2474 126 IV1 = JC( 1 , IK ) 2474
2475 127 IV2 = JC( 2 , IK ) 2475
2476 128 IV3 = JC( 3 , IK ) 2476
2477 129 IV4 = JC( 4 , IK ) 2477
2478 130 C 2478
2479 131 C SIDE INFORMATION, FOR EACH CELL CENTROID OF CELL 2479
2480 132 C 2480
2481 133 XC( 1 , IK ) = ( XV( 1 , IV1 ) + XV( 1 , IV2 ) + 2481
2482 134 XV( 1 , IV3 ) + XV( 1 , IV4 ) ) * .25 2482
2483 135 XC( 2 , IK ) = ( XV( 2 , IV1 ) + XV( 2 , IV2 ) + 2483
2484 136 XV( 2 , IV3 ) + XV( 2 , IV4 ) ) * .25 2484
2485 137 XC( 3 , IK ) = ( XV( 3 , IV1 ) + XV( 3 , IV2 ) + 2485
2486 138 XV( 3 , IV3 ) + XV( 3 , IV4 ) ) * .25 2486
2487 139 C 2487
2488 140 C SIDE INFORMATION, FOR EACH CELL VOLUME OF CELL 2488
2489 141 C 2489
2490 142 XPIJ = XV( 1 , IV2 ) - XV( 1 , IV1 ) 2490
2491 143 YPIJ = XV( 2 , IV2 ) - XV( 2 , IV1 ) 2491
2492 144 ZPIJ = XV( 3 , IV2 ) - XV( 3 , IV1 ) 2492
2493 145 C 2493
2494 146 XPIK = XV( 1 , IV3 ) - XV( 1 , IV1 ) 2494
2495 147 YPIK = XV( 2 , IV3 ) - XV( 2 , IV1 ) 2495
2496 148 ZPIK = XV( 3 , IV3 ) - XV( 3 , IV1 ) 2496
2497 149 C 2497
2498 150 XNIK = YPIJ * ZPIK - ZPIJ * YPIK 2498
2499 151 YNIK = ZPIJ * XPIK - XPIJ * ZPIK 2499
2500 152 ZNIK = XPIJ * YPIK - YPIJ * XPIK 2500
2501 153 C 2501
2502 154 XPIJ = XV( 1 , IV4 ) - XV( 1 , IV1 ) 2502
2503 155 YPIJ = XV( 2 , IV4 ) - XV( 2 , IV1 ) 2503
2504 156 ZPIJ = XV( 3 , IV4 ) - XV( 3 , IV1 ) 2504
2505 157 C 2505
2506 158 VOL = ( XNIK * XPIJ + YNIK * YPIJ + 2506
2507 159 ZNIK * ZPIJ ) / 6. 2507
2508 160 XC( 4 , IK ) = VOL 2508
2509 161 IF( VOL .LT. 0. ) PRINT * ,IK,VOL 2509
2510 162 1200 CONTINUE 2510
2511 163 PRINT * , NC 2511
2512 164 C 2512
2513 165 C READ IN SIDE (TRIANGLE) INFORMATION. 2513
2514 166 C 2514
2515 167 C SIDE INFORMATION, FOR EACH FACE THREE VERTICES 2515
2516 168 C 2516
2517 169 DO 1150 IK = 1 , NS 2517
2518 170 READ (16,*) IJ,JS(1,IK),JS(2,IK),JS(3,IK) 2518
2519 171 1150 CONTINUE 2519
2520 172 PRINT * , NS,NC 2520
2521 173 C 2521
2522 174 C SIDE INFORMATION, FOR EACH FACE THREE EDGES 2522
2523 175 C 2523
2524 176 DO 1160 IK = 1 , NS 2524
2525 177 READ (16,*) IJ,JS(4,IK),JS(5,IK),JS(6,IK) 2525
2526 178 1160 CONTINUE 2526
2527 179 PRINT * , NS,NC,NV 2527
2528 180 C 2528
2529 181 C CELL INFORMATION, FOR EACH CELL FOUR EDGES 2529
2530 182 C 2530

```

```

2531 183      DO 1140 IK = 1 , NC
2532 184      READ (16,*) IJ,JC5,IDIR1,JC6,IDIR2,
2533 185      JC7,IDIR3,JC8,IDIR4
2534 186      JC(5,IK) = IABS( JC5 )
2535 187      JC(6,IK) = IABS( JC6 )
2536 188      JC(7,IK) = IABS( JC7 )
2537 189      JC(8,IK) = IABS( JC8 )
2538 190      1140 CONTINUE
2539 191      PRINT * , NS,NC,NV,NE
2540 192      C
2541 193      C SIDE INFORMATION, FOR EACH FACE LEFT AND RIGHT TETREHEDRA
2542 194      C
2543 195      DO 1170 IK = 1 , NS
2544 196      READ (16,*) IJ,JS(7,IK),JS(8,IK)
2545 197      JS( 9 , IK ) = 0
2546 198      1170 CONTINUE
2547 199      PRINT * , NC,NV,NE
2548 200      C
2549 201      C SIDE INFORMATION, FOR EACH FACE BOUNDARY CONDITION
2550 202      C
2551 203      1180 CONTINUE
2552 204      READ (16,*,END=1210) IJ,IDUMY,JS(9,IJ)
2553 205      GO TO 1180
2554 206      1210 CONTINUE
2555 207      PRINT * , NV,NE,NS,NC
2556 208      C
2557 209      DO 1190 IK = 1 , NS
2558 210      IV1 = JS( 1 , IK )
2559 211      IV2 = JS( 2 , IK )
2560 212      IV3 = JS( 3 , IK )
2561 213      C
2562 214      C SIDE INFORMATION, FOR EACH FACE TANGENTIAL VECTOR
2563 215      C
2564 216      XP( IK ) = XV( 1 , IV2 ) - XV( 1 , IV1 )
2565 217      YP( IK ) = XV( 2 , IV2 ) - XV( 2 , IV1 )
2566 218      ZP( IK ) = XV( 3 , IV2 ) - XV( 3 , IV1 )
2567 219      XPDUMY = XV( 1 , IV3 ) - XV( 1 , IV1 )
2568 220      YPDUMY = XV( 2 , IV3 ) - XV( 2 , IV1 )
2569 221      ZPDUMY = XV( 3 , IV3 ) - XV( 3 , IV1 )
2570 222      C
2571 223      C SIDE INFORMATION, FOR EACH FACE NORMAL UNIT VECTOR
2572 224      C
2573 225      XN( IK ) = YP( IK ) * ZPDUMY - ZP( IK ) * YPDUMY
2574 226      YN( IK ) = ZP( IK ) * XPDUMY - XP( IK ) * ZPDUMY
2575 227      ZN( IK ) = XP( IK ) * YPDUMY - YP( IK ) * XPDUMY
2576 228      C
2577 229      C SIDE INFORMATION, FOR EACH FACE TANGENTIAL VECTOR
2578 230      C
2579 231      XT( IK ) = - YP( IK ) * ZN( IK ) + ZP( IK ) * YN( IK )
2580 232      YT( IK ) = - ZP( IK ) * XN( IK ) + XP( IK ) * ZN( IK )
2581 233      ZT( IK ) = - XP( IK ) * YN( IK ) + YP( IK ) * XN( IK )
2582 234      C
2583 235      XYZDUM = XN( IK ) * XN( IK ) + YN( IK ) * YN( IK ) + ZN( IK ) * ZN( IK )
2584 236      IF( XYZDUM.EQ.0.) PRINT * , IK
2585 237      XYZDUM = 1. / SQRT( XYZDUM )
2586 238      C
2587 239      C SIDE INFORMATION, FOR EACH FACE AREA OF FACE
2588 240      C
2589 241      XS( 4 , IK ) = .5 / XYZDUM
2590 242      C
2591 243      C SIDE INFORMATION, FOR EACH FACE CENTROID OF FACE
2592 244      C
2593 245      XS( 1 , IK ) = ( XV( 1 , IV1 ) + XV( 1 , IV2 ) +
2594 246      XV( 1 , IV3 ) ) / 3.
2595 247      XS( 2 , IK ) = ( XV( 2 , IV1 ) + XV( 2 , IV2 ) +
2596 248      XV( 2 , IV3 ) ) / 3.
2597 249      XS( 3 , IK ) = ( XV( 3 , IV1 ) + XV( 3 , IV2 ) +
2598 250      XV( 3 , IV3 ) ) / 3.
2599 251      XN( IK ) = XN( IK ) * XYZDUM
2600 252      YN( IK ) = YN( IK ) * XYZDUM
2601 253      ZN( IK ) = ZN( IK ) * XYZDUM
2602 254      XYZDUM = XP( IK ) * XP( IK ) + YP( IK ) * YP( IK ) + ZP( IK ) * ZP( IK )
2603 255      XYZDUM = 1. / SQRT( XYZDUM )
2604 256      XP( IK ) = XP( IK ) * XYZDUM

```

2605	257		YP(IK) = YP(IK) * XYZDUM	2605
2606	258		ZP(IK) = ZP(IK) * XYZDUM	2606
2607	259		XYZDUM = XT(IK) * XT(IK) + YT(IK) * YT(IK) + ZT(IK) * ZT(IK)	2607
2608	260		XYZDUM = 1. / SORT(XYZDUM)	2608
2609	261		XT(IK) = XT(IK) * XYZDUM	2609
2610	262		YT(IK) = YT(IK) * XYZDUM	2610
2611	263		ZT(IK) = ZT(IK) * XYZDUM	2611
2612	264	1190	CONTINUE	2612
2613	265		PRINT * , NS	2613
2614	266	C		2614
2615	267		NVECV = NV / 128	2615
2616	268		NREMV = NV - NVECV * 128	2616
2617	269		NVECE = NE / 128	2617
2618	270		NREME = NE - NVECE * 128	2618
2619	271		NVECS = NS / 128	2619
2620	272		NREMS = NS - NVECS * 128	2620
2621	273		NVECC = NC / 128	2621
2622	274		NREMC = NC - NVECC * 128	2622
2623	275	C		2623
2624	276		DO 125 INV = 1 , NVECV	2624
2625	277		NOFVEV(INV) = 128	2625
2626	278	125	CONTINUE	2626
2627	279		NVEEV = NVECV	2627
2628	280		IF(NREMV . GT . 0) THEN	2628
2629	281		NVEEV = NVEEV + 1	2629
2630	282		NOFVEV(NVEEV) = NREMV	2630
2631	283		END IF	2631
2632	284	C		2632
2633	285		DO 105 INE = 1 , NVECE	2633
2634	286		NOFVEE(INE) = 128	2634
2635	287	105	CONTINUE	2635
2636	288		NVEEE = NVECE	2636
2637	289		IF(NREME . GT . 0) THEN	2637
2638	290		NVEEE = NVEEE + 1	2638
2639	291		NOFVEE(NVEEE) = NREME	2639
2640	292		END IF	2640
2641	293	C		2641
2642	294		DO 115 INS = 1 , NVECS	2642
2643	295		NOFVES(INS) = 128	2643
2644	296	115	CONTINUE	2644
2645	297		NVEES = NVECS	2645
2646	298		IF(NREMS . GT . 0) THEN	2646
2647	299		NVEES = NVEES + 1	2647
2648	300		NOFVES(NVEES) = NREMS	2648
2649	301		END IF	2649
2650	302	C		2650
2651	303		DO 135 INC = 1 , NVECC	2651
2652	304		NOFVEC(INC) = 128	2652
2653	305	135	CONTINUE	2653
2654	306		NVEEC = NVECC	2654
2655	307		IF(NREMC . GT . 0) THEN	2655
2656	308		NVEEC = NVEEC + 1	2656
2657	309		NOFVEC(NVEEC) = NREMC	2657
2658	310		END IF	2658
2659	311	C		2659
2660	312		PRINT * , NV , NE , NS , NC	2660
2661	313		PRINT * , NVEEV , NVEEE , NVEES , NVEEC	2661
2662	314		PRINT * , NREMV , NREME , NREMS , NREMC	2662
2663	315	1001	FORMAT(4I7)	2663
2664	316	1002	FORMAT(17,3E20.12)	2664
2665	317	C		2665
2666	318		CALL GEOMTR	2666
2667	319	C		2667
2668	320		RETURN	2668
2669	321		END	2669
2670	322	C		2670

```

2671 1      SUBROUTINE UPGRAD                                2671
2672 2      C                                                2672
2673 3      C-----I                                         2673
2674 4      C                                                I                                         2674
2675 5      C      UPGRAD COMPUTE THE DUAL MESH AFTER ADAPTING THE GRID  I                                         2675
2676 6      C-----I                                         2676
2677 7      C-----I                                         2677
2678 8      C                                                2678
2679 9      include 'dmsh00.h'                                2679
2680 10     include 'dhyd00.h'                                2680
2681 11     include 'dphsm0.h'                               2681
2682 12     include 'dmtri0.h'                               2682
2683 13     C                                                2683
2684 14     REAL XELEFT(128),YELEFT(128),XERIGT(128),YERIGT(128) 2684
2685 15     C                                                2685
2686 16     C --- DEFINING BOUNDARY EDGES                      2686
2687 17     C                                                2687
2688 18     READ(8) NV,NE,NS,NC,NTINE                          2688
2689 19     READ(8) ((XV(IK,IV),IK=1,3),IV=1,NV)              2689
2690 20     READ(8) ((JE(KK,IE),KK=1,2),IE=1,NE)              2690
2691 21     READ(8) ((JS(KK,IS),KK=1,9),(XS(KI,IS),KI=1,5),    2691
2692 22     .          XN(IS),YN(IS),ZN(IS),XP(IS),YP(IS),ZP(IS),  2692
2693 23     .          XT(IS),YT(IS),ZT(IS),IS=1,NS)           2693
2694 24     READ(8) ((XYZMDL(KI,IS),KI=1,4),IS=1,NS)          2694
2695 25     READ(8) ((JC(KK,IC),KK=1,8),(XC(KI,IC),KI=1,4),IC=1,NC) 2695
2696 26     READ(8) ((RGRAD(IC,KI),UGRAD(IC,KI),VGRAD(IC,KI),  2696
2697 27     .          WGRAD(IC,KI),PGRAD(IC,KI),KI=1,3),IC=1,NC) 2697
2698 28     READ(8) SAREVG,                                     2698
2699 29     .          NVECE,NREME,NVECV,NREMV,NVECS,NREMS,NVECC,NREMC 2699
2700 30     PRINT *, NE,NS                                     2700
2701 31     C                                                2701
2702 32     DO 100 IC = 1 , NC                                  2702
2703 33     SVOLM( IC ) = 1. / XC( 4 , IC )                    2703
2704 34     100 CONTINUE                                       2704
2705 35     C                                                2705
2706 36     DO 105 INE = 1 , NVECE                              2706
2707 37     NOFVEE( INE ) = 128                                2707
2708 38     105 CONTINUE                                       2708
2709 39     NVEEE = NVECE                                       2709
2710 40     IF( NREME . GT . 0 ) THEN                          2710
2711 41     NVEEE = NVEEE + 1                                  2711
2712 42     NOFVEE( NVEEE ) = NREME                          2712
2713 43     END IF                                             2713
2714 44     C                                                2714
2715 45     DO 115 INS = 1 , NVECS                              2715
2716 46     NOFVES( INS ) = 128                                2716
2717 47     115 CONTINUE                                       2717
2718 48     NVEES = NVECS                                       2718
2719 49     IF( NREMS . GT . 0 ) THEN                          2719
2720 50     NVEES = NVEES + 1                                  2720
2721 51     NOFVES( NVEES ) = NREMS                          2721
2722 52     END IF                                             2722
2723 53     C                                                2723
2724 54     DO 125 INV = 1 , NVECV                              2724
2725 55     NOFVEV( INV ) = 128                                2725
2726 56     125 CONTINUE                                       2726
2727 57     NVEEV = NVECV                                       2727
2728 58     IF( NREMV . GT . 0 ) THEN                          2728
2729 59     NVEEV = NVEEV + 1                                  2729
2730 60     NOFVEV( NVEEV ) = NREMV                          2730
2731 61     END IF                                             2731
2732 62     C                                                2732
2733 63     DO 135 INC = 1 , NVECC                              2733
2734 64     NOFVEC( INC ) = 128                                2734
2735 65     135 CONTINUE                                       2735
2736 66     NVEEC = NVECC                                       2736
2737 67     IF( NREMC . GT . 0 ) THEN                          2737
2738 68     NVEEC = NVEEC + 1                                  2738
2739 69     NOFVEC( NVEEC ) = NREMC                          2739
2740 70     END IF                                             2740
2741 71     C                                                2741
2742 72     PRINT *, NV,NE,NS,NC,NVECV,NREMV,NVECE,NREME,NVECS,NREMS, 2742
2743 73     NVECC,NREMC                                         2743
2744 74     C                                                2744

```

2745 75 RETURN
 2746 76 END
 2747 77 C

2745
 2746
 2747

```

2748 1 SUBROUTINE GRADNT 2748
2749 2 C 2749
2750 3 C-----I 2750
2751 4 C I 2751
2752 5 C GRADNT COMPUTE THE GRADIENT FOR SECOND ORDER CALCULATION I 2752
2753 6 C I 2753
2754 7 C-----I 2754
2755 8 C 2755
2756 9 include 'dmsh00.h' 2756
2757 10 include 'dhyd0.h' 2757
2758 11 include 'dphsm0.h' 2758
2759 12 include 'dmtr10.h' 2759
2760 13 C 2760
2761 14 REAL RRMIDL(128),PPMIDL(128),UUMIDL(128),VVMIDL(128), 2761
2762 15 WMIDL(128),AAMIDL(128) 2762
2763 16 REAL RIGRAD(128),PIGRAD(128),UIGRAD(128),VIGRAD(128), 2763
2764 17 WIGRAD(128),AIGRAD(128) 2764
2765 18 REAL RJGRAD(128),PJGRAD(128),UJGRAD(128),VJGRAD(128), 2765
2766 19 WJGRAD(128),AJGRAD(128) 2766
2767 20 REAL RKGRAD(128),PKGRAD(128),UKGRAD(128),VKGRAD(128), 2767
2768 21 WKGRAD(128),AKGRAD(128) 2768
2769 22 REAL RMAX(128),PMAX(128),UMAX(128),VMAX(128),WMAX(128), 2769
2770 23 AMAX(128) 2770
2771 24 REAL RMIN(128),PMIN(128),UMIN(128),VMIN(128),WMIN(128), 2771
2772 25 AMIN(128) 2772
2773 26 REAL ROR(4),UOR(4),VOR(4),WOR(4),POR(4),AOR(4) 2773
2774 27 REAL ROL(4),UOL(4),VOL(4),WOL(4),POL(4),AOL(4) 2774
2775 28 C 2775
2776 29 DO 120 IH = 1, 3 2776
2777 30 C 2777
2778 31 DO 120 IC = 1, NC 2778
2779 32 C 2779
2780 33 RGRAD( IC, IH ) = 0. 2780
2781 34 UGRAD( IC, IH ) = 0. 2781
2782 35 VGRAD( IC, IH ) = 0. 2782
2783 36 WGRAD( IC, IH ) = 0. 2783
2784 37 PGRAD( IC, IH ) = 0. 2784
2785 38 C 2785
2786 39 120 CONTINUE 2786
2787 40 C 2787
2788 41 C --- BEGIN LOOP OVER ALL EDGES IN THE DOMAIN ----- 2788
2789 42 C 2789
2790 43 NS1 = 1 2790
2791 44 NS2 = NOFVES( 1 ) 2791
2792 45 DO 90 INS = 1, NVEES 2792
2793 46 C 2793
2794 47 C --- FETCH HYDRO QUANTITIES ----- 2794
2795 48 C 2795
2796 49 DO 105 IS = NS1, NS2 2796
2797 50 KS = IS - NS1 + 1 2797
2798 51 C 2798
2799 52 ICL = JS( 7, IS ) 2799
2800 53 ICR = JS( 8, IS ) 2800
2801 54 C 2801
2802 55 IATRB = JS( 9, IS ) 2802
2803 56 IF( IATRB .EQ. 0 ) THEN 2803
2804 57 C 2804
2805 58 XYZ = XYZMDL( 4, IS ) 2805
2806 59 RRMIDL( KS ) = HYDV( ICL, 1 ) + XYZ * ( HYDV( ICR, 1 ) - 2806
2807 60 HYDV( ICL, 1 ) ) 2807
2808 61 UUMIDL( KS ) = HYDV( ICL, 2 ) + XYZ * ( HYDV( ICR, 2 ) - 2808
2809 62 HYDV( ICL, 2 ) ) 2809
2810 63 VVMIDL( KS ) = HYDV( ICL, 3 ) + XYZ * ( HYDV( ICR, 3 ) - 2810
2811 64 HYDV( ICL, 3 ) ) 2811
2812 65 WMIDL( KS ) = HYDV( ICL, 4 ) + XYZ * ( HYDV( ICR, 4 ) - 2812
2813 66 HYDV( ICL, 4 ) ) 2813
2814 67 PPMIDL( KS ) = HYDV( ICL, 5 ) + XYZ * ( HYDV( ICR, 5 ) - 2814
2815 68 HYDV( ICL, 5 ) ) 2815

```

```

2816 69 C
2817 70 ELSE
2818 71 C
2819 72 RRMIDL( KS ) = HYDV( ICL , 1 )
2820 73 UUMIDL( KS ) = HYDV( ICL , 2 )
2821 74 VVMIDL( KS ) = HYDV( ICL , 3 )
2822 75 WWMIDL( KS ) = HYDV( ICL , 4 )
2823 76 PPMIDL( KS ) = HYDV( ICL , 5 )
2824 77 C
2825 78 END IF
2826 79 C
2827 80 105 CONTINUE
2828 81 C
2829 82 DO 110 IS = NS1 , NS2
2830 83 KS = IS - NS1 + 1
2831 84 C
2832 85 XEXN = XS( 4 , IS ) * XN( IS )
2833 86 XEYN = XS( 4 , IS ) * YN( IS )
2834 87 XEZN = XS( 4 , IS ) * ZN( IS )
2835 88 C
2836 89 RIGRAD( KS ) = RRMIDL( KS ) * XEXN
2837 90 UIGRAD( KS ) = UUMIDL( KS ) * XEXN
2838 91 VIGRAD( KS ) = VVMIDL( KS ) * XEXN
2839 92 WIGRAD( KS ) = WWMIDL( KS ) * XEXN
2840 93 PIGRAD( KS ) = PPMIDL( KS ) * XEXN
2841 94 C
2842 95 RJGRAD( KS ) = RRMIDL( KS ) * XEYN
2843 96 UJGRAD( KS ) = UUMIDL( KS ) * XEYN
2844 97 VJGRAD( KS ) = VVMIDL( KS ) * XEYN
2845 98 WJGRAD( KS ) = WWMIDL( KS ) * XEYN
2846 99 PJGRAD( KS ) = PPMIDL( KS ) * XEYN
2847 100 C
2848 101 RKGRAD( KS ) = RRMIDL( KS ) * XEZN
2849 102 UKGRAD( KS ) = UUMIDL( KS ) * XEZN
2850 103 VKGRAD( KS ) = VVMIDL( KS ) * XEZN
2851 104 WKGRAD( KS ) = WWMIDL( KS ) * XEZN
2852 105 PKGRAD( KS ) = PPMIDL( KS ) * XEZN
2853 106 C
2854 107 110 CONTINUE
2855 108 C
2856 109 DO 130 IS = NS1 , NS2
2857 110 KS = IS - NS1 + 1
2858 111 C
2859 112 ICL = JS( 7 , IS )
2860 113 ICR = JS( 8 , IS )
2861 114 C
2862 115 RGRAD( ICL , 1 ) = RGRAD( ICL , 1 ) + RIGRAD( KS )
2863 116 RGRAD( ICL , 2 ) = RGRAD( ICL , 2 ) + RJGRAD( KS )
2864 117 RGRAD( ICL , 3 ) = RGRAD( ICL , 3 ) + RKGRAD( KS )
2865 118 UGRAD( ICL , 1 ) = UGRAD( ICL , 1 ) + UIGRAD( KS )
2866 119 UGRAD( ICL , 2 ) = UGRAD( ICL , 2 ) + UJGRAD( KS )
2867 120 UGRAD( ICL , 3 ) = UGRAD( ICL , 3 ) + UKGRAD( KS )
2868 121 VGRAD( ICL , 1 ) = VGRAD( ICL , 1 ) + VIGRAD( KS )
2869 122 VGRAD( ICL , 2 ) = VGRAD( ICL , 2 ) + VJGRAD( KS )
2870 123 VGRAD( ICL , 3 ) = VGRAD( ICL , 3 ) + VKGRAD( KS )
2871 124 WGRAD( ICL , 1 ) = WGRAD( ICL , 1 ) + WIGRAD( KS )
2872 125 WGRAD( ICL , 2 ) = WGRAD( ICL , 2 ) + WJGRAD( KS )
2873 126 WGRAD( ICL , 3 ) = WGRAD( ICL , 3 ) + WKGRAD( KS )
2874 127 PGRAD( ICL , 1 ) = PGRAD( ICL , 1 ) + PIGRAD( KS )
2875 128 PGRAD( ICL , 2 ) = PGRAD( ICL , 2 ) + PJGRAD( KS )
2876 129 PGRAD( ICL , 3 ) = PGRAD( ICL , 3 ) + PKGRAD( KS )
2877 130 C
2878 131 IATRB = JS( 9 , IS )
2879 132 IF( IATRB .EQ. 0 ) THEN
2880 133 C
2881 134 C ... GRADIENT OF DENSITY ( U V W DIRECTION ) .....
2882 135 C
2883 136 RGRAD( ICR , 1 ) = RGRAD( ICR , 1 ) - RIGRAD( KS )
2884 137 RGRAD( ICR , 2 ) = RGRAD( ICR , 2 ) - RJGRAD( KS )
2885 138 RGRAD( ICR , 3 ) = RGRAD( ICR , 3 ) - RKGRAD( KS )
2886 139 C
2887 140 C ... GRADIENT OF U VELOCITY ( U V W DIRECTION ) .....
2888 141 C
2889 142 UGRAD( ICR , 1 ) = UGRAD( ICR , 1 ) - UIGRAD( KS )

```

```

2890 143      UGRAD( ICR , 2 ) = UGRAD( ICR , 2 ) - UJGRAD( KS )      2890
2891 144      UGRAD( ICR , 3 ) = UGRAD( ICR , 3 ) - UKGRAD( KS )      2891
2892 145      C      2892
2893 146      C ... GRADIENT OF V VELOCITY ( U V W DIRECTION ) ..... 2893
2894 147      C      2894
2895 148      VGRAD( ICR , 1 ) = VGRAD( ICR , 1 ) - VIGRAD( KS )      2895
2896 149      VGRAD( ICR , 2 ) = VGRAD( ICR , 2 ) - VJGRAD( KS )      2896
2897 150      VGRAD( ICR , 3 ) = VGRAD( ICR , 3 ) - VKGRAD( KS )      2897
2898 151      C      2898
2899 152      C ... GRADIENT OF W VELOCITY ( U V W DIRECTION ) ..... 2899
2900 153      C      2900
2901 154      WGRAD( ICR , 1 ) = WGRAD( ICR , 1 ) - WIGRAD( KS )      2901
2902 155      WGRAD( ICR , 2 ) = WGRAD( ICR , 2 ) - WJGRAD( KS )      2902
2903 156      WGRAD( ICR , 3 ) = WGRAD( ICR , 3 ) - WKGRAD( KS )      2903
2904 157      C      2904
2905 158      C ... GRADIENT OF PRESSURE ( U V W DIRECTION ) ..... 2905
2906 159      C      2906
2907 160      PGRAD( ICR , 1 ) = PGRAD( ICR , 1 ) - PIGRAD( KS )      2907
2908 161      PGRAD( ICR , 2 ) = PGRAD( ICR , 2 ) - PJGRAD( KS )      2908
2909 162      PGRAD( ICR , 3 ) = PGRAD( ICR , 3 ) - PKGRAD( KS )      2909
2910 163      C      2910
2911 164      END IF      2911
2912 165      C      2912
2913 166      130 CONTINUE      2913
2914 167      C      2914
2915 168      NS1 = NS2 + 1      2915
2916 169      NS2 = NS2 + NOFVES( INS + 1 )      2916
2917 170      90 CONTINUE      2917
2918 171      C      2918
2919 172      DO 140 IH = 1 , 3      2919
2920 173      C      2920
2921 174      DO 140 IC = 1 , NC      2921
2922 175      C      2922
2923 176      RGRAD( IC , IH ) = RGRAD( IC , IH ) * SVOLM( IC )      2923
2924 177      UGRAD( IC , IH ) = UGRAD( IC , IH ) * SVOLM( IC )      2924
2925 178      VGRAD( IC , IH ) = VGRAD( IC , IH ) * SVOLM( IC )      2925
2926 179      WGRAD( IC , IH ) = WGRAD( IC , IH ) * SVOLM( IC )      2926
2927 180      PGRAD( IC , IH ) = PGRAD( IC , IH ) * SVOLM( IC )      2927
2928 181      C      2928
2929 182      140 CONTINUE      2929
2930 183      C      2930
2931 184      NC1 = 1      2931
2932 185      NC2 = NOFVEC( 1 )      2932
2933 186      DO 80 INC = 1 , NVEEC      2933
2934 187      C      2934
2935 188      DO 150 IC = NC1 , NC2      2935
2936 189      KC = IC - NC1 + 1      2936
2937 190      C      2937
2938 191      IS = JC( 5 , IC )      2938
2939 192      C      2939
2940 193      ICL = JS( 7 , IS )      2940
2941 194      ICR = JS( 8 , IS )      2941
2942 195      C      2942
2943 196      RROL = HYDV( ICL , 1 )      2943
2944 197      UUOL = HYDV( ICL , 2 )      2944
2945 198      VVOL = HYDV( ICL , 3 )      2945
2946 199      WWOL = HYDV( ICL , 4 )      2946
2947 200      PPOL = HYDV( ICL , 5 )      2947
2948 201      C      2948
2949 202      IATRB = JS( 9 , IS )      2949
2950 203      IF( IATRB .EQ. 0 ) THEN      2950
2951 204      C      2951
2952 205      RROR = HYDV( ICR , 1 )      2952
2953 206      UUOR = HYDV( ICR , 2 )      2953
2954 207      VVOR = HYDV( ICR , 3 )      2954
2955 208      WWOR = HYDV( ICR , 4 )      2955
2956 209      PPOR = HYDV( ICR , 5 )      2956
2957 210      C      2957
2958 211      ELSE      2958
2959 212      C      2959
2960 213      RROR = RROL      2960
2961 214      UUOR = UUOL      2961
2962 215      VVOR = VVOL      2962
2963 216      WWOR = WWOL      2963

```


2964	217		PPOR = PPOL	2964
2965	218	C		2965
2966	219		END IF	2966
2967	220	C		2967
2968	221		ROL(1) = RROL	2968
2969	222		UOL(1) = UUOL	2969
2970	223		VOL(1) = VVOL	2970
2971	224		WOL(1) = WWOL	2971
2972	225		POL(1) = PPOL	2972
2973	226	C		2973
2974	227		ROR(1) = RROR	2974
2975	228		UOR(1) = UOR	2975
2976	229		VOR(1) = VVOR	2976
2977	230		WOR(1) = WWOR	2977
2978	231		POR(1) = PPOR	2978
2979	232	C		2979
2980	233		IS = JC(6 , IC)	2980
2981	234	C		2981
2982	235		ICL = JS(7 , IS)	2982
2983	236		ICR = JS(8 , IS)	2983
2984	237	C		2984
2985	238		RROL = HYDV(ICL , 1)	2985
2986	239		UUOL = HYDV(ICL , 2)	2986
2987	240		VVOL = HYDV(ICL , 3)	2987
2988	241		WWOL = HYDV(ICL , 4)	2988
2989	242		PPOL = HYDV(ICL , 5)	2989
2990	243	C		2990
2991	244		IATRB = JS(9 , IS)	2991
2992	245		IF(IATRB . EQ . 0) THEN	2992
2993	246	C		2993
2994	247		RROR = HYDV(ICR , 1)	2994
2995	248		UOR = HYDV(ICR , 2)	2995
2996	249		VVOR = HYDV(ICR , 3)	2996
2997	250		WWOR = HYDV(ICR , 4)	2997
2998	251		PPOR = HYDV(ICR , 5)	2998
2999	252	C		2999
3000	253		ELSE	3000
3001	254	C		3001
3002	255		RROR = RROL	3002
3003	256		UOR = UUOL	3003
3004	257		VVOR = VVOL	3004
3005	258		WWOR = WWOL	3005
3006	259		PPOR = PPOL	3006
3007	260	C		3007
3008	261		END IF	3008
3009	262	C		3009
3010	263		ROL(2) = RROL	3010
3011	264		UOL(2) = UUOL	3011
3012	265		VOL(2) = VVOL	3012
3013	266		WOL(2) = WWOL	3013
3014	267		POL(2) = PPOL	3014
3015	268	C		3015
3016	269		ROR(2) = RROR	3016
3017	270		UOR(2) = UOR	3017
3018	271		VOR(2) = VVOR	3018
3019	272		WOR(2) = WWOR	3019
3020	273		POR(2) = PPOR	3020
3021	274	C		3021
3022	275		IS = JC(7 , IC)	3022
3023	276	C		3023
3024	277		ICL = JS(7 , IS)	3024
3025	278		ICR = JS(8 , IS)	3025
3026	279	C		3026
3027	280		RROL = HYDV(ICL , 1)	3027
3028	281		UUOL = HYDV(ICL , 2)	3028
3029	282		VVOL = HYDV(ICL , 3)	3029
3030	283		WWOL = HYDV(ICL , 4)	3030
3031	284		PPOL = HYDV(ICL , 5)	3031
3032	285	C		3032
3033	286		IATRB = JS(9 , IS)	3033
3034	287		IF(IATRB . EQ . 0) THEN	3034
3035	288	C		3035
3036	289		RROR = HYDV(ICR , 1)	3036
3037	290		UOR = HYDV(ICR , 2)	3037

```

3038 291      VVOR = HYDV( ICR , 3 )
3039 292      WWOR = HYDV( ICR , 4 )
3040 293      PPOR = HYDV( ICR , 5 )
3041 294      C
3042 295      ELSE
3043 296      C
3044 297      RROR = RROL
3045 298      UUOR = UUOL
3046 299      VVOR = VVOL
3047 300      WWOR = WWOL
3048 301      PPOR = PPOL
3049 302      C
3050 303      END IF
3051 304      C
3052 305      ROL( 3 ) = RROL
3053 306      UOL( 3 ) = UUOL
3054 307      VOL( 3 ) = VVOL
3055 308      WOL( 3 ) = WWOL
3056 309      POL( 3 ) = PPOL
3057 310      C
3058 311      ROR( 3 ) = RROR
3059 312      UOR( 3 ) = UUOR
3060 313      VOR( 3 ) = VVOR
3061 314      WOR( 3 ) = WWOR
3062 315      POR( 3 ) = PPOR
3063 316      C
3064 317      IS = JC( 8 , IC )
3065 318      C
3066 319      ICL = JS( 7 , IS )
3067 320      ICR = JS( 8 , IS )
3068 321      C
3069 322      RROL = HYDV( ICL , 1 )
3070 323      UUOL = HYDV( ICL , 2 )
3071 324      VVOL = HYDV( ICL , 3 )
3072 325      WWOL = HYDV( ICL , 4 )
3073 326      PPOL = HYDV( ICL , 5 )
3074 327      C
3075 328      IATRB = JS( 9 , IS )
3076 329      IF( IATRB .EQ. 0 ) THEN
3077 330      C
3078 331      RROR = HYDV( ICR , 1 )
3079 332      UUOR = HYDV( ICR , 2 )
3080 333      VVOR = HYDV( ICR , 3 )
3081 334      WWOR = HYDV( ICR , 4 )
3082 335      PPOR = HYDV( ICR , 5 )
3083 336      C
3084 337      ELSE
3085 338      C
3086 339      RROR = RROL
3087 340      UUOR = UUOL
3088 341      VVOR = VVOL
3089 342      WWOR = WWOL
3090 343      PPOR = PPOL
3091 344      C
3092 345      END IF
3093 346      C
3094 347      ROL( 4 ) = RROL
3095 348      UOL( 4 ) = UUOL
3096 349      VOL( 4 ) = VVOL
3097 350      WOL( 4 ) = WWOL
3098 351      POL( 4 ) = PPOL
3099 352      C
3100 353      ROR( 4 ) = RROR
3101 354      UOR( 4 ) = UUOR
3102 355      VOR( 4 ) = VVOR
3103 356      WOR( 4 ) = WWOR
3104 357      POR( 4 ) = PPOR
3105 358      C
3106 359      RMAX( KC ) = AMAX1( ROL( 1 ) , ROL( 2 ) , ROL( 3 ) , ROL( 4 ) ,
3107 360      . UOR( 1 ) , UOR( 2 ) , UOR( 3 ) , UOR( 4 ) )
3108 361      UMAX( KC ) = AMAX1( UOL( 1 ) , UOL( 2 ) , UOL( 3 ) , UOL( 4 ) ,
3109 362      . VOR( 1 ) , VOR( 2 ) , VOR( 3 ) , VOR( 4 ) )
3110 363      VMAX( KC ) = AMAX1( VOL( 1 ) , VOL( 2 ) , VOL( 3 ) , VOL( 4 ) ,
3111 364      . WOR( 1 ) , WOR( 2 ) , WOR( 3 ) , WOR( 4 ) )

```

```

3112 365      WMAX( KC ) = AMAX1( WOL( 1 ) , WOL( 2 ) , WOL( 3 ) , WOL( 4 ) ,
3113 366      .      WOR( 1 ) , WOR( 2 ) , WOR( 3 ) , WOR( 4 ) )
3114 367      PMAX( KC ) = AMAX1( POL( 1 ) , POL( 2 ) , POL( 3 ) , POL( 4 ) ,
3115 368      .      POR( 1 ) , POR( 2 ) , POR( 3 ) , POR( 4 ) )
3116 369      C
3117 370      RMIN( KC ) = AMIN1( ROL( 1 ) , ROL( 2 ) , ROL( 3 ) , ROL( 4 ) ,
3118 371      .      ROR( 1 ) , ROR( 2 ) , ROR( 3 ) , ROR( 4 ) )
3119 372      UMIN( KC ) = AMIN1( UOL( 1 ) , UOL( 2 ) , UOL( 3 ) , UOL( 4 ) ,
3120 373      .      UOR( 1 ) , UOR( 2 ) , UOR( 3 ) , UOR( 4 ) )
3121 374      VMIN( KC ) = AMIN1( VOL( 1 ) , VOL( 2 ) , VOL( 3 ) , VOL( 4 ) ,
3122 375      .      VOR( 1 ) , VOR( 2 ) , VOR( 3 ) , VOR( 4 ) )
3123 376      WMIN( KC ) = AMIN1( WOL( 1 ) , WOL( 2 ) , WOL( 3 ) , WOL( 4 ) ,
3124 377      .      WOR( 1 ) , WOR( 2 ) , WOR( 3 ) , WOR( 4 ) )
3125 378      PMIN( KC ) = AMIN1( POL( 1 ) , POL( 2 ) , POL( 3 ) , POL( 4 ) ,
3126 379      .      POR( 1 ) , POR( 2 ) , POR( 3 ) , POR( 4 ) )
3127 380      C
3128 381      150 CONTINUE
3129 382      C
3130 383      DO 180 IC = NC1 , NC2
3131 384      KC = IC - NC1 + 1
3132 385      C
3133 386      RRR( KC ) = RMAX( KC ) - HYDV( IC , 1 )
3134 387      RRL( KC ) = RMIN( KC ) - HYDV( IC , 1 )
3135 388      UUR( KC ) = UMAX( KC ) - HYDV( IC , 2 )
3136 389      UUL( KC ) = UMIN( KC ) - HYDV( IC , 2 )
3137 390      VVR( KC ) = VMAX( KC ) - HYDV( IC , 3 )
3138 391      VVL( KC ) = VMIN( KC ) - HYDV( IC , 3 )
3139 392      WWR( KC ) = WMAX( KC ) - HYDV( IC , 4 )
3140 393      WWL( KC ) = WMIN( KC ) - HYDV( IC , 4 )
3141 394      PPR( KC ) = PMAX( KC ) - HYDV( IC , 5 )
3142 395      PPL( KC ) = PMIN( KC ) - HYDV( IC , 5 )
3143 396      C
3144 397      180 CONTINUE
3145 398      C
3146 399      DO 170 IC = NC1 , NC2
3147 400      KC = IC - NC1 + 1
3148 401      C
3149 402      IS = JC( 5 , IC )
3150 403      C
3151 404      ICL = JS( 7 , IS )
3152 405      ICR = JS( 8 , IS )
3153 406      C
3154 407      XML = XYZMDL( 1 , IS ) - XC( 1 , ICL )
3155 408      YML = XYZMDL( 2 , IS ) - XC( 2 , ICL )
3156 409      ZML = XYZMDL( 3 , IS ) - XC( 3 , ICL )
3157 410      C
3158 411      RROL = 1.E-16 + RGRAD( ICL , 1 ) * XML +
3159 412      .      RGRAD( ICL , 2 ) * YML + RGRAD( ICL , 3 ) * ZML
3160 413      UUOL = 1.E-16 + UGRAD( ICL , 1 ) * XML +
3161 414      .      UGRAD( ICL , 2 ) * YML + UGRAD( ICL , 3 ) * ZML
3162 415      VVOL = 1.E-16 + VGRAD( ICL , 1 ) * XML +
3163 416      .      VGRAD( ICL , 2 ) * YML + VGRAD( ICL , 3 ) * ZML
3164 417      WWOL = 1.E-16 + WGRAD( ICL , 1 ) * XML +
3165 418      .      WGRAD( ICL , 2 ) * YML + WGRAD( ICL , 3 ) * ZML
3166 419      PPOL = 1.E-16 + PGRAD( ICL , 1 ) * XML +
3167 420      .      PGRAD( ICL , 2 ) * YML + PGRAD( ICL , 3 ) * ZML
3168 421      C
3169 422      IATRB = JS( 9 , IS )
3170 423      IF( IATRB .EQ. 0 ) THEN
3171 424      C
3172 425      XMR = XYZMDL( 1 , IS ) - XC( 1 , ICR )
3173 426      YMR = XYZMDL( 2 , IS ) - XC( 2 , ICR )
3174 427      ZMR = XYZMDL( 3 , IS ) - XC( 3 , ICR )
3175 428      C
3176 429      RROR = 1.E-16 + RGRAD( ICR , 1 ) * XMR +
3177 430      .      RGRAD( ICR , 2 ) * YMR + RGRAD( ICR , 3 ) * ZMR
3178 431      UUOR = 1.E-16 + UGRAD( ICR , 1 ) * XMR +
3179 432      .      UGRAD( ICR , 2 ) * YMR + UGRAD( ICR , 3 ) * ZMR
3180 433      VVOR = 1.E-16 + VGRAD( ICR , 1 ) * XMR +
3181 434      .      VGRAD( ICR , 2 ) * YMR + VGRAD( ICR , 3 ) * ZMR
3182 435      WWOR = 1.E-16 + WGRAD( ICR , 1 ) * XMR +
3183 436      .      WGRAD( ICR , 2 ) * YMR + WGRAD( ICR , 3 ) * ZMR
3184 437      PPOR = 1.E-16 + PGRAD( ICR , 1 ) * XMR +
3185 438      .      PGRAD( ICR , 2 ) * YMR + PGRAD( ICR , 3 ) * ZMR

```

```

3186 439 C
3187 440 ELSE
3188 441 C
3189 442 RROR = RROL
3190 443 UUOR = UUOL
3191 444 VVOR = VVOL
3192 445 WWOR = WWOL
3193 446 PPOR = PPOL
3194 447 C
3195 448 END IF
3196 449 C
3197 450 ROL( 1 ) = 1. / RROL
3198 451 UOL( 1 ) = 1. / UUOL
3199 452 VOL( 1 ) = 1. / VVOL
3200 453 WOL( 1 ) = 1. / WWOL
3201 454 POL( 1 ) = 1. / PPOL
3202 455 C
3203 456 ROR( 1 ) = 1. / RROR
3204 457 UOR( 1 ) = 1. / UUOR
3205 458 VOR( 1 ) = 1. / VVOR
3206 459 WOR( 1 ) = 1. / WWOR
3207 460 POR( 1 ) = 1. / PPOR
3208 461 C
3209 462 IS = JC( 6 , IC )
3210 463 C
3211 464 ICL = JS( 7 , IS )
3212 465 ICR = JS( 8 , IS )
3213 466 C
3214 467 XML = XYZMDL( 1 , IS ) - XC( 1 , ICL )
3215 468 YML = XYZMDL( 2 , IS ) - XC( 2 , ICL )
3216 469 ZML = XYZMDL( 3 , IS ) - XC( 3 , ICL )
3217 470 C
3218 471 RROL = 1.E-16 + RGRAD( ICL , 1 ) * XML +
3219 472 RGRAD( ICL , 2 ) * YML + RGRAD( ICL , 3 ) * ZML
3220 473 UUOL = 1.E-16 + UGRAD( ICL , 1 ) * XML +
3221 474 UGRAD( ICL , 2 ) * YML + UGRAD( ICL , 3 ) * ZML
3222 475 VVOL = 1.E-16 + VGRAD( ICL , 1 ) * XML +
3223 476 VGRAD( ICL , 2 ) * YML + VGRAD( ICL , 3 ) * ZML
3224 477 WWOL = 1.E-16 + WGRAD( ICL , 1 ) * XML +
3225 478 WGRAD( ICL , 2 ) * YML + WGRAD( ICL , 3 ) * ZML
3226 479 PPOL = 1.E-16 + PGRAD( ICL , 1 ) * XML +
3227 480 PGRAD( ICL , 2 ) * YML + PGRAD( ICL , 3 ) * ZML
3228 481 C
3229 482 IATRB = JS( 9 , IS )
3230 483 IF( IATRB .EQ. 0 ) THEN
3231 484 C
3232 485 XMR = XYZMDL( 1 , IS ) - XC( 1 , ICR )
3233 486 YMR = XYZMDL( 2 , IS ) - XC( 2 , ICR )
3234 487 ZMR = XYZMDL( 3 , IS ) - XC( 3 , ICR )
3235 488 C
3236 489 RROR = 1.E-16 + RGRAD( ICR , 1 ) * XMR +
3237 490 RGRAD( ICR , 2 ) * YMR + RGRAD( ICR , 3 ) * ZMR
3238 491 UUOR = 1.E-16 + UGRAD( ICR , 1 ) * XMR +
3239 492 UGRAD( ICR , 2 ) * YMR + UGRAD( ICR , 3 ) * ZMR
3240 493 VVOR = 1.E-16 + VGRAD( ICR , 1 ) * XMR +
3241 494 VGRAD( ICR , 2 ) * YMR + VGRAD( ICR , 3 ) * ZMR
3242 495 WWOR = 1.E-16 + WGRAD( ICR , 1 ) * XMR +
3243 496 WGRAD( ICR , 2 ) * YMR + WGRAD( ICR , 3 ) * ZMR
3244 497 PPOR = 1.E-16 + PGRAD( ICR , 1 ) * XMR +
3245 498 PGRAD( ICR , 2 ) * YMR + PGRAD( ICR , 3 ) * ZMR
3246 499 C
3247 500 ELSE
3248 501 C
3249 502 RROR = RROL
3250 503 UUOR = UUOL
3251 504 VVOR = VVOL
3252 505 WWOR = WWOL
3253 506 PPOR = PPOL
3254 507 C
3255 508 END IF
3256 509 C
3257 510 ROL( 2 ) = 1. / RROL
3258 511 UOL( 2 ) = 1. / UUOL
3259 512 VOL( 2 ) = 1. / VVOL

```

```

3260 513      WOL( 2 ) = 1. / WWOL
3261 514      POL( 2 ) = 1. / PPOL
3262 515      C
3263 516      ROR( 2 ) = 1. / RROR
3264 517      UOR( 2 ) = 1. / UUOR
3265 518      VOR( 2 ) = 1. / VVOR
3266 519      WOR( 2 ) = 1. / WWOR
3267 520      POR( 2 ) = 1. / PPOR
3268 521      C
3269 522      IS = JC( 7 , IC )
3270 523      C
3271 524      ICL = JS( 7 , IS )
3272 525      ICR = JS( 8 , IS )
3273 526      C
3274 527      XML = XYZMDL( 1 , IS ) - XC( 1 , ICL )
3275 528      YML = XYZMDL( 2 , IS ) - XC( 2 , ICL )
3276 529      ZML = XYZMDL( 3 , IS ) - XC( 3 , ICL )
3277 530      C
3278 531      RROL = 1.E-16 + RGRAD( ICL , 1 ) * XML +
3279 532      .      RGRAD( ICL , 2 ) * YML + RGRAD( ICL , 3 ) * ZML
3280 533      .      UUOL = 1.E-16 + UGRAD( ICL , 1 ) * XML +
3281 534      .      UGRAD( ICL , 2 ) * YML + UGRAD( ICL , 3 ) * ZML
3282 535      .      VVOL = 1.E-16 + VGRAD( ICL , 1 ) * XML +
3283 536      .      VGRAD( ICL , 2 ) * YML + VGRAD( ICL , 3 ) * ZML
3284 537      .      WWOL = 1.E-16 + WGRAD( ICL , 1 ) * XML +
3285 538      .      WGRAD( ICL , 2 ) * YML + WGRAD( ICL , 3 ) * ZML
3286 539      .      PPOL = 1.E-16 + PGRAD( ICL , 1 ) * XML +
3287 540      .      PGRAD( ICL , 2 ) * YML + PGRAD( ICL , 3 ) * ZML
3288 541      C
3289 542      IATRB = JS( 9 , IS )
3290 543      IF( IATRB .EQ. 0 ) THEN
3291 544      C
3292 545      XMR = XYZMDL( 1 , IS ) - XC( 1 , ICR )
3293 546      YMR = XYZMDL( 2 , IS ) - XC( 2 , ICR )
3294 547      ZMR = XYZMDL( 3 , IS ) - XC( 3 , ICR )
3295 548      C
3296 549      RROR = 1.E-16 + RGRAD( ICR , 1 ) * XMR +
3297 550      .      RGRAD( ICR , 2 ) * YMR + RGRAD( ICR , 3 ) * ZMR
3298 551      .      UUOR = 1.E-16 + UGRAD( ICR , 1 ) * XMR +
3299 552      .      UGRAD( ICR , 2 ) * YMR + UGRAD( ICR , 3 ) * ZMR
3300 553      .      VVOR = 1.E-16 + VGRAD( ICR , 1 ) * XMR +
3301 554      .      VGRAD( ICR , 2 ) * YMR + VGRAD( ICR , 3 ) * ZMR
3302 555      .      WWOR = 1.E-16 + WGRAD( ICR , 1 ) * XMR +
3303 556      .      WGRAD( ICR , 2 ) * YMR + WGRAD( ICR , 3 ) * ZMR
3304 557      .      PPOR = 1.E-16 + PGRAD( ICR , 1 ) * XMR +
3305 558      .      PGRAD( ICR , 2 ) * YMR + PGRAD( ICR , 3 ) * ZMR
3306 559      C
3307 560      ELSE
3308 561      C
3309 562      RROR = RROL
3310 563      UUOR = UUOL
3311 564      VVOR = VVOL
3312 565      WWOR = WWOL
3313 566      PPOR = PPOL
3314 567      C
3315 568      END IF
3316 569      C
3317 570      ROL( 3 ) = 1. / RROR
3318 571      UOL( 3 ) = 1. / UUOL
3319 572      VOL( 3 ) = 1. / VVOL
3320 573      WOL( 3 ) = 1. / WWOL
3321 574      POL( 3 ) = 1. / PPOL
3322 575      C
3323 576      ROR( 3 ) = 1. / RROR
3324 577      UOR( 3 ) = 1. / UUOR
3325 578      VOR( 3 ) = 1. / VVOR
3326 579      WOR( 3 ) = 1. / WWOR
3327 580      POR( 3 ) = 1. / PPOR
3328 581      C
3329 582      IS = JC( 8 , IC )
3330 583      C
3331 584      ICL = JS( 7 , IS )
3332 585      ICR = JS( 8 , IS )
3333 586      C

```

```

3334 587 XML = XYZMDL( 1 , IS ) - XC( 1 , ICL ) 3334
3335 588 YML = XYZMDL( 2 , IS ) - XC( 2 , ICL ) 3335
3336 589 ZML = XYZMDL( 3 , IS ) - XC( 3 , ICL ) 3336
3337 590 C 3337
3338 591 RROR = 1.E-16 + RGRAD( ICL , 1 ) * XML + 3338
3339 592 RGRAD( ICL , 2 ) * YML + RGRAD( ICL , 3 ) * ZML 3339
3340 593 . UUOL = 1.E-16 + UGRAD( ICL , 1 ) * XML + 3340
3341 594 UGRAD( ICL , 2 ) * YML + UGRAD( ICL , 3 ) * ZML 3341
3342 595 . VVOL = 1.E-16 + VGRAD( ICL , 1 ) * XML + 3342
3343 596 VGRAD( ICL , 2 ) * YML + VGRAD( ICL , 3 ) * ZML 3343
3344 597 . WWOL = 1.E-16 + WGRAD( ICL , 1 ) * XML + 3344
3345 598 WGRAD( ICL , 2 ) * YML + WGRAD( ICL , 3 ) * ZML 3345
3346 599 . PPOL = 1.E-16 + PGRAD( ICL , 1 ) * XML + 3346
3347 600 PGRAD( ICL , 2 ) * YML + PGRAD( ICL , 3 ) * ZML 3347
3348 601 C 3348
3349 602 IATR = JS(.9 , IS ) 3349
3350 603 IF( IATR .EQ. 0 ) THEN 3350
3351 604 C 3351
3352 605 XMR = XYZMDL( 1 , IS ) - XC( 1 , ICR ) 3352
3353 606 YMR = XYZMDL( 2 , IS ) - XC( 2 , ICR ) 3353
3354 607 ZMR = XYZMDL( 3 , IS ) - XC( 3 , ICR ) 3354
3355 608 C 3355
3356 609 RROR = 1.E-16 + RGRAD( ICR , 1 ) * XMR + 3356
3357 610 RGRAD( ICR , 2 ) * YMR + RGRAD( ICR , 3 ) * ZMR 3357
3358 611 . UUOR = 1.E-16 + UGRAD( ICR , 1 ) * XMR + 3358
3359 612 UGRAD( ICR , 2 ) * YMR + UGRAD( ICR , 3 ) * ZMR 3359
3360 613 . VVOR = 1.E-16 + VGRAD( ICR , 1 ) * XMR + 3360
3361 614 VGRAD( ICR , 2 ) * YMR + VGRAD( ICR , 3 ) * ZMR 3361
3362 615 . WWOR = 1.E-16 + WGRAD( ICR , 1 ) * XMR + 3362
3363 616 WGRAD( ICR , 2 ) * YMR + WGRAD( ICR , 3 ) * ZMR 3363
3364 617 . PPOR = 1.E-16 + PGRAD( ICR , 1 ) * XMR + 3364
3365 618 PGRAD( ICR , 2 ) * YMR + PGRAD( ICR , 3 ) * ZMR 3365
3366 619 C 3366
3367 620 ELSE 3367
3368 621 C 3368
3369 622 RROR = RROR 3369
3370 623 UUOR = UUOL 3370
3371 624 VVOR = VVOL 3371
3372 625 WWOR = WWOL 3372
3373 626 PPOR = PPOL 3373
3374 627 C 3374
3375 628 END IF 3375
3376 629 C 3376
3377 630 ROL( 4 ) = 1. / RROR 3377
3378 631 UOL( 4 ) = 1. / UUOL 3378
3379 632 VOL( 4 ) = 1. / VVOL 3379
3380 633 WOL( 4 ) = 1. / WWOL 3380
3381 634 POL( 4 ) = 1. / PPOL 3381
3382 635 C 3382
3383 636 ROR( 4 ) = 1. / RROR 3383
3384 637 UOR( 4 ) = 1. / UUOR 3384
3385 638 VOR( 4 ) = 1. / VVOR 3385
3386 639 WOR( 4 ) = 1. / WWOR 3386
3387 640 POR( 4 ) = 1. / PPOR 3387
3388 641 C 3388
3389 642 ISNR = SIGN( 1. , ROR( 1 ) ) 3389
3390 643 ISNL = SIGN( 1. , ROL( 1 ) ) 3390
3391 644 C 3391
3392 645 TEMPR = ( 1 + ISNR ) * RRR( KC ) + 3392
3393 646 ( 1 - ISNR ) * RRL( KC ) 3393
3394 647 RUVPR1 = 0.5 * TEMPR * ROR( 1 ) 3394
3395 648 C 3395
3396 649 TEMPL = ( 1 + ISNL ) * RRR( KC ) + 3396
3397 650 ( 1 - ISNL ) * RRL( KC ) 3397
3398 651 RUVPL1 = 0.5 * TEMPL * ROL( 1 ) 3398
3399 652 C 3399
3400 653 ISNR = SIGN( 1. , ROR( 2 ) ) 3400
3401 654 ISNL = SIGN( 1. , ROL( 2 ) ) 3401
3402 655 C 3402
3403 656 TEMPR = ( 1 + ISNR ) * RRR( KC ) + 3403
3404 657 ( 1 - ISNR ) * RRL( KC ) 3404
3405 658 RUVPR2 = 0.5 * TEMPR * ROR( 2 ) 3405
3406 659 C 3406
3407 660 TEMPL = ( 1 + ISNL ) * RRR( KC ) + 3407

```

```

3408 661      ( 1 - ISNL ) * RRL( KC )
3409 662      RUVPL2 = 0.5 * TEMPL * ROL( 2 )
3410 663      C
3411 664      ISNR = SIGN( 1. , ROR( 3 ) )
3412 665      ISNL = SIGN( 1. , ROL( 3 ) )
3413 666      C
3414 667      TEMPR = ( 1 + ISNR ) * RRR( KC ) +
3415 668      ( 1 - ISNR ) * RRL( KC )
3416 669      RUVPR3 = 0.5 * TEMPR * ROR( 3 )
3417 670      C
3418 671      TEMPL = ( 1 + ISNL ) * RRR( KC ) +
3419 672      ( 1 - ISNL ) * RRL( KC )
3420 673      RUVPL3 = 0.5 * TEMPL * ROL( 3 )
3421 674      C
3422 675      ISNR = SIGN( 1. , ROR( 4 ) )
3423 676      ISNL = SIGN( 1. , ROL( 4 ) )
3424 677      C
3425 678      TEMPR = ( 1 + ISNR ) * RRR( KC ) +
3426 679      ( 1 - ISNR ) * RRL( KC )
3427 680      RUVPR4 = 0.5 * TEMPR * ROR( 4 )
3428 681      C
3429 682      TEMPL = ( 1 + ISNL ) * RRR( KC ) +
3430 683      ( 1 - ISNL ) * RRL( KC )
3431 684      RUVPL4 = 0.5 * TEMPL * ROL( 4 )
3432 685      C
3433 686      RMIN( KC ) = AMINI( 1. , RUVPR1 , RUVPL1 , RUVPR2 , RUVPL2 ,
3434 687      RUVPR3 , RUVPL3 , RUVPR4 , RUVPL4 )
3435 688      C
3436 689      ISNR = SIGN( 1. , UOR( 1 ) )
3437 690      ISNL = SIGN( 1. , UOL( 1 ) )
3438 691      C
3439 692      TEMPR = ( 1 + ISNR ) * UUR( KC ) +
3440 693      ( 1 - ISNR ) * UUL( KC )
3441 694      RUVPR1 = 0.5 * TEMPR * UOR( 1 )
3442 695      C
3443 696      TEMPL = ( 1 + ISNL ) * UUR( KC ) +
3444 697      ( 1 - ISNL ) * UUL( KC )
3445 698      RUVPL1 = 0.5 * TEMPL * UOL( 1 )
3446 699      C
3447 700      ISNR = SIGN( 1. , UOR( 2 ) )
3448 701      ISNL = SIGN( 1. , UOL( 2 ) )
3449 702      C
3450 703      TEMPR = ( 1 + ISNR ) * UUR( KC ) +
3451 704      ( 1 - ISNR ) * UUL( KC )
3452 705      RUVPR2 = 0.5 * TEMPR * UOR( 2 )
3453 706      C
3454 707      TEMPL = ( 1 + ISNL ) * UUR( KC ) +
3455 708      ( 1 - ISNL ) * UUL( KC )
3456 709      RUVPL2 = 0.5 * TEMPL * UOL( 2 )
3457 710      C
3458 711      ISNR = SIGN( 1. , UOR( 3 ) )
3459 712      ISNL = SIGN( 1. , UOL( 3 ) )
3460 713      C
3461 714      TEMPR = ( 1 + ISNR ) * UUR( KC ) +
3462 715      ( 1 - ISNR ) * UUL( KC )
3463 716      RUVPR3 = 0.5 * TEMPR * UOR( 3 )
3464 717      C
3465 718      TEMPL = ( 1 + ISNL ) * UUR( KC ) +
3466 719      ( 1 - ISNL ) * UUL( KC )
3467 720      RUVPL3 = 0.5 * TEMPL * UOL( 3 )
3468 721      C
3469 722      ISNR = SIGN( 1. , UOR( 4 ) )
3470 723      ISNL = SIGN( 1. , UOL( 4 ) )
3471 724      C
3472 725      TEMPR = ( 1 + ISNR ) * UUR( KC ) +
3473 726      ( 1 - ISNR ) * UUL( KC )
3474 727      RUVPR4 = 0.5 * TEMPR * UOR( 4 )
3475 728      C
3476 729      TEMPL = ( 1 + ISNL ) * UUR( KC ) +
3477 730      ( 1 - ISNL ) * UUL( KC )
3478 731      RUVPL4 = 0.5 * TEMPL * UOL( 4 )
3479 732      C
3480 733      UMIN( KC ) = AMINI( 1. , RUVPR1 , RUVPL1 , RUVPR2 , RUVPL2 ,
3481 734      RUVPR3 , RUVPL3 , RUVPR4 , RUVPL4 )

```

```

3482 735 C
3483 736     ISNR = SIGN( 1. , VOR( 1 ) )
3484 737     ISNL = SIGN( 1. , VOL( 1 ) )
3485 738 C
3486 739     TEMPR = ( 1 + ISNR ) * VVR( KC ) +
3487 740           ( 1 - ISNR ) * VVL( KC )
3488 741     RUVPR1 = 0.5 * TEMPR * VOR( 1 )
3489 742 C
3490 743     TEMPL = ( 1 + ISNL ) * VVR( KC ) +
3491 744           ( 1 - ISNL ) * VVL( KC )
3492 745     RUVPL1 = 0.5 * TEMPL * VOL( 1 )
3493 746 C
3494 747     ISNR = SIGN( 1. , VOR( 2 ) )
3495 748     ISNL = SIGN( 1. , VOL( 2 ) )
3496 749 C
3497 750     TEMPR = ( 1 + ISNR ) * VVR( KC ) +
3498 751           ( 1 - ISNR ) * VVL( KC )
3499 752     RUVPR2 = 0.5 * TEMPR * VOR( 2 )
3500 753 C
3501 754     TEMPL = ( 1 + ISNL ) * VVR( KC ) +
3502 755           ( 1 - ISNL ) * VVL( KC )
3503 756     RUVPL2 = 0.5 * TEMPL * VOL( 2 )
3504 757 C
3505 758     ISNR = SIGN( 1. , VOR( 3 ) )
3506 759     ISNL = SIGN( 1. , VOL( 3 ) )
3507 760 C
3508 761     TEMPR = ( 1 + ISNR ) * VVR( KC ) +
3509 762           ( 1 - ISNR ) * VVL( KC )
3510 763     RUVPR3 = 0.5 * TEMPR * VOR( 3 )
3511 764 C
3512 765     TEMPL = ( 1 + ISNL ) * VVR( KC ) +
3513 766           ( 1 - ISNL ) * VVL( KC )
3514 767     RUVPL3 = 0.5 * TEMPL * VOL( 3 )
3515 768 C
3516 769     ISNR = SIGN( 1. , VOR( 4 ) )
3517 770     ISNL = SIGN( 1. , VOL( 4 ) )
3518 771 C
3519 772     TEMPR = ( 1 + ISNR ) * VVR( KC ) +
3520 773           ( 1 - ISNR ) * VVL( KC )
3521 774     RUVPR4 = 0.5 * TEMPR * VOR( 4 )
3522 775 C
3523 776     TEMPL = ( 1 + ISNL ) * VVR( KC ) +
3524 777           ( 1 - ISNL ) * VVL( KC )
3525 778     RUVPL4 = 0.5 * TEMPL * VOL( 4 )
3526 779 C
3527 780     VMIN( KC ) = AMIN1( 1. , RUVPR1 , RUVPL1 , RUVPR2 , RUVPL2 ,
3528 781           RUVPR3 , RUVPL3 , RUVPR4 , RUVPL4 )
3529 782 C
3530 783     ISNR = SIGN( 1. , WOR( 1 ) )
3531 784     ISNL = SIGN( 1. , WOL( 1 ) )
3532 785 C
3533 786     TEMPR = ( 1 + ISNR ) * WVR( KC ) +
3534 787           ( 1 - ISNR ) * WWL( KC )
3535 788     RUVPR1 = 0.5 * TEMPR * WOR( 1 )
3536 789 C
3537 790     TEMPL = ( 1 + ISNL ) * WVR( KC ) +
3538 791           ( 1 - ISNL ) * WWL( KC )
3539 792     RUVPL1 = 0.5 * TEMPL * WOL( 1 )
3540 793 C
3541 794     ISNR = SIGN( 1. , WOR( 2 ) )
3542 795     ISNL = SIGN( 1. , WOL( 2 ) )
3543 796 C
3544 797     TEMPR = ( 1 + ISNR ) * WVR( KC ) +
3545 798           ( 1 - ISNR ) * WWL( KC )
3546 799     RUVPR2 = 0.5 * TEMPR * WOR( 2 )
3547 800 C
3548 801     TEMPL = ( 1 + ISNL ) * WVR( KC ) +
3549 802           ( 1 - ISNL ) * WWL( KC )
3550 803     RUVPL2 = 0.5 * TEMPL * WOL( 2 )
3551 804 C
3552 805     ISNR = SIGN( 1. , WOR( 3 ) )
3553 806     ISNL = SIGN( 1. , WOL( 3 ) )
3554 807 C
3555 808     TEMPR = ( 1 + ISNR ) * WVR( KC ) +

```



```

3556 809      .      ( 1 - ISNR ) * WVL( KC )
3557 810      .      RUVPR3 = 0.5 * TEMPR * WOR( 3 )
3558 811      C
3559 812      .      TEMPL = ( 1 + ISNL ) * WWR( KC ) +
3560 813      .      ( 1 - ISNL ) * WWL( KC )
3561 814      .      RUVPL3 = 0.5 * TEMPL * WOL( 3 )
3562 815      C
3563 816      .      ISNR = SIGN( 1. , WOR( 4 ) )
3564 817      .      ISNL = SIGN( 1. , WOL( 4 ) )
3565 818      C
3566 819      .      TEMPR = ( 1 + ISNR ) * WWR( KC ) +
3567 820      .      ( 1 - ISNR ) * WWL( KC )
3568 821      .      RUVPR4 = 0.5 * TEMPR * WOR( 4 )
3569 822      C
3570 823      .      TEMPL = ( 1 + ISNL ) * WWR( KC ) +
3571 824      .      ( 1 - ISNL ) * WWL( KC )
3572 825      .      RUVPL4 = 0.5 * TEMPL * WOL( 4 )
3573 826      C
3574 827      .      WMIN( KC ) = AMIN1( 1. , RUVPR1 , RUVPL1 , RUVPR2 , RUVPL2 ,
3575 828      .      RUVPR3 , RUVPL3 , RUVPR4 , RUVPL4 )
3576 829      C
3577 830      .      ISNR = SIGN( 1. , POR( 1 ) )
3578 831      .      ISNL = SIGN( 1. , POL( 1 ) )
3579 832      C
3580 833      .      TEMPR = ( 1 + ISNR ) * PPR( KC ) +
3581 834      .      ( 1 - ISNR ) * PPL( KC )
3582 835      .      RUVPR1 = 0.5 * TEMPR * POR( 1 )
3583 836      C
3584 837      .      TEMPL = ( 1 + ISNL ) * PPR( KC ) +
3585 838      .      ( 1 - ISNL ) * PPL( KC )
3586 839      .      RUVPL1 = 0.5 * TEMPL * POL( 1 )
3587 840      C
3588 841      .      ISNR = SIGN( 1. , POR( 2 ) )
3589 842      .      ISNL = SIGN( 1. , POL( 2 ) )
3590 843      C
3591 844      .      TEMPR = ( 1 + ISNR ) * PPR( KC ) +
3592 845      .      ( 1 - ISNR ) * PPL( KC )
3593 846      .      RUVPR2 = 0.5 * TEMPR * POR( 2 )
3594 847      C
3595 848      .      TEMPL = ( 1 + ISNL ) * PPR( KC ) +
3596 849      .      ( 1 - ISNL ) * PPL( KC )
3597 850      .      RUVPL2 = 0.5 * TEMPL * POL( 2 )
3598 851      C
3599 852      .      ISNR = SIGN( 1. , POR( 3 ) )
3600 853      .      ISNL = SIGN( 1. , POL( 3 ) )
3601 854      C
3602 855      .      TEMPR = ( 1 + ISNR ) * PPR( KC ) +
3603 856      .      ( 1 - ISNR ) * PPL( KC )
3604 857      .      RUVPR3 = 0.5 * TEMPR * POR( 3 )
3605 858      C
3606 859      .      TEMPL = ( 1 + ISNL ) * PPR( KC ) +
3607 860      .      ( 1 - ISNL ) * PPL( KC )
3608 861      .      RUVPL3 = 0.5 * TEMPL * POL( 3 )
3609 862      C
3610 863      .      ISNR = SIGN( 1. , POR( 4 ) )
3611 864      .      ISNL = SIGN( 1. , POL( 4 ) )
3612 865      C
3613 866      .      TEMPR = ( 1 + ISNR ) * PPR( KC ) +
3614 867      .      ( 1 - ISNR ) * PPL( KC )
3615 868      .      RUVPR4 = 0.5 * TEMPR * POR( 4 )
3616 869      C
3617 870      .      TEMPL = ( 1 + ISNL ) * PPR( KC ) +
3618 871      .      ( 1 - ISNL ) * PPL( KC )
3619 872      .      RUVPL4 = 0.5 * TEMPL * POL( 4 )
3620 873      C
3621 874      .      PMIN( KC ) = AMIN1( 1. , RUVPR1 , RUVPL1 , RUVPR2 , RUVPL2 ,
3622 875      .      RUVPR3 , RUVPL3 , RUVPR4 , RUVPL4 )
3623 876      C
3624 877      170 CONTINUE
3625 878      C
3626 879      DO 330 IH = 1 , 3
3627 880      C
3628 881      DO 330 IC = NC1 , NC2
3629 882      KC = IC - NC1 + 1

```

```

3630 883 C
3631 884 RGRAD( IC , IH ) = RGRAD( IC , IH ) * RMIN( KC )
3632 885 UGRAD( IC , IH ) = UGRAD( IC , IH ) * UMIN( KC )
3633 886 VGRAD( IC , IH ) = VGRAD( IC , IH ) * VMIN( KC )
3634 887 WGRAD( IC , IH ) = WGRAD( IC , IH ) * WMIN( KC )
3635 888 PGRAD( IC , IH ) = PGRAD( IC , IH ) * PMIN( KC )
3636 889 C
3637 890 330 CONTINUE
3638 891 C
3639 892 NC1 = NC2 + 1
3640 893 NC2 = NC2 + NOFVEC( INC + 1 )
3641 894 80 CONTINUE
3642 895 C
3643 896 CALL FCHART
3644 897 C
3645 898 RETURN
3646 899 END
3647 900 C

```

```

3648 1 SUBROUTINE FIRST
3649 2 C
3650 3 C-----I
3651 4 C I
3652 5 C FIRST IS TO BY PASS GRADIENT AND CHARACTERSTIC COMPUTATION I
3653 6 C I
3654 7 C-----I
3655 8 C
3656 9 include 'dmsh00.h'
3657 10 include 'dhyd0.h'
3658 11 include 'dphsm0.h'
3659 12 include 'dmtr10.h'
3660 13 C
3661 14 DO 110 IS = 1 , NS
3662 15 C
3663 16 ICL = JS( 7 , IS )
3664 17 ICR = JS( 8 , IS )
3665 18 C
3666 19 RL( IS ) = HYDV( ICL , 1 )
3667 20 UL( IS ) = HYDV( ICL , 2 ) * XN( IS ) +
3668 21 . HYDV( ICL , 3 ) * YN( IS ) +
3669 22 . HYDV( ICL , 4 ) * ZN( IS )
3670 23 VL( IS ) = HYDV( ICL , 2 ) * XP( IS ) +
3671 24 . HYDV( ICL , 3 ) * YP( IS ) +
3672 25 . HYDV( ICL , 4 ) * ZP( IS )
3673 26 WL( IS ) = HYDV( ICL , 2 ) * XT( IS ) +
3674 27 . HYDV( ICL , 3 ) * YT( IS ) +
3675 28 . HYDV( ICL , 4 ) * ZT( IS )
3676 29 PL( IS ) = HYDV( ICL , 5 )
3677 30 AL( IS ) = HYDV( ICL , 6 )
3678 31 GL( IS ) = HYDV( ICL , 7 )
3679 32 EL( IS ) = HYDV( ICL , 8 )
3680 33 C
3681 34 IATRB = JS( 9 , IS )
3682 35 IF( IATRB .EQ. 0 ) THEN
3683 36 C
3684 37 RR( IS ) = HYDV( ICR , 1 )
3685 38 UR( IS ) = HYDV( ICR , 2 ) * XN( IS ) +
3686 39 . HYDV( ICR , 3 ) * YN( IS ) +
3687 40 . HYDV( ICR , 4 ) * ZN( IS )
3688 41 VR( IS ) = HYDV( ICR , 2 ) * XP( IS ) +
3689 42 . HYDV( ICR , 3 ) * YP( IS ) +
3690 43 . HYDV( ICR , 4 ) * ZP( IS )
3691 44 WR( IS ) = HYDV( ICR , 2 ) * XT( IS ) +
3692 45 . HYDV( ICR , 3 ) * YT( IS ) +
3693 46 . HYDV( ICR , 4 ) * ZT( IS )
3694 47 PR( IS ) = HYDV( ICR , 5 )
3695 48 AR( IS ) = HYDV( ICR , 6 )
3696 49 GR( IS ) = HYDV( ICR , 7 )
3697 50 ER( IS ) = HYDV( ICR , 8 )
3698 51 C
3699 52 ELSE
3700 53 C

```

```
3701 54      IF( IATRB . EQ . 8 ) THEN
3702 55      C
3703 56          RR( IS ) = RIN
3704 57          UR( IS ) = UIN * XN( IS ) + VIN * YN( IS ) + WIN * ZN( IS )
3705 58          VR( IS ) = UIN * XP( IS ) + VIN * YP( IS ) + WIN * ZP( IS )
3706 59          WR( IS ) = UIN * XT( IS ) + VIN * YT( IS ) + WIN * ZT( IS )
3707 60          PR( IS ) = PIN
3708 61          AR( IS ) = AL( IS )
3709 62          GR( IS ) = GL( IS )
3710 63          ER( IS ) = EL( IS )
3711 64      C
3712 65      END IF
3713 66      C
3714 67      IF( IATRB . EQ . 7 ) THEN
3715 68      C
3716 69          RR( IS ) = RL( IS )
3717 70          UR( IS ) = UL( IS )
3718 71          VR( IS ) = VL( IS )
3719 72          WR( IS ) = WL( IS )
3720 73          PR( IS ) = PL( IS )
3721 74          AR( IS ) = AL( IS )
3722 75          GR( IS ) = GL( IS )
3723 76          ER( IS ) = EL( IS )
3724 77      C
3725 78      END IF
3726 79      C
3727 80      IF( IATRB . EQ . 6 ) THEN
3728 81      C
3729 82          RR( IS ) = RL( IS )
3730 83          UR( IS ) = - UL( IS )
3731 84          VR( IS ) = VL( IS )
3732 85          WR( IS ) = WL( IS )
3733 86          PR( IS ) = PL( IS )
3734 87          AR( IS ) = AL( IS )
3735 88          GR( IS ) = GL( IS )
3736 89          ER( IS ) = EL( IS )
3737 90      C
3738 91      END IF
3739 92      C
3740 93      END IF
3741 94      C
3742 95      110 CONTINUE
3743 96      C
3744 97      RETURN
3745 98      END
3746 99      C
```

```

3747 1 SUBROUTINE FCHART 3747
3748 2 C 3748
3749 3 C-----I 3749
3750 4 C 3750
3751 5 C CHARCT INTRODUCE CORRECTION FOR SECOND ORDER CALCULATION I 3751
3752 6 C I 3752
3753 7 C-----I 3753
3754 8 C 3754
3755 9 include 'dmsh00.h' 3755
3756 10 include 'dhyd0.h' 3756
3757 11 include 'dphsm0.h' 3757
3758 12 include 'dmtr10.h' 3758
3759 13 C 3759
3760 14 REAL ZZLEFT(128),ZOLEFT(128),ZPLEFT(128),ZMLEFT(128) 3760
3761 15 REAL ZZRIGHT(128),ZORIGT(128),ZPRIGT(128),ZMRIGT(128) 3761
3762 16 REAL UPLEFT(128),UMLEFT(128),URLEFT(128) 3762
3763 17 REAL UPRIGT(128),UMRIGHT(128),URRIGHT(128) 3763
3764 18 REAL UVLEFT(128),UVRIGT(128),CNLEFT(128),CNRIGHT(128) 3764
3765 19 REAL RLEFTT(128),ULEFTT(128),VLEFTT(128),PLEFTT(128), 3765
3766 20 ALEFTT(128) 3766
3767 21 REAL RRIGHT(128),URIGHT(128),VRIGHT(128),PRIGHT(128), 3767
3768 22 ARIGHT(128) 3768
3769 23 C 3769
3770 24 NS1 = 1 3770
3771 25 NS2 = NOFVES( 1 ) 3771
3772 26 DO 90 IS = 1 , NVEES 3772
3773 27 C 3773
3774 28 DO 110 IS = NS1 , NS2 3774
3775 29 KS = IS - NS1 + 1 3775
3776 30 C 3776
3777 31 ICL = JS( 7 , IS ) 3777
3778 32 ICR = JS( 8 , IS ) 3778
3779 33 C 3779
3780 34 GL( IS ) = HYDV( ICL , 7 ) 3780
3781 35 CNLFST = GL( IS ) * HYDV( ICL , 5 ) / HYDV( ICL , 1 ) 3781
3782 36 CNLFT = SQRT( CNLFST ) 3782
3783 37 C 3783
3784 38 IATRB = JS( 9 , IS ) 3784
3785 39 IF( IATRB .EQ. 0 ) THEN 3785
3786 40 C 3786
3787 41 XYZ = 1. / XS( 5 , IS ) 3787
3788 42 XXN = ( XC( 1 , ICR ) - XC( 1 , ICL ) ) * XYZ 3788
3789 43 YYN = ( XC( 2 , ICR ) - XC( 2 , ICL ) ) * XYZ 3789
3790 44 ZZN = ( XC( 3 , ICR ) - XC( 3 , ICL ) ) * XYZ 3790
3791 45 C 3791
3792 46 UVLFT = HYDV( ICL , 2 ) * XXN + 3792
3793 47 HYDV( ICL , 3 ) * YYN + 3793
3794 48 HYDV( ICL , 4 ) * ZZN 3794
3795 49 C 3795
3796 50 GR( IS ) = HYDV( ICR , 7 ) 3796
3797 51 CNRGTS = GR( IS ) * HYDV( ICR , 5 ) / HYDV( ICR , 1 ) 3797
3798 52 CNRGT = SQRT( CNRGTS ) 3798
3799 53 C 3799
3800 54 UVRGT = HYDV( ICR , 2 ) * XXN + 3800
3801 55 HYDV( ICR , 3 ) * YYN + 3801
3802 56 HYDV( ICR , 4 ) * ZZN 3802
3803 57 C 3803
3804 58 ELSE 3804
3805 59 C 3805
3806 60 CNRGT = CNLFT 3806
3807 61 C 3807
3808 62 XYZ = 1. / XS( 5 , IS ) 3808
3809 63 XXN = ( XYZMDL( 1 , IS ) - XC( 1 , ICL ) ) * XYZ 3809
3810 64 YYN = ( XYZMDL( 2 , IS ) - XC( 2 , ICL ) ) * XYZ 3810
3811 65 ZZN = ( XYZMDL( 3 , IS ) - XC( 3 , ICL ) ) * XYZ 3811
3812 66 C 3812
3813 67 UVLFT = HYDV( ICL , 2 ) * XXN + 3813
3814 68 HYDV( ICL , 3 ) * YYN + 3814
3815 69 HYDV( ICL , 4 ) * ZZN 3815
3816 70 C 3816
3817 71 UVRGT = UVLFT 3817
3818 72 GR( IS ) = GL( IS ) 3818
3819 73 C 3819
3820 74 END IF 3820

```

```

3821 75 C
3822 76 CNLEFT( KS ) = CNLFT
3823 77 CNRIGHT( KS ) = CNRGT
3824 78 C
3825 79 UVLEFT( KS ) = UVLFT
3826 80 UVRIGHT( KS ) = UVRGT
3827 81 C
3828 82 110 CONTINUE
3829 83 C
3830 84 DO 130 KS = 1 , NOFVES( INS )
3831 85 C
3832 86 ZZLEFT( KS ) = .5 * ( UVLEFT( KS ) + CNLEFT( KS ) ) * DTT
3833 87 ZZRIGHT( KS ) = -.5 * ( UVRIGHT( KS ) - CNRIGHT( KS ) ) * DTT
3834 88 C
3835 89 130 CONTINUE
3836 90 C
3837 91 C CHARACTERISTICS LOCATIONS
3838 92 C
3839 93 DO 140 KS = 1 , NOFVES( INS )
3840 94 C
3841 95 IF( ZZLEFT( KS ) . LT . 0. ) ZZLEFT( KS ) = 0.
3842 96 IF( ZZRIGHT( KS ) . LT . 0. ) ZZRIGHT( KS ) = 0.
3843 97 C
3844 98 140 CONTINUE
3845 99 C
3846 100 DO 150 KS = 1 , NOFVES( INS )
3847 101 C
3848 102 ZOLEFT( KS ) = .5 * UVLEFT( KS ) * DTT
3849 103 ZORIGHT( KS ) = -.5 * UVRIGHT( KS ) * DTT
3850 104 ZPLEFT( KS ) = .5 * ( UVLEFT( KS ) + CNLEFT( KS ) ) * DTT
3851 105 ZPRIGHT( KS ) = -.5 * ( UVRIGHT( KS ) + CNRIGHT( KS ) ) * DTT
3852 106 ZMLEFT( KS ) = .5 * ( UVLEFT( KS ) - CNLEFT( KS ) ) * DTT
3853 107 ZMRIGHT( KS ) = -.5 * ( UVRIGHT( KS ) - CNRIGHT( KS ) ) * DTT
3854 108 C
3855 109 150 CONTINUE
3856 110 C
3857 111 C FIRST GUESS LEFT AND RIGHT VARIABLES, LINEAR INTERPOLATON
3858 112 C
3859 113 DO 160 IS = NS1 , NS2
3860 114 KS = IS - NS1 + 1
3861 115 C
3862 116 ICL = JS( 7 , IS )
3863 117 ICR = JS( 8 , IS )
3864 118 C
3865 119 IATRB = JS( 9 , IS )
3866 120 IF( IATRB . EQ . 0 ) THEN
3867 121 C
3868 122 XYZ = 1. / XS( 5 , IS )
3869 123 XXN = ( XC( 1 , ICR ) - XC( 1 , ICL ) ) * XYZ
3870 124 YYN = ( XC( 2 , ICR ) - XC( 2 , ICL ) ) * XYZ
3871 125 ZZN = ( XC( 3 , ICR ) - XC( 3 , ICL ) ) * XYZ
3872 126 C
3873 127 XXL = ( XYZMDL( 1 , IS ) - XC( 1 , ICL ) )
3874 128 YYL = ( XYZMDL( 2 , IS ) - XC( 2 , ICL ) )
3875 129 ZZL = ( XYZMDL( 3 , IS ) - XC( 3 , ICL ) )
3876 130 C
3877 131 XX = XXL - ZZLEFT( KS ) * XXN
3878 132 YY = YYL - ZZLEFT( KS ) * YYN
3879 133 ZZ = ZZL - ZZLEFT( KS ) * ZZN
3880 134 C
3881 135 HRRL = HYDV( ICL , 1 ) + RGRAD( ICL , 1 ) * XX +
3882 136 . RGRAD( ICL , 2 ) * YY + RGRAD( ICL , 3 ) * ZZ
3883 137 HUUL = HYDV( ICL , 2 ) + UGRAD( ICL , 1 ) * XX +
3884 138 . UGRAD( ICL , 2 ) * YY + UGRAD( ICL , 3 ) * ZZ
3885 139 HVVL = HYDV( ICL , 3 ) + VGRAD( ICL , 1 ) * XX +
3886 140 . VGRAD( ICL , 2 ) * YY + VGRAD( ICL , 3 ) * ZZ
3887 141 HWWL = HYDV( ICL , 4 ) + WGRAD( ICL , 1 ) * XX +
3888 142 . WGRAD( ICL , 2 ) * YY + WGRAD( ICL , 3 ) * ZZ
3889 143 HPPL = HYDV( ICL , 5 ) + PGRAD( ICL , 1 ) * XX +
3890 144 . PGRAD( ICL , 2 ) * YY + PGRAD( ICL , 3 ) * ZZ
3891 145 C
3892 146 GMTLFT = GL( IS ) * HRRL * HPPL
3893 147 SQGMTL = SQRT( GMTLFT )
3894 148 C

```

```

3895 149      XX = ( ZLEFT( KS ) - ZZLEFT( KS ) ) * XXN      3895
3896 150      YY = ( ZLEFT( KS ) - ZZLEFT( KS ) ) * YYN      3896
3897 151      ZZ = ( ZLEFT( KS ) - ZZLEFT( KS ) ) * ZZN      3897
3898 152      UUU = UGRAD( ICL , 1 ) * XX + UGRAD( ICL , 2 ) * YY +      3898
3899 153      UGRAD( ICL , 3 ) * ZZ      3899
3900 154      PPP = PGRAD( ICL , 1 ) * XX + PGRAD( ICL , 2 ) * YY +      3900
3901 155      PGRAD( ICL , 3 ) * ZZ      3901
3902 156      UPLFT = -.5 * ( UUU + PPP / SQGRTL ) / SQGRTL      3902
3903 157      C      3903
3904 158      XX = ( ZLEFT( KS ) - ZZLEFT( KS ) ) * XXN      3904
3905 159      YY = ( ZLEFT( KS ) - ZZLEFT( KS ) ) * YYN      3905
3906 160      ZZ = ( ZLEFT( KS ) - ZZLEFT( KS ) ) * ZZN      3906
3907 161      UUU = UGRAD( ICL , 1 ) * XX + UGRAD( ICL , 2 ) * YY +      3907
3908 162      UGRAD( ICL , 3 ) * ZZ      3908
3909 163      PPP = PGRAD( ICL , 1 ) * XX + PGRAD( ICL , 2 ) * YY +      3909
3910 164      PGRAD( ICL , 3 ) * ZZ      3910
3911 165      UMLFT = .5 * ( UUU - PPP / SQGRTL ) / SQGRTL      3911
3912 166      C      3912
3913 167      XX = ( ZLEFT( KS ) - ZZLEFT( KS ) ) * XXN      3913
3914 168      YY = ( ZLEFT( KS ) - ZZLEFT( KS ) ) * YYN      3914
3915 169      ZZ = ( ZLEFT( KS ) - ZZLEFT( KS ) ) * ZZN      3915
3916 170      PPP = PGRAD( ICL , 1 ) * XX + PGRAD( ICL , 2 ) * YY +      3916
3917 171      PGRAD( ICL , 3 ) * ZZ      3917
3918 172      C      3918
3919 173      XX = XXL - ZLEFT( KS ) * XXN      3919
3920 174      YY = YYL - ZLEFT( KS ) * YYN      3920
3921 175      ZZ = ZZL - ZLEFT( KS ) * ZZN      3921
3922 176      C      3922
3923 177      RRRR = HYDV( ICL , 1 ) + RGRAD( ICL , 1 ) * XX +      3923
3924 178      RGRAD( ICL , 2 ) * YY + RGRAD( ICL , 3 ) * ZZ      3924
3925 179      URLFT = PPP / GMTLFT + 1. / HRRL - 1. / RRRR      3925
3926 180      C      3926
3927 181      XXR = ( XYZMDL( 1 , IS ) - XC( 1 , ICR ) )      3927
3928 182      YYR = ( XYZMDL( 2 , IS ) - XC( 2 , ICR ) )      3928
3929 183      ZZR = ( XYZMDL( 3 , IS ) - XC( 3 , ICR ) )      3929
3930 184      C      3930
3931 185      XX = XXR + ZZRIGHT( KS ) * XXN      3931
3932 186      YY = YYR + ZZRIGHT( KS ) * YYN      3932
3933 187      ZZ = ZZR + ZZRIGHT( KS ) * ZZN      3933
3934 188      C      3934
3935 189      HRRR = HYDV( ICR , 1 ) + RGRAD( ICR , 1 ) * XX +      3935
3936 190      RGRAD( ICR , 2 ) * YY + RGRAD( ICR , 3 ) * ZZ      3936
3937 191      HUUR = HYDV( ICR , 2 ) + UGRAD( ICR , 1 ) * XX +      3937
3938 192      UGRAD( ICR , 2 ) * YY + UGRAD( ICR , 3 ) * ZZ      3938
3939 193      HVVR = HYDV( ICR , 3 ) + VGRAD( ICR , 1 ) * XX +      3939
3940 194      VGRAD( ICR , 2 ) * YY + VGRAD( ICR , 3 ) * ZZ      3940
3941 195      HWWR = HYDV( ICR , 4 ) + WGRAD( ICR , 1 ) * XX +      3941
3942 196      WGRAD( ICR , 2 ) * YY + WGRAD( ICR , 3 ) * ZZ      3942
3943 197      HPPR = HYDV( ICR , 5 ) + PGRAD( ICR , 1 ) * XX +      3943
3944 198      PGRAD( ICR , 2 ) * YY + PGRAD( ICR , 3 ) * ZZ      3944
3945 199      C      3945
3946 200      GMTRGT = GR( IS ) * HRRR * HPPR      3946
3947 201      SQMTR = SQRT( GMTRGT )      3947
3948 202      C      3948
3949 203      XX = ( ZZRIGHT( KS ) - ZPRIGT( KS ) ) * XXN      3949
3950 204      YY = ( ZZRIGHT( KS ) - ZPRIGT( KS ) ) * YYN      3950
3951 205      ZZ = ( ZZRIGHT( KS ) - ZPRIGT( KS ) ) * ZZN      3951
3952 206      UUU = UGRAD( ICR , 1 ) * XX + UGRAD( ICR , 2 ) * YY +      3952
3953 207      UGRAD( ICR , 3 ) * ZZ      3953
3954 208      PPP = PGRAD( ICR , 1 ) * XX + PGRAD( ICR , 2 ) * YY +      3954
3955 209      PGRAD( ICR , 3 ) * ZZ      3955
3956 210      UPRGT = -.5 * ( UUU + PPP / SQMTR ) / SQMTR      3956
3957 211      C      3957
3958 212      XX = ( ZZRIGHT( KS ) - ZMRIGT( KS ) ) * XXN      3958
3959 213      YY = ( ZZRIGHT( KS ) - ZMRIGT( KS ) ) * YYN      3959
3960 214      ZZ = ( ZZRIGHT( KS ) - ZMRIGT( KS ) ) * ZZN      3960
3961 215      UUU = UGRAD( ICR , 1 ) * XX + UGRAD( ICR , 2 ) * YY +      3961
3962 216      UGRAD( ICR , 3 ) * ZZ      3962
3963 217      PPP = PGRAD( ICR , 1 ) * XX + PGRAD( ICR , 2 ) * YY +      3963
3964 218      PGRAD( ICR , 3 ) * ZZ      3964
3965 219      UMRGT = .5 * ( UUU - PPP / SQMTR ) / SQMTR      3965
3966 220      C      3966
3967 221      XX = ( ZZRIGHT( KS ) - ZORIGT( KS ) ) * XXN      3967
3968 222      YY = ( ZZRIGHT( KS ) - ZORIGT( KS ) ) * YYN      3968

```

```

3969 223      ZZ = ( ZZRIGT( KS ) - ZORIGT( KS ) ) * ZZN      3969
3970 224      PPP = PGRAD( ICR , 1 ) * XX + PGRAD( ICR , 2 ) * YY +      3970
3971 225      .      PGRAD( ICR , 3 ) * ZZ      3971
3972 226      C      3972
3973 227      XX = XXR + ZORIGT( KS ) * XXN      3973
3974 228      YY = YYR + ZORIGT( KS ) * YYN      3974
3975 229      ZZ = ZZR + ZORIGT( KS ) * ZZN      3975
3976 230      C      3976
3977 231      RRRR = HYDV( ICR , 1 ) + RGRAD( ICR , 1 ) * XX +      3977
3978 232      .      RGRAD( ICR , 2 ) * YY + RGRAD( ICR , 3 ) * ZZ      3978
3979 233      URRGT = PPP / GMTRGT + 1. / HRRR - 1. / RRRR      3979
3980 234      C      3980
3981 235      ELSE      3981
3982 236      C      3982
3983 237      XYZ = 1. / XS( 5 , IS )      3983
3984 238      XXN = ( XYZMDL( 1 , IS ) - XC( 1 , ICL ) ) * XYZ      3984
3985 239      YYN = ( XYZMDL( 2 , IS ) - XC( 2 , ICL ) ) * XYZ      3985
3986 240      ZZN = ( XYZMDL( 3 , IS ) - XC( 3 , ICL ) ) * XYZ      3986
3987 241      C      3987
3988 242      XXL = ( XYZMDL( 1 , IS ) - XC( 1 , ICL ) )      3988
3989 243      YYL = ( XYZMDL( 2 , IS ) - XC( 2 , ICL ) )      3989
3990 244      ZZL = ( XYZMDL( 3 , IS ) - XC( 3 , ICL ) )      3990
3991 245      C      3991
3992 246      XX = XXL - ZZLEFT( KS ) * XXN      3992
3993 247      YY = YYL - ZZLEFT( KS ) * YYN      3993
3994 248      ZZ = ZZL - ZZLEFT( KS ) * ZZN      3994
3995 249      C      3995
3996 250      HRRL = HYDV( ICL , 1 ) + RGRAD( ICL , 1 ) * XX +      3996
3997 251      .      RGRAD( ICL , 2 ) * YY + RGRAD( ICL , 3 ) * ZZ      3997
3998 252      HUUL = HYDV( ICL , 2 ) + UGRAD( ICL , 1 ) * XX +      3998
3999 253      .      UGRAD( ICL , 2 ) * YY + UGRAD( ICL , 3 ) * ZZ      3999
4000 254      HVVL = HYDV( ICL , 3 ) + VGRAD( ICL , 1 ) * XX +      4000
4001 255      .      VGRAD( ICL , 2 ) * YY + VGRAD( ICL , 3 ) * ZZ      4001
4002 256      HWML = HYDV( ICL , 4 ) + WGRAD( ICL , 1 ) * XX +      4002
4003 257      .      WGRAD( ICL , 2 ) * YY + WGRAD( ICL , 3 ) * ZZ      4003
4004 258      HPPL = HYDV( ICL , 5 ) + PGRAD( ICL , 1 ) * XX +      4004
4005 259      .      PGRAD( ICL , 2 ) * YY + PGRAD( ICL , 3 ) * ZZ      4005
4006 260      C      4006
4007 261      GMTLFT = GL( IS ) * HRRL * HPPL      4007
4008 262      SQGRTL = SQRT( GMTLFT )      4008
4009 263      C      4009
4010 264      XX = ( ZPLEFT( KS ) - ZZLEFT( KS ) ) * XXN      4010
4011 265      YY = ( ZPLEFT( KS ) - ZZLEFT( KS ) ) * YYN      4011
4012 266      ZZ = ( ZPLEFT( KS ) - ZZLEFT( KS ) ) * ZZN      4012
4013 267      UUU = UGRAD( ICL , 1 ) * XX + UGRAD( ICL , 2 ) * YY +      4013
4014 268      .      UGRAD( ICL , 3 ) * ZZ      4014
4015 269      PPP = PGRAD( ICL , 1 ) * XX + PGRAD( ICL , 2 ) * YY +      4015
4016 270      .      PGRAD( ICL , 3 ) * ZZ      4016
4017 271      UPLFT = .5 * ( UUU + PPP / SQGRTL ) / SQGRTL      4017
4018 272      C      4018
4019 273      XX = ( ZMLEFT( KS ) - ZZLEFT( KS ) ) * XXN      4019
4020 274      YY = ( ZMLEFT( KS ) - ZZLEFT( KS ) ) * YYN      4020
4021 275      ZZ = ( ZMLEFT( KS ) - ZZLEFT( KS ) ) * ZZN      4021
4022 276      UUU = UGRAD( ICL , 1 ) * XX + UGRAD( ICL , 2 ) * YY +      4022
4023 277      .      UGRAD( ICL , 3 ) * ZZ      4023
4024 278      PPP = PGRAD( ICL , 1 ) * XX + PGRAD( ICL , 2 ) * YY +      4024
4025 279      .      PGRAD( ICL , 3 ) * ZZ      4025
4026 280      UMLFT = .5 * ( UUU - PPP / SQGRTL ) / SQGRTL      4026
4027 281      C      4027
4028 282      XX = ( ZOLEFT( KS ) - ZZLEFT( KS ) ) * XXN      4028
4029 283      YY = ( ZOLEFT( KS ) - ZZLEFT( KS ) ) * YYN      4029
4030 284      ZZ = ( ZOLEFT( KS ) - ZZLEFT( KS ) ) * ZZN      4030
4031 285      PPP = PGRAD( ICL , 1 ) * XX + PGRAD( ICL , 2 ) * YY +      4031
4032 286      .      PGRAD( ICL , 3 ) * ZZ      4032
4033 287      C      4033
4034 288      XX = XXL - ZOLEFT( KS ) * XXN      4034
4035 289      YY = YYL - ZOLEFT( KS ) * YYN      4035
4036 290      ZZ = ZZL - ZOLEFT( KS ) * ZZN      4036
4037 291      C      4037
4038 292      RRRR = HYDV( ICL , 1 ) + RGRAD( ICL , 1 ) * XX +      4038
4039 293      .      RGRAD( ICL , 2 ) * YY + RGRAD( ICL , 3 ) * ZZ      4039
4040 294      URLFT = PPP / GMTLFT + 1. / HRRL - 1. / RRRR      4040
4041 295      C      4041
4042 296      HRRR = HRRL      4042

```

```

4043 297      HUUR = HUUL      4043
4044 298      HVVR = HVVL      4044
4045 299      HWR = HWL      4045
4046 300      HPPR = HPPL      4046
4047 301      C      4047
4048 302      GMTRGT = GMTLFT      4048
4049 303      SQGMTR = SQGRTL      4049
4050 304      C      4050
4051 305      UPRGT = UPLFT      4051
4052 306      UMRGT = UMLFT      4052
4053 307      URRGT = URLFT      4053
4054 308      C      4054
4055 309      END IF      4055
4056 310      C      4056
4057 311      RRL( KS ) = HRRL      4057
4058 312      UUL( KS ) = HUUL      4058
4059 313      VVL( KS ) = HVVL      4059
4060 314      WWL( KS ) = HWL      4060
4061 315      PPL( KS ) = HPPL      4061
4062 316      C      4062
4063 317      RRR( KS ) = HRRR      4063
4064 318      UUR( KS ) = HUUR      4064
4065 319      VVR( KS ) = HVVR      4065
4066 320      WWR( KS ) = HWR      4066
4067 321      PPR( KS ) = HPPR      4067
4068 322      C      4068
4069 323      UPLEFT( KS ) = UPLFT      4069
4070 324      UMLEFT( KS ) = UMLFT      4070
4071 325      URLEFT( KS ) = URLFT      4071
4072 326      C      4072
4073 327      UPRIGT( KS ) = UPRGT      4073
4074 328      UMRIGT( KS ) = UMRGT      4074
4075 329      URRIGT( KS ) = URRGT      4075
4076 330      C      4076
4077 331      160 CONTINUE      4077
4078 332      C      4078
4079 333      C CORRECTION OF THE FIRST GUESS      4079
4080 334      C      4080
4081 335      DO 170 KS = 1 , NOFVES( INS )      4081
4082 336      C      4082
4083 337      IF( UVLEFT( KS ) + CNLEFT( KS ) . LE . 0. ) UPLEFT( KS ) = 0.      4083
4084 338      IF( UVLEFT( KS ) - CNLEFT( KS ) . LE . 0. ) UMLEFT( KS ) = 0.      4084
4085 339      IF( UVLEFT( KS ) . LE . 0. ) URLEFT( KS ) = 0.      4085
4086 340      C      4086
4087 341      IF( UVRIGT( KS ) + CNRIGHT( KS ) . GE . 0. ) UPRIGT( KS ) = 0.      4087
4088 342      IF( UVRIGT( KS ) - CNRIGHT( KS ) . GE . 0. ) UMRIGT( KS ) = 0.      4088
4089 343      IF( UVRIGT( KS ) . GE . 0. ) URRIGT( KS ) = 0.      4089
4090 344      C      4090
4091 345      170 CONTINUE      4091
4092 346      C      4092
4093 347      C FINAL VALUES FOR RIGHT AND LEFT STATES      4093
4094 348      C      4094
4095 349      DO 180 KS = 1 , NOFVES( INS )      4095
4096 350      IS = KS + NSI - 1      4096
4097 351      C      4097
4098 352      GMTLFT = GL( IS ) * RRL( KS ) * PPL( KS )      4098
4099 353      SQGRTL = SQRT( GMTLFT )      4099
4100 354      C      4100
4101 355      GMTRGT = GR( IS ) * RRR( KS ) * PPR( KS )      4101
4102 356      SQGMTR = SQRT( GMTRGT )      4102
4103 357      C      4103
4104 358      RRL( KS ) = 1. / ( 1. / RRL( KS ) - ( UPLEFT( KS ) +      4104
4105 359      . UMLEFT( KS ) + URLEFT( KS ) ) )      4105
4106 360      UUL( KS ) = UUL( KS ) + SQGRTL * ( UPLEFT( KS ) -      4106
4107 361      . UMLEFT( KS ) )      4107
4108 362      VVL( KS ) = VVL( KS ) + SQGRTL * ( UPLEFT( KS ) -      4108
4109 363      . UMLEFT( KS ) )      4109
4110 364      WWL( KS ) = WWL( KS ) + SQGRTL * ( UPLEFT( KS ) -      4110
4111 365      . UMLEFT( KS ) )      4111
4112 366      PPL( KS ) = PPL( KS ) + GMTLFT * ( UPLEFT( KS ) +      4112
4113 367      . UMLEFT( KS ) )      4113
4114 368      C      4114
4115 369      RRR( KS ) = 1. / ( 1. / RRR( KS ) - ( UPRIGT( KS ) +      4115
4116 370      . UMRIGT( KS ) + URRIGT( KS ) ) )      4116

```



```

4117 371      UUR( KS ) = UUR( KS ) + SQGMTR * ( UPRIGT( KS ) -
4118 372      .      UMRIGT( KS ) )
4119 373      VVR( KS ) = VVR( KS ) + SQGMTR * ( UPRIGT( KS ) -
4120 374      .      UMRIGT( KS ) )
4121 375      WWR( KS ) = WWR( KS ) + SQGMTR * ( UPRIGT( KS ) -
4122 376      .      UMRIGT( KS ) )
4123 377      PPR( KS ) = PPR( KS ) + GMTRGT * ( UPRIGT( KS ) +
4124 378      .      UMRIGT( KS ) )
4125 379      C
4126 380      180 CONTINUE
4127 381      C
4128 382      DO 200 IS = NS1 , NS2
4129 383      .      KS = IS - NS1 + 1
4130 384      C
4131 385      .      ICL = JS( 7 , IS )
4132 386      .      ICR = JS( 8 , IS )
4133 387      C
4134 388      .      RL( IS ) = RRL( KS )
4135 389      .      UL( IS ) = UUL( KS ) * XN( IS ) +
4136 390      .      .      VVL( KS ) * YN( IS ) +
4137 391      .      .      WVL( KS ) * ZN( IS )
4138 392      .      VL( IS ) = UUL( KS ) * XP( IS ) +
4139 393      .      .      VVL( KS ) * YP( IS ) +
4140 394      .      .      WVL( KS ) * ZP( IS )
4141 395      .      WL( IS ) = UUL( KS ) * XT( IS ) +
4142 396      .      .      VVL( KS ) * YT( IS ) +
4143 397      .      .      WVL( KS ) * ZT( IS )
4144 398      .      PL( IS ) = PPL( KS )
4145 399      .      AL( IS ) = HYDV( ICL , 6 )
4146 400      .      GL( IS ) = HYDV( ICL , 7 )
4147 401      .      EL( IS ) = HYDV( ICL , 8 )
4148 402      C
4149 403      .      IATRB = JS( 9 , IS )
4150 404      .      IF( IATRB .EQ. 0 ) THEN
4151 405      C
4152 406      .      RR( IS ) = RRR( KS )
4153 407      .      UR( IS ) = UUR( KS ) * XN( IS ) +
4154 408      .      .      VVR( KS ) * YN( IS ) +
4155 409      .      .      WVR( KS ) * ZN( IS )
4156 410      .      VR( IS ) = UUR( KS ) * XP( IS ) +
4157 411      .      .      VVR( KS ) * YP( IS ) +
4158 412      .      .      WVR( KS ) * ZP( IS )
4159 413      .      WR( IS ) = UUR( KS ) * XT( IS ) +
4160 414      .      .      VVR( KS ) * YT( IS ) +
4161 415      .      .      WVR( KS ) * ZT( IS )
4162 416      .      PR( IS ) = PPR( KS )
4163 417      .      AR( IS ) = HYDV( ICR , 6 )
4164 418      .      GR( IS ) = HYDV( ICR , 7 )
4165 419      .      ER( IS ) = HYDV( ICR , 8 )
4166 420      C
4167 421      .      ELSE
4168 422      C
4169 423      .      IF( IATRB .EQ. 8 ) THEN
4170 424      C
4171 425      .      RR( IS ) = RIN
4172 426      .      UR( IS ) = UIN * XN( IS ) + VIN * YN( IS ) + WIN * ZN( IS )
4173 427      .      VR( IS ) = UIN * XP( IS ) + VIN * YP( IS ) + WIN * ZP( IS )
4174 428      .      WR( IS ) = UIN * XT( IS ) + VIN * YT( IS ) + WIN * ZT( IS )
4175 429      .      PR( IS ) = PIN
4176 430      .      AR( IS ) = AL( IS )
4177 431      .      GR( IS ) = GL( IS )
4178 432      .      ER( IS ) = EL( IS )
4179 433      C
4180 434      .      END IF
4181 435      C
4182 436      .      IF( IATRB .EQ. 7 ) THEN
4183 437      C
4184 438      .      RR( IS ) = RL( IS )
4185 439      .      UR( IS ) = UL( IS )
4186 440      .      VR( IS ) = VL( IS )
4187 441      .      WR( IS ) = WL( IS )
4188 442      .      PR( IS ) = PL( IS )
4189 443      .      AR( IS ) = AL( IS )
4190 444      .      GR( IS ) = GL( IS )

```

```

4191 445      ER( IS ) = EL( IS )
4192 446      C
4193 447      END IF
4194 448      C
4195 449      IF( IATRB . EQ . 6 ) THEN
4196 450      C
4197 451          RR( IS ) = RL( IS )
4198 452          UR( IS ) = - UL( IS )
4199 453          VR( IS ) = VL( IS )
4200 454          WR( IS ) = WL( IS )
4201 455          PR( IS ) = PL( IS )
4202 456          AR( IS ) = AL( IS )
4203 457          GR( IS ) = GL( IS )
4204 458          ER( IS ) = EL( IS )
4205 459      C
4206 460      END IF
4207 461      C
4208 462      END IF
4209 463      C
4210 464      200 CONTINUE
4211 465      C
4212 466          NS1 = NS2 + 1
4213 467          NS2 = NS2 + NOFVES( INS + 1 )
4214 468      90 CONTINUE
4215 469      C
4216 470      RETURN
4217 471      END

```

```

4218 1      SUBROUTINE EOS1 (RRR,EEE,N,GAMMA)
4219 2      C
4220 3      C AIR IS ASSUMED TO BE CALORICALLY IMPERFECT, THERMALLY PERFECT.
4221 4      C THEREFORE, INCLUDE IMPERFECTIONS VIA A VARIABLE GAMMA DEPENDENT
4222 5      C ON DENSITY AND INTERNAL ENERGY. THIS ROUTINE PERFORMS A TABLE
4223 6      C LOOK UP FOR GAMMA.
4224 7      C
4225 8      C INPUT VARIABLE DEFINITIONS.
4226 9      C RRR = MASS DENSITY
4227 10     C EEE = INTERNAL ENERGY PER UNIT VOLUME
4228 11     C (CONVERTED FOR INTERNAL *CALL TO ENERGY PER UNIT MASS)
4229 12     C N = NUMBER OF ENTRIES IN ARRAYS RRR & EEE
4230 13     C
4231 14     C PARAMETER (M = 64)
4232 15     C
4233 16     C DIMENSION RRR(N), EEE(N), GAMMA(N)
4234 17     C DIMENSION T11(M), T12(M), T21(M), T22(M), RHO(M), E(M)
4235 18     C DIMENSION OMP(M), Q(M), I(M), J(M)
4236 19     C DIMENSION G1(168),G2(112),G3(112),G4(112),G5(112),
4237 20     C G6(112),G7(112),GF(840)
4238 21     C
4239 22     C NOTE: THE TABLE LOOK UP TREATS ARRAY GF AS THOUGH IT
4240 23     C WERE DIMENSIONED (8,105).
4241 24     C
4242 25     C EQUIVALENCE (G1(1),GF( 1)), (G2(1),GF(169)), (G3(1),GF(281)),
4243 26     C (G4(1),GF(393)), (G5(1),GF(505)), (G6(1),GF(617)),
4244 27     C (G7(1),GF(729))
4245 28     C
4246 29     C DATA XL16E /2.7725887222397744835689081810414791107177734375/
4247 30     C -----
4248 31     C G = GAMMA - 1.0 IS STORED FOR 32 BIT WORD MACHINES IN POWERS OF
4249 32     C 16 ACROSS FOR MASS DENSITY VARIATION AND INTERMEDIATE VALUES
4250 33     C 1 - 16 FOR POWERS OF 16 VERTICALLY WHICH REPRESENT THE INTERNAL
4251 34     C ENERGY VARIATION.
4252 35     C
4253 36     C 16**(?) .GE. RHO .GE. 16**(-6)
4254 37     C 16**(15) .GE. E .GE. 16**(8)
4255 38     C -----
4256 39     C DATA G1 /8*.4222,8*.4152,8*.4110,8*.4081,8*.4058,8*.4040,
4257 40     C 8*.4024,8*.4011,8*.3998,8*.3988,8*.3978,8*.3969,
4258 41     C 8*.3961,8*.3953,8*.3935,8*.3918,
4259 42     C .3723,.3715,.3707,.3699,.3690,.3680,.3663,.3637,
4260 43     C .3555,.3538,.3522,.3502,.3476,.3430,.3344,.3238,
4261 44     C .3370,.3370,.3370,.3364,.3347,.3277,.3099,.2885,

```

4262	45		.3257, .3227, .3201, .3134, .3062, .3014, .2884, .2591,	4262
4263	46		.3166, .3110, .3063, .2946, .2831, .2783, .2677, .2358/	4263
4264	47		DATA G2/ .3111, .3006, .2940, .2787, .2635, .2588, .2502, .2236,	4264
4265	48		.3075, .2906, .2810, .2665, .2466, .2418, .2350, .2131,	4265
4266	49		.3043, .2819, .2695, .2554, .2317, .2269, .2216, .2038,	4266
4267	50		.2929, .2740, .2593, .2455, .2206, .2136, .2097, .1955,	4267
4268	51		.2840, .2672, .2500, .2366, .2166, .2015, .1988, .1879,	4268
4269	52		.2764, .2611, .2429, .2285, .2125, .1890, .1890, .1811,	4269
4270	53		.2714, .2555, .2384, .2210, .2079, .1818, .1799, .1747,	4270
4271	54		.2669, .2500, .2343, .2141, .2037, .1822, .1709, .1689,	4271
4272	55		.2624, .2473, .2304, .2096, .1998, .1828, .1684, .1639,	4272
4273	56		.2590, .2446, .2268, .2087, .1961, .1834, .1673, .1601,	4273
4274	57		.2401, .2191, .1972, .1775, .1592, .1444, .1358, .1203,	4274
4275	58		.2002, .1960, .1749, .1536, .1376, .1252, .1107, .1044,	4275
4276	59		.1911, .1829, .1633, .1420, .1266, .1101, .1012, .0933,	4276
4277	60		.1950, .1781, .1566, .1415, .1241, .1118, .1009, .0948/	4277
4278	61		DATA G3/ .2001, .1789, .1594, .1443, .1306, .1189, .1095, .1013,	4278
4279	62		.2040, .1826, .1657, .1494, .1338, .1177, .1081, .0980,	4279
4280	63		.2034, .1854, .1683, .1497, .1322, .1169, .1051, .0946,	4280
4281	64		.1969, .1855, .1685, .1487, .1304, .1149, .1024, .0916,	4281
4282	65		.1899, .1837, .1677, .1475, .1287, .1126, .1002, .0900,	4282
4283	66		.1841, .1817, .1667, .1464, .1272, .1109, .0983, .0888,	4283
4284	67		.1800, .1800, .1659, .1455, .1262, .1097, .0965, .0878,	4284
4285	68		.1779, .1787, .1657, .1450, .1254, .1087, .0949, .0868,	4285
4286	69		.1773, .1778, .1656, .1447, .1250, .1080, .0939, .0859,	4286
4287	70		.1783, .1778, .1658, .1448, .1248, .1076, .0933, .0851,	4287
4288	71		.1808, .1781, .1667, .1451, .1248, .1074, .0930, .0843,	4288
4289	72		.2134, .2040, .1978, .1782, .1565, .1368, .1206, .1074,	4289
4290	73		.2210, .2072, .1957, .1739, .1516, .1312, .1137, .1000,	4290
4291	74		.2245, .2109, .1989, .1772, .1563, .1390, .1247, .1133/	4291
4292	75		DATA G4/ .2299, .2132, .2017, .1795, .1579, .1384, .1221, .1090,	4292
4293	76		.2350, .2157, .2023, .1798, .1575, .1370, .1197, .1057,	4293
4294	77		.2397, .2194, .2034, .1796, .1572, .1372, .1205, .1070,	4294
4295	78		.2452, .2227, .2050, .1805, .1576, .1379, .1236, .1118,	4295
4296	79		.2510, .2256, .2069, .1814, .1581, .1383, .1231, .1103,	4296
4297	80		.2560, .2282, .2091, .1822, .1585, .1385, .1226, .1083,	4297
4298	81		.2605, .2312, .2111, .1829, .1588, .1386, .1222, .1070,	4298
4299	82		.2677, .2358, .2129, .1836, .1592, .1386, .1218, .1071,	4299
4300	83		.2759, .2403, .2145, .1857, .1598, .1389, .1219, .1078,	4300
4301	84		.2834, .2445, .2160, .1878, .1603, .1394, .1223, .1084,	4301
4302	85		.2905, .2484, .2175, .1898, .1613, .1399, .1226, .1090,	4302
4303	86		.2963, .2531, .2199, .1918, .1625, .1407, .1230, .1096,	4303
4304	87		.4323, .3582, .3109, .2889, .2803, .2706, .2410, .2224,	4304
4305	88		.4610, .4026, .3624, .3212, .2926, .2551, .2375, .2015/	4305
4306	89		DATA G5/ .4199, .3837, .3401, .2979, .2623, .2318, .2108, .1854,	4306
4307	90		.3924, .3642, .3194, .2760, .2427, .2157, .1902, .1721,	4307
4308	91		.3794, .3479, .3025, .2673, .2311, .2019, .1842, .1613,	4308
4309	92		.3674, .3448, .2961, .2593, .2255, .1994, .1785, .1594,	4309
4310	93		.3573, .3443, .2910, .2517, .2293, .2006, .1843, .1679,	4310
4311	94		.3661, .3438, .2935, .2597, .2336, .2225, .2143, .2116,	4311
4312	95		.3674, .3435, .3080, .2728, .2606, .2577, .2573, .2573,	4312
4313	96		.3685, .3453, .3210, .3014, .2942, .2933, .2932, .2932,	4313
4314	97		.3814, .3612, .3341, .3276, .3257, .3253, .3252, .3252,	4314
4315	98		.3903, .3752, .3570, .3522, .3513, .3510, .3506, .3496,	4315
4316	99		.4012, .3899, .3782, .3751, .3743, .3741, .3734, .3713,	4316
4317	100		.4155, .4057, .3956, .3930, .3920, .3913, .3907, .3890,	4317
4318	101		.4290, .4205, .4118, .4092, .4077, .4065, .4059, .4047,	4318
4319	102		.5411, .5385, .5359, .5353, .5351, .5350, .5350, .5350/	4319
4320	103		DATA G6/ .5823, .5812, .5801, .5797, .5796, .5797, .5797, .5797,	4320
4321	104		.6096, .6090, .6085, .6082, .6082, .6083, .6083, .6083,	4321
4322	105		.6308, .6306, .6305, .6303, .6303, .6305, .6305, .6305,	4322
4323	106		.6481, .6483, .6485, .6483, .6484, .6486, .6487, .6487,	4323
4324	107		.6627, .6632, .6637, .6636, .6637, .6640, .6640, .6640,	4324
4325	108		.6754, .6761, .6769, .6768, .6770, .6773, .6773, .6773,	4325
4326	109		.6866, .6875, .6885, .6884, .6886, .6890, .6890, .6890,	4326
4327	110		.6966, .6977, .6989, .6989, .6991, .6995, .6995, .6995,	4327
4328	111		.7056, .7070, .7083, .7083, .7085, .7090, .7090, .7090,	4328
4329	112		.7139, .7154, .7169, .7169, .7172, .7176, .7177, .7177,	4329
4330	113		.7214, .7231, .7248, .7248, .7251, .7256, .7256, .7256,	4330
4331	114		.7285, .7303, .7321, .7321, .7325, .7330, .7330, .7330,	4331
4332	115		.7350, .7370, .7390, .7390, .7393, .7398, .7399, .7399,	4332
4333	116		.7411, .7432, .7453, .7454, .7457, .7463, .7463, .7463/	4333
4334	117		DATA G7/ .8069, .8103, .8138, .8139, .8145, .8152, .8153, .8153,	4334
4335	118		.8454, .8496, .8538, .8540, .8547, .8556, .8557, .8557,	4335

```

4336 119      | .8727,.8774,.8822,.8825,.8832,.8842,.8843,.8843,      4336
4337 120      | .8938,.8990,.9042,.9046,.9054,.9064,.9065,.9065,      4337
4338 121      | .9111,.9166,.9222,.9226,.9235,.9246,.9247,.9247,      4338
4339 122      | .9258,.9316,.9374,.9379,.9387,.9399,.9400,.9400,      4339
4340 123      | .9384,.9445,.9506,.9511,.9520,.9532,.9533,.9533,      4340
4341 124      | .9496,.9559,.9622,.9627,.9637,.9649,.9650,.9650,      4341
4342 125      | .9596,.9661,.9727,.9731,.9741,.9754,.9755,.9755,      4342
4343 126      | .9686,.9753,.9821,.9826,.9836,.9849,.9850,.9850,      4343
4344 127      | .9769,.9837,.9906,.9912,.9922,.9936,.9937,.9937,      4344
4345 128      | .9845,.9915,.9986,.9991,.9999,.9999,.9999,.9999,      4345
4346 129      | .9915,.9987,.9999,.9999,.9999,.9999,.9999,.9999,      4346
4347 130      | .9981,.9999,.9999,.9999,.9999,.9999,.9999,.9999/      4347
4348 131      C -----
4349 132      C REAL AIR EOS, TABLE LOOKUP ON GILMORE DATA. (NO TEMP. MODEL)      4349
4350 133      C TO AVOID COSTLY LOGARITHMIC FUNCTIONS THE TABLE "G" IS STORED IN A      4350
4351 134      C FORM SO THAT THE HEXADECIMAL WORD STRUCTURE OF A 32 BIT MACHINE      4351
4352 135      C MAY BE EXPLOITED.
4353 136      C THIS LOGIC MAY BE TRANSFERED TO OTHER MACHINES BY RECALCULATING      4353
4354 137      C THE TABLE "G" APPROPRIATE TO THE WORD ARCITECTURE OF THAT MACHINE.      4354
4355 138      C MACHINE DEPENDENT FUNCTIONS AND KEY NUMBERS MUST ALSO BE CHANGED.      4355
4356 139      C -----
4357 140      RL16E = 1./XL16E
4358 141      IST = 0
4359 142      NR = N
4360 143      C
4361 144      10 CONTINUE
4362 145      NST = MINO(NR,M)
4363 146      C
4364 147      DO 20 IRE=1,NST
4365 148      RHO(IRE) = .774413*RRR(IST+IRE)
4366 149      E(IRE) = AMAX1(3.e8,10000.*EEE(IST+IRE)/RRR(IST+IRE))
4367 150      C
4368 151      C CALCULATE MASS DENSITY VARIATION INDEX "I".
4369 152      C
4370 153      TEM = ALOG(RHO(IRE))*RL16E + 500.0
4371 154      I(IRE) = AINT(TEM)
4372 155      OMP(IRE) = TEM - FLOAT(I(IRE))
4373 156      I(IRE) = 502 - I(IRE)
4374 157      I(IRE) = MAXO(I(IRE),1)
4375 158      C
4376 159      C CALCULATE INTERNAL ENERGY VARIATION INDEX "J".
4377 160      C
4378 161      TEM = ALOG(E(IRE))*RL16E
4379 162      JCY = AINT(TEM)
4380 163      TEM = TEM - FLOAT(JCY)
4381 164      TEM = EXP(XL16E*TEM)
4382 165      JCY = JCY - 7
4383 166      JS = AINT(TEM)
4384 167      Q(IRE) = TEM - FLOAT(JS)
4385 168      J(IRE) = JS + 15*JCY
4386 169      J(IRE) = MINO(J(IRE),104)
4387 170      J(IRE) = I(IRE) + 8*J(IRE)
4388 171      I(IRE) = J(IRE) - 8
4389 172      20 CONTINUE
4390 173      C
4391 174      DO 30 IRE=1,NST
4392 175      T11(IRE) = GF(I(IRE))
4393 176      T21(IRE) = GF(I(IRE)+1)
4394 177      T12(IRE) = GF(J(IRE))
4395 178      T22(IRE) = GF(J(IRE)+1)
4396 179      30 CONTINUE
4397 180      C
4398 181      C CALCULATE GAMMA BY LINEAR INTERPOLATION.
4399 182      C
4400 183      DO 40 IRE=1,NST
4401 184      T12(IRE) = T12(IRE) - T11(IRE)
4402 185      T22(IRE) = T22(IRE) - T21(IRE)
4403 186      GAMMA(IST+IRE) = OMP(IRE) *(T11(IRE) + Q(IRE)*T12(IRE))
4404 187      + (1. - OMP(IRE))*(T21(IRE) + Q(IRE)*T22(IRE))
4405 188      + 1.
4406 189      40 CONTINUE
4407 190      C
4408 191      NR = NR - NST
4409 192      IST = IST + NST

```

```

4410 193      IF(NR.GT.0) GO TO 10
4411 194      C
4412 195      RETURN
4413 196      END
4414 197      C

```

```

4415 1      SUBROUTINE MATRLA
4416 2      C
4417 3      C READS MATERIAL PROPERTIES
4418 4      C EVALUATES MATERIAL RELATED CONTANTS
4419 5      C
4420 6      include 'dmtr10.h'
4421 7      CHARACTER*8 PRODC(15), PHASE(15), GG, SS, VV, NASAP
4422 8      REAL X(15), KVOL(15), M(15), RHOS(15), CVS(15)
4423 9      REAL CF(7,2), CV(0:5,2)
4424 10     C
4425 11     DATA KVOL/15*0./
4426 12     DATA GG/'G'/. SS/'S'/. VV/'VVVV'/
4427 13     DATA CV/12*0./
4428 14     DATA PRODC/'O2','N2',13*'VVVV' '/'
4429 15     DATA PHASE/2*'G',13*' '/'
4430 16     C
4431 17     C DATA PRODC/'H2O','H2','CO2','CO','NH3','CH4','N2','Cs'
4432 18     C I ,7*'VVVV' '/'
4433 19     C DATA PHASE/7*'G','S',7*' '/'
4434 20     C
4435 21     ALFAA=.5
4436 22     BETAA=.09585
4437 23     THETAA=400.
4438 24     CAPPAA=12.685
4439 25     NNASA=100
4440 26     X(1)=21.
4441 27     X(2)=79.
4442 28     KVOL(1)=350.
4443 29     KVOL(2)=380.
4444 30     M(1)=32.
4445 31     M(2)=28.016
4446 32     CVS(1)=0.
4447 33     CVS(2)=0.
4448 34     RHOS(1)=0.
4449 35     RHOS(2)=0.
4450 36     NS = 0
4451 37     NG = 0
4452 38     C
4453 39     TMS = 0.
4454 40     COVA = 0.
4455 41     GML = 0.
4456 42     SML = 0.
4457 43     SV = 0.
4458 44     SCVA = 0.
4459 45     C
4460 46     REWIND 4
4461 47     DO 110 I = 1, 15
4462 48     IF ( PRODC(I) .EQ. VV ) GO TO 10
4463 49     NS = I
4464 50     C
4465 51     IF ( PHASE(I) .EQ. GG ) THEN
4466 52     NG = NG + 1
4467 53     GML = GML + X(I)
4468 54     TMS = TMS + X(I)*M(I)
4469 55     COVA = COVA + X(I)*KVOL(I)
4470 56     C
4471 57     ELSE IF ( PHASE(I) .EQ. SS ) THEN
4472 58     PHASE(I) = VV
4473 59     SML = SML + X(I)
4474 60     TMS = TMS + X(I)*M(I)
4475 61     SCVA = SCVA + X(I)*CVS(I)
4476 62     SV = SV + X(I)*M(I)/RHOS(I)
4477 63     C
4478 64     ELSE
4479 65     STOP ' PRODUCTS EITHER SOLID, S, OR GAS, G'
4480 66     C

```

```

4481      67      END IF                                4481
4482      68      110 CONTINUE                          4482
4483      69      C                                      4483
4484      70      10  IF ( NS .LT. 1 ) STOP ' NO PRODUCTS ? ' 4484
4485      71      C                                      4485
4486      72      COVA = COVA * CAPPAA / GML              4486
4487      73      FSA = TMS/AMAX1(SV,1.E-15)             4487
4488      74      TML = GML + SML                        4488
4489      75      XGA = GML/TML                          4489
4490      76      SCVA = SCVA/TML                        4490
4491      77      WMA = TMS/TML                          4491
4492      78      C                                      4492
4493      79      DO 130 INASA = 1, NNASA                 4493
4494      80      IF ( NG .EQ. 0 ) GO TO 20              4494
4495      81      1  READ (4,1001) NASAP, ID             4495
4496      82      1001 FORMAT(A8,71X,11)                4496
4497      83      IF ( ID .NE. 1 ) GO TO 1              4497
4498      84      C                                      4498
4499      85      DO 120 I = 1, NS                       4499
4500      86      IF ( NASAP .EQ. PRODC(I) .AND. PHASE(I) .EQ. GG ) THEN 4500
4501      87      PHASE(I) = VV                          4501
4502      88      NG = NG - 1                             4502
4503      89      READ (4,1002) ((CF(K,KK),K=1,7),KK=1,2) 4503
4504      90      1002 FORMAT(5E15.8)                   4504
4505      91      C                                      4505
4506      92      CF(1,1) = CF(1,1) - 1.                 4506
4507      93      CF(1,2) = CF(1,2) - 1.                 4507
4508      94      DO 115 K = 0, 5                       4508
4509      95      CV(K,1) = CV(K,1) + (X(I)/GML)*CF(K+1,1) 4509
4510      96      115 CV(K,2) = CV(K,2) + (X(I)/GML)*CF(K+1,2) 4510
4511      97      C                                      4511
4512      98      END IF                                4512
4513      99      120 CONTINUE                          4513
4514      100     C                                      4514
4515      101     130 CONTINUE                          4515
4516      102     C                                      4516
4517      103     20  DO 140 I = 1, NS                   4517
4518      104     IF ( PHASE(I) .NE. VV ) STOP ' SPECIES NOT FOUND IN NASA ' 4518
4519      105     140 CONTINUE                          4519
4520      106     C                                      4520
4521      107     DO 150 I = 3, 50                       4521
4522      108     TA(I) = FLOAT(100*I)                   4522
4523      109     C                                      4523
4524      110     CALL PSM ( CV(0,2),4, TA(3),8, CVMA(3) ) 4524
4525      111     CALL PSM ( CV(0,1),4, TA(11),40, CVMA(11) ) 4525
4526      112     C                                      4526
4527      113     DO 155 K = 1, 4                       4527
4528      114     CV(K,1) = CV(K,1)/FLOAT(K+1)          4528
4529      115     155 CV(K,2) = CV(K,2)/FLOAT(K+1)          4529
4530      116     C                                      4530
4531      117     CALL PSM ( CV(0,2),4, TA(3),8, EMOEA(3) ) 4531
4532      118     CALL PSM ( CV(0,1),4, TA(11),40, EMOEA(11) ) 4532
4533      119     C                                      4533
4534      120     DO 160 I = 3, 10                      4534
4535      121     EMOEA(I) = TA(I)*EMOEA(I)              4535
4536      122     DO 161 I = 11, 50                    4536
4537      123     161 EMOEA(I) = TA(I)*EMOEA(I)          4537
4538      124     C                                      4538
4539      125     DO 180 I = 3, 50                      4539
4540      126     180 EMOEA(I) = EMOEA(I)*XGA + TA(I)*SCVA 4540
4541      127     C                                      4541
4542      128     CALL BILD (EMOEA,48,RANGEA,DYA)        4542
4543      129     C                                      4543
4544      130     RETURN                                  4544
4545      131     END                                    4545

```

```

Thu Jul 1 14:17:00 1993   threed.f           SUBROUTINE PSM           page 64
4546      1      SUBROUTINE PSM (A,NPOL, T,N, SMM)
4547      2      C
4548      3      REAL      A(0:NPOL), T(N), SMM(N)
4549      4      C
4550      5      DO 10 J = 1, N
4551      6      10     SMM(J) = A(NPOL)
4552      7      C
4553      8      DO 20 K = NPOL-1, 0, -1
4554      9      C
4555     10      DO 15 J = 1, N
4556     11     15     SMM(J) = SMM(J) * T(J) + A(K)
4557     12     C
4558     13     20     CONTINUE
4559     14     C
4560     15     RETURN
4561     16     END
4562     17     C

```

```

Thu Jul 1 14:17:00 1993   threed.f           SUBROUTINE BILD
4563      1      SUBROUTINE BILD(Y,N,RANGE,DY)
4564      2      C
4565      3      REAL Y(N),RANGE,DY(200)
4566      4      IF( N .GT. 201 ) STOP ' ONLY 201 POINTS ALLOWED '
4567      5      C
4568      6      RANGE = (Y(N+2) - Y(3)) / (N-1)
4569      7      DO 10 I = 1, N-1
4570      8      10     DY(I+2) = Y(I+3) - Y(I+2)
4571      9      10     CONTINUE
4572     10     C
4573     11     RETURN
4574     12     END

```

```

Thu Jul 1 14:17:00 1993   threed.f           SUBROUTINE MATRLX
4575      1      SUBROUTINE MATRLX
4576      2      C
4577      3      C      READS MATERIAL PROPERTIES
4578      4      C      EVALUATES MATERIAL RELATED CONTANTS
4579      5      C
4580      6      include 'dmtr10.h'
4581      7      CHARACTER*8  PRODC(15), PHASE(15), GG, SS, VV, NASAP
4582      8      REAL      X(15), KVOL(15), M(15), RHOS(15), CVS(15)
4583      9      REAL      CF(7,2), CV(0:5,2)
4584     10     C
4585     11     DATA      KVOL/15*0./
4586     12     DATA      GG/'G'/, SS/'S'/, VV/'VVVV'/
4587     13     DATA      CV/12*0./
4588     14     C
4589     15     DATA PRODC/'H2O','CO2','CO','N2','Cs',10*'VVVV'  '/'
4590     16     DATA PHASE/4*'G','S',10*'  '/
4591     17     C
4592     18     ALFAX=.5
4593     19     BETAX=.09585
4594     20     THETAX=400.
4595     21     CAPPAX=12.685
4596     22     NNASA=100
4597     23     X(1)=2.5
4598     24     X(2)=1.66
4599     25     X(3)=.188
4600     26     X(4)=1.5
4601     27     X(5)=5.15
4602     28     KVOL(1)=250.
4603     29     KVOL(2)=600.
4604     30     KVOL(3)=390.
4605     31     KVOL(4)=380.
4606     32     KVOL(5)=0.
4607     33     M(1)=18.
4608     34     M(2)=44.
4609     35     M(3)=28.
4610     36     M(4)=28.
4611     37     M(5)=12.
4612     38     CVS(5)=1.1
4613     39     RHOS(5)=2.6

```

```

4614 40      NS = 0
4615 41      NG = 0
4616 42      C
4617 43      TMS = 0.
4618 44      COVX = 0.
4619 45      GML = 0.
4620 46      SML = 0.
4621 47      SV = 0.
4622 48      SCVX = 0.
4623 49      C
4624 50      REWIND 4
4625 51      DO 110 I = 1, 15
4626 52      IF ( PRODC(T) .EQ. VV ) GO TO 10
4627 53      NS = I
4628 54      C
4629 55      IF ( PHASE(I) .EQ. GG ) THEN
4630 56      NG = NG + 1
4631 57      GML = GML + X(I)
4632 58      TMS = TMS + X(I)*M(I)
4633 59      COVX = COVX + X(I)*KVOL(I)
4634 60      C
4635 61      ELSE IF ( PHASE(I) .EQ. SS ) THEN
4636 62      PHASE(I) = VV
4637 63      SML = SML + X(I)
4638 64      TMS = TMS + X(I)*M(I)
4639 65      SCVX = SCVX + X(I)*CVS(I)
4640 66      SV = SV + X(I)*M(I)/RHOS(I)
4641 67      C
4642 68      ELSE
4643 69      STOP ' PRODUCTS EITHER SOLID, S, OR GAS, G'
4644 70      C
4645 71      END IF
4646 72 110 CONTINUE
4647 73      C
4648 74 10 IF ( NS .LT. 1 ) STOP ' NO PRODUCTS ?'
4649 75      C
4650 76      COVX = COVX * CAPPAX / GML
4651 77      FSX = TMS/AMAX1(SV,1.E-15)
4652 78      TML = GML + SML
4653 79      XGX = GML/TML
4654 80      SCVX = SCVX / TML
4655 81      WMX = TMS/TML
4656 82      C
4657 83      DO 130 INASA = 1, NNASA
4658 84      IF ( NG .EQ. 0 ) GO TO 20
4659 85 1 READ (4,1001) NASAP, ID
4660 86 1001 FORMAT(A8,71X,I1)
4661 87      IF ( ID .NE. 1 ) GO TO 1
4662 88      C
4663 89      DO 120 I = 1, NS
4664 90      IF ( NASAP .EQ. PRODC(T) .AND. PHASE(I) .EQ. GG ) THEN
4665 91      PHASE(I) = VV
4666 92      NG = NG - 1
4667 93      READ (4,1002) ((CF(K,KK),K=1,7),KK=1,2)
4668 94 1002 FORMAT(5E15.8)
4669 95      C
4670 96      CF(1,1) = CF(1,1) - 1.
4671 97      CF(1,2) = CF(1,2) - 1.
4672 98      DO 115 K = 0, 5
4673 99      CV(K,1) = CV(K,1) + (X(I)/GML)*CF(K+1,1)
4674 100 115 CV(K,2) = CV(K,2) + (X(I)/GML)*CF(K+1,2)
4675 101      C
4676 102      END IF
4677 103 120 CONTINUE
4678 104      C
4679 105 130 CONTINUE
4680 106      C
4681 107 20 DO 140 I = 1, NS
4682 108      IF ( PHASE(I) .NE. VV ) STOP ' SPECIES NOT FOUND IN NASA'
4683 109 140 CONTINUE
4684 110      C
4685 111      DO 150 I = 3, 50
4686 112 150 TX(I) = FLOAT(100*I)
4687 113      C

```



```

4688 114 CALL PSM ( CV(0,2),4, TX(3),8, CVMX(3) )
4689 115 CALL PSM ( CV(0,1),4, TX(11),40, CVMX(11) )
4690 116 C
4691 117 DO 155 K = 1, 4
4692 118 CV(K,1) = CV(K,1)/FLOAT(K+1)
4693 119 155 CV(K,2) = CV(K,2)/FLOAT(K+1)
4694 120 C
4695 121 CALL PSM ( CV(0,2),4, TX(3),8, EMEOX(3) )
4696 122 CALL PSM ( CV(0,1),4, TX(11),40, EMEOX(11) )
4697 123 C
4698 124 DO 160 I = 3, 10
4699 125 160 EMEOX(I) = TX(I)*EMEOX(I)
4700 126 DO 161 I = 11, 50
4701 127 161 EMEOX(I) = TX(I)*EMEOX(I)
4702 128 C
4703 129 DO 180 I = 3, 50
4704 130 180 EMEOX(I) = EMEOX(I)*XGX + TX(I)*SCVX
4705 131 C
4706 132 CALL BILD (EMEOX,48,RANGEX,DYX)
4707 133 C
4708 134 RETURN
4709 135 END

```

```

4710 1 SUBROUTINE VOLMTETC ( I1, I2, I3, X, Y, Z, VOLUMT )
4711 2 C
4712 3 C-----I
4713 4 C I
4714 5 C VOLMTETC FINDS THE VOLUME OF THE TETRAHEDRON DEFINED BY THE I
4715 6 C GRID VERTICES I1, I2, I3, AND THE POINT (X, Y, Z). I
4716 7 C THE CODE ASSUMES THAT THE AREAL VECTOR OF THE BASE TRIANGLE I
4717 8 C FORMED BY I1, I2 AND I3 POINTS IN THE DIRECTION OF (X, Y, Z): I
4718 9 C BY THE RIGHT HAND RULE, IF I1, I2 AND I3 ARE ARRANGED I
4719 10 C COUNTER-CLOCKWISE AS VIEWED FROM ABOVE THE PLANE OF THE I
4720 11 C TRIANGLE, (X, Y, Z) ALSO LIES ABOVE THE PLANE). BUT NOTE -- I
4721 12 C I
4722 13 C THE VOLUME RETURNED IS A SIGNED QUANTITY - IE. I
4723 14 C IF THE VERTICES ARE NOT ORDERED BY THE RIGHT I
4724 15 C HAND RULE THE VOLUME WILL BE NEGATIVE. I
4725 16 C I
4726 17 C I
4727 18 C DECEMBER, 1991: M. FRITTS, FRITTS%MCL.SAINET@CCC.NERSC.GOV, I
4728 19 C (301) 266-0992 I
4729 20 C I
4730 21 C-----I
4731 22 C I
4732 23 C-----I
4733 24 C I
4734 25 DOUBLE PRECISION R21X,R21Y,R21Z,R31X,R31Y,R31Z,R41X,R41Y,R41Z I
4735 26 DOUBLE PRECISION VOLUMT,X,Y,Z I
4736 27 C I
4737 28 C I
4738 29 include 'dmsh00.h' I
4739 30 C I
4740 31 C-----I
4741 32 C I
4742 33 C --- FIND THE VOLUME OF THE TETRAHEDRON.-----I
4743 34 C I
4744 35 R21X = XV(1,I2) - XV(1,I1) I
4745 36 R21Y = XV(2,I2) - XV(2,I1) I
4746 37 R21Z = XV(3,I2) - XV(3,I1) I
4747 38 R31X = XV(1,I3) - XV(1,I1) I
4748 39 R31Y = XV(2,I3) - XV(2,I1) I
4749 40 R31Z = XV(3,I3) - XV(3,I1) I
4750 41 R41X = X - XV(1,I1) I
4751 42 R41Y = Y - XV(2,I1) I
4752 43 R41Z = Z - XV(3,I1) I
4753 44 C I
4754 45 VOLUMT = ( R41X*( R21Y*R31Z - R21Z*R31Y ) - I
4755 46 1 R41Y*( R21X*R31Z - R21Z*R31X ) + I
4756 47 2 R41Z*( R21X*R31Y - R21Y*R31X ) )/6.00 I
4757 48 C I
4758 49 C I

```

```
4759 50 C -----  
4760 51 C RETURN  
4761 52 C -----  
4762 53 C  
4763 54 C ---  
4764 55 C END  
4765 56 C
```

```
4759  
4760  
4761  
4762  
4763  
4764  
4765
```

Thu Jul 1 14:15:40 1993

mainhd.f

Module List - order of occurrence

page i

#	routine	page
1	AUGUST	1
2	HYDRFL	13
3	HYDRMN	18
4	GEOMTR	25
5	UPDATE	29
6	UPGRAD	30

Thu Jul 1 14:15:40 1993

mainhd.f

Module List - alphabetical order

#	routine	page
1	AUGUST	1
2	GEOMTR	25
3	HYDRFL	13
4	HYDRMN	18
5	UPDATE	29
6	UPGRAD	30

```

1      1      PROGRAM AUGUST
2      2      C-----
3      3      C
4      4      C      The AUGUST Code
5      5      C
6      6      C          - Adaptive
7      7      C          - Unstructured
8      8      C          - Godunov
9      9      C          - Upwind
10     10     C          - Second order
11     11     C          - Triangular
12     12     C
13     13     C          The geometry structure comes from BERMUDA
14     14     C          The solver is based on FUGGS
15     15     C
16     16     C          Version:  2.00      june 17, 1991
17     17     C
18     18     C          Authors:  Itzhak Lottati  (703)749-8648
19     19     C                   Shmuel Eidelman (703)448-6491
20     20     C                   Adam Drobot   (703)734-5840
21     21     C
22     22     C          Science Applications International Corporation
23     23     C          Applied Physics Operation
24     24     C          1710 Goodridge Drive
25     25     C          McLean, Virginia 22102
26     26     C
27     27     C-----
28     28     C
29     29     C
30     30     C
31     31     C-----I
32     32     C          I
33     33     C          BERMUDA IS A MULTIDIMENSIONAL CODE WHICH IS BASED ON THE I
34     34     C          USE OF TRIANGULAR GRIDS AS THE FUNDAMENTAL MESH I
35     35     C          FOR FIELD LIKE QUANTITIES. THE CODE REQUIRES I
36     36     C          THAT ALL QUANTITIES ARE BASED AT THE BARICENTER I
37     37     C          OF SIDES/TRIANGLES. I
38     38     C          I
39     39     C          THE QUIP IS THAT THOSE WHO WORK ON BERMUDA I
40     40     C          TRIANGLES ARE NEVER HEARD FROM AGAIN. I
41     41     C          I
42     42     C          THE BASIC MODULES IN BERMUDA INCLUDE: I
43     43     C          I
44     44     C          A HYDRODYNAMICS CODE I
45     45     C          .BASED ON A FIRST ORDER GODUNOV I
46     46     C          METHOD OR A SECOND ORDER GODUNOV I
47     47     C          WITH MESH ADAPTATION. I
48     48     C          I
49     49     C-----I
50     50     C
51     51     C          GRID SETUP TABLES AND THEIR MEANING:
52     52     C
53     53     C          +-----+
54     54     C          +
55     55     C          + LIST OF VERTICES
56     56     C          +
57     57     C          + IV - VERTEX INDEX
58     58     C          + JV(1,IV) - S STATUS OF THE POINT
59     59     C          + S=0 FREE POINT WHICH MAY BE
60     60     C          + DELETED/MOVED
61     61     C          + S=1 POINT RESTRICTED TO A SURFACE
62     62     C          + S=2 POINT RESTRICTED TO A LINE
63     63     C          + S=3 POINT WHICH IS FIXED AND MAY
64     64     C          + NOT BE REMOVED
65     65     C          + JV(2,IV) - INDEX OF A LINE WHICH INCLUDE THE
66     66     C          + POINT
67     67     C          + NEGATIVE MEANS THE POINT IS ON A
68     68     C          + BOUNDARY LINE
69     69     C          + XV(1,IV) - X POSITION OF VERTEX
70     70     C          + XV(2,IV) - Y POSITION OF VERTEX
71     71     C          +
72     72     C          +-----+
73     73     C

```

```

74 74 C +-----+ 74
75 75 C + 75
76 76 C + LIST OF EDGES + 76
77 77 C + + 77
78 78 C + IE - EDGE INDEX + 78
79 79 C + JE(1,IE) - INDEX OF LOWER EDGE VERTEX + 79
80 80 C + JE(2,IE) - INDEX OF UPPER EDGE VERTEX + 80
81 81 C + + 81
82 82 C + FOR TWO DIMENSIONAL PROBLEMS + 82
83 83 C + + 83
84 84 C + JE(3,IE) - INDEX OF LEFT SIDE + 84
85 85 C + JE(4,IE) - INDEX OF RIGHT SIDE + 85
86 86 C + + 86
87 87 C + IF JE(3-4,IE) IS NEGATIVE THIS INDICATES THAT THE + 87
88 88 C + EDGE LIES ALONG A BOUNDARY. + 88
89 89 C + + 89
90 90 C + JE(5,IE) - INDEX DEFINING BOUNDARY CONDITION + 90
91 91 C + 0 - ORDINARY EDGE INTERIOR + 91
92 92 C + 6 - WALL V-PERPENDICULAR=0 + 92
93 93 C + 7 - SUPERSONIC OUTFLOW + 93
94 94 C + 8 - INFLOW BOUNDARY + 94
95 95 C + + 95
96 96 C + XE(1,IE) - LENGTH OF EDGE + 96
97 97 C + XE(2,IE) - DISTANCE BETWEEN ADJOINING SIDE + 97
98 98 C + POINTS. + 98
99 99 C + + 99
100 100 C +-----+ 100
101 101 C 101
102 102 C +-----+ 102
103 103 C + LIST OF SIDES + 103
104 104 C + + 104
105 105 C + IS - SIDE INDEX + 105
106 106 C + JS(1,IS) - INDEX OF FIRST VERTEX + 106
107 107 C + JS(2,IS) - INDEX OF SECOND VERTEX + 107
108 108 C + JS(3,IS) - INDEX OF THIRD VERTEX + 108
109 109 C + + 109
110 110 C + THE VERTICES RUN AROUND THE SIDE IN ORDER + 110
111 111 C + COUNTER-CLOCKWISE FASHION + 111
112 112 C + + 112
113 113 C + JS(4,IS) - INDEX OF THE FIRST EDGE + 113
114 114 C + JS(5,IS) - INDEX OF THE SECOND EDGE + 114
115 115 C + JS(6,IS) - INDEX OF THE THIRO EDGE + 115
116 116 C + + 116
117 117 C + THE EDGES ARE ARRANGED IN COUNTER-CLOCKWISE + 117
118 118 C + FASHION. EDGE ONE RUNS FROM VERTEX-ONE TO + 118
119 119 C + VERTEX-TWO ETC.. THE SIGN OF JS(4-6,IS) INDICATES + 119
120 120 C + IF EDGE DATA IS STORED THE SAME WAY. IF IT IS + 120
121 121 C + JS>0 AND IT IS REVERSED JS<0 + 121
122 122 C + + 122
123 123 C + + 123
124 124 C + XS(1,IS) - X POSITION OF SIDE POINT + 124
125 125 C + XS(2,IS) - Y POSITION OF SIDE POINT + 125
126 126 C + XS(3,IS) - AREA OF SIDE + 126
127 127 C + + 127
128 128 C + + 128
129 129 C +-----+ 129
130 130 C 130
131 131 C ----- 131
132 132 C 132
133 133 C --- DEFINITION FOR ALL HYDRODYNAMIC QUANTITIES --- 133
134 134 C 134
135 135 C -----I 135
136 136 C I 136
137 137 C USE OF PARAMETERS: I 137
138 138 C I 138
139 139 C MHQ - MAXIMUM NUMBER OF HYDRO QUANTITIES. I 139
140 140 C I 140
141 141 C I 141
142 142 C -----I 142
143 143 C 143
144 144 C ----- 144
145 145 C 145
146 146 C include 'cmsh00.h' 146
147 147 C include 'chyd00.h' 147

```

```

148 148 include 'cint00.h' 148
149 149 include 'cphs10.h' 149
150 150 include 'cphs20.h' 150
151 151 C 151
152 152 C----- 152
153 153 C 153
154 154 NAMELIST /DATA/ ICOND,ICONP,ITRIGR,IOPTN, 154
155 155 . XMCHIN,RIN,PIN,ALFA,HRGG,IHRN,NTIME,MDUMP,NDUMP, 155
156 156 . KDUMP,IOSPCL,IOPFLT,IOPRCN,IOPORD,IOPBYN,IAXSYM, 156
157 157 . IOPEOS,MPRTCL,IOPINT,IOPADD,IOPDEL,AREADD,AREDEL, 157
158 158 . IWINDW,ISTATC 158
159 159 C 159
160 160 C----- 160
161 161 C 161
162 162 C 162
163 163 C --- MEANING OF NAMELIST VARIABLES: 163
164 164 C 164
165 165 C ICOND = 0 READ INPUT GRID FOR A NEW RUN 165
166 166 C = 1 READ THE GRID FROM PREVIOUS RUN 166
167 167 C ICONP = 0 PRIMITIVE VARIABLES SET TO ZERO 167
168 168 C = 1 VARIABLES READ FROM PREVIOUS RUN 168
169 169 C ITRIGR = 0 USING THE INPUT GRID AS THE INITIAL GRID 169
170 170 C = 1 THE INPUT GRID TRIPLED BY ADDING AN EXTRA VERTEX IN 170
171 171 C EACH TRIANGLE 171
172 172 C IOPTN = 1 SOLUTION FOR STEADY STATE, 172
173 173 C = 2 SOLUTION FOR TRANSIENT PHENOMENA 173
174 174 C 174
175 175 C XMCHIN = FOR TRANSIENT SHOCK CALCULATIONS(IOPTN=2)THIS VARIABLE 175
176 176 C IS USED TO SPECIFY THE UPSTREAM MACH NUMBER 176
177 177 C 177
178 178 C RIN = THE AMBIENT DENSITY IN THE CHAMBER 178
179 179 C 179
180 180 C PIN = THE AMBIENT PRESSURE IN THE CHAMBER 180
181 181 C 181
182 182 C APPLYING NORMAL SHOCK WAVES RELATIONS FOR AN ADIABATIC 182
183 183 C FLOW RELATION STATIC-PRESSURE RATIO ACROSS THE SHOCK 183
184 184 C AS WELL AS THE DENSITY RATIO AND MACH NUMBER RATIO 184
185 185 C ARE COMPUTED TO SET CORRECTLY THE CONDITION AT THE 185
186 186 C INLET EDGES( EDGE BOUNDARY 8 ) OF THE COMPUTATIONAL 186
187 187 C DOMAIN 187
188 188 C 188
189 189 C FOR STEADY STATE SHOCK CALCULATIONS(IOPTN=1)THIS IS THE 189
190 190 C INFLOW MACH NUMBER. ALL DOMAIN VELOCITIES ARE THEN 190
191 191 C INITIALIZED WITH THIS VALUE. 191
192 192 C 192
193 193 C RIN = THE AMBIENT DENSITY AT INFINITY 193
194 194 C 194
195 195 C PIN = THE AMBIENT PRESSURE AT INFINITY 195
196 196 C 196
197 197 C ALL COMPUTATIONAL DOMAIN ARE THEN INITIALIZED WITH 197
198 198 C THOSE VALUES. 198
199 199 C 199
200 200 C ALFA = THE DIRECTION OF INFLOW IN DEGREES RELATIVE TO A RIGHT 200
201 201 C HAND COORDINATE SYSTEM. ALFA=0 MEANS FLOW FROM LEFT TO 201
202 202 C RIGHT. ALFA=90 MEANS FROM BOTTOM TO TOP. ALFA=-90 OR 270 202
203 203 C MEANS FLOW FROM TOP TO BOTTOM ETC. 203
204 204 C HRGG = INITIAL GAMMA IN THE EQUATION OF STATE 204
205 205 C THE CODE RUNS USING THE AIR EQUATION AS A BASELINE AND 205
206 206 C SHOULD BE MODIFIED IF SOMETHING ELSE IS DESIRED. 206
207 207 C IHRN = NUMBER OF ITERATIONS IN THE RIEMANN SOLVER TO FIND THE 207
208 208 C DIAPHRAGM SOLUTION.(3 to 4 SHOULD BE USED AND INCREASED 208
209 209 C ONLY FOR HIGH MACH NUMBER CASES). 209
210 210 C 210
211 211 C NTIME = NUMBER OF REPEATS FOR THE INTEGRATION/REFINEMENT/ 211
212 212 C COARSENING SEQUENCE 212
213 213 C AN OUTPUT DUMP IS DONE EVERY SEQUENCE REPEAT. 213
214 214 C MDUMP = NUMBER OF OUTERMOST LOOP ITERATIONS IN THE CALCULATION 214
215 215 C WHERE COARSENING OF THE GRID IS PERFORMED EVERY SEQUENCE 215
216 216 C REPEAT. 216
217 217 C NDUMP = NUMBER OF OUTER LOOP ITERATIONS IN THE CALCULATION WHERE 217
218 218 C REFINING IS DONE EVERY SEQUENCE REPEAT WITHOUT COARSENING 218
219 219 C KDUMP = NUMBER OF ITERATIONS PERFORMED WITH NO REFINEMENT OR 219
220 220 C COARSENING. IT IS THE INNER LOOP OF THE CALCULATION. 220
221 221 C IF KDUMP=0 IS READ IN, KDUMP WILL BE SET BY THE 221

```

222	222	C	CODE AUTOMATICALLY ACCORDING TO THE VARIABLE AREADD.	I	222
223	223	C		I	223
224	224	C	+ _____ o NTIME - DUMPING DATA	I	224
225	225	C		I	225
226	226	C	+ _____ o MDUMP - COARSENING	I	226
227	227	C		I	227
228	228	C	+ _____ o NDUMP - REFINEMENT	I	228
229	229	C		I	229
230	230	C	+ _____ o KDUMP - INTEGRATION	I	230
231	231	C		I	231
232	232	C		I	232
233	233	C	+ _____ o INNER LOOP	I	233
234	234	C		I	234
235	235	C	+ _____ o OUTER LOOP	I	235
236	236	C		I	236
237	237	C	+ _____ o OUTERMOST LOOP	I	237
238	238	C		I	238
239	239	C	+ _____ o DUMPING LOOP	I	239
240	240	C		I	240
241	241	C	IOSPCL = 0 NOT USING REDEFINITION OF POINTS ON THE BOUNDARY	I	241
242	242	C	= 1 USING REDEFINITION OF POINTS ON THE BOUNDARY	I	242
243	243	C	IOPFLT = 0 THE COMPUTATION OF LIFT DRAG AND MOMENT TURNED OFF	I	243
244	244	C	= 1 THE COMPUTATION OF LIFT DRAG AND MOMENT TURNED ON	I	244
245	245	C	IOPRCN = 0 A GLOBAL SWAPING (RECONNECTION) PROCEDURE IS OFF	I	245
246	246	C	= 1 A GLOBAL SWAPING (RECONNECTION) PROCEDURE IS ON	I	246
247	247	C	IOPORD = 1 THE CODE WILL RUN FIRST ORDER GODUNOV METHOD	I	247
248	248	C	= 2 THE CODE WILL RUN SECOND ORDER GODUNOV METHOD	I	248
249	249	C	IOPBYN = 0 NO BUOYANCY EFFECT ARE COMPUTED	I	249
250	250	C	= 1 BUOYANCY EFFECT IN THE Y DIRECTION ARE COMPUTED	I	250
251	251	C	IAXSYM = 0 THE CODE WILL RUN IN A PURE TWO DIMENSIONAL MODE	I	251
252	252	C	= 1 THE CODE WILL RUN IN AN AXI SYMMETRICAL MODE (X AXIS)	I	252
253	253	C	= 2 THE CODE WILL RUN IN AN AXI SYMMETRICAL MODE (Y AXIS)	I	253
254	254	C	IOPEOS = 0 THE CODE WILL RUN WITH CONSTANT GAMA	I	254
255	255	C	= 1 THE CODE WILL RUN WITH VARIABLE GAMA USING EQUATION	I	255
256	256	C	OF STATE FOR AIR	I	256
257	257	C		I	257
258	258	C	MPRTCL = 0 NO PARTICLE TRACING	I	258
259	259	C	= 1 THE CODE WILL TRACE PARTICLES	I	259
260	260	C	IOPINT = 0 NOT REFINING INITIALLY THE EDGE BOUNDARY NO 8	I	260
261	261	C	= 1 REFINING INITIALLY THE EDGE BOUNDARY NO 8	I	261
262	262	C	IOPADD = 0 THE REFINEMENT PROCEDURE IS TURNED OFF	I	262
263	263	C	= 1 THE REFINEMENT PROCEDURE IS TURNED ON	I	263
264	264	C	IOPDEL = 0 THE COARSENING PROCEDURE IS TURNED OFF	I	264
265	265	C	= 1 THE COARSENING PROCEDURE IS TURNED ON	I	265
266	266	C	AREADD = SPECIFY THE MINIMUM VALUE THAT A TRIANGLE SHOULD HAVE	I	266
267	267	C	AFTER REFINEMENT AS A FRACTION OF AVERAGE TRIANGLE AREA	I	267
268	268	C	AREDEL = SPECIFY THE MAXIMUM VALUE THAT A TRIANGLE SHOULD HAVE	I	268
269	269	C	AFTER COARSENING AS A FRACTION OF AVERAGE TRIANGLE AREA	I	269
270	270	C		I	270
271	271	C	IWINDOW = 0 NO RESTRICTION ON THE REGION FOR REFINING THE GRID	I	271
272	272	C	= 1 SETTING A WINDOW FOR REFINING THE GRID	I	272
273	273	C	ISTATC = 0 THE ADAPTATION WILL BE DONE ON A MOVING WAVE	I	273
274	274	C	= 1 THE ADAPTATION WILL BE DONE ON A STEADY STATE	I	274
275	275	C	CONDITION	I	275
276	276	C		I	276
277	277	C	-----	I	277
278	278	C		I	278
279	279	C	CHARACTER*15 ZHEADER,MNAME,MVNAME	I	279
280	280	C	CHARACTER*1 FILLCH	I	280
281	281	C	INTEGER NUMQUADS	I	281
282	282	C		I	282
283	283	C	--- OPEN ALL FILES FOR THIS RUN -----	I	283
284	284	C		I	284
285	285	C	OPEN(4,FILE='naca4' ,FORM='UNFORMATTED')	I	285
286	286	C	OPEN(88,FILE='naca82' ,FORM='UNFORMATTED')	I	286
287	287	C	OPEN(8,FILE='naca2' ,FORM='UNFORMATTED')	I	287
288	288	C	OPEN(9,FILE='naca3' ,FORM='UNFORMATTED')	I	288
289	289	C	OPEN(2,FILE='data.d' ,FORM='FORMATTED')	I	289
290	290	C	OPEN(16,FILE='wedge45.zon',STATUS='OLD')	I	290
291	291	C	OPEN(18,FILE='nacaa' ,FORM='UNFORMATTED')	I	291
292	292	C		I	292
293	293	C	-----	I	293
294	294	C		I	294
295	295	C	--- DEFAULT VALUES FOR INPUT DATA -----	I	295

```

296 296 C
297 297   THIRD = 1. / 3.
298 298 C
299 299   ICOND = 0
300 300   ICONP = 0
301 301   ITRIGR = 0
302 302   IOPTN = 1
303 303 C
304 304   XMCHIN = 25.
305 305   RIN = 1.
306 306   PIN = 1.
307 307 C
308 308   ALFA = 0.
309 309   HRGG = 1.4
310 310   IHRN = 4
311 311   NTIME = 1
312 312   MDUMP = 80
313 313   NDUMP = 1
314 314   KDUMP = 0
315 315   IOSPLC = 0
316 316   IOPLFT = 0
317 317   IOPRCN = 0
318 318   IOPORD = 2
319 319   IOPBYN = 0
320 320   IAXSYM = 0
321 321   IOPEOS = 0
322 322 C
323 323   MPRTCL = 0
324 324   IOPINT = 0
325 325   IOPADD = 0
326 326   IOPDEL = 0
327 327   AREADD = 0.005
328 328   AREDEL = 1.
329 329   IWINDW = 0
330 330   ISTATC = 0
331 331 C
332 332 C --- READ THE INPUT DATA -----
333 333 C
334 334   READ (2,DATA)
335 335 C
336 336 C --- PRINTOUT THE RUN PARAMETERS -----
337 337 C
338 338   PRINT 101,   ICOND,ICONP,ITRIGR,IOPTN,
339 339   .           XMCHIN,RIN,PIN,ALFA,HRGG,IHRN,NTIME,MDUMP,NDUMP,
340 340   .           KDUMP,IOSPLC,IOPLFT,IOPRCN,IOPORD,IOPBYN,IAXSYM,
341 341   .           IOPEOS,MPRTCL,IOPINT,IOPADD,IOPDEL,AREADD,AREDEL,
342 342   .           IWINDW,ISTATC
343 343 C
344 344 C --- SET RUN CONDITIONS AND PRINTOUT TO CONSOLE -----
345 345 C
346 346   XREADD = 1. / AREADD
347 347   NAREAD = ALOG( XREADD ) / ALOG( 3. ) + 1
348 348   IF( NAREAD . LT . 3 ) NAREAD = 3
349 349   IF( NAREAD . GT . 5 ) NAREAD = 5
350 350   IF( ISTATC . EQ . 1 ) NAREAD = 3
351 351   PRINT*,AREADD,AREDEL,NAREAD
352 352   PRINT * ,ICOND,ICONP
353 353 C
354 354   NPT = 0
355 355   IJKINT = 3
356 356   IF( ICOND . EQ . 0 ) THEN
357 357     DO 122 IS = 1 , MSM
358 358     KSDEL( IS ) = 0
359 359   122 CONTINUE
360 360   END IF
361 361   HYDMOM( 1 ) = 0.
362 362   HYDMOM( 2 ) = 0.
363 363   HYDMOM( 4 ) = 0.
364 364 C
365 365   DO 124 IK = 1 , MBP
366 366   GAMAG( IK ) = HRGG
367 367   124 CONTINUE
368 368 C
369 369 C-----

```



```

370 370 C 370
371 371 C --- READ IN THE MESH DATA ----- 371
372 372 C 372
373 373 C(1)>>>> 373
374 374 IF( ICOND . EQ . 0 ) THEN 374
375 375 IF( ICONP . EQ . 1 ) CALL UPGRAD 375
376 376 C 376
377 377 C----- 377
378 378 C A "SMART" FORMAT MESH FILE IS READ. THE FILE IS SELECTED BY THE 378
379 379 C NORMAL MACINTOSH FILE DIALOG BECAUSE OF THE '*' IN PLACE OF THE 379
380 380 C FILE NAME. VERTICES OF EACH TRIANGLE ARE FORMED FROM THE INPUT. 380
381 381 C----- 381
382 382 READ (16,900) ZHEADER 382
383 383 900 FORMAT(A15) 383
384 384 IF (ZHEADER .NE. 'SMaRT-Z-T-(003)') THEN 384
385 385 C----- THIS ROUTINE CANNOT READ ANY OTHER INPUT 385
386 386 PRINT *, 'MESH FILE IS NOT THE CORRECT KIND OR VERSION' 386
387 387 CALL EXIT 387
388 388 ENDIF 388
389 389 READ (16,910) FILLCH,NV,NVMK 389
390 390 PRINT *,NV,NVMK 390
391 391 READ (16,910) FILLCH,NE,NEMK 391
392 392 PRINT *,NE,NEMK 392
393 393 READ (16,910) FILLCH,NS 393
394 394 PRINT *,NS 394
395 395 READ (16,910) FILLCH,NUMQUADS 395
396 396 PRINT *,NUMQUADS 396
397 397 910 FORMAT(A1,2I7) 397
398 398 READ (16,920) FILLCH,NZMK,NSMK,NNMK 398
399 399 PRINT *,NZMK,NSMK,NNMK 399
400 400 920 FORMAT(A1,1X,3I3) 400
401 401 IF (NV .GT. MVM) THEN 401
402 402 C----- CHECK NODE (I.E., VERTEX) STORAGE SIZE 402
403 403 PRINT 1020,NV,MVM,NVMK 403
404 404 1010 FORMAT(1X,'TOO MANY NODES. ',I9,', MAX = ',I5) 404
405 405 CALL EXIT 405
406 406 ENDIF 406
407 407 IF (NE .GT. MEM) THEN 407
408 408 C----- CHECK SIDE (I.E., EDGE) STORAGE SIZE 408
409 409 PRINT 1020,NE,MEM,NEMK 409
410 410 1020 FORMAT(1X,'TOO MANY SIDES. ',I9,', MAX = ',I5) 410
411 411 CALL EXIT 411
412 412 ENDIF 412
413 413 IF (NS .GT. MSM) THEN 413
414 414 C----- CHECK ZONE (I.E., SIDE OR TRIANGLE) STORAGE SIZE 414
415 415 PRINT 1030,NS,MSM 415
416 416 1030 FORMAT(1X,'TOO MANY ZONES. ',I9,', MAX = ',I5) 416
417 417 CALL EXIT 417
418 418 ENDIF 418
419 419 IF (NUMQUADS .GT. 0) THEN 419
420 420 C----- CHECK FOR QUADRILATERALS IN THE INPUT 420
421 421 PRINT 1040 421
422 422 1040 FORMAT(1X,'NO QUADRILATERALS ARE ALLOWED.') 422
423 423 CALL EXIT 423
424 424 ENDIF 424
425 425 C----- READ MARKER DEFINITIONS 425
426 426 C THE FOLLOWING JUST READS THE VARIABLES WITHOUT STORING 426
427 427 C THEM INTO PERMANENT ARRAYS. EFFECTIVELY JUST READING 427
428 428 C PAST THE MARKER DEFINITION INFORMATION. 428
429 429 DO 21 NZM = 1,NZMK 429
430 430 READ (16,1050) NMN,MNAME,NVAL 430
431 431 DO 20 NZMV = 1,NVAL 431
432 432 READ (16,1050) NMV,MVNAME 432
433 433 20 CONTINUE 433
434 434 21 CONTINUE 434
435 435 1050 FORMAT(3X,I2,1X,A15,1X,I2) 435
436 436 DO 31 NZM = 1,NSMK 436
437 437 READ (16,1050) NMN,MNAME,NVAL 437
438 438 DO 30 NSMV = 1,NVAL 438
439 439 READ (16,1050) NMV,MVNAME 439
440 440 30 CONTINUE 440
441 441 31 CONTINUE 441
442 442 DO 41 NZM = 1,NNMK 442
443 443 READ (16,1050) NMN,MNAME,NVAL 443

```

```

444 444 DO 40 NNMV = 1,NVAL 444
445 445 READ (16,1050) NMV,MVNAME 445
446 446 40 CONTINUE 446
447 447 41 CONTINUE 447
448 448 C----- READ IN VERTEX INFORMATION 448
449 449 DO 51 IV = 1 , NV 449
450 450 IS = IV 450
451 451 READ (16,1210) IK,XV(1,IS),XV(2,IS) 451
452 452 JV(1,IV) = 0 452
453 453 C INITIALIZE ANY VERTEX MARKER STORAGE, I.E. JV(*,IV) 453
454 454 51 CONTINUE 454
455 455 PRINT 1060,NV 455
456 456 1060 FORMAT(15,' NODES (VERTICES) READ IN.') 456
457 457 1210 FORMAT(17,E15.9,1X,E15.9) 457
458 458 IF (NVMK .GT. 0) THEN 458
459 459 C----- READ IN VERTEX MARKER INFORMATION 459
460 460 DO 55 IV = 1,NVMK 460
461 461 READ (16,*) IXV,MV1,MV2,MV3,MV4 461
462 462 JV(1,IXV) = MV1 462
463 463 C STORE THESE MARKERS IN JV(*, IXV) AS DESIRED 463
464 464 55 CONTINUE 464
465 465 PRINT 1070,NVMK 465
466 466 1070 FORMAT(15,' NODE (VERTEX) MARKERS READ IN.') 466
467 467 ENDIF 467
468 468 C----- READ IN EDGE INFORMATION ( EDGES OF TRIANGLES). 468
469 469 DO 60 IE = 1 , NE 469
470 470 IS = IE 470
471 471 READ (16,*) IJ,JE(1,IS),JE(2,IS),JE(3,IS),JE(4,IS) 471
472 472 C INITIALIZE ANY MARKER STORAGE. 472
473 473 JE(5,IE) = 0 473
474 474 60 CONTINUE 474
475 475 PRINT 1080,NE 475
476 476 1080 FORMAT(15,' SIDES (EDGES) READ IN.') 476
477 477 IF (NEMK .GT. 0) THEN 477
478 478 C----- READ IN EDGE MARKER INFORMATION 478
479 479 DO 65 IV = 1,NEMK 479
480 480 READ (16,*) IXE,MV1,MV2,MV3,MV4 480
481 481 JE(5,IXE) = MV1 481
482 482 65 CONTINUE 482
483 483 PRINT 1090,NEMK 483
484 484 1090 FORMAT(15,' SIDE (EDGE) MARKERS READ IN.') 484
485 485 ENDIF 485
486 486 C----- READ IN SIDE (TRIANGLE) INFORMATION. 486
487 487 DO 81 IS = 1 , NS 487
488 488 IE = IS 488
489 489 READ (16,1100) IJ, MV1,MV2,MV3,MV4, 489
490 490 IV1,IV2,IV3,IV4 490
491 491 1100 FORMAT (17,4I3,3(17,I2)) 491
492 492 JS(4,IE) = IV1 * ID1 492
493 493 JS(5,IE) = IV2 * ID2 493
494 494 JS(6,IE) = IV3 * ID3 494
495 495 C JS(7,IE) = MV1 495
496 496 C 496
497 497 C STORE THESE MARKERS IN JS(*, IS) AS DESIRED 497
498 498 C 498
499 499 81 CONTINUE 499
500 500 PRINT 1110,NS 500
501 501 1110 FORMAT(15,' ZONES (SIDES) READ IN.') 501
502 502 CLOSE (16) 502
503 503 C----- FORM VERTEX INDICES FOR EACH SIDE (TRIANGLE). 503
504 504 DO 85 IS = 1 , NS 504
505 505 DO 85 J = 1 , 3 505
506 506 IE = JS( J + 3 , IS ) 506
507 507 IEABS = IABS( IE ) 507
508 508 IF ( IE .GT. 0 ) THEN 508
509 509 JS( J , IS ) = JE( 1 , IEABS ) 509
510 510 ELSE 510
511 511 JS( J , IS ) = JE( 2 , IEABS ) 511
512 512 END IF 512
513 513 85 CONTINUE 513
514 514 C 514
515 515 C----- 515
516 516 C 516
517 517 IF(IOSPCL.EQ.1)THEN 517

```

```

518 518 C 518
519 519 C --- SPECIAL CASE FOR HALF CIRCLE BOUNDARY DATA ----- 519
520 520 C 520
521 521 DO 382 IE = 1 , NE 521
522 522 IJE5 = JE( 5 , IE ) 522
523 523 IF( IJE5 .EQ. 6 ) THEN 523
524 524 C 524
525 525 IV1 = JE( 1 , IE ) 525
526 526 IV2 = JE( 2 , IE ) 526
527 527 C 527
528 528 XXS1 = XV( 1 , IV1 ) 528
529 529 YYS1 = XV( 2 , IV1 ) 529
530 530 XXS2 = XV( 1 , IV2 ) 530
531 531 YYS2 = XV( 2 , IV2 ) 531
532 532 DXX = XXS1 - 1.50 532
533 533 ANGL = 1.570796327 533
534 534 IF( DXX .NE. 0 ) ANGL = ATAN2( YYS1 , DXX ) 534
535 535 XV( 1 , IV1 ) = COS( ANGL ) + 1.5 535
536 536 XV( 2 , IV1 ) = SIN( ANGL ) 536
537 537 DXX = XXS2 - 1.50 537
538 538 ANGL = 1.570796327 538
539 539 IF( DXX .NE. 0 ) ANGL = ATAN2( YYS2 , DXX ) 539
540 540 XV( 1 , IV2 ) = COS( ANGL ) + 1.5 540
541 541 XV( 2 , IV2 ) = SIN( ANGL ) 541
542 542 C XXS = XV( 1 , IV1 ) * 1.008930411364 542
543 543 C YYS = .6 * ( .2969 * SQRT( XXS ) - .126 * XXS - 543
544 544 C .3516 * XXS * XXS + .2843 * XXS * XXS * XXS - 544
545 545 C .1015 * XXS * XXS * XXS * XXS ) 545
546 546 C XV( 2 , IV1 ) = SIGN( 1. , XV( 2 , IV1 ) ) * YYS 546
547 547 C IF( XXS .GT. .3 .AND. XXS .LT. .7 ) JV( 1 , IV1 ) = 0 547
548 548 C 548
549 549 C XXS = XV( 1 , IV2 ) * 1.008930411364 549
550 550 C YYS = .6 * ( .2969 * SQRT( XXS ) - .126 * XXS - 550
551 551 C .3516 * XXS * XXS + .2843 * XXS * XXS * XXS - 551
552 552 C .1015 * XXS * XXS * XXS * XXS ) 552
553 553 C XV( 2 , IV2 ) = SIGN( 1. , XV( 2 , IV2 ) ) * YYS 553
554 554 C IF( XXS .GT. .3 .AND. XXS .LT. .7 ) JV( 1 , IV2 ) = 0 554
555 555 C IF( XE( 1 , IE ).GT. .2 ) CALL DISECT ( IE , IDONE , IJKINT ) 555
556 556 END IF 556
557 557 382 CONTINUE 557
558 558 END IF 558
559 559 C 559
560 560 C ----- 560
561 561 C 561
562 562 C 562
563 563 C --- CALCULATE GRID QUANTITIES THROUGH GEOMTR ----- 563
564 564 C 564
565 565 CALL UPDATE 565
566 566 C 566
567 567 C ----- 567
568 568 C 568
569 569 C --- REFINE THE INITIAL GRID BY A FACTOR OF THREE IF CALLED FOR ----- 569
570 570 C 570
571 571 C>>>>> 571
572 572 IF( ITRIGR .EQ. 1 ) THEN 572
573 573 NSS = NS 573
574 574 DO 110 IS = 1 , NSS 574
575 575 CALL VERCEN( IS ) 575
576 576 110 CONTINUE 576
577 577 NEE = NE 577
578 578 DO 120 IE = 1 , NEE 578
579 579 IF( JE( 5 , IE ) .NE. 0 ) THEN 579
580 580 CALL DISECT ( IE , IDONE , IJKINT ) 580
581 581 ENDIF 581
582 582 120 CONTINUE 582
583 583 DO 130 IK = 1 , 3 583
584 584 PRINT*,NV,NE,NS,IK 584
585 585 DO 130 IE = 1 , NE 585
586 586 CALL RECNC( IE , IDONE , ITL , ITR , JA , JB , JC , JD ) 586
587 587 CALL RECNC( JA , JADONE , ITL , ITR , JAA , JAB , JAC , JAD ) 587
588 588 CALL RECNC( JB , JBDONE , ITL , ITR , JBA , JBB , JBC , JBD ) 588
589 589 CALL RECNC( JC , JCDONE , ITL , ITR , JCA , JCB , JCC , JCD ) 589
590 590 CALL RECNC( JD , JDDONE , ITL , ITR , JDA , JDB , JDC , JDD ) 590
591 591 130 CONTINUE 591

```

```

592 592      END IF
593 593 C<<<<<<
594 594 C
595 595 C-----
596 596 C
597 597 C --- FIND AVERAGE TRIANGLE AREA -----
598 598 C
599 599      SAREMN = 1000000.
600 600      SAREMX = 0.
601 601      SAREVG = 0.
602 602      DO 105 IS = 1 , NS
603 603          AREASS = XS( 3 , IS )
604 604          SAREMX = AMAX1( SAREMX , AREASS )
605 605          SAREMN = AMINI( SAREMN , AREASS )
606 606          SAREVG = SAREVG + AREASS
607 607 105 CONTINUE
608 608      AVGARE = SAREVG
609 609      SAREVG = SAREVG / NS
610 610      FMINVG = SAREVG * AREADD
611 611      SAREMN = SAREMN / SAREVG
612 612      SAREMX = SAREMX / SAREVG
613 613      PRINT*, SAREVG, SAREMX, SAREMN
614 614 C
615 615 C --- DO INITIAL REFINEMENT FOR ALL INFLOW BOUNDARIES DEFINED -----
616 616 C BY EDGES THAT CONTAIN BOUNDARY CONDITION 8(INFLOW)
617 617 C
618 618      IF(IOPINT.EQ.1)THEN
619 619          NOFDIV = 2
620 620          CALL INTPTN( AREADD , NOFDIV , 1 , LTRIG )
621 621          NOFDIV = 2
622 622          CALL DYYPTN( AREADD , NOFDIV , 1 , LTRIG )
623 623          NOFDIV = 2
624 624          CALL INTPTN( AREADD , NOFDIV , 2 , LTRIG )
625 625          NOFDIV = 2
626 626          CALL DYYPTN( AREADD , NOFDIV , 2 , LTRIG )
627 627          NOFDIV = 2
628 628          CALL INTPTN( AREADD , NOFDIV , 3 , LTRIG )
629 629          NOFDIV = 2
630 630          CALL DYYPTN( AREADD , NOFDIV , 3 , LTRIG )
631 631 C
632 632          PRINT*, NV, NE, NS
633 633          ENDIF
634 634 C
635 635 C-----
636 636 C
637 637 C --- FOR ICOND>0 READ IN PREVIOUS RUN'S DATA -----
638 638 C
639 639 C(1)----
640 640      ELSE
641 641          CALL UPGRAD
642 642 C          CALL GEOMTR
643 643          IF( ICONP . EQ . 0 ) THEN
644 644              READ (88) RIN, PIN, RINL, PINL, UVIN, UIN, VIN, TT,
645 645                  HYDMOM(1), HYDMOM(2), HYDMOM(4)
646 646              PRINT *, RIN, PIN, UVIN, UIN, VIN, TT
647 647              READ (88) ((HYDV(IS, IK), IK=1, 5), IS=1, NS)
648 648              READ (88) ((HYDVVV(IV, IK), IK=1, 5), IV=1, NV)
649 649              READ (88) IJKINT, (KSDEL(IS), IS=1, NS)
650 650              IF( MPRTCL . EQ . 1 )
651 651                  .   READ (88) NPT, ((XPRTCL(IK, IPT), IK=1, 2), IPT=1, NPT),
652 652                  .   (IJKPRT(IPT), IPT=1, NPT)
653 653              ENDIF
654 654          ENDIF
655 655 C(1)<<<<
656 656 C
657 657 C --- INITIALIZATION OF THE PROBLEM -----
658 658 C
659 659      SARERV = 1. / SAREVG
660 660      SARESQ = SORT( SAREVG )
661 661      FMINVG = SAREVG * AREADD
662 662      HRSM = 1.E-8
663 663      HRGP = HRGG + 1.
664 664      HRGM = HRGG - 1.
665 665      CF = HRGP / ( 2. * HRGG )

```

```

666 666 C
667 667 JDUMP = 9
668 668 IF(KDUMP.EQ.0)THEN
669 669 KDUMP = JDUMP
670 670 ENDIF
671 671 C
672 672 TT = 0.
673 673 C
674 674 PIRAD = ATAN( 1. ) / 45.
675 675 ALPHA = ALFA * PIRAD
676 676 PRINT *,ALFA,PIRAD,ALPHA
677 677 PRINT *,XMCHIN,PIN,RIN
678 678 C
679 679 COSS = COS( ALPHA )
680 680 SINN = SIN( ALPHA )
681 681 TANN = TAN( ALPHA )
682 682 C
683 683 C-----
684 684 C
685 685 C --- SET THE INITIAL VALUE FOR PRIMITIVE VARIABLES -----
686 686 C
687 687 C(2)>>>>
688 688 IF( IOPTN . EQ . 1 ) THEN
689 689 UVIN = XMCHIN * SQRT( HRGG * PIN / RIN )
690 690 UIN = UVIN * COSS
691 691 VIN = UVIN * SINN
692 692 RIN = 1.
693 693 PIN = 1.
694 694 C
695 695 DO 150 IS = 1 , NS
696 696 HYDV( IS , 1 ) = RIN
697 697 HYDV( IS , 2 ) = 0.
698 698 HYDV( IS , 3 ) = 0.
699 699 HYDV( IS , 4 ) = PIN
700 700 HYDV( IS , 5 ) = HRGG
701 701 XSS = XS( 1 , IS )
702 702 IF( XSS . LT . .0 ) THEN
703 703 HYDV( IS , 1 ) = .125 * RIN
704 704 HYDV( IS , 4 ) = .100 * PIN
705 705 END IF
706 706 150 CONTINUE
707 707 C
708 708 DO 176 IV = 1 , NV
709 709 HYDVVV( IV , 1 ) = RIN
710 710 HYDVVV( IV , 2 ) = 0.
711 711 HYDVVV( IV , 3 ) = 0.
712 712 HYDVVV( IV , 4 ) = PIN / HRGM
713 713 HYDVVV( IV , 5 ) = HRGG
714 714 XSS = XV( 1 , IV )
715 715 IF( XSS . LT . -.0 ) THEN
716 716 HYDVVV( IV , 1 ) = RIN
717 717 HYDVVV( IV , 4 ) = PIN / HRGM
718 718 END IF
719 719 176 CONTINUE
720 720 C(2)----
721 721 ELSE
722 722 C
723 723 XMSQR = XMCHIN * XMCHIN
724 724 IF( ICOND . EQ . 1 . AND . ICONP . EQ . 0 ) THEN
725 725 ELSE
726 726 PINL = PIN
727 727 RINL = RIN
728 728 RINRTO = ( HRGG + 1. ) * XMSQR /
729 729 ( ( HRGG - 1. ) * XMSQR + 2. )
730 730 PINRTO = ( 2. * HRGG * XMSQR - ( HRGG - 1. ) ) /
731 731 ( HRGG + 1. )
732 732 PIN = PINRTO * PINL
733 733 RIN = RINRTO * RINL
734 734 YMCHIN = SQRT( ( ( HRGG - 1. ) * XMSQR + 2. ) /
735 735 ( 2. * HRGG * XMSQR - ( HRGG - 1. ) ) )
736 736 PRINT*,HRGG,RIN,PIN,YMCHIN
737 737 PRINT*,HRGG,RINL,PINL,XMCHIN
738 738 UVIN = XMCHIN * SQRT( HRGG * PINL / RINL ) -
739 739 YMCHIN * SQRT( HRGG * PIN / RIN )

```

```

740 740      END IF
741 741      DO 175 IV = 1 , NV
742 742      HYDVVV( IV , 1 ) = RINL
743 743      HYDVVV( IV , 2 ) = 0.
744 744      HYDVVV( IV , 3 ) = 0.
745 745      HYDVVV( IV , 4 ) = PINL / HRGM
746 746      HYDVVV( IV , 5 ) = HRGG
747 747      XSS = XV( 1 , IV )
748 748      IF( XSS . LT . -.0 ) THEN
749 749      HYDVVV( IV , 1 ) = RIN
750 750      HYDVVV( IV , 2 ) = UVIN * RIN
751 751      HYDVVV( IV , 4 ) = PIN / HRGM + .5 * RIN * UVIN * UVIN
752 752      END IF
753 753 175  CONTINUE
754 754      DO 170 IS = 1 , NS
755 755      HYDV( IS , 1 ) = RINL
756 756      HYDV( IS , 2 ) = 0.
757 757      HYDV( IS , 3 ) = 0.
758 758      HYDV( IS , 4 ) = PINL
759 759      HYDV( IS , 5 ) = HRGG
760 760      XSS = XS( 1 , IS )
761 761      IF( XSS . LT . -.0 ) THEN
762 762      HYDV( IS , 1 ) = RIN
763 763      HYDV( IS , 2 ) = UVIN
764 764      HYDV( IS , 4 ) = PIN
765 765      END IF
766 766 170  CONTINUE
767 767  C
768 768      IF( IOPEOS . EQ . 1 ) THEN
769 769      HRGGN = HRGG
770 770      HRGGL = HRGG
771 771      RINRTO = ( HRGGN + 1. ) * XMSQR /
772 772      . ( ( HRGGN - 1. ) * XMSQR + 2. )
773 773      PINRTO = ( 2. * HRGGN * XMSQR - ( HRGGN - 1. ) ) /
774 774      . ( HRGGN + 1. )
775 775      PIN = PINRTO * PINL
776 776      RIN = RINRTO * RINL
777 777      TTNN = PIN / ( HRGGN - 1. )
778 778      RRNN = RIN
779 779      TTNL = PINL / ( HRGGL - 1. )
780 780      RRNL = RINL
781 781      DO 1122 KI = 1 , 9
782 782      CALL EOS( RRNN , TTNN , 1 , HRGGN )
783 783      CALL EOS( RRNL , TTNL , 1 , HRGGL )
784 784      RINRTO = ( HRGGN + 1. ) * XMSQR /
785 785      . ( ( HRGGN - 1. ) * XMSQR + 2. )
786 786      PINRTO = ( 2. * HRGGN * XMSQR - ( HRGGN - 1. ) ) /
787 787      . ( HRGGN + 1. )
788 788      RIN = RINRTO * RINL
789 789      PIN = PINRTO * PINL
790 790      TTNN = PIN / ( HRGGN - 1. )
791 791      RRNN = RIN
792 792      TTNL = PINL / ( HRGGL - 1. )
793 793      RRNL = RINL
794 794      YMCHIN = SQRT( ( ( HRGGN - 1. ) * XMSQR + 2. ) /
795 795      . ( 2. * HRGGN * XMSQR - ( HRGGN - 1. ) ) )
796 796      PRINT*,HRGGN,RIN,PIN,YMCHIN
797 797      PRINT*,HRGGL,RINL,PINL,YMCHIN
798 798 1122 CONTINUE
799 799      UVIN = XMCHIN * SQRT( HRGGL * PINL / RINL ) -
800 800      . YMCHIN * SQRT( HRGGN * PIN / RIN )
801 801      DO 172 IS = 1 , NS
802 802      HYDV( IS , 5 ) = HRGGL
803 803 172  CONTINUE
804 804      END IF
805 805      UIN = UVIN * COSS
806 806      VIN = UVIN * SINN
807 807  C
808 808      ENDIF
809 809  C(2)<<<<
810 810  C
811 811      IF( MPRTCL . EQ . 1 ) THEN
812 812      IKXY = 0
813 813      DO 190 IKX = 1 , 30

```

```

814 814 DO 190 IKY = 1 , 15 814
815 815 IKXY = IKXY + 1 815
816 816 XPRTCL( 1 , IKXY ) = ( IKX - 1 ) * .1 + .05 816
817 817 XPRTCL( 2 , IKXY ) = ( IKY - 1 ) * .1 + .05 817
818 818 190 CONTINUE 818
819 819 NPT = IKXY 819
820 820 PRINT *, NPT 820
821 821 CALL PRLCTN 821
822 822 PRINT *, NPT 822
823 823 ENDIF 823
824 824 C 824
825 825 C----- 825
826 826 C 826
827 827 C --- READ INPUT DATA FROM THE PREVIOUS RUN ----- 827
828 828 C 828
829 829 PRINT * , ICOND, ICOMP 829
830 830 IF( ICOMP . EQ . 1 ) THEN 830
831 831 READ (88) RIN, PIN, RINL, PINL, UVIN, UIN, VIN, TT, 831
832 832 HYDMOM(1), HYDMOM(2), HYDMOM(4) 832
833 833 PRINT * , RIN, PIN, UVIN, UIN, VIN, TT 833
834 834 READ (88) ((HYDV(IS, IK), IK=1,5), IS=1, NS) 834
835 835 READ (88) ((HYDVVV(IV, IK), IK=1,5), IV=1, NV) 835
836 836 READ (88) IJKINT, (KSDEL(TS), IS=1, NS) 836
837 837 IF( MPRTCL . EQ . 1 ) 837
838 838 READ (88) NPT, ((XPRTCL(IK, IPT), IK=1,2), IPT=1, NPT), 838
839 839 (IJKPRT(IPT), IPT=1, NPT) 839
840 840 ENDIF 840
841 841 C 841
842 842 C----- 842
843 843 C 843
844 844 C --- PERFORM THE ACTUAL CALCULATION ----- 844
845 845 C 845
846 846 CALL HYDRMN 846
847 847 C 847
848 848 C----- 848
849 849 C 849
850 850 C --- EXIT POINT FROM PROGRAM ----- 850
851 851 C 851
852 852 C ----- 852
853 853 STOP 777 853
854 854 C ----- 854
855 855 C 855
856 856 C --- FORMATS ----- 856
857 857 C 857
858 858 101 FORMAT(1H , ' ICOND=' , I2, 5X, ' ICOMP=' , I2, 5X, ' ITRIGR=' , I2, 5X, 858
859 859 ' IOPTN=' , I2, /, 1X, 859
860 860 ' XMCHIN=' , F13.6, 5X, ' RIN=' , F13.6, 5X, ' PIN=' , F13.6, /, 1X, 860
861 861 ' ALFA=' , F13.6, 5X, ' HRGG=' , F13.6, 5X, ' IHRN=' , I2, 5X, /, 1X, 861
862 862 ' NTIME=' , I2, 5X, ' MDUMP=' , I5, 5X, ' NDUMP=' , I5, 5X, /, 1X 862
863 863 ' KDUMP=' , I5, 5X, ' IOSPCL=' , I2, 5X, ' IOPLFT=' , I2, 5X, /, 1X, 863
864 864 ' IOPRCN=' , I2, 5X, ' IOPORD=' , I2, 5X, ' IOPBYN=' , I2, 5X, /, 1X 864
865 865 ' IAXSYM=' , I2, 5X, ' IOPEOS=' , I2, 5X, ' MPRTCL=' , I6, 5X, /, 1X, 865
866 866 ' IOPINT=' , I2, 5X, ' IOPADD=' , I2, 5X, ' IOPDEL=' , I2, 5X, /, 1X, 866
867 867 ' AREADD=' , F13.6, 5X, ' AREDEL=' , F13.6, 5X, /, 1X, 867
868 868 ' IWINDW=' , I2, 5X, ' ISTATC=' , I2) 868
869 869 C 869
870 870 C --- 870
871 871 END 871

```

```

872      1      SUBROUTINE HYDRFL
873      2      C
874      3      C-----I
875      4      C
876      5      C      HYDRFL IS A 2 DIMENSIONAL RIEMANN SOLVER THAT COMPUTES THE I
877      6      C      FLUXES ACROSS NORMAL INTERFACES FOR UPDATING SIDE I
878      7      C      OR TRIANGLE BASED QUANTITIES. I
879      8      C I
880      9      C-----I
881     10      C
882     11      include      'cmsh00.h'
883     12      include      'chyd00.h'
884     13      include      'cint00.h'
885     14      include      'cphs10.h'
886     15      include      'cphs20.h'
887     16      C
888     17      C-----
889     18      C
890     19      REAL DELP(MBP),WSQP(MBP),WSOM(MBP),WSOO(MBP),
891     20      . RSTAR(MBP),CSTAR(MBP),PMAX(MBP),PMIN(MBP)
892     21      REAL RRIGHT(MBP),URIGHT(MBP),VRIGHT(MBP),PRIGHT(MBP)
893     22      REAL RLEFT(MBP),ULEFT(MBP),VLEFT(MBP),PLEFT(MBP)
894     23      C
895     24      C-----
896     25      C
897     26      C
898     27      C --- BEGIN LOOP OVER ALL EDGES IN THE DOMAIN -----
899     28      C
900     29      DO 280 IH = 1 , 4
901     30      DO 280 IS = 1 , NS
902     31      HYDFLX( IS , IH ) = 0.
903     32      280 CONTINUE
904     33      C
905     34      NE1 = 1
906     35      NE2 = NOFVEE( 1 )
907     36      DO 110 INE = 1 , NVEEE
908     37      C
909     38      C --- FETCH HYDRO QUANTITIES -----
910     39      C FOR LEFT AND RIGHT SIDE OF THE INTERFACE ON WHICH THE
911     40      C RIEMANN PROBLEM IS SOLVED
912     41      C
913     42      DO 120 IE = NE1 , NE2
914     43      KE = IE - NE1 + 1
915     44      C
916     45      RRR( KE ) = RR( IE )
917     46      UUR( KE ) = UR( IE )
918     47      VVR( KE ) = VR( IE )
919     48      PPR( KE ) = PR( IE )
920     49      C
921     50      RRL( KE ) = RL( IE )
922     51      UUL( KE ) = UL( IE )
923     52      VVL( KE ) = VL( IE )
924     53      PPL( KE ) = PL( IE )
925     54      120 CONTINUE
926     55      C
927     56      C --- ASSIGN GAMA A VALUE -----
928     57      C
929     58      DO 130 KE = 1 , NOFVEE( INE )
930     59      IE = KE + NE1 - 1
931     60      ISL = JE( 3 , IE )
932     61      ISR = JE( 4 , IE )
933     62      GAMAL( KE ) = HYDV( ISL , 5 )
934     63      IF( ISR . NE . 0 ) THEN
935     64      GAMAR( KE ) = HYDV( ISR , 5 )
936     65      ELSE
937     66      GAMAR( KE ) = GAMAL( KE )
938     67      END IF
939     68      C
940     69      C --- THIS SECTION OF CODE SOLVES FOR "PSTAR" AND "USTAR" IN -----
941     70      C THE RIEMANN PROBLEM USING NEWTON'S METHOD.
942     71      C
943     72      WLEFT( KE ) = SQRT( GAMAL( KE ) * PPL( KE ) * RRL( KE ) )
944     73      WRIGT( KE ) = SQRT( GAMAR( KE ) * PPR( KE ) * RRR( KE ) )
945     74      C

```



```

946 75      WLESQ( KE ) = WLEFT( KE ) * WLEFT( KE )      946
947 76      WRISQ( KE ) = WRIGT( KE ) * WRIGT( KE )      947
948 77      C      948
949 78      PMIN( KE ) = AMINI( PPL( KE ) , PPR( KE ) )    949
950 79      PSML( KE ) = HRSM * PMIN( KE )                950
951 80      C      951
952 81      C --- FORM THE STARTING GUESS FOR THE SOLUTION ----- 952
953 82      C      953
954 83      PSTAR( KE ) = ( WLEFT( KE ) * PPR( KE ) +      954
955 84      .      WRIGT( KE ) * PPL( KE ) -            955
956 85      .      WLEFT( KE ) * WRIGT( KE ) *          956
957 86      .      ( UUR( KE ) - UUL( KE ) ) ) /         957
958 87      .      ( WLEFT( KE ) + WRIGT( KE ) )         958
959 88      PSTAR( KE ) = AMAX1( PSTAR( KE ) , PSML( KE ) ) 959
960 89      130 CONTINUE                                  960
961 90      C      961
962 91      DO 140 I = 1 , IHRN                            962
963 92      C      963
964 93      C --- BEGIN THE NEWTON ITERATION ----- 964
965 94      C      965
966 95      DO 150 KE = 1 , NOFVEE( INE )                  966
967 96      C      967
968 97      CF = ( GAMAL( KE ) + 1 . ) / GAMAL( KE ) * .5  968
969 98      WLEFS( KE ) = ( 1. + CF * ( PSTAR( KE ) /      969
970 99      .      PPL( KE ) - 1. ) ) * WLESQ( KE )        970
971 100     WLEFT( KE ) = SQRT( WLEFS( KE ) )             971
972 101     ZLEFT( KE ) = 2. * WLEFT( KE ) * WLEFS( KE ) / 972
973 102     .      ( WLESQ( KE ) + WLEFS( KE ) )          973
974 103     USTL( KE ) = UUL( KE ) -                     974
975 104     .      ( PSTAR( KE ) - PPL( KE ) ) / WLEFT( KE ) 975
976 105     150 CONTINUE                                  976
977 106     C      977
978 107     DO 152 KE = 1 , NOFVEE( INE )                  978
979 108     C      979
980 109     CF = ( GAMAR( KE ) + 1 . ) / GAMAR( KE ) * .5  980
981 110     WRIFS( KE ) = ( 1. + CF * ( PSTAR( KE ) /      981
982 111     .      PPR( KE ) - 1. ) ) * WRISQ( KE )        982
983 112     WRIGT( KE ) = SQRT( WRIFS( KE ) )             983
984 113     ZRIGT( KE ) = 2. * WRIGT( KE ) * WRIFS( KE ) / 984
985 114     .      ( WRISQ( KE ) + WRIFS( KE ) )          985
986 115     USTR( KE ) = UUR( KE ) +                     986
987 116     .      ( PSTAR( KE ) - PPR( KE ) ) / WRIGT( KE ) 987
988 117     152 CONTINUE                                  988
989 118     C      989
990 119     DO 160 KE = 1 , NOFVEE( INE )                  990
991 120     DPST( KE ) = ZLEFT( KE ) * ZRIGT( KE ) *      991
992 121     .      ( USTR( KE ) - USTL( KE ) ) /          992
993 122     .      ( ZLEFT( KE ) + ZRIGT( KE ) )          993
994 123     PSTAR( KE ) = PSTAR( KE ) - DPST( KE )        994
995 124     PSTAR( KE ) = AMAX1( PSTAR( KE ) , PSML( KE ) ) 995
996 125     160 CONTINUE                                  996
997 126     140 CONTINUE                                  997
998 127     C      998
999 128     C --- FORM FINAL SOLUTIONS ----- 999
1000 129     C      1000
1001 130     DO 170 KE = 1 , NOFVEE( INE )                  1001
1002 131     C      1002
1003 132     CF = ( GAMAL( KE ) + 1 . ) / GAMAL( KE ) * .5  1003
1004 133     WLEFT( KE ) = SQRT( WLESQ( KE ) * ( 1. +      1004
1005 134     .      CF * ( PSTAR( KE ) / PPL( KE ) - 1. ) ) ) 1005
1006 135     170 CONTINUE                                  1006
1007 136     C      1007
1008 137     DO 172 KE = 1 , NOFVEE( INE )                  1008
1009 138     C      1009
1010 139     CF = ( GAMAR( KE ) + 1 . ) / GAMAR( KE ) * .5  1010
1011 140     WRIGT( KE ) = SQRT( WRISQ( KE ) * ( 1. +      1011
1012 141     .      CF * ( PSTAR( KE ) / PPR( KE ) - 1. ) ) ) 1012
1013 142     172 CONTINUE                                  1013
1014 143     C      1014
1015 144     DO 180 KE = 1 , NOFVEE( INE )                  1015
1016 145     USTAR( KE ) = ( PPL( KE ) - PPR( KE ) +      1016
1017 146     .      WLEFT( KE ) * UUL( KE ) +              1017
1018 147     .      WRIGT( KE ) * UUR( KE ) ) /            1018
1019 148     .      ( WLEFT( KE ) + WRIGT( KE ) )          1019

```

```

1020 149 180 CONTINUE
1021 150 C
1022 151 C --- BEGIN PROCEDURE TO OBTAIN FLUXES FROM REIMANN FORMALISM -----
1023 152 C
1024 153 DO 190 KE = 1 , NOFVEE( INE )
1025 154 IF( USTAR( KE ) . LE . 0.0 ) THEN
1026 155 C
1027 156 RO( KE ) = RRR( KE )
1028 157 PO( KE ) = PPR( KE )
1029 158 UO( KE ) = UUR( KE )
1030 159 CO( KE ) = SQRT( GAMAR( KE ) * PPR( KE ) / RRR( KE ) )
1031 160 WO( KE ) = WRIGT( KE )
1032 161 ISN( KE ) = 1
1033 162 C
1034 163 VGDNV( KE ) = VVR( KE )
1035 164 C
1036 165 ELSE
1037 166 C
1038 167 RO( KE ) = RRL( KE )
1039 168 PO( KE ) = PPL( KE )
1040 169 UO( KE ) = UUL( KE )
1041 170 CO( KE ) = SQRT( GAMAL( KE ) * PPL( KE ) / RRL( KE ) )
1042 171 WO( KE ) = WLEFT( KE )
1043 172 ISN( KE ) = - 1
1044 173 C
1045 174 VGDNV( KE ) = VVL( KE )
1046 175 END IF
1047 176 190 CONTINUE
1048 177 C
1049 178 DO 200 KE = 1 , NOFVEE( INE )
1050 179 DELP( KE ) = PSTAR( KE ) - PO( KE )
1051 180 WSOP( KE ) = ISN( KE ) * UO( KE ) + WO( KE ) / RO( KE )
1052 181 WSOM( KE ) = ISN( KE ) * UO( KE ) + CO( KE )
1053 182 200 CONTINUE
1054 183 C
1055 184 DO 210 KE = 1 , NOFVEE( INE )
1056 185 IF( DELP( KE ) . GT . 0. ) THEN
1057 186 WSOO( KE ) = WSOP( KE )
1058 187 ELSE
1059 188 WSOO( KE ) = WSOM( KE )
1060 189 END IF
1061 190 210 CONTINUE
1062 191 C
1063 192 C --- USE OUTER STATE SOLUTION -----
1064 193 C
1065 194 DO 220 KE = 1 , NOFVEE( INE )
1066 195 PGDNV( KE ) = PO( KE )
1067 196 UGDNV( KE ) = UO( KE )
1068 197 CGDNV( KE ) = CO( KE )
1069 198 RGDNV( KE ) = RO( KE )
1070 199 220 CONTINUE
1071 200 C
1072 201 C --- COMPUTE STARRED VALUES -----
1073 202 C
1074 203 DO 230 KE = 1 , NOFVEE( INE )
1075 204 IE = KE + NE1 - 1
1076 205 ISL = JE( 3 , IE )
1077 206 ISR = JE( 4 , IE )
1078 207 IF( ISR . NE . 0 ) THEN
1079 208 GAMAG( KE ) = .5 * ( HYDV( ISL , 5 ) + HYDV( ISR , 5 ) )
1080 209 ELSE
1081 210 GAMAG( KE ) = HYDV( ISL , 5 )
1082 211 END IF
1083 212 C
1084 213 RSTAR( KE ) = 1. / ( 1. / RO( KE ) - DELP( KE ) /
1085 214 ( WO( KE ) * WG( KE ) ) )
1086 215 C
1087 216 CSTAR( KE ) = SQRT( GAMAG( KE ) * PSTAR( KE ) / RSTAR( KE ) )
1088 217 WSOM( KE ) = ISN( KE ) * USTAR( KE ) + CSTAR( KE )
1089 218 230 CONTINUE
1090 219 C
1091 220 DO 240 KE = 1 , NOFVEE( INE )
1092 221 IF( DELP( KE ) . GT . 0. ) THEN
1093 222 SPIN( KE ) = WSOP( KE )

```

```

1094 223      ELSE
1095 224      SPIN( KE ) = WSOM( KE )
1096 225      END IF
1097 226 240 CONTINUE
1098 227 C
1099 228      DO 250 KE = 1 , NOFVEE( INE )
1100 229 C
1101 230      IF( WSOO( KE ) . GE . 0. ) THEN
1102 231      IF( SPIN( KE ) . GE . 0. ) THEN
1103 232 C
1104 233 C --- USE THE STARRED STATE RESULTS -----
1105 234 C
1106 235      RGDNV( KE ) = RSTAR( KE )
1107 236      UGDNV( KE ) = USTAR( KE )
1108 237      CGDNV( KE ) = CSTAR( KE )
1109 238      PGDNV( KE ) = PSTAR( KE )
1110 239      ELSE
1111 240 C
1112 241 C --- EVALUATE THE INSIDE RAREFACTION WAVE -----
1113 242 C
1114 243      HRGG = GAMAG( KE )
1115 244      HRGM = GAMAG( KE ) - 1.
1116 245      HRGP = GAMAG( KE ) + 1.
1117 246      CGDNV( KE ) = ( CSTAR( KE ) * 2. -
1118 247      . ISN( KE ) * USTAR( KE ) * HRGM ) / HRGP
1119 248      UGDNV( KE ) = - ISN( KE ) * CGDNV( KE )
1120 249      RGDNV( KE ) = ( CGDNV( KE ) / CO( KE ) ) **
1121 250      . ( 2. / HRGM ) * RO( KE )
1122 251      PGDNV( KE ) = CGDNV( KE ) * CGDNV( KE ) * RGDNV( KE ) / HRGG
1123 252 C
1124 253      END IF
1125 254 C
1126 255      END IF
1127 256 250 CONTINUE
1128 257 C
1129 258      DO 142 IE = NE1 , NE2
1130 259      KE = IE - NE1 + 1
1131 260 C
1132 261      RRR( KE ) = XN( IE )
1133 262      UUR( KE ) = YN( IE )
1134 263      VVR( KE ) = XXN( IE )
1135 264      PPR( KE ) = YYN( IE )
1136 265      PPL( KE ) = XE( 2 , IE )
1137 266      RRL( KE ) = XE( 1 , IE )
1138 267      UUL( KE ) = XYMIDL( IE )
1139 268 C
1140 269 142 CONTINUE
1141 270 C
1142 271 C --- SEARCH FOR MINIMUM VALUE OF TIMESTEP ...DTT... -----
1143 272 C
1144 273      DO 260 KE = 1 , NOFVEE( INE )
1145 274      CTT = SQRT( GAMAG( KE ) * PGDNV( KE ) / RGDNV( KE ) )
1146 275      VEL = UGDNV( KE )
1147 276 C
1148 277      PROJECT = RRR( KE ) * VVR( KE ) + UUR( KE ) * PPR( KE )
1149 278      DTU = PPL( KE ) * ABS( PROJECT ) / ( CTT + ABS( VEL ) )
1150 279      DT1 = DTU * UUL( KE )
1151 280      DT2 = DTU - DT1
1152 281      DTT = AMIN1( DTT , DT1 , DT2 )
1153 282 260 CONTINUE
1154 283 C
1155 284 C --- NOW FIND THE FLUXES AT EACH INTERFACE -----
1156 285 C
1157 286      DO 270 KE = 1 , NOFVEE( INE )
1158 287      HRGG = GAMAG( KE )
1159 288      HRGM = GAMAG( KE ) - 1.
1160 289      HRGP = GAMAG( KE ) + 1.
1161 290 C
1162 291 C ... FLUX FOR DENSITY .....
1163 292 C
1164 293      RO( KE ) = RGDNV( KE ) * UGDNV( KE )
1165 294 C
1166 295 C ... FLUX FOR MOMENTUM DENSITY .....
1167 296 C

```

```

1168 297      UO( KE ) = PGDNV( KE ) * RRR( KE ) +
1169 298      .          RO( KE ) * ( UGDNV( KE ) * RRR( KE ) -
1170 299      .          VGDNV( KE ) * UUR( KE ) )
1171 300      WO( KE ) = PGDNV( KE ) * UUR( KE ) +
1172 301      .          RO( KE ) * ( UGDNV( KE ) * UUR( KE ) +
1173 302      .          VGDNV( KE ) * RRR( KE ) )
1174 303      C
1175 304      C ... FLUX FOR ENERGY DENSITY .....
1176 305      C
1177 306      PO( KE ) = UGDNV( KE ) * ( PGDNV( KE ) * HRGG / HRGM +
1178 307      .          .5 * RGDNV( KE ) * ( UGDNV( KE ) * UGDNV( KE ) +
1179 308      .          VGDNV( KE ) * VGDNV( KE ) ) )
1180 309      C
1181 310      270 CONTINUE
1182 311      C
1183 312      C --- COLLECT INTERFACE FLUXES FOR EACH TRIANGLE -----
1184 313      C
1185 314      DO 290 IE = NE1 , NE2
1186 315      KE = IE - NE1 + 1
1187 316      C
1188 317      ISL = JE( 3 , IE )
1189 318      ISR = JE( 4 , IE )
1190 319      C
1191 320      DFLUX = RRL( KE )
1192 321      C
1193 322      IF( JE( 5 , IE ) .EQ. 0 ) THEN
1194 323      C
1195 324      C ... FLUX FOR DENSITY .....
1196 325      C
1197 326      HYDFLX( ISL , 1 ) = HYDFLX( ISL , 1 ) + DFLUX * RO( KE )
1198 327      HYDFLX( ISR , 1 ) = HYDFLX( ISR , 1 ) - DFLUX * RO( KE )
1199 328      C
1200 329      C ... FLUX FOR MOMENTUM DENSITY ( U DIRECTION ) .....
1201 330      C
1202 331      HYDFLX( ISL , 2 ) = HYDFLX( ISL , 2 ) + DFLUX * UO( KE )
1203 332      HYDFLX( ISR , 2 ) = HYDFLX( ISR , 2 ) - DFLUX * UO( KE )
1204 333      C
1205 334      C ... FLUX FOR MOMENTUM DENSITY ( V DIRECTION ) .....
1206 335      C
1207 336      HYDFLX( ISL , 3 ) = HYDFLX( ISL , 3 ) + DFLUX * WO( KE )
1208 337      HYDFLX( ISR , 3 ) = HYDFLX( ISR , 3 ) - DFLUX * WO( KE )
1209 338      C
1210 339      C ... FLUX FOR ENERGY DENSITY .....
1211 340      C
1212 341      HYDFLX( ISL , 4 ) = HYDFLX( ISL , 4 ) + DFLUX * PO( KE )
1213 342      HYDFLX( ISR , 4 ) = HYDFLX( ISR , 4 ) - DFLUX * PO( KE )
1214 343      C
1215 344      ELSE
1216 345      C
1217 346      C ... FLUX FOR DENSITY .....
1218 347      C
1219 348      HYDFLX( ISL , 1 ) = HYDFLX( ISL , 1 ) + DFLUX * RO( KE )
1220 349      C
1221 350      C ... FLUX FOR MOMENTUM DENSITY ( U DIRECTION ) .....
1222 351      C
1223 352      HYDFLX( ISL , 2 ) = HYDFLX( ISL , 2 ) + DFLUX * UO( KE )
1224 353      C
1225 354      C ... FLUX FOR MOMENTUM DENSITY ( V DIRECTION ) .....
1226 355      C
1227 356      HYDFLX( ISL , 3 ) = HYDFLX( ISL , 3 ) + DFLUX * WO( KE )
1228 357      C
1229 358      C ... FLUX FOR ENERGY DENSITY .....
1230 359      C
1231 360      HYDFLX( ISL , 4 ) = HYDFLX( ISL , 4 ) + DFLUX * PO( KE )
1232 361      C
1233 362      END IF
1234 363      290 CONTINUE
1235 364      C
1236 365      NE1 = NE2 + 1
1237 366      NE2 = NE2 + NOFVEE( INE + 1 )
1238 367      110 CONTINUE
1239 368      C
1240 369      C-----
1241 370      C

```

```

1242 371 C --- EXIT POINT FROM SUBROUTINE ----- 1242
1243 372 C 1243
1244 373 C ----- 1244
1245 374 C RETURN 1245
1246 375 C ----- 1246
1247 376 C 1247
1248 377 C --- 1248
1249 378 C END 1249

```

```

1250 1 SUBROUTINE HYDRMN 1250
1251 2 C 1251
1252 3 C -----I 1252
1253 4 C I 1253
1254 5 C HYDRMN IS THE MAIN SUBROUTINE FOR THE UNSTRUCTURED GRID I 1254
1255 6 C HYDRODYNAMIC SOLVER. THIS SUBROUTINE OBTAINS THE I 1255
1256 7 C EDGE BASED FLUXES FOR EACH TRIANGLE/SIDE FROM I 1256
1257 8 C SUBROUTINE --- HYORFL --- . IT ALSO CONTROLS I 1257
1258 9 C THE REFINEMENT AND COARSENING OF THE GRID. I 1258
1259 10 C THE SUBROUTINE GENERATES THE OUTPUT THAT IS USED I 1259
1260 11 C FOR POST-PROCESSING. I 1260
1261 12 C I 1261
1262 13 C I 1262
1263 14 C -----I 1263
1264 15 C 1264
1265 16 C include 'cmsh00.h' 1265
1266 17 C include 'chyd00.h' 1266
1267 18 C include 'cint00.h' 1267
1268 19 C include 'cphs10.h' 1268
1269 20 C include 'cphs20.h' 1269
1270 21 C 1270
1271 22 C ----- 1271
1272 23 C 1272
1273 24 C REAL RRN(MBP),URN(MBP),VRN(MBP),EPN(MBP),XSAR(MBP), 1273
1274 25 C TTN(MBP),XYRAD(MBP) 1274
1275 26 C INTEGER IEDIST(2) 1275
1276 27 C 1276
1277 28 C ----- 1277
1278 29 C 1278
1279 30 C CFL = 0.90 1279
1280 31 C 1280
1281 32 C --- SET SPECIFIC TIME FOR A DUMP ----- 1281
1282 33 C 1282
1283 34 C TLIMIT=30. 1283
1284 35 C FLATDR = .9 1284
1285 36 C LDUMP = KDUMP 1285
1286 37 C IF( IJKINT .EQ. 3 ) THEN 1286
1287 38 C LDUMP = 6 1287
1288 39 C IF( LDUMP .LT. KDUMP ) LDUMP = KDUMP 1288
1289 40 C END IF 1289
1290 41 C 1290
1291 42 C DO 120 JT = 1 , NTIME 1291
1292 43 C DO 130 IT = 1 , MDUMP 1292
1293 44 C 1293
1294 45 C DO 140 ITT = 1 , NDUMP 1294
1295 46 C IJKKJI = ( JT - 1 ) * NDUMP * MDUMP + ( IT - 1 ) * NDUMP + ITT 1295
1296 47 C IJKIJK = IJKINT + IJKKJI 1296
1297 48 C 1297
1298 49 C DO 142 IKT = 1 , LDUMP 1298
1299 50 C 1299
1300 51 C --- SELECT ORDER OF INTEGRATION ----- 1300
1301 52 C 1301
1302 53 C IF(IOPORD.EQ.1)THEN 1302
1303 54 C CALL FIRST 1303
1304 55 C ELSEIF(IOPORD.EQ.2)THEN 1304
1305 56 C CALL GRADNG 1305
1306 57 C ENDF 1306
1307 58 C 1307
1308 59 C --- SET TIMESTEP TO HIGH VALUE IT WILL BE CALCULATED PROPERLY ----- 1308
1309 60 C IN THE FLUX SUBROUTINE 1309
1310 61 C 1310
1311 62 C DTT = 1.E24 1311
1312 63 C 1312

```

```

1313 64 C --- FIND THE FLUXES -----
1314 65 C
1315 66 CALL HYDRFL
1316 67 C
1317 68 DTT = DTT * CFL
1318 69 TT = TT + DTT
1319 70 PRINT *,JT,IT,ITT,IKT,DTT,TT,NS
1320 71 C
1321 72 C --- INITIALIZE THE VERTEX BASED QUANTITIES NEEDED FOR COARSENING AND -
1322 73 C FOR REFINEMENT, AND FOR POST-PROCESSING
1323 74 C
1324 75 DO 210 IV = 1 , NV
1325 76 PR( IV ) = 0.
1326 77 DO 210 IR = 1 , MHQ
1327 78 HYDVVV( IV , IR ) = 0.
1328 79 210 CONTINUE
1329 80 C
1330 81 NS1 = 1
1331 82 NS2 = NOFVES( 1 )
1332 83 DO 110 INS = 1 , NVEES
1333 84 C
1334 85 DO 150 IS = NS1 , NS2
1335 86 KS = IS - NS1 + 1
1336 87 RRR( KS ) = HYDV( IS , 1 )
1337 88 UUR( KS ) = HYDV( IS , 2 )
1338 89 VVR( KS ) = HYDV( IS , 3 )
1339 90 PPR( KS ) = HYDV( IS , 4 )
1340 91 C
1341 92 RRL( KS ) = HYDFLX( IS , 1 )
1342 93 UUL( KS ) = HYDFLX( IS , 2 )
1343 94 VVL( KS ) = HYDFLX( IS , 3 )
1344 95 PPL( KS ) = HYDFLX( IS , 4 )
1345 96 C
1346 97 XSAR( KS ) = SAREA( IS )
1347 98 150 CONTINUE
1348 99 C
1349 100 DO 170 KS = 1 , NOFVES( INS )
1350 101 IS = KS + NS1 - 1
1351 102 GAMAG( KS ) = HYOV( IS , 5 )
1352 103 HRGM = GAMAG( KS ) - 1.
1353 104 C
1354 105 RRN( KS ) = RRR( KS )
1355 106 URN( KS ) = RRR( KS ) * UUR( KS )
1356 107 VRN( KS ) = RRR( KS ) * VVR( KS )
1357 108 EPN( KS ) = PPR( KS ) / HRGM + .5 * RRR( KS ) *
1358 109 ( UUR( KS ) * UUR( KS ) +
1359 110 VVR( KS ) * VVR( KS ) )
1360 111 170 CONTINUE
1361 112 C
1362 113 C-----
1363 114 C
1364 115 C --- COMPUTING THE SOURCE TERM ASSOCIATED WITH AXI-SYMMETRIC CASE -----
1365 116 C
1366 117 XYDUMY = 1. / 6.283185307
1367 118 DO 188 KS = 1 , NOFVES( INS )
1368 119 XYRAD( KS ) = XYDUMY
1369 120 188 CONTINUE
1370 121 C
1371 122 C --- Y-AXIS IS AXIS OF SYMMETRY -----
1372 123 C
1373 124 IF( IAXSYM . EQ . 2 )THEN
1374 125 DO 180 KS = 1 , NOFVES( INS )
1375 126 IS = KS + NS1 - 1
1376 127 XS2S = XS( 1 , IS )
1377 128 XYRAD( KS ) = XS2S
1378 129 IF( XS2S . GT . .0005 ) THEN
1379 130 DTA = DTT * UUR( KS ) / XS2S
1380 131 RRN( KS ) = RRN( KS ) * ( 1. - DTA )
1381 132 URN( KS ) = URN( KS ) * ( 1. - DTA )
1382 133 VRN( KS ) = VRN( KS ) * ( 1. - DTA )
1383 134 EPN( KS ) = EPN( KS ) * ( 1. - DTA ) - PPR( KS ) * DTA
1384 135 END IF
1385 136 180 CONTINUE
1386 137 C

```

```

1387 138 C --- X-AXIS IS AXIS OF SYMMETRY -----
1388 139 C
1389 140     ELSEIF( IAXSYM . EQ . 1 )THEN
1390 141     DO 182 KS = 1 , NOFVES( INS )
1391 142     IS = KS + NS1 - 1
1392 143     XS2S = XS( 2 , IS )
1393 144     XYRAD( KS ) = XS2S
1394 145     IF( XS2S . GT . .0005 ) THEN
1395 146         DTA = DTT * VVR( KS ) / XS2S
1396 147         RRN( KS ) = RRN( KS ) * ( 1. - DTA )
1397 148         URN( KS ) = URN( KS ) * ( 1. - DTA )
1398 149         VRN( KS ) = VRN( KS ) * ( 1. - DTA )
1399 150         EPN( KS ) = EPN( KS ) * ( 1. - DTA ) - PPR( KS ) * DTA
1400 151     END IF
1401 152 182 CONTINUE
1402 153     ENDIF
1403 154 C
1404 155 C --- COMPUTE THE EFFECT OF THE BOUYANCY(GRAVITY) TERM -----
1405 156 C
1406 157     GRAVTY = 9.81
1407 158 C
1408 159     IF( IOPBYN . EQ . 2 )THEN
1409 160     DO 184 KS = 1 , NOFVES( INS )
1410 161     DTA = DTT * RRR( KS ) * GRAVTY
1411 162     VRN( KS ) = VRN( KS ) - DTA
1412 163     EPN( KS ) = EPN( KS ) - DTA * VVR( KS )
1413 164 184 CONTINUE
1414 165 C
1415 166     ELSEIF( IOPBYN . EQ . 1 )THEN
1416 167     DO 186 KS = 1 , NOFVES( INS )
1417 168     DTA = DTT * RRR( KS ) * GRAVTY
1418 169     URN( KS ) = URN( KS ) - DTA
1419 170     EPN( KS ) = EPN( KS ) - DTA * UUR( KS )
1420 171 186 CONTINUE
1421 172     END IF
1422 173 C
1423 174 -----
1424 175 C
1425 176 C --- UPDATE THE HYDRODYNAMIC QUANTITIES -----
1426 177 C     STORING THE FLUXES FOR THE REFINEMENT/COARSENING STEPS
1427 178 C
1428 179     DO 190 KS = 1 , NOFVES( INS )
1429 180     IS = KS + NS1 - 1
1430 181     DTA = DTT * XSAR( KS )
1431 182 C
1432 183     RRLL = RRL( KS )
1433 184     UULL = UUL( KS )
1434 185     VVLL = VVL( KS )
1435 186     RRN( KS ) = RRN( KS ) - RRLL * DTA
1436 187     URN( KS ) = URN( KS ) - UULL * DTA
1437 188     VRN( KS ) = VRN( KS ) - VVLL * DTA
1438 189 C
1439 190     PPLL = PPL( KS )
1440 191     HYDFLX( IS , 4 ) = ABS( PPLL ) / EPN( KS ) * DTA
1441 192     EPN( KS ) = EPN( KS ) - PPLL * DTA
1442 193 C
1443 194 190 CONTINUE
1444 195 C
1445 196     DO 202 IS = NS1 , NS2
1446 197     KS = IS - NS1 + 1
1447 198     ENERGY = 1. / RRN( KS ) * ( URN( KS ) * URN( KS ) +
1448 199         VRN( KS ) * VRN( KS ) )
1449 200     TTN( KS ) = EPN( KS ) - .5 * ENERGY
1450 201     HYDFLX( IS , 1 ) = ENERGY / TTN( KS )
1451 202     HYDFLX( IS , 2 ) = RRN( KS )
1452 203 C
1453 204 202 CONTINUE
1454 205 C --- EQUATION OF STATE FOR AIR -----
1455 206 C
1456 207     IF( IOPEOS . EQ . 1 )THEN
1457 208         CALL EOS( RRN , TTN , NOFVES( INS ) , GAMAG )
1458 209     ELSE
1459 210     ENDIF
1460 211 C

```

```

1461 212 C --- ACCUMULATE VALUES AT THE VERTICES FOR ADAPTATION AND ALSO ----- 1461
1462 213 C FOR POST-PROCESSING 1462
1463 214 C 1463
1464 215 DO 220 KS = 1 , NOFVES( INS ) 1464
1465 216 IS = KS + NS1 - 1 1465
1466 217 C 1466
1467 218 IV1 = JS( 1 , IS ) 1467
1468 219 IV2 = JS( 2 , IS ) 1468
1469 220 IV3 = JS( 3 , IS ) 1469
1470 221 C 1470
1471 222 VOLUME = 6.283185307 * XYRAD( KS ) 1471
1472 223 C 1472
1473 224 XYAREA = XS( 3 , IS ) * VOLUME 1473
1474 225 XYFDR = XYAREA * RRN( KS ) 1474
1475 226 XYFDU = XYAREA * URN( KS ) 1475
1476 227 XYFDV = XYAREA * VRN( KS ) 1476
1477 228 XYFDP = XYAREA * EPN( KS ) 1477
1478 229 XYFDG = XYAREA * GAMAG( KS ) 1478
1479 230 C 1479
1480 231 HYDVVV( IV1 , 1 ) = HYDVVV( IV1 , 1 ) + XYFDR 1480
1481 232 HYDVVV( IV1 , 2 ) = HYDVVV( IV1 , 2 ) + XYFDU 1481
1482 233 HYDVVV( IV1 , 3 ) = HYDVVV( IV1 , 3 ) + XYFDV 1482
1483 234 HYDVVV( IV1 , 4 ) = HYDVVV( IV1 , 4 ) + XYFDP 1483
1484 235 HYDVVV( IV1 , 5 ) = HYDVVV( IV1 , 5 ) + XYFDG 1484
1485 236 PR( IV1 ) = PR( IV1 ) + XYAREA 1485
1486 237 C 1486
1487 238 HYDVVV( IV2 , 1 ) = HYDVVV( IV2 , 1 ) + XYFDR 1487
1488 239 HYDVVV( IV2 , 2 ) = HYDVVV( IV2 , 2 ) + XYFDU 1488
1489 240 HYDVVV( IV2 , 3 ) = HYDVVV( IV2 , 3 ) + XYFDV 1489
1490 241 HYDVVV( IV2 , 4 ) = HYDVVV( IV2 , 4 ) + XYFDP 1490
1491 242 HYDVVV( IV2 , 5 ) = HYDVVV( IV2 , 5 ) + XYFDG 1491
1492 243 PR( IV2 ) = PR( IV2 ) + XYAREA 1492
1493 244 C 1493
1494 245 HYDVVV( IV3 , 1 ) = HYDVVV( IV3 , 1 ) + XYFDR 1494
1495 246 HYDVVV( IV3 , 2 ) = HYDVVV( IV3 , 2 ) + XYFDU 1495
1496 247 HYDVVV( IV3 , 3 ) = HYDVVV( IV3 , 3 ) + XYFDV 1496
1497 248 HYDVVV( IV3 , 4 ) = HYDVVV( IV3 , 4 ) + XYFDP 1497
1498 249 HYDVVV( IV3 , 5 ) = HYDVVV( IV3 , 5 ) + XYFDG 1498
1499 250 PR( IV3 ) = PR( IV3 ) + XYAREA 1499
1500 251 C 1500
1501 252 IENUMR = 0 1501
1502 253 IE1 = IABS( JS( 4 , IS ) ) 1502
1503 254 IJE5 = JE( 5 , IE1 ) 1503
1504 255 IF( IJE5 . NE . 0 ) THEN 1504
1505 256 IENUMR = IENUMR + 1 1505
1506 257 IEDIST( IENUMR ) = IE1 1506
1507 258 END IF 1507
1508 259 IE2 = IABS( JS( 5 , IS ) ) 1508
1509 260 IJE5 = JE( 5 , IE2 ) 1509
1510 261 IF( IJE5 . NE . 0 ) THEN 1510
1511 262 IENUMR = IENUMR + 1 1511
1512 263 IEDIST( IENUMR ) = IE2 1512
1513 264 END IF 1513
1514 265 IE3 = IABS( JS( 6 , IS ) ) 1514
1515 266 IJE5 = JE( 5 , IE3 ) 1515
1516 267 IF( IJE5 . NE . 0 ) THEN 1516
1517 268 IENUMR = IENUMR + 1 1517
1518 269 IEDIST( IENUMR ) = IE3 1518
1519 270 END IF 1519
1520 271 C 1520
1521 272 IF( IENUMR . NE . 0 ) THEN 1521
1522 273 DO 322 IK = 1 , IENUMR 1522
1523 274 IEK = IEDIST( IENUMR ) 1523
1524 275 IJE55 = JE( 5 , IEK ) 1524
1525 276 RRNN = RRN( KS ) 1525
1526 277 URNN = URN( KS ) 1526
1527 278 VRNN = VRN( KS ) 1527
1528 279 EPNN = EPN( KS ) 1528
1529 280 IF( IJE55 . EQ . 6 . OR . IJE55 . EQ . 5 ) THEN 1529
1530 281 UUVV = - ( URN( KS ) * XN( IEK ) + 1530
1531 282 VRN( KS ) * YN( IEK ) ) 1531
1532 283 VVUU = - URN( KS ) * YN( IEK ) + 1532
1533 284 VRN( KS ) * XN( IEK ) 1533
1534 285 URNN = UUVV * XN( IEK ) - VVUU * YN( IEK ) 1534

```



```

1535 286 C      VRNN = UVVV * YN( IEK ) + VVVU * XN( IEK )      1535
1536 287 C      ELSE IF( IJESS . EQ . 8 ) THEN                1536
1537 288 C      RRNN = RIN                                       1537
1538 289 C      URNN = RIN * UIN                                   1538
1539 290 C      VRNN = RIN * VIN                                   1539
1540 291 C      EPNN = PIN / HRGM + .5 * RIN * UVIN * UVIN     1540
1541 292 C      END IF                                           1541
1542 293 C                                                         1542
1543 294 C      XYFDR = XYAREA * RRNN                               1543
1544 295 C      XYFDU = XYAREA * URNN                               1544
1545 296 C      XYFDV = XYAREA * VRNN                               1545
1546 297 C      XYFDP = XYAREA * EPNN                               1546
1547 298 C      XYFDG = XYAREA * GAMAG( KS )                     1547
1548 299 C                                                         1548
1549 300 C      IV1 = JE( 1 , IEK )                                 1549
1550 301 C      IV2 = JE( 2 , IEK )                                 1550
1551 302 C      HYDVVV( IV1 , 1 ) = HYDVVV( IV1 , 1 ) + XYFDR   1551
1552 303 C      HYDVVV( IV1 , 2 ) = HYDVVV( IV1 , 2 ) + XYFDU   1552
1553 304 C      HYDVVV( IV1 , 3 ) = HYDVVV( IV1 , 3 ) + XYFDV   1553
1554 305 C      HYDVVV( IV1 , 4 ) = HYDVVV( IV1 , 4 ) + XYFDP   1554
1555 306 C      HYDVVV( IV1 , 5 ) = HYDVVV( IV1 , 5 ) + XYFDG   1555
1556 307 C      PR( IV1 ) = PR( IV1 ) + XYAREA                    1556
1557 308 C                                                         1557
1558 309 C      HYDVVV( IV2 , 1 ) = HYDVVV( IV2 , 1 ) + XYFDR   1558
1559 310 C      HYDVVV( IV2 , 2 ) = HYDVVV( IV2 , 2 ) + XYFDU   1559
1560 311 C      HYDVVV( IV2 , 3 ) = HYDVVV( IV2 , 3 ) + XYFDV   1560
1561 312 C      HYDVVV( IV2 , 4 ) = HYDVVV( IV2 , 4 ) + XYFDP   1561
1562 313 C      HYDVVV( IV2 , 5 ) = HYDVVV( IV2 , 5 ) + XYFDG   1562
1563 314 C      PR( IV2 ) = PR( IV2 ) + XYAREA                    1563
1564 315 322 CONTINUE                                           1564
1565 316 C      END IF                                           1565
1566 317 C                                                         1566
1567 318 220 CONTINUE                                           1567
1568 319 C                                                         1568
1569 320 C --- CONSTRUCT NONCONSERVED HYDRODYNAMIC QUANTITIES ----- 1569
1570 321 C                                                         1570
1571 322 C      DO 195 IS = NS1 , NS2                               1571
1572 323 C      KS = IS - NS1 + 1                                   1572
1573 324 C      HDUM = 1. / RRN( KS )                               1573
1574 325 C      HYDV( IS , 1 ) = RRN( KS )                         1574
1575 326 C      HYDV( IS , 2 ) = URN( KS ) * HDUM                 1575
1576 327 C      HYDV( IS , 3 ) = VRN( KS ) * HDUM                 1576
1577 328 C      HYDV( IS , 5 ) = GAMAG( KS )                       1577
1578 329 C      HYDV( IS , 4 ) = TTN( KS ) * ( HYDV( IS , 5 ) - 1. ) 1578
1579 330 195 CONTINUE                                           1579
1580 331 C                                                         1580
1581 332 C      NS1 = NS2 + 1                                       1581
1582 333 C      NS2 = NS2 + NOFVES( INS + 1 )                     1582
1583 334 110 CONTINUE                                           1583
1584 335 C                                                         1584
1585 336 C --- END OF LOOP OVER TRIANGLES -----                1585
1586 337 C                                                         1586
1587 338 C -----                                                1587
1588 339 C                                                         1588
1589 340 C --- CALL FOR PARTICLE TRACERS -----                1589
1590 341 C                                                         1590
1591 342 C      IF( MPRTCL . EQ . 1 ) THEN                          1591
1592 343 C                                                         1592
1593 344 C      CALL PRPTHC                                         1593
1594 345 C                                                         1594
1595 346 C      ENDDIF                                             1595
1596 347 C                                                         1596
1597 348 C --- END OF INNER LOOP OVER ... KDUMP ... -----    1597
1598 349 C                                                         1598
1599 350 C -----                                                1599
1600 351 C                                                         1600
1601 352 C --- NORMALIZE CONSERVATIVE VERTEX BASED QUANTITIES ----- 1601
1602 353 C                                                         1602
1603 354 C      DO 230 IV = 1 , NV                                    1603
1604 355 C      VAREA = 1. / PR( IV )                                1604
1605 356 C      DO 230 IR = 1 , MHQ                                  1605
1606 357 C      HYDVVV( IV , IR ) = HYDVVV( IV , IR ) * VAREA     1606
1607 358 230 CONTINUE                                           1607
1608 359 C                                                         1608

```

```

1609 360 142 CONTINUE 1609
1610 361 C --- WRITE THE DUMP FILE DATA FOR POST-PROCESSING ----- 1610
1611 362 C 1611
1612 363 IF( IT . EQ . MDUMP . AND . ITT . EQ . NDUMP ) THEN 1612
1613 364 WRITE (9) NV,NE,NS,NPT,NTIME 1613
1614 365 WRITE (9) ((XV(IK,IV),IK=1,2),IV=1,NV) 1614
1615 366 WRITE (9) (JV(2,IV),IV=1,NV) 1615
1616 367 WRITE (9) ((JE(KK,IE),KK=1,5),IE=1,NE) 1616
1617 368 WRITE (9) ((JS(KK,IS),KK=1,6),IS=1,NS) 1617
1618 369 WRITE (9) ((XS(KK,IS),KK=1,2),IS=1,NS) 1618
1619 370 WRITE (9) RIN,PIN,UVIN,UIV,VIN,TT,IOPFLT 1619
1620 371 WRITE (9) ((HYDV(IS,IK),IK=1,5),IS=1,NS) 1620
1621 372 C 1621
1622 373 C --- WRITE OUT PARTICLE TRACER DATA ----- 1622
1623 374 C 1623
1624 375 IF( MPRTCL . EQ . 1 ) THEN 1624
1625 376 WRITE (9) ((XPRTCL(IK,IPT),IK=1,2),IPT=1,NPT), 1625
1626 377 ((WPRTCL(IK,IPT),IK=1,2),IPT=1,NPT) 1626
1627 378 ENDF 1627
1628 379 C 1628
1629 380 C --- PRINT CONSOLE MESSAGE AT END OF LOOP ----- 1629
1630 381 C 1630
1631 382 PRINT * , JT,NV,NE,NS 1631
1632 383 C 1632
1633 384 END IF 1633
1634 385 C 1634
1635 386 C----- 1635
1636 387 C 1636
1637 388 C I-----I 1637
1638 389 C I REFINEMENT/ADDITION OF POINTS I 1638
1639 390 C I-----I 1639
1640 391 C 1640
1641 392 C 1641
1642 393 C --- CALCULATE THE GRADIENT OF THE MACH NUMBER FOR STEADY STATE ----- 1642
1643 394 C ADAPTIVE STEP AND GENERATE THE QUANTITIES ON WHICH WE ADAPT. 1643
1644 395 C 1644
1645 396 C --- ADAPTATION TO STATIC QUANTITIES BASED ON GRADIENTS OF ----- 1645
1646 397 C MACH NUMBER, PRESSURE, AND DENSITY. OVERRIDES 1646
1647 398 C ADAPTATION ON DYNAMIC FLUXES OF ENERGY AND DENSITY 1647
1648 399 C 1648
1649 400 IF( ISTATC . EQ . 1 ) THEN 1649
1650 401 CALL GRDFLX 1650
1651 402 DO 240 IS = 1 , NS 1651
1652 403 HYDFLX( IS , 1 ) = ABS( PL( IS ) ) + ABS( PR( IS ) ) 1652
1653 404 HYDFLX( IS , 2 ) = ABS( RL( IS ) ) + ABS( RR( IS ) ) 1653
1654 405 HYDFLX( IS , 4 ) = ABS( VL( IS ) ) + ABS( VR( IS ) ) 1654
1655 406 240 CONTINUE 1655
1656 407 C 1656
1657 408 ELSE 1657
1658 409 C 1658
1659 410 CALL GRDENG 1659
1660 411 DO 242 IS = 1 , NS 1660
1661 412 HYDFLX( IS , 1 ) = ( UL( IS ) * UL( IS ) + UR( IS ) * UR( IS ) ) / 1661
1662 413 ( HYDFLX( IS , 1 ) + 1.E-12 ) 1662
1663 414 HYDFLX( IS , 2 ) = ( RL( IS ) * RL( IS ) + RR( IS ) * RR( IS ) ) / 1663
1664 415 ( HYDFLX( IS , 2 ) + 1.E-12 ) 1664
1665 416 HYDFLX( IS , 4 ) = ( VL( IS ) * VL( IS ) + VR( IS ) * VR( IS ) ) / 1665
1666 417 ( HYDFLX( IS , 4 ) + 1.E-12 ) 1666
1667 418 242 CONTINUE 1667
1668 419 END IF 1668
1669 420 C 1669
1670 421 DYDMOM = HYDFLX( 1 , 4 ) 1670
1671 422 DO 250 IS = 1 , NS 1671
1672 423 DYDMOM = AMAX1( DYDMOM , HYDFLX( IS , 4 ) ) 1672
1673 424 250 CONTINUE 1673
1674 425 HYDMOM( 4 ) = .5 * ( DYDMOM + HYDMOM( 4 ) ) 1674
1675 426 PRINT*,HYDMOM( 4 ) 1675
1676 427 C 1676
1677 428 DYDMOM = HYDFLX( 1 , 2 ) 1677
1678 429 DO 260 IS = 1 , NS 1678
1679 430 DYDMOM = AMAX1( DYDMOM , HYDFLX( IS , 2 ) ) 1679
1680 431 260 CONTINUE 1680
1681 432 HYDMOM( 2 ) = .5 * ( DYDMOM + HYDMOM( 2 ) ) 1681
1682 433 PRINT*,HYDMOM( 2 ) 1682

```

```

1683 434 C 1683
1684 435 C DYDMOM = HYDFLX( 1 , 1 ) 1684
1685 436 C DO 270 IS = 1 , NS 1685
1686 437 C DYDMOM = AMAX1( DYDMOM , HYDFLX( IS , 1 ) ) 1686
1687 438 270 CONTINUE 1687
1688 439 C HYDMOM( 1 ) = .5 * ( DYDMOM + HYDMOM( 1 ) ) 1688
1689 440 C PRINT*,HYDMOM( 1 ) 1689
1690 441 C 1690
1691 442 C --- REFINEMENT STEP DONE HERE ----- 1691
1692 443 C 1692
1693 444 C IF(IOPADD.EQ.1) THEN 1693
1694 445 C NOFDIV = 4 1694
1695 446 C CALL DYNPTN( AREADD , NOFDIV , IJKIJK , LTRIG ) 1695
1696 447 C NOFDIV = 2 1696
1697 448 C CALL DYYPTN( AREADD , NOFDIV , IJKIJK , LTRIG ) 1697
1698 449 C NOFDIV = 1 1698
1699 450 C CALL DYYPTN( AREADD , NOFDIV , IJKIJK , LTRIG ) 1699
1700 451 C CALL DYYPTN( AREADD , NOFDIV , IJKIJK , LTRIG ) 1700
1701 452 C 1701
1702 453 C PRINT*,NV,NE,NS 1702
1703 454 C ENDIF 1703
1704 455 140 CONTINUE 1704
1705 456 C 1705
1706 457 C --- END OF OUTER LOOP DEFINED BY ...NDUMP... ----- 1706
1707 458 C 1707
1708 459 C ----- 1708
1709 460 C 1709
1710 461 C I-----I 1710
1711 462 C I COARSENING/DELETION OF POINTS I 1711
1712 463 C I-----I 1712
1713 464 C 1713
1714 465 C 1714
1715 466 C IF(IOPDEL.EQ.1)THEN 1715
1716 467 C IF( IJKIJK . GT . 19 ) CALL DELPTNT( AREDEL , IJKIJK ) 1716
1717 468 C PRINT*,NV,NE,NS 1717
1718 469 C ENDIF 1718
1719 470 130 CONTINUE 1719
1720 471 C 1720
1721 472 C --- END OF OUTERMOST LOOP DEFINED BY ...MDUMP... ----- 1721
1722 473 C 1722
1723 474 C ----- 1723
1724 475 C 1724
1725 476 C I-----I 1725
1726 477 C I DIAGNOSTIC FOR LIFT/DORAG I 1726
1727 478 C I-----I 1727
1728 479 C 1728
1729 480 C IF(IOPLFT.EQ.1)THEN 1729
1730 481 C CALL LIFTDR 1730
1731 482 C ENDIF 1731
1732 483 C 1732
1733 484 C ----- 1733
1734 485 C 1734
1735 486 C I-----I 1735
1736 487 C I OUTPUT FILE FOR RESTARTS I 1736
1737 488 C I-----I 1737
1738 489 C 1738
1739 490 C 1739
1740 491 C REWIND 88 1740
1741 492 C ITERAT = ITERAT + 1 1741
1742 493 C WRITE (88) NV,NVMK,NE,NEMK,NS,NSMK,ITERAT 1742
1743 494 C WRITE (88) ((JW(KK,IV),KK=1,2),(XV(IK,IV),IK=1,2),IV=1,NV) 1743
1744 495 C WRITE (88) ((JE(KK,IE),KK=1,5),(XE(KI,IE),KI=1,2),IE=1,NE) 1744
1745 496 C WRITE (88) (XN(IE),YN(IE),XXN(IE),YYN(IE),IE=1,NE) 1745
1746 497 C WRITE (88) ((JS(KK,IS),KK=1,6),(XS(KI,IS),KI=1,3),IS=1,NS) 1746
1747 498 C WRITE (88) (XMIDL(IE),YMIDL(IE),XYMIDL(IE),IE=1,NE) 1747
1748 499 C WRITE (88) SAREVG,NVECE,NREME,NVECV,NREMV,NVECS,NREMS 1748
1749 500 C WRITE (88) RIN,PIN,RINL,PINL,UVIN,UVIN,VIN,TT, 1749
1750 501 C HYDMOM(1),HYDMOM(2),HYDMOM(4) 1750
1751 502 C WRITE (88) ((HYDV(IS,IK),IK=1,5),IS=1,NS) 1751
1752 503 C WRITE (88) ((HYDVVV(IV,IK),IK=1,5),IV=1,NV) 1752
1753 504 C WRITE (88) IJKIJK,(KSDEL(IS),IS=1,NS) 1753
1754 505 C IF( MPRTCL . EQ . 1 ) 1754
1755 506 C WRITE (88) NPT,((XPRTCL(IK,IPT),IK=1,2),IPT=1,NPT), 1755
1756 507 C (IJKPRT(IPT),IPT=1,NPT) 1756

```

```

1757 508 C
1758 509     REWIND 8
1759 510     WRITE (8) NV,NVMK,NE,NEMK,NS,NSMK,ITERAT
1760 511     WRITE (8) ((JV(KK,IV),KK=1,2),(XV(IK,IV),IK=1,2),IV=1,NV)
1761 512     WRITE (8) ((JE(KK,IE),KK=1,5),(XE(KI,IE),KI=1,2),IE=1,NE)
1762 513     WRITE (8) (XN(IE),YN(IE),XXN(IE),YYN(IE),IE=1,NE)
1763 514     WRITE (8) ((JS(KK,IS),KK=1,6),(XS(KI,IS),KI=1,3),IS=1,NS)
1764 515     WRITE (8) (XMIDL(IE),YMIDL(IE),XYMIDL(IE),IE=1,NE)
1765 516     WRITE (8) SAREVG,NVECE,NREME,NVECV,NREMV,NVECS,NREMS
1766 517     WRITE (8) RIN,PIN,RINL,PINL,UVIN,UVIN,TT,
1767 518     . HYDMOM(1),HYDMOM(2),HYDMOM(4)
1768 519     WRITE (8) ((HYDV(IS,IK),IK=1,5),IS=1,NS)
1769 520     WRITE (8) ((HYDVV(IV,IK),IK=1,5),IV=1,NV)
1770 521     WRITE (8) IJKIJK,(KSDLT(IS),IS=1,NS)
1771 522     IF( MPRTCL .EQ. 1 )
1772 523     . WRITE (8) NPT,((XPRCTL(IK,IPT),IK=1,2),IPT=1,NPT),
1773 524     . (IJKPRT(IPT),IPT=1,NPT)
1774 525 C
1775 526 120 CONTINUE
1776 527 C
1777 528 C --- END MAIN SEQUENCE LOOP DEFINED BY ...NTIME...-----
1778 529 C
1779 530 C-----
1780 531 C
1781 532 C --- EXIT POINT FROM SUBROUTINE -----
1782 533 C
1783 534 C -----
1784 535     RETURN
1785 536 C -----
1786 537 C
1787 538 C --- FORMATS -----
1788 539 C
1789 540 C ---
1790 541     END

```

```

1791 1 SUBROUTINE GEOMTR
1792 2 C
1793 3 C-----I
1794 4 C I
1795 5 C GEOMTR COMPUTE GEOMETRICAL PARAMETERS TO COMPLETE THE GRID I
1796 6 C DEFINITION NEEDED BY THE CODE. I
1797 7 C I
1798 8 C I
1799 9 C-----I
1800 10 C
1801 11 include 'cmsh00.h'
1802 12 include 'chyd00.h'
1803 13 include 'cint00.h'
1804 14 include 'cphs10.h'
1805 15 include 'cphs20.h'
1806 16 C
1807 17 C-----
1808 18 C
1809 19 REAL XELEFT(MBP),YELEFT(MBP),XERIGT(MBP),YERIGT(MBP)
1810 20 C
1811 21 C-----
1812 22 C
1813 23 C
1814 24 C --- MAKE SURE THAT THE DOMAIN IS ALWAYS TO THE LEFT OF BOUNDARY -----
1815 25 C EDGES BY ORIENTING THEM CORRECTLY.
1816 26 C
1817 27 DO 105 IE = 1, NE
1818 28 IJE5 = JE( 5, IE )
1819 29 IF( IJE5 .NE. 0 ) THEN
1820 30 ISR = JE( 4, IE )
1821 31 IF( ISR .NE. 0 ) THEN
1822 32 IV2 = JE( 1, IE )
1823 33 IV1 = JE( 2, IE )
1824 34 JE( 1, IE ) = IV1
1825 35 JE( 2, IE ) = IV2
1826 36 JE( 3, IE ) = ISR
1827 37 JE( 4, IE ) = 0

```

```

1828 38 DO 106 IR = 1, 3 1828
1829 39 JEE = IABS( JS( IR + 3, ISR ) ) 1829
1830 40 IF( JEE .EQ. IE ) JS( IR + 3, ISR ) = IE 1830
1831 41 106 CONTINUE 1831
1832 42 ENDIF 1832
1833 43 ENDIF 1833
1834 44 105 CONTINUE 1834
1835 45 C 1835
1836 46 C --- FIND UNIT VECTORS NORMAL TO THE EDGES ----- 1836
1837 47 C AND EDGE CROSSING OF LINE BETWEEN TRIANGLE BARICENTERS 1837
1838 48 C AND ALSO TRIANGLE AREAS. 1838
1839 49 C 1839
1840 50 NE1 = 1 1840
1841 51 NE2 = NOFVEE( 1 ) 1841
1842 52 DO 110 INE = 1, NVEEE 1842
1843 53 DO 140 IE = NE1, NE2 1843
1844 54 KE = IE - NE1 + 1 1844
1845 55 IV1 = JE( 1, IE ) 1845
1846 56 IV2 = JE( 2, IE ) 1846
1847 57 ISL = JE( 3, IE ) 1847
1848 58 ISR = JE( 4, IE ) 1848
1849 59 C 1849
1850 60 C --- FIND UNIT VECTOR NORMAL TO AN EDGE ----- 1850
1851 61 C STORED IN XN(IE),YN(IE) 1851
1852 62 C 1852
1853 63 DXD = XV( 1, IV2 ) - XV( 1, IV1 ) 1853
1854 64 DYD = XV( 2, IV2 ) - XV( 2, IV1 ) 1854
1855 65 XE( 1, IE ) = SQRT( DXD * DXD + DYD * DYD ) 1855
1856 66 XEY = 1. / XE( 1, IE ) 1856
1857 67 XD = DXD * XEY 1857
1858 68 YD = DYD * XEY 1858
1859 69 C 1859
1860 70 XN( IE ) = YD 1860
1861 71 YN( IE ) = - XD 1861
1862 72 C 1862
1863 73 IJES = JE( 5, IE ) 1863
1864 74 C 1864
1865 75 C ----- 1865
1866 76 C 1866
1867 77 C --- BOUNDARY TRIANGLES ----- 1867
1868 78 C 1868
1869 79 IF( IJES .NE. 0 ) THEN 1869
1870 80 IV3 = JS( 1, ISL ) 1870
1871 81 IF( IV3 .EQ. IV1 .OR. IV3 .EQ. IV2 ) IV3 = JS( 2, ISL ) 1871
1872 82 IF( IV3 .EQ. IV1 .OR. IV3 .EQ. IV2 ) IV3 = JS( 3, ISL ) 1872
1873 83 XELEFT( KE ) = ( XV( 1, IV3 ) + XV( 1, IV2 ) + 1873
1874 84 XV( 1, IV1 ) ) * THIRD 1874
1875 85 YELEFT( KE ) = ( XV( 2, IV3 ) + XV( 2, IV2 ) + 1875
1876 86 XV( 2, IV1 ) ) * THIRD 1876
1877 87 C 1877
1878 88 AA = XV( 1, IV2 ) - XV( 1, IV1 ) 1878
1879 89 BB = XV( 2, IV2 ) - XV( 2, IV1 ) 1879
1880 90 CC = XELEFT( KE ) - XV( 1, IV1 ) 1880
1881 91 DD = YELEFT( KE ) - XV( 2, IV1 ) 1881
1882 92 EE = ( AA * CC + BB * DD ) * XEY * XEY 1882
1883 93 XERIGT( KE ) = XV( 1, IV1 ) + AA * EE 1883
1884 94 YERIGT( KE ) = XV( 2, IV1 ) + BB * EE 1884
1885 95 C 1885
1886 96 DXD = XERIGT( KE ) - XELEFT( KE ) 1886
1887 97 DYD = YERIGT( KE ) - YELEFT( KE ) 1887
1888 98 XE( 2, IE ) = SQRT( DXD * DXD + DYD * DYD ) 1888
1889 99 C 1889
1890 100 C --- UNIT VECTOR FROM LEFT TO RIGHT BARI-CENTER ----- 1890
1891 101 C STORED IN XXN(IE),YYN(IE) 1891
1892 102 C 1892
1893 103 XY = 1. / XE( 2, IE ) 1893
1894 104 XXN( IE ) = DXD * XY 1894
1895 105 YYN( IE ) = DYD * XY 1895
1896 106 C 1896
1897 107 C --- LENGTH OF LINE BETWEEN BARI-CENTERS ----- 1897
1898 108 C STORED IN XE(2,IE) 1898
1899 109 C 1899
1900 110 XE( 2, IE ) = 2. * XE( 2, IE ) 1900
1901 111 C 1901

```

```

1902 112 C --- COORDINATES OF BARI-CENTERS FOR EACH TRIANGLE ----- 1902
1903 113 C   STORED IN XS(1,IS),XS(2,IS) 1903
1904 114 C 1904
1905 115     XS( 1 , ISL ) = XELEFT( KE ) 1905
1906 116     XS( 2 , ISL ) = YELEFT( KE ) 1906
1907 117 C 1907
1908 118 C --- INTERSECTION POINT ON INTERFACE FOR LINE CONNECTING BARI-CENTERS - 1908
1909 119 C   STORED IN XMIDL(IE),YMIDL(IE) AND FRACTION OF LENGHT BETWEEN 1909
1910 120 C   LEFT BARI-CENTER TO INTERSECTION POINT IN XYMIDL(IE). 1910
1911 121 C 1911
1912 122     XYMIDL( IE ) = .5 1912
1913 123     XMIDL( IE ) = XERIGT( KE ) 1913
1914 124     YMIDL( IE ) = YERIGT( KE ) 1914
1915 125 C 1915
1916 126 C ----- 1916
1917 127 C 1917
1918 128 C 1918
1919 129 C --- REGULAR TRIANGLES ----- 1919
1920 130 C 1920
1921 131     ELSE 1921
1922 132     IV3 = JS( 1 , ISL ) 1922
1923 133     IF( IV3 . EQ . IV1 . OR . IV3 . EQ . IV2 ) IV3 = JS( 2 , ISL ) 1923
1924 134     IF( IV3 . EQ . IV1 . OR . IV3 . EQ . IV2 ) IV3 = JS( 3 , ISL ) 1924
1925 135     XELEFT( KE ) = ( XV( 1 , IV3 ) + XV( 1 , IV2 ) + 1925
1926 136     . XV( 1 , IV1 ) ) * THIRD 1926
1927 137     YELEFT( KE ) = ( XV( 2 , IV3 ) + XV( 2 , IV2 ) + 1927
1928 138     . XV( 2 , IV1 ) ) * THIRD 1928
1929 139 C 1929
1930 140     IV3 = JS( 1 , ISR ) 1930
1931 141     IF( IV3 . EQ . IV1 . OR . IV3 . EQ . IV2 ) IV3 = JS( 2 , ISR ) 1931
1932 142     IF( IV3 . EQ . IV1 . OR . IV3 . EQ . IV2 ) IV3 = JS( 3 , ISR ) 1932
1933 143     XERIGT( KE ) = ( XV( 1 , IV3 ) + XV( 1 , IV2 ) + 1933
1934 144     . XV( 1 , IV1 ) ) * THIRD 1934
1935 145     YERIGT( KE ) = ( XV( 2 , IV3 ) + XV( 2 , IV2 ) + 1935
1936 146     . XV( 2 , IV1 ) ) * THIRD 1936
1937 147 C 1937
1938 148     DXD = XERIGT( KE ) - XELEFT( KE ) 1938
1939 149     DYD = YERIGT( KE ) - YELEFT( KE ) 1939
1940 150 C 1940
1941 151 C --- LENGTH OF LINE BETWEEN BARI-CENTERS ----- 1941
1942 152 C   STORED IN XE(2,IE) 1942
1943 153 C 1943
1944 154     XE( 2 , IE ) = SQRT( DXD * DXD + DYD * DYD ) 1944
1945 155 C 1945
1946 156 C --- UNIT VECTOR FROM LEFT TO RIGHT BARI-CENTER ----- 1946
1947 157 C   STORED IN XXN(IE),YYN(IE) 1947
1948 158 C 1948
1949 159     XY = 1. / XE( 2 , IE ) 1949
1950 160     XXN( IE ) = DXD * XY 1950
1951 161     YYN( IE ) = DYD * XY 1951
1952 162 C 1952
1953 163 C --- COORDINATES OF BARI-CENTERS FOR EACH TRIANGLE ----- 1953
1954 164 C   STORED IN XS(1,IS),XS(2,IS) 1954
1955 165 C 1955
1956 166     XS( 1 , ISL ) = XELEFT( KE ) 1956
1957 167     XS( 2 , ISL ) = YELEFT( KE ) 1957
1958 168     XS( 1 , ISR ) = XERIGT( KE ) 1958
1959 169     XS( 2 , ISR ) = YERIGT( KE ) 1959
1960 170 C 1960
1961 171     AA = XV( 1 , IV2 ) - XV( 1 , IV1 ) 1961
1962 172     BB = XV( 2 , IV2 ) - XV( 2 , IV1 ) 1962
1963 173     CC = XELEFT( KE ) - XERIGT( KE ) 1963
1964 174     DD = YELEFT( KE ) - YERIGT( KE ) 1964
1965 175     ACA = XERIGT( KE ) - XV( 1 , IV1 ) 1965
1966 176     DBD = YERIGT( KE ) - XV( 2 , IV1 ) 1966
1967 177     EE = ( ACA * DD - DBD * CC ) / ( AA * DD - BB * CC ) 1967
1968 178 C 1968
1969 179 C --- INTERSECTION POINT ON INTERFACE FOR LINE CONNECTING BARI-CENTERS - 1969
1970 180 C   STORED IN XMIDL(IE),YMIDL(IE) AND FRACTION OF LENGHT BETWEEN 1970
1971 181 C   LEFT BARI-CENTER TO INTERSECTION POINT IN XYMIDL(IE). 1971
1972 182 C 1972
1973 183     XMIDL( IE ) = XV( 1 , IV1 ) + AA * EE 1973
1974 184     YMIDL( IE ) = XV( 2 , IV1 ) + BB * EE 1974
1975 185 C 1975

```

```

1976 186 XEMID = XMIDL( IE ) - XELEFT( KE )
1977 187 YEMID = YMIDL( IE ) - YELEFT( KE )
1978 188 C
1979 189 XYMIDL( IE ) = SQRT( XEMID * XEMID + YEMID * YEMID ) * XY
1980 190 C
1981 191 ENDIF
1982 192 140 CONTINUE
1983 193 C
1984 194 NE1 = NE2 + 1
1985 195 NE2 = NE2 + NOFVEE( INE + 1 )
1986 196 110 CONTINUE
1987 197 C
1988 198 C --- CALCULATE AREA OF TRIANGLES -----
1989 199 C
1990 200 DO 150 IS = 1 , NS
1991 201 IV1 = JS( 1 , IS )
1992 202 IV2 = JS( 2 , IS )
1993 203 IV3 = JS( 3 , IS )
1994 204 DX = XV( 1 , IV2 ) - XV( 1 , IV1 )
1995 205 DXX = XV( 1 , IV3 ) - XV( 1 , IV1 )
1996 206 DY = XV( 2 , IV2 ) - XV( 2 , IV1 )
1997 207 DYY = XV( 2 , IV3 ) - XV( 2 , IV1 )
1998 208 XS( 3 , IS ) = .5 * ( DX * DYY - DXX * DY )
1999 209 150 CONTINUE
2000 210 C
2001 211 PRINT * , NE,NS
2002 212 C
2003 213 C --- FIND AN EDGE ASSOCIATED WITH A VERTEX -----
2004 214 C THE VALUE WILL BE NEGATIVE IF ON THE BOUNDARY
2005 215 C
2006 216 DO 180 IV = 1 , NV
2007 217 JV( 2 , IV ) = 0
2008 218 180 CONTINUE
2009 219 C
2010 220 DO 160 IE = 1 , NE
2011 221 IV1 = JE( 1 , IE )
2012 222 IJES = JE( 5 , IE )
2013 223 IF( IJES . NE . 0 ) THEN
2014 224 JV( 2 , IV1 ) = - IE
2015 225 END IF
2016 226 160 CONTINUE
2017 227 C
2018 228 DO 170 IE = 1 , NE
2019 229 C
2020 230 IV1 = JE( 1 , IE )
2021 231 IV2 = JE( 2 , IE )
2022 232 C
2023 233 IF( JV( 2 , IV1 ) . EQ . 0 ) THEN
2024 234 JV( 2 , IV1 ) = IE
2025 235 END IF
2026 236 C
2027 237 IF( JV( 2 , IV2 ) . EQ . 0 ) THEN
2028 238 JV( 2 , IV2 ) = IE
2029 239 END IF
2030 240 C
2031 241 170 CONTINUE
2032 242 C
2033 243 DO 190 IS = 1 , NS
2034 244 SAREA( IS ) = 1. / XS( 3 , IS )
2035 245 190 CONTINUE
2036 246 C
2037 247 -----
2038 248 C
2039 249 C --- OPTION FOR GLOBAL RECONNECTION -----
2040 250 C
2041 251 IF( IOPRCN.EQ.1)THEN
2042 252 DO 200 IE = 1 , NE
2043 253 CALL RECNC( IE , IDONE , ITL , ITR , JA , JB , JC , JD )
2044 254 CALL RECNC( JA , JADONE , ITL , ITR , JAA , JAB , JAC , JAD )
2045 255 CALL RECNC( JB , JBDONE , ITL , ITR , JBA , JBB , JBC , JBD )
2046 256 CALL RECNC( JC , JCDONE , ITL , ITR , JCA , JCB , JCC , JCD )
2047 257 CALL RECNC( JD , JDDONE , ITL , ITR , JDA , JDB , JDC , JDD )
2048 258 200 CONTINUE
2049 259 ENDIF

```

```

2050 260 C
2051 261 C -----
2052 262 C
2053 263 C --- EXIT POINT FROM SUBROUTINE -----
2054 264 C
2055 265 C -----
2056 266 C RETURN
2057 267 C -----
2058 268 C
2059 269 C ---
2060 270 C END

```

```

2061 1 SUBROUTINE UPDATE
2062 2 C
2063 3 C ----- I
2064 4 C I
2065 5 C UPDATE BUFFERS IN A BLOCK OF NODES, EDGES, AND CELLS TO I
2066 6 C CONSTRUCT THE NEW GEOMETRY FOR THE MESH. I
2067 7 C THE BLOCKING SIZE IS DETERMINED BY PARAMETER ...MBL... I
2068 8 C I
2069 9 C ----- I
2070 10 C
2071 11 C -----
2072 12 C
2073 13 C include 'cmsh00.h'
2074 14 C include 'chyd00.h'
2075 15 C include 'cint00.h'
2076 16 C include 'cphs10.h'
2077 17 C include 'cphs20.h'
2078 18 C
2079 19 C -----
2080 20 C
2081 21 C --- BREAK UP THE VERTEX, EDGE, AND TRIANGLE DATA INTO BLOCKS -----
2082 22 C
2083 23 NVECE = NE / MBL
2084 24 NREME = NE - NVECE * MBL
2085 25 NVECS = NS / MBL
2086 26 NREMS = NS - NVECS * MBL
2087 27 NVECV = NV / MBL
2088 28 NREMV = NV - NVECV * MBL
2089 29 PRINT *, NV, NE, NS, NVECE, NREME, NVECV, NREMV, NVECS, NREMS
2090 30 C
2091 31 DO 105 INE = 1, NVECE
2092 32 NOFVEE( INE ) = MBL
2093 33 105 CONTINUE
2094 34 NVEEE = NVECE
2095 35 IF( NREME .GT. 0 ) THEN
2096 36 NVEEE = NVECE + 1
2097 37 NOFVEE( NVEEE ) = NREME
2098 38 END IF
2099 39 C
2100 40 DO 115 INS = 1, NVECS
2101 41 NOFVES( INS ) = MBL
2102 42 115 CONTINUE
2103 43 NVEES = NVECS
2104 44 IF( NREMS .GT. 0 ) THEN
2105 45 NVEES = NVECS + 1
2106 46 NOFVES( NVEES ) = NREMS
2107 47 END IF
2108 48 C
2109 49 DO 125 INV = 1, NVECV
2110 50 NOFVEV( INV ) = MBL
2111 51 125 CONTINUE
2112 52 NVEEV = NVECV
2113 53 IF( NREMV .GT. 0 ) THEN
2114 54 NVEEV = NVECV + 1
2115 55 NOFVEV( NVEEV ) = NREMV
2116 56 END IF
2117 57 C
2118 58 C --- CALL TO THE GEOMETRY DEFINITION SUBROUTINE -----
2119 59 C
2120 60 CALL GEOMTR

```


2121	61	C		2121
2122	62	C	-----	2122
2123	63	C		2123
2124	64	C	--- EXIT POINT FROM SUBROUTINE -----	2124
2125	65	C		2125
2126	66	C	-----	2126
2127	67	C	RETURN	2127
2128	68	C	-----	2128
2129	69	C		2129
2130	70	C	---	2130
2131	71	C	END	2131

2132	1	C	SUBROUTINE UPGRAD	2132
2133	2	C		2133
2134	3	C	-----	2134
2135	4	C		2135
2136	5	C	UPGRAD READS THE RESTART FILE FROM A PREVIOUS RUN	2136
2137	6	C	AND BREAKS THE DATA INTO BLOCKS AS DEFINED BY THE	2137
2138	7	C	PARAMETER ...MBL...	2138
2139	8	C		2139
2140	9	C	-----	2140
2141	10	C		2141
2142	11	C	include 'cmsh00.h'	2142
2143	12	C	include 'chyd00.h'	2143
2144	13	C	include 'cint00.h'	2144
2145	14	C	include 'cphs10.h'	2145
2146	15	C	include 'cphs20.h'	2146
2147	16	C		2147
2148	17	C	-----	2148
2149	18	C		2149
2150	19	C	MVM MAX NUMBER OF VERTICES (POINTS)	2150
2151	20	C	MEM MAX NUMBER OF EDGES (INTERFACES)	2151
2152	21	C	MSM MAX NUMBER OF SIDES (TRIANGLES)	2152
2153	22	C		2153
2154	23	C	-----	2154
2155	24	C		2155
2156	25	C		2156
2157	26	C	READ (88) NV,NVMK,NE,NEMK,NS,NSMK,ITERAT	2157
2158	27	C		2158
2159	28	C	--- READ IN VERTEX INFORMATION -----	2159
2160	29	C		2160
2161	30	C	READ (88) ((JV(IK,IV),IK=1,2),(XV(IK,IV),IK=1,2),IV=1,NV)	2161
2162	31	C		2162
2163	32	C	--- READ IN EDGE INFORMATION (EDGES OF TRIANGLES).-----	2163
2164	33	C		2164
2165	34	C	READ (88) ((JE(KK,IE),KK=1,5),(XE(KI,IE),KI=1,2),IE=1,NE)	2165
2166	35	C	READ (88) (XN(IE),YN(IE),XXN(IE),YYN(IE),IE=1,NE)	2166
2167	36	C		2167
2168	37	C	--- READ IN SIDE (TRIANGLE) INFORMATION.-----	2168
2169	38	C		2169
2170	39	C	READ (88) ((JS(KK,IS),KK=1,6),(XS(KI,IS),KI=1,3),IS=1,NS)	2170
2171	40	C	READ (88) (XMIDL(IE),YMIDL(IE),XYMIDL(IE),IE=1,NE)	2171
2172	41	C	READ (88) SAREVG,NVECE,NREME,NVECV,NREMV,NVECS,NREMS	2172
2173	42	C		2173
2174	43	C	--- PRINT PROMT TO CONSOLE -----	2174
2175	44	C		2175
2176	45	C	PRINT * , NE,NS	2176
2177	46	C		2177
2178	47	C	--- DEFINE INVERSE AREA OF TRIANGLES -----	2178
2179	48	C		2179
2180	49	C	DO 100 IS = 1 , NS	2180
2181	50	C	SAREA(IS) = 1. / XS(3 , IS)	2181
2182	51	C	100 CONTINUE	2182
2183	52	C		2183
2184	53	C	--- BREAKUP THE DATA STRUCTURE INTO BLOCKS -----	2184
2185	54	C		2185
2186	55	C	DO 105 INE = 1 , NVECE	2186
2187	56	C	NOFVEE(INE) = MBL	2187
2188	57	C	105 CONTINUE	2188
2189	58	C	NVEEE = NVECE	2189
2190	59	C	IF(NREME . GT . 0) THEN	2190
2191	60	C	NVEEE = NVECE + 1	2191

```
2192 61      NOFVEE( NVEEE ) = NREME                2192
2193 62      END IF                                2193
2194 63      C                                     2194
2195 64      DO 115 INS = 1 , NVECS                2195
2196 65      NOFVES( INS ) = MBL                  2196
2197 66      115 CONTINUE                          2197
2198 67      NVEES = NVECS                          2198
2199 68      IF( NREMS .GT. 0 ) THEN              2199
2200 69      NVEES = NVECS + 1                      2200
2201 70      NOFVES( NVEES ) = NREMS              2201
2202 71      END IF                                2202
2203 72      C                                     2203
2204 73      DO 125 INV = 1 , NVECV                2204
2205 74      NOFVEV( INV ) = MBL                  2205
2206 75      125 CONTINUE                          2206
2207 76      NVEEV = NVECV                          2207
2208 77      IF( NREMV .GT. 0 ) THEN              2208
2209 78      NVEEV = NVECV + 1                      2209
2210 79      NOFVEV( NVEEV ) = NREMV              2210
2211 80      END IF                                2211
2212 81      C                                     2212
2213 82      C --- PRINTOUT THE VERTEX,EDGE, AND TRIANGLE BLOCK DATA ----- 2213
2214 83      C                                     2214
2215 84      PRINT *,NV,NE,NS,NVECE,NREME,NVECV,NREMV,NVECS,NREMS 2215
2216 85      C                                     2216
2217 86      C----- 2217
2218 87      C                                     2218
2219 88      C --- EXIT POINT FROM SUBROUTINE ----- 2219
2220 89      C                                     2220
2221 90      C ----- 2221
2222 91      RETURN                                2222
2223 92      C ----- 2223
2224 93      C                                     2224
2225 94      C ----- 2225
2226 95      END                                  2226
```

Thu Jul 1 14:15:55 1993

gradhd.f

Module List - order of occurrence

page i

#	routine	page
1	GRDFLX	1
2	GRDENG	4
3	GRADNL	7
4	GRADNT	11
5	MONOTN	15
6	GRADNG	23
7	GRADNO	25
8	GRADNS	27
9	LUDCMP	29
10	LUBKSB	30
11	FIRST	31
12	FCHART	32
13	PRLCTN	36
14	PRPTH	37

Thu Jul 1 14:15:55 1993

gradhd.f

Module List - alphabetical order

#	routine	page
1	FCHART	32
2	FIRST	31
3	GRADNG	23
4	GRADNL	7
5	GRADNO	25
6	GRADNS	27
7	GRADNT	11
8	GRDENG	4
9	GRDFLX	1
10	LUBKSB	30
11	LUDCMP	29
12	MONOTN	15
13	PRLCTN	36
14	PRPTH	37

```

1      1      SUBROUTINE GRDFLX                                1
2      2      C                                              2
3      3      C-----I                                        3
4      4      C                                              4
5      5      C      GRAFLX COMPUTE THE GRADIENT FOR ERROR INDICATOR  I
6      6      C      CALCULATION FOR STEADY STATE                I
7      7      C                                              I
8      8      C-----I                                        8
9      9      C                                              9
10     10     include 'cmsh00.h'                                10
11     11     include 'chyd00.h'                                11
12     12     include 'cint00.h'                                12
13     13     include 'cphs10.h'                                13
14     14     include 'cphs20.h'                                14
15     15     C                                              15
16     16     C-----I                                        16
17     17     C                                              17
18     18     REAL AA(3,3),BB(3,4),B(3),INDX(3),ATEMP(3,3,3),BTEMP(3,4,3)  18
19     19     REAL AAO(3,3),BB0(3,4)                            19
20     20     C                                              20
21     21     C-----I                                        21
22     22     C                                              22
23     23     C                                              23
24     24     C --- BEGIN LOOP OVER ALL CELLS IN THE DOMAIN ----- 24
25     25     C                                              25
26     26     NS1 = 1                                           26
27     27     NS2 = NOFVES( 1 )                                  27
28     28     DO 90 INS = 1 , NVEES                             28
29     29     C                                              29
30     30     C --- FETCH HYDRO QUANTITIES -----              30
31     31     C                                              31
32     32     DO 105 IS = NS1 , NS2                              32
33     33     KS = IS - NS1 + 1                                  33
34     34     C                                              34
35     35     XSM = XS( 1 , IS )                                  35
36     36     YSM = XS( 2 , IS )                                  36
37     37     XSM2 = XSM * XSM                                    37
38     38     YSM2 = YSM * YSM                                    38
39     39     XYSM = XSM * YSM                                    39
40     40     C                                              40
41     41     AAO( 1 , 1 ) = 1.0                                  41
42     42     AAO( 1 , 2 ) = XSM                                  42
43     43     AAO( 1 , 3 ) = YSM                                  43
44     44     C                                              44
45     45     AAO( 2 , 1 ) = XSM                                  45
46     46     AAO( 2 , 2 ) = XSM2                                46
47     47     AAO( 2 , 3 ) = XYSM                                47
48     48     C                                              48
49     49     AAO( 3 , 1 ) = YSM                                  49
50     50     AAO( 3 , 2 ) = XYSM                                50
51     51     AAO( 3 , 3 ) = YSM2                                51
52     52     C                                              52
53     53     BB1 = HYDV( IS , 4 )                                53
54     54     BB2 = SQRT ( ( HYDV( IS , 2 ) * HYDV( IS , 2 ) +      54
55     55     .          HYDV( IS , 3 ) * HYDV( IS , 3 ) ) *      55
56     56     .          HYDV( IS , 1 ) / HYDV( IS , 4 ) / HYDV( IS , 5 ) )  56
57     57     C                                              57
58     58     BB1X = BB1 * XSM                                    58
59     59     BB2X = BB2 * XSM                                    59
60     60     C                                              60
61     61     BB1Y = BB1 * YSM                                    61
62     62     BB2Y = BB2 * YSM                                    62
63     63     C                                              63
64     64     BB0( 1 , 1 ) = BB1                                  64
65     65     BB0( 1 , 2 ) = BB2                                  65
66     66     C                                              66
67     67     BB0( 2 , 1 ) = BB1X                                  67
68     68     BB0( 2 , 2 ) = BB2X                                  68
69     69     C                                              69
70     70     BB0( 3 , 1 ) = BB1Y                                  70
71     71     BB0( 3 , 2 ) = BB2Y                                  71
72     72     C                                              72
73     73     DO 115 IK = 1 , 3                                  73

```

```

74 74 IE = JS( IK + 3 , IS ) 74
75 75 IF( IE . GT . 0 ) THEN 75
76 76 ISS = JE( 4 , IE ) 76
77 77 ELSE 77
78 78 ISS = JE( 3 , -IE ) 78
79 79 END IF 79
80 80 C 80
81 81 IF( ISS . NE . 0 ) THEN 81
82 82 XSS = XS( 1 , ISS ) 82
83 83 YSS = XS( 2 , ISS ) 83
84 84 C 84
85 85 HYDVP = HYDV( ISS , 4 ) 85
86 86 HYDVR = SQRT ( ( HYDV( ISS , 2 ) * HYDV( ISS , 2 ) + 86
87 87 HYDV( ISS , 3 ) * HYDV( ISS , 3 ) ) * 87
88 88 HYDV( ISS , 1 ) / HYDV( ISS , 4 ) / HYDV( ISS , 5 ) ) 88
89 89 C 89
90 90 ELSE 90
91 91 C 91
92 92 IE = IABS( IE ) 92
93 93 XSS = 2. * XMIDL( IE ) - XSM 93
94 94 YSS = 2. * YMIDL( IE ) - YSM 94
95 95 C 95
96 96 HYDVP = BB1 96
97 97 HYDVR = BB2 97
98 98 IJES = JE( 5 , IE ) 98
99 99 IF( IJES . EQ . 8 ) THEN 99
100 100 HYDVP = PIN 100
101 101 HYDVR = SQRT( UVIN * UVIN * RIN / PIN / HRGG ) 101
102 102 END IF 102
103 103 C 103
104 104 END IF 104
105 105 C 105
106 106 XSS2 = XSS * XSS 106
107 107 YSS2 = YSS * YSS 107
108 108 XYSS = XSS * YSS 108
109 109 C 109
110 110 ATEMP( 1 , 1 , IK ) = 1.0 110
111 111 ATEMP( 1 , 2 , IK ) = XSS 111
112 112 ATEMP( 1 , 3 , IK ) = YSS 112
113 113 C 113
114 114 ATEMP( 2 , 1 , IK ) = XSS 114
115 115 ATEMP( 2 , 2 , IK ) = XSS2 115
116 116 ATEMP( 2 , 3 , IK ) = XYSS 116
117 117 C 117
118 118 ATEMP( 3 , 1 , IK ) = YSS 118
119 119 ATEMP( 3 , 2 , IK ) = XYSS 119
120 120 ATEMP( 3 , 3 , IK ) = YSS2 120
121 121 C 121
122 122 BTEMP( 1 , 1 , IK ) = HYDVP 122
123 123 BTEMP( 1 , 2 , IK ) = HYDVR 123
124 124 C 124
125 125 BTEMP( 2 , 1 , IK ) = HYDVP * XSS 125
126 126 BTEMP( 2 , 2 , IK ) = HYDVR * XSS 126
127 127 C 127
128 128 BTEMP( 3 , 1 , IK ) = HYDVP * YSS 128
129 129 BTEMP( 3 , 2 , IK ) = HYDVR * YSS 129
130 130 C 130
131 131 115 CONTINUE 130
132 132 C 131
133 133 AA( 1 , 1 ) = AAO( 1 , 1 ) + ATEMP( 1 , 1 , 1 ) + 132
134 134 ATEMP( 1 , 1 , 2 ) + ATEMP( 1 , 1 , 3 ) 133
135 135 AA( 1 , 2 ) = AAO( 1 , 2 ) + ATEMP( 1 , 2 , 1 ) + 134
136 136 ATEMP( 1 , 2 , 2 ) + ATEMP( 1 , 2 , 3 ) 135
137 137 AA( 1 , 3 ) = AAO( 1 , 3 ) + ATEMP( 1 , 3 , 1 ) + 136
138 138 ATEMP( 1 , 3 , 2 ) + ATEMP( 1 , 3 , 3 ) 137
139 139 C 138
140 140 AA( 2 , 1 ) = AAO( 2 , 1 ) + ATEMP( 2 , 1 , 1 ) + 139
141 141 ATEMP( 2 , 1 , 2 ) + ATEMP( 2 , 1 , 3 ) 140
142 142 AA( 2 , 2 ) = AAO( 2 , 2 ) + ATEMP( 2 , 2 , 1 ) + 141
143 143 ATEMP( 2 , 2 , 2 ) + ATEMP( 2 , 2 , 3 ) 142
144 144 AA( 2 , 3 ) = AAO( 2 , 3 ) + ATEMP( 2 , 3 , 1 ) + 143
145 145 ATEMP( 2 , 3 , 2 ) + ATEMP( 2 , 3 , 3 ) 144
146 146 C 145
147 147 AA( 3 , 1 ) = AAO( 3 , 1 ) + ATEMP( 3 , 1 , 1 ) + 146

```

```

148 148 . ATEMP( 3 , 1 , 2 ) + ATEMP( 3 , 1 , 3 ) 148
149 149 . AA( 3 , 2 ) = AAO( 3 , 2 ) + ATEMP( 3 , 2 , 1 ) + 149
150 150 . ATEMP( 3 , 2 , 2 ) + ATEMP( 3 , 2 , 3 ) 150
151 151 . AA( 3 , 3 ) = AAO( 3 , 3 ) + ATEMP( 3 , 3 , 1 ) + 151
152 152 . ATEMP( 3 , 3 , 2 ) + ATEMP( 3 , 3 , 3 ) 152
153 153 C 153
154 154 . BB( 1 , 1 ) = BBO( 1 , 1 ) + BTEMP( 1 , 1 , 1 ) + 154
155 155 . BTEMP( 1 , 1 , 2 ) + BTEMP( 1 , 1 , 3 ) 155
156 156 . BB( 1 , 2 ) = BBO( 1 , 2 ) + BTEMP( 1 , 2 , 1 ) + 156
157 157 . BTEMP( 1 , 2 , 2 ) + BTEMP( 1 , 2 , 3 ) 157
158 158 C 158
159 159 . BB( 2 , 1 ) = BBO( 2 , 1 ) + BTEMP( 2 , 1 , 1 ) + 159
160 160 . BTEMP( 2 , 1 , 2 ) + BTEMP( 2 , 1 , 3 ) 160
161 161 . BB( 2 , 2 ) = BBO( 2 , 2 ) + BTEMP( 2 , 2 , 1 ) + 161
162 162 . BTEMP( 2 , 2 , 2 ) + BTEMP( 2 , 2 , 3 ) 162
163 163 C 163
164 164 . BB( 3 , 1 ) = BBO( 3 , 1 ) + BTEMP( 3 , 1 , 1 ) + 164
165 165 . BTEMP( 3 , 1 , 2 ) + BTEMP( 3 , 1 , 3 ) 165
166 166 . BB( 3 , 2 ) = BBO( 3 , 2 ) + BTEMP( 3 , 2 , 1 ) + 166
167 167 . BTEMP( 3 , 2 , 2 ) + BTEMP( 3 , 2 , 3 ) 167
168 168 C 168
169 169 DETERM = AA( 1 , 1 ) * ( AA( 2 , 2 ) * AA( 3 , 3 ) - 169
170 170 . AA( 3 , 2 ) * AA( 2 , 3 ) ) + 170
171 171 . AA( 2 , 1 ) * ( AA( 3 , 2 ) * AA( 1 , 3 ) - 171
172 172 . AA( 1 , 2 ) * AA( 3 , 3 ) ) + 172
173 173 . AA( 3 , 1 ) * ( AA( 1 , 2 ) * AA( 2 , 3 ) - 173
174 174 . AA( 2 , 2 ) * AA( 1 , 3 ) ) 174
175 175 C 175
176 176 DTRMIN = 1. / DETERM 176
177 177 C 177
178 178 AAA1 = AA( 2 , 3 ) * AA( 3 , 1 ) - AA( 2 , 1 ) * AA( 3 , 3 ) 178
179 179 AAA2 = AA( 3 , 3 ) * AA( 1 , 1 ) - AA( 3 , 1 ) * AA( 1 , 3 ) 179
180 180 AAA3 = AA( 1 , 3 ) * AA( 2 , 1 ) - AA( 1 , 1 ) * AA( 2 , 3 ) 180
181 181 C 181
182 182 AAA4 = AA( 2 , 1 ) * AA( 3 , 2 ) - AA( 3 , 1 ) * AA( 2 , 2 ) 182
183 183 AAA5 = AA( 3 , 1 ) * AA( 1 , 2 ) - AA( 1 , 1 ) * AA( 3 , 2 ) 183
184 184 AAA6 = AA( 1 , 1 ) * AA( 2 , 2 ) - AA( 2 , 1 ) * AA( 1 , 2 ) 184
185 185 C 185
186 186 PL( IS ) = DTRMIN * ( BB( 1 , 1 ) * AAA1 + 186
187 187 . BB( 2 , 1 ) * AAA2 + 187
188 188 . BB( 3 , 1 ) * AAA3 ) 188
189 189 C 189
190 190 PR( IS ) = DTRMIN * ( BB( 1 , 1 ) * AAA4 + 190
191 191 . BB( 2 , 1 ) * AAA5 + 191
192 192 . BB( 3 , 1 ) * AAA6 ) 192
193 193 C 193
194 194 RL( IS ) = DTRMIN * ( BB( 1 , 2 ) * AAA1 + 194
195 195 . BB( 2 , 2 ) * AAA2 + 195
196 196 . BB( 3 , 2 ) * AAA3 ) 196
197 197 C 197
198 198 RR( IS ) = DTRMIN * ( BB( 1 , 2 ) * AAA4 + 198
199 199 . BB( 2 , 2 ) * AAA5 + 199
200 200 . BB( 3 , 2 ) * AAA6 ) 200
201 201 C 201
202 202 105 CONTINUE 202
203 203 C 203
204 204 NS1 = NS2 + 1 204
205 205 NS2 = NS2 + NOFVES( INS + 1 ) 205
206 206 90 CONTINUE 206
207 207 C 207
208 208 C----- 208
209 209 C 209
210 210 C --- EXIT POINT FROM SUBROUTINE ----- 210
211 211 C 211
212 212 C 212
213 213 RETURN 213
214 214 C ----- 214
215 215 C 215
216 216 C --- 216
217 217 END 217

```

```

218      1      SUBROUTINE GRDENG
219      2      C
220      3      C-----]
221      4      C
222      5      C      GRAENG COMPUTE THE GRADIENT FOR SECOND ORDER CALCULATION
223      6      C
224      7      C-----]
225      8      C
226      9      include 'cmsh00.h'
227     10      include 'chyd00.h'
228     11      include 'cint00.h'
229     12      include 'cphs10.h'
230     13      include 'cphs20.h'
231     14      C
232     15      C-----]
233     16      C
234     17      REAL RRMIDL(MBP),PPMIDL(MBP),UUMIDL(MBP),VVMIDL(MBP)
235     18      REAL RIGRAD(MBP),PIGRAD(MBP),UIGRAD(MBP),VIGRAD(MBP)
236     19      REAL RJGRAD(MBP),PJGRAD(MBP),UJGRAD(MBP),VJGRAD(MBP)
237     20      REAL AA(3,3),BB(3,4),B(3),INDX(3),ATEMP(3,3,3),BTEMP(3,4,3)
238     21      REAL AAO(3,3),BBO(3,4)
239     22      C
240     23      C-----]
241     24      C
242     25      C --- BEGIN LOOP OVER ALL CELLS IN THE DOMAIN -----]
243     26      C
244     27      NS1 = 1
245     28      NS2 = NOFVES( 1 )
246     29      DO 90 INS = 1 , NVEES
247     30      C
248     31      C --- FETCH HYDRO QUANTITIES -----]
249     32      C
250     33      DO 105 IS = NS1 , NS2
251     34      C
252     35      XSM = XS( 1 , IS )
253     36      YSM = XS( 2 , IS )
254     37      XSM2 = XSM * XSM
255     38      YSM2 = YSM * YSM
256     39      XYSM = XSM * YSM
257     40      C
258     41      AAO( 1 , 1 ) = 1.0
259     42      AAO( 1 , 2 ) = XSM
260     43      AAO( 1 , 3 ) = YSM
261     44      C
262     45      AAO( 2 , 1 ) = XSM
263     46      AAO( 2 , 2 ) = XSM2
264     47      AAO( 2 , 3 ) = XYSM
265     48      C
266     49      AAO( 3 , 1 ) = YSM
267     50      AAO( 3 , 2 ) = XYSM
268     51      AAO( 3 , 3 ) = YSM2
269     52      C
270     53      BB1 = HYDFLX( IS , 1 )
271     54      BB2 = HYDFLX( IS , 2 )
272     55      BB3 = HYDFLX( IS , 4 )
273     56      C
274     57      BB1X = BB1 * XSM
275     58      BB2X = BB2 * XSM
276     59      BB3X = BB3 * XSM
277     60      C
278     61      BB1Y = BB1 * YSM
279     62      BB2Y = BB2 * YSM
280     63      BB3Y = BB3 * YSM
281     64      C
282     65      BBO( 1 , 1 ) = BB1
283     66      BBO( 1 , 2 ) = BB2
284     67      BBO( 1 , 3 ) = BB3
285     68      C
286     69      BBO( 2 , 1 ) = BB1X
287     70      BBO( 2 , 2 ) = BB2X
288     71      BBO( 2 , 3 ) = BB3X
289     72      C
290     73      BBO( 3 , 1 ) = BB1Y
291     74      BBO( 3 , 2 ) = BB2Y

```

292	75		BB0(3 , 3) = BB3Y	292
293	76	C		293
294	77		DO 115 IK = 1 , 3	294
295	78		IE = JS(IK + 3 , IS)	295
296	79		IF(IE . GT . 0) THEN	296
297	80		ISS = JE(4 , IE)	297
298	81		ELSE	298
299	82		ISS = JE(3 , - IE)	299
300	83		END IF	300
301	84	C		301
302	85		IF(ISS . NE . 0) THEN	302
303	86		XSS = XS(1 , ISS)	303
304	87		YSS = XS(2 , ISS)	304
305	88	C		305
306	89		HYDVR = HYDFLX(ISS , 1)	306
307	90		HYDVU = HYDFLX(ISS , 2)	307
308	91		HYDVV = HYDFLX(ISS , 4)	308
309	92	C		309
310	93		ELSE	310
311	94	C		311
312	95		IE = IABS(IE)	312
313	96		HYDVR = BB1	313
314	97		HYDVU = BB2	314
315	98		HYDVV = BB3	315
316	99	C		316
317	100		XSS = 2. * XMIDL(IE) - XSM	317
318	101		YSS = 2. * YMIDL(IE) - YSM	318
319	102	C		319
320	103		END IF	320
321	104	C		321
322	105		XSS2 = XSS * XSS	322
323	106		YSS2 = YSS * YSS	323
324	107		XYSS = XSS * YSS	324
325	108	C		325
326	109		ATEMP(1 , 1 , IK) = 1.0	326
327	110		ATEMP(1 , 2 , IK) = XSS	327
328	111		ATEMP(1 , 3 , IK) = YSS	328
329	112	C		329
330	113		ATEMP(2 , 1 , IK) = XSS	330
331	114		ATEMP(2 , 2 , IK) = XSS2	331
332	115		ATEMP(2 , 3 , IK) = XYSS	332
333	116	C		333
334	117		ATEMP(3 , 1 , IK) = YSS	334
335	118		ATEMP(3 , 2 , IK) = XYSS	335
336	119		ATEMP(3 , 3 , IK) = YSS2	336
337	120	C		337
338	121		BTEMP(1 , 1 , IK) = HYDVR	338
339	122		BTEMP(1 , 2 , IK) = HYDVU	339
340	123		BTEMP(1 , 3 , IK) = HYDVV	340
341	124	C		341
342	125		BTEMP(2 , 1 , IK) = HYDVR * XSS	342
343	126		BTEMP(2 , 2 , IK) = HYDVU * XSS	343
344	127		BTEMP(2 , 3 , IK) = HYDVV * XSS	344
345	128	C		345
346	129		BTEMP(3 , 1 , IK) = HYDVR * YSS	346
347	130		BTEMP(3 , 2 , IK) = HYDVU * YSS	347
348	131		BTEMP(3 , 3 , IK) = HYDVV * YSS	348
349	132	C		349
350	133		115 CONTINUE	350
351	134	C		351
352	135		AA(1 , 1) = AAO(1 , 1) + ATEMP(1 , 1 , 1) +	352
353	136		ATEMP(1 , 1 , 2) + ATEMP(1 , 1 , 3)	353
354	137		AA(1 , 2) = AAO(1 , 2) + ATEMP(1 , 2 , 1) +	354
355	138		ATEMP(1 , 2 , 2) + ATEMP(1 , 2 , 3)	355
356	139		AA(1 , 3) = AAO(1 , 3) + ATEMP(1 , 3 , 1) +	356
357	140		ATEMP(1 , 3 , 2) + ATEMP(1 , 3 , 3)	357
358	141	C		358
359	142		AA(2 , 1) = AAO(2 , 1) + ATEMP(2 , 1 , 1) +	359
360	143		ATEMP(2 , 1 , 2) + ATEMP(2 , 1 , 3)	360
361	144		AA(2 , 2) = AAO(2 , 2) + ATEMP(2 , 2 , 1) +	361
362	145		ATEMP(2 , 2 , 2) + ATEMP(2 , 2 , 3)	362
363	146		AA(2 , 3) = AAO(2 , 3) + ATEMP(2 , 3 , 1) +	363
364	147		ATEMP(2 , 3 , 2) + ATEMP(2 , 3 , 3)	364
365	148	C		365


```

366 149      AA( 3 , 1 ) = AAO( 3 , 1 ) + ATEMP( 3 , 1 , 1 ) + 366
367 150      . ATEMP( 3 , 1 , 2 ) + ATEMP( 3 , 1 , 3 ) 367
368 151      AA( 3 , 2 ) = AAO( 3 , 2 ) + ATEMP( 3 , 2 , 1 ) + 368
369 152      . ATEMP( 3 , 2 , 2 ) + ATEMP( 3 , 2 , 3 ) 369
370 153      AA( 3 , 3 ) = AAO( 3 , 3 ) + ATEMP( 3 , 3 , 1 ) + 370
371 154      . ATEMP( 3 , 3 , 2 ) + ATEMP( 3 , 3 , 3 ) 371
372 155      C 372
373 156      BB( 1 , 1 ) = BBO( 1 , 1 ) + BTEMP( 1 , 1 , 1 ) + 373
374 157      . BTEMP( 1 , 1 , 2 ) + BTEMP( 1 , 1 , 3 ) 374
375 158      BB( 1 , 2 ) = BBO( 1 , 2 ) + BTEMP( 1 , 2 , 1 ) + 375
376 159      . BTEMP( 1 , 2 , 2 ) + BTEMP( 1 , 2 , 3 ) 376
377 160      BB( 1 , 3 ) = BBO( 1 , 3 ) + BTEMP( 1 , 3 , 1 ) + 37.
378 161      . BTEMP( 1 , 3 , 2 ) + BTEMP( 1 , 3 , 3 ) 378
379 162      C 379
380 163      BB( 2 , 1 ) = BBO( 2 , 1 ) + BTEMP( 2 , 1 , 1 ) + 380
381 164      . BTEMP( 2 , 1 , 2 ) + BTEMP( 2 , 1 , 3 ) 381
382 165      BB( 2 , 2 ) = BBO( 2 , 2 ) + BTEMP( 2 , 2 , 1 ) + 382
383 166      . BTEMP( 2 , 2 , 2 ) + BTEMP( 2 , 2 , 3 ) 383
384 167      BB( 2 , 3 ) = BBO( 2 , 3 ) + BTEMP( 2 , 3 , 1 ) + 384
385 168      . BTEMP( 2 , 3 , 2 ) + BTEMP( 2 , 3 , 3 ) 385
386 169      C 386
387 170      BB( 3 , 1 ) = BBO( 3 , 1 ) + BTEMP( 3 , 1 , 1 ) + 387
388 171      . BTEMP( 3 , 1 , 2 ) + BTEMP( 3 , 1 , 3 ) 388
389 172      BB( 3 , 2 ) = BBO( 3 , 2 ) + BTEMP( 3 , 2 , 1 ) + 389
390 173      . BTEMP( 3 , 2 , 2 ) + BTEMP( 3 , 2 , 3 ) 390
391 174      BB( 3 , 3 ) = BBO( 3 , 3 ) + BTEMP( 3 , 3 , 1 ) + 391
392 175      . BTEMP( 3 , 3 , 2 ) + BTEMP( 3 , 3 , 3 ) 392
393 176      C 393
394 177      DETERM = AA( 1 , 1 ) * ( AA( 2 , 2 ) * AA( 3 , 3 ) - 394
395 178      . AA( 3 , 2 ) * AA( 2 , 3 ) ) + 395
396 179      . AA( 2 , 1 ) * ( AA( 1 , 3 ) * AA( 3 , 2 ) - 396
397 180      . AA( 3 , 3 ) * AA( 1 , 2 ) ) + 397
398 181      . AA( 3 , 1 ) * ( AA( 1 , 2 ) * AA( 2 , 3 ) - 398
399 182      . AA( 2 , 2 ) * AA( 1 , 3 ) ) 399
400 183      C 400
401 184      DTRMIN = 1. / DETERM 401
402 185      C 402
403 186      AAA1 = AA( 2 , 3 ) * AA( 3 , 1 ) - AA( 2 , 1 ) * AA( 3 , 3 ) 403
404 187      AAA2 = AA( 3 , 3 ) * AA( 1 , 1 ) - AA( 3 , 1 ) * AA( 1 , 3 ) 404
405 188      AAA3 = AA( 1 , 3 ) * AA( 2 , 1 ) - AA( 1 , 1 ) * AA( 2 , 3 ) 405
406 189      C 406
407 190      AAA4 = AA( 2 , 1 ) * AA( 3 , 2 ) - AA( 3 , 1 ) * AA( 2 , 2 ) 407
408 191      AAA5 = AA( 3 , 1 ) * AA( 1 , 2 ) - AA( 1 , 1 ) * AA( 3 , 2 ) 408
409 192      AAA6 = AA( 1 , 1 ) * AA( 2 , 2 ) - AA( 2 , 1 ) * AA( 1 , 2 ) 409
410 193      C 410
411 194      RR( IS ) = DTRMIN * ( BB( 1 , 1 ) * AAA1 + 411
412 195      . BB( 2 , 1 ) * AAA2 + 412
413 196      . BB( 3 , 1 ) * AAA3 ) 413
414 197      C 414
415 198      RL( IS ) = DTRMIN * ( BB( 1 , 1 ) * AAA4 + 415
416 199      . BB( 2 , 1 ) * AAA5 + 416
417 200      . BB( 3 , 1 ) * AAA6 ) 417
418 201      C 418
419 202      UR( IS ) = DTRMIN * ( BB( 1 , 2 ) * AAA1 + 419
420 203      . BB( 2 , 2 ) * AAA2 + 420
421 204      . BB( 3 , 2 ) * AAA3 ) 421
422 205      C 422
423 206      UL( IS ) = DTRMIN * ( BB( 1 , 2 ) * AAA4 + 423
424 207      . BB( 2 , 2 ) * AAA5 + 424
425 208      . BB( 3 , 2 ) * AAA6 ) 425
426 209      C 426
427 210      VR( IS ) = DTRMIN * ( BB( 1 , 3 ) * AAA1 + 427
428 211      . BB( 2 , 3 ) * AAA2 + 428
429 212      . BB( 3 , 3 ) * AAA3 ) 429
430 213      C 430
431 214      VL( IS ) = DTRMIN * ( BB( 1 , 3 ) * AAA4 + 431
432 215      . BB( 2 , 3 ) * AAA5 + 432
433 216      . BB( 3 , 3 ) * AAA6 ) 433
434 217      C 434
435 218      105 CONTINUE 435
436 219      C 436
437 220      NS1 = NS2 + 1 437
438 221      NS2 = NS2 + NOFVES( INS + 1 ) 438
439 222      90 CONTINUE 439

```

```

440 223 C
441 224 C -----
442 225 C
443 226 C --- EXIT POINT FROM SUBROUTINE -----
444 227 C
445 228 C -----
446 229 C RETURN
447 230 C -----
448 231 C
449 232 C ---
450 233 C END

```

```

451 1 SUBROUTINE GRADNL 451
452 2 C 452
453 3 C ----- I 453
454 4 C I 454
455 5 C GRADNL COMPUTE THE GRADIENT FOR SECOND ORDER CALCULATION I 455
456 6 C SEARCH FOR ALL TRIANGLES SURROUNDING THE TARGET I 456
457 7 C CELL FOR COMPUTING THE GRADIENT APPLYING LEAST I 457
458 8 C SQUARE TECHNIQUE I 458
459 9 C I 459
460 10 C ----- I 460
461 11 C 461
462 12 C include 'cmsh00.h' 462
463 13 C include 'chyd00.h' 463
464 14 C include 'cint00.h' 464
465 15 C include 'cphs10.h' 465
466 16 C include 'cphs20.h' 466
467 17 C 467
468 18 C ----- 468
469 19 C 469
470 20 REAL RRMIDL(MBP),PPMIDL(MBP),UUMIDL(MBP),VVMIDL(MBP) 470
471 21 REAL RIGRAD(MBP),PIGRAD(MBP),UIGRAD(MBP),VIGRAD(MBP) 471
472 22 REAL RJGRAD(MBP),PJGRAD(MBP),UJGRAD(MBP),VJGRAD(MBP) 472
473 23 REAL RMAX(MBP),PMAX(MBP),UMAX(MBP),VMAX(MBP) 473
474 24 REAL RMIN(MBP),PMIN(MBP),UMIN(MBP),VMIN(MBP) 474
475 25 REAL RLEFTT(MBP),ULEFTT(MBP),VLEFTT(MBP),PLEFTT(MBP) 475
476 26 REAL RRIGHT(MBP),URIGHT(MBP),VRIGHT(MBP),PRIGHT(MBP) 476
477 27 REAL ROR(3),UOR(3),VOR(3),POR(3) 477
478 28 REAL ROL(3),UOL(3),VOL(3),POL(3) 478
479 29 REAL AA(3,3),BB(3,4),B(3),INDX(3),ATEMP(3,3),BTEMP(3,4) 479
480 30 C 480
481 31 C ----- 481
482 32 C 482
483 33 C --- BEGIN LOOP OVER ALL CELLS IN THE DOMAIN ----- 483
484 34 C 484
485 35 NS1 = 1 485
486 36 NS2 = NOFVES( 1 ) 486
487 37 DO 90 INS = 1 , NVEES 487
488 38 C 488
489 39 C --- FETCH HYDRO QUANTITIES ----- 489
490 40 C 490
491 41 DO 105 IS = NS1 , NS2 491
492 42 C 492
493 43 JJCOLR = 0 493
494 44 C 494
495 45 DO 115 IK = 1 , 3 495
496 46 IVV = JS( IK , IS ) 496
497 47 IEE = JV( 2 , IVV ) 497
498 48 C 498
499 49 IF( IEE . GT . 0 ) THEN 499
500 50 C 500
501 51 IV1 = JE( 1 , IEE ) 501
502 52 IF( IV1 . EQ . IVV ) THEN 502
503 53 ISI = JE( 3 , IEE ) 503
504 54 ELSE 504
505 55 ISI = JE( 4 , IEE ) 505
506 56 END IF 506
507 57 ISS = ISI 507
508 58 IE = IEE 508
509 59 C 509
510 60 150 CONTINUE 510

```

511	61	C		511
512	62		JJCOLR = JJCOLR + 1	512
513	63		IICOLR(JJCOLR) = ISS	513
514	64	C		514
515	65		DO 160 IR = 1 , 3	515
516	66		JR = MOD(IR , 3) + 1	516
517	67		IEA = IABS(JS(JR + 3 , ISS))	517
518	68		IF(IEA . EQ . IE) THEN	518
519	69		JJR = MOD(JR + 1 , 3) + 4	519
520	70		IER = IABS(JS(JJR , ISS))	520
521	71	C		521
522	72		IVI = JE(1 , IER)	522
523	73		IF(IVI . EQ . IVV) THEN	523
524	74		ISR = JE(3 , IER)	524
525	75		ELSE	525
526	76		ISR = JE(4 , IER)	526
527	77		END IF	527
528	78		END IF	528
529	79	C		529
530	80	160	CONTINUE	530
531	81	C		531
532	82		IF(ISR . NE . ISI) THEN	532
533	83		ISS = ISR	533
534	84		IE = IER	534
535	85		GO TO 150	535
536	86		END IF	536
537	87	C		537
538	88		ELSE	538
539	89	C		539
540	90		IEE = - IEE	540
541	91		IVI = JE(1 , IEE)	541
542	92		IF(IVI . EQ . IVV) THEN	542
543	93		ISI = JE(3 , IEE)	543
544	94		ELSE	544
545	95		ISI = JE(4 , IEE)	545
546	96		END IF	546
547	97		ISS = ISI	547
548	98		ISI = 0	548
549	99		IE = IEE	549
550	100	C		550
551	101	170	CONTINUE	551
552	102	C		552
553	103		JJCOLR = JJCOLR + 1	553
554	104		IICOLR(JJCOLR) = ISS	554
555	105	C		555
556	106		DO 180 IR = 1 , 3	556
557	107		JR = MOD(IR , 3) + 1	557
558	108		IEA = IABS(JS(JR + 3 , ISS))	558
559	109		IF(IEA . EQ . IE) THEN	559
560	110		JJR = MOD(JR + 1 , 3) + 4	560
561	111		IER = IABS(JS(JJR , ISS))	561
562	112	C		562
563	113		IVI = JE(1 , IER)	563
564	114		IF(IVI . EQ . IVV) THEN	564
565	115		ISR = JE(3 , IER)	565
566	116		ELSE	566
567	117		ISR = JE(4 , IER)	567
568	118		END IF	568
569	119		END IF	569
570	120	C		570
571	121	180	CONTINUE	571
572	122	C		572
573	123		IF(ISR . NE . ISI) THEN	573
574	124		ISS = ISR	574
575	125		IE = IER	575
576	126		GO TO 170	576
577	127		END IF	577
578	128	C		578
579	129		END IF	579
580	130	115	CONTINUE	580
581	131	C		581
582	132		ATEMP(1 , 1) = 0.	582
583	133		ATEMP(1 , 2) = 0.	583
584	134		ATEMP(1 , 3) = 0.	584

```

585 135 C
586 136 ATEMP( 2 , 1 ) = 0.
587 137 ATEMP( 2 , 2 ) = 0.
588 138 ATEMP( 2 , 3 ) = 0.
589 139 C
590 140 ATEMP( 3 , 1 ) = 0.
591 141 ATEMP( 3 , 2 ) = 0.
592 142 ATEMP( 3 , 3 ) = 0.
593 143 C
594 144 BTEMP( 1 , 1 ) = 0.
595 145 BTEMP( 1 , 2 ) = 0.
596 146 BTEMP( 1 , 3 ) = 0.
597 147 BTEMP( 1 , 4 ) = 0.
598 148 C
599 149 BTEMP( 2 , 1 ) = 0.
600 150 BTEMP( 2 , 2 ) = 0.
601 151 BTEMP( 2 , 3 ) = 0.
602 152 BTEMP( 2 , 4 ) = 0.
603 153 C
604 154 BTEMP( 3 , 1 ) = 0.
605 155 BTEMP( 3 , 2 ) = 0.
606 156 BTEMP( 3 , 3 ) = 0.
607 157 BTEMP( 3 , 4 ) = 0.
608 158 C
609 159 DO 225 KK = 2 , JJCCLR
610 160 ISS = IICCLR( KK )
611 161 IF( ISS .NE. 0 ) THEN
612 162 XSS = XS( 1 , ISS )
613 163 YSS = XS( 2 , ISS )
614 164 C
615 165 HYDVR = HYDV( ISS , 1 )
616 166 HYDVU = HYDV( ISS , 2 )
617 167 HYDVV = HYDV( ISS , 3 )
618 168 HYDVP = HYDV( ISS , 4 )
619 169 C
620 170 XSS2 = XSS * XSS
621 171 YSS2 = YSS * YSS
622 172 XYSS = XSS * YSS
623 173 C
624 174 ATEMP( 1 , 1 ) = ATEMP( 1 , 1 ) + 1.0
625 175 ATEMP( 1 , 2 ) = ATEMP( 1 , 2 ) + XSS
626 176 ATEMP( 1 , 3 ) = ATEMP( 1 , 3 ) + YSS
627 177 C
628 178 ATEMP( 2 , 1 ) = ATEMP( 2 , 1 ) + XSS
629 179 ATEMP( 2 , 2 ) = ATEMP( 2 , 2 ) + XSS2
630 180 ATEMP( 2 , 3 ) = ATEMP( 2 , 3 ) + XYSS
631 181 C
632 182 ATEMP( 3 , 1 ) = ATEMP( 3 , 1 ) + YSS
633 183 ATEMP( 3 , 2 ) = ATEMP( 3 , 2 ) + XYSS
634 184 ATEMP( 3 , 3 ) = ATEMP( 3 , 3 ) + YSS2
635 185 C
636 186 BTEMP( 1 , 1 ) = BTEMP( 1 , 1 ) + HYDVR
637 187 BTEMP( 1 , 2 ) = BTEMP( 1 , 2 ) + HYDVU
638 188 BTEMP( 1 , 3 ) = BTEMP( 1 , 3 ) + HYDVV
639 189 BTEMP( 1 , 4 ) = BTEMP( 1 , 4 ) + HYDVP
640 190 C
641 191 BTEMP( 2 , 1 ) = BTEMP( 2 , 1 ) + HYDVR * XSS
642 192 BTEMP( 2 , 2 ) = BTEMP( 2 , 2 ) + HYDVU * XSS
643 193 BTEMP( 2 , 3 ) = BTEMP( 2 , 3 ) + HYDVV * XSS
644 194 BTEMP( 2 , 4 ) = BTEMP( 2 , 4 ) + HYDVP * XSS
645 195 C
646 196 BTEMP( 3 , 1 ) = BTEMP( 3 , 1 ) + HYDVR * YSS
647 197 BTEMP( 3 , 2 ) = BTEMP( 3 , 2 ) + HYDVU * YSS
648 198 BTEMP( 3 , 3 ) = BTEMP( 3 , 3 ) + HYDVV * YSS
649 199 BTEMP( 3 , 4 ) = BTEMP( 3 , 4 ) + HYDVP * YSS
650 200 C
651 201 END IF
652 202 C
653 203 225 CONTINUE
654 204 C
655 205 AA( 1 , 1 ) = ATEMP( 1 , 1 )
656 206 AA( 1 , 2 ) = ATEMP( 1 , 2 )
657 207 AA( 1 , 3 ) = ATEMP( 1 , 3 )
658 208 C

```

```

659 209 AA( 2 , 1 ) = ATEMP( 2 , 1 ) 659
660 210 AA( 2 , 2 ) = ATEMP( 2 , 2 ) 660
661 211 AA( 2 , 3 ) = ATEMP( 2 , 3 ) 661
662 212 C 662
663 213 AA( 3 , 1 ) = ATEMP( 3 , 1 ) 663
664 214 AA( 3 , 2 ) = ATEMP( 3 , 2 ) 664
665 215 AA( 3 , 3 ) = ATEMP( 3 , 3 ) 665
666 216 C 666
667 217 BB( 1 , 1 ) = BTEMP( 1 , 1 ) 667
668 218 BB( 1 , 2 ) = BTEMP( 1 , 2 ) 668
669 219 BB( 1 , 3 ) = BTEMP( 1 , 3 ) 669
670 220 BB( 1 , 4 ) = BTEMP( 1 , 4 ) 670
671 221 C 671
672 222 BB( 2 , 1 ) = BTEMP( 2 , 1 ) 672
673 223 BB( 2 , 2 ) = BTEMP( 2 , 2 ) 673
674 224 BB( 2 , 3 ) = BTEMP( 2 , 3 ) 674
675 225 BB( 2 , 4 ) = BTEMP( 2 , 4 ) 675
676 226 C 676
677 227 BB( 3 , 1 ) = BTEMP( 3 , 1 ) 677
678 228 BB( 3 , 2 ) = BTEMP( 3 , 2 ) 678
679 229 BB( 3 , 3 ) = BTEMP( 3 , 3 ) 679
680 230 BB( 3 , 4 ) = BTEMP( 3 , 4 ) 680
681 231 C 681
682 232 DETERM = AA( 1 , 1 ) * ( AA( 2 , 2 ) * AA( 3 , 3 ) -
683 233 . AA( 3 , 2 ) * AA( 2 , 3 ) ) +
684 234 . AA( 2 , 1 ) * ( AA( 1 , 3 ) * AA( 3 , 2 ) -
685 235 . AA( 3 , 3 ) * AA( 1 , 2 ) ) +
686 236 . AA( 3 , 1 ) * ( AA( 1 , 2 ) * AA( 2 , 3 ) -
687 237 . AA( 2 , 2 ) * AA( 1 , 3 ) )
688 238 C 688
689 239 DTRMIN = 1. / DETERM 689
690 240 C 690
691 241 AAA1 = AA( 2 , 3 ) * AA( 3 , 1 ) - AA( 2 , 1 ) * AA( 3 , 3 ) 691
692 242 AAA2 = AA( 3 , 3 ) * AA( 1 , 1 ) - AA( 3 , 1 ) * AA( 1 , 3 ) 692
693 243 AAA3 = AA( 1 , 3 ) * AA( 2 , 1 ) - AA( 1 , 1 ) * AA( 2 , 3 ) 693
694 244 C 694
695 245 AAA4 = AA( 2 , 1 ) * AA( 3 , 2 ) - AA( 3 , 1 ) * AA( 2 , 2 ) 695
696 246 AAA5 = AA( 3 , 1 ) * AA( 1 , 2 ) - AA( 1 , 1 ) * AA( 3 , 2 ) 696
697 247 AAA6 = AA( 1 , 1 ) * AA( 2 , 2 ) - AA( 2 , 1 ) * AA( 1 , 2 ) 697
698 248 C 698
699 249 RGRAD( IS , 1 ) = DTRMIN * ( BB( 1 , 1 ) * AAA1 +
700 250 . BB( 2 , 1 ) * AAA2 +
701 251 . BB( 3 , 1 ) * AAA3 ) 701
702 252 C 702
703 253 RGRAD( IS , 2 ) = DTRMIN * ( BB( 1 , 1 ) * AAA4 +
704 254 . BB( 2 , 1 ) * AAA5 +
705 255 . BB( 3 , 1 ) * AAA6 ) 705
706 256 C 706
707 257 UGRAD( IS , 1 ) = DTRMIN * ( BB( 1 , 2 ) * AAA1 +
708 258 . BB( 2 , 2 ) * AAA2 +
709 259 . BB( 3 , 2 ) * AAA3 ) 709
710 260 C 710
711 261 UGRAD( IS , 2 ) = DTRMIN * ( BB( 1 , 2 ) * AAA4 +
712 262 . BB( 2 , 2 ) * AAA5 +
713 263 . BB( 3 , 2 ) * AAA6 ) 713
714 264 C 714
715 265 VGRAD( IS , 1 ) = DTRMIN * ( BB( 1 , 3 ) * AAA1 +
716 266 . BB( 2 , 3 ) * AAA2 +
717 267 . BB( 3 , 3 ) * AAA3 ) 717
718 268 C 718
719 269 VGRAD( IS , 2 ) = DTRMIN * ( BB( 1 , 3 ) * AAA4 +
720 270 . BB( 2 , 3 ) * AAA5 +
721 271 . BB( 3 , 3 ) * AAA6 ) 721
722 272 C 722
723 273 PGRAD( IS , 1 ) = DTRMIN * ( BB( 1 , 4 ) * AAA1 +
724 274 . BB( 2 , 4 ) * AAA2 +
725 275 . BB( 3 , 4 ) * AAA3 ) 725
726 276 C 726
727 277 PGRAD( IS , 2 ) = DTRMIN * ( BB( 1 , 4 ) * AAA4 +
728 278 . BB( 2 , 4 ) * AAA5 +
729 279 . BB( 3 , 4 ) * AAA6 ) 729
730 280 C 730
731 281 105 CONTINUE 731
732 282 C 732

```

```

733 283      NS1 = NS2 + 1                                733
734 284      NS2 = NS2 + NOFVES( INS + 1 )              734
735 285      90 CONTINUE                                  735
736 286      C                                          736
737 287      C-----I-----                            737
738 288      C                                          738
739 289      C --- CALL THE MONOTONICITY LIMITER ----- 739
740 290      C                                          740
741 291      CALL MONOTN                                  741
742 292      C                                          742
743 293      C-----I-----                            743
744 294      C                                          744
745 295      C                                          745
746 296      C --- EXIT POINT FROM SUBROUTINE ----- 746
747 297      C                                          747
748 298      C -----I-----                            748
749 299      RETURN                                       749
750 300      C -----I-----                            750
751 301      C                                          751
752 302      C ---I-----                               752
753 303      END                                          753

```

```

754 1      SUBROUTINE GRADNT                                754
755 2      C                                          755
756 3      C-----I-----                            756
757 4      C                                          757
758 5      C      GRADNT COMPUTE THE GRADIENT FOR SECOND ORDER CALCULATION I 758
759 6      C      USE THE INFORMATION IN THE THREE NEIGHBOURING I 759
760 7      C      TRIANGLES THAT HAVE COMMON EDGES TO COMPUTE I 760
761 8      C      GRADIENT APPLYING LEAST SQUARE TECHNIQUE I 761
762 9      C                                          I 762
763 10     C-----I-----                            763
764 11     C                                          764
765 12     include 'cmsh00.h'                               765
766 13     include 'chyd00.h'                               766
767 14     include 'cint00.h'                               767
768 15     include 'cphs10.h'                               768
769 16     include 'cphs20.h'                               769
770 17     C                                          770
771 18     C-----I-----                            771
772 19     C                                          772
773 20     REAL RRMIDL(MBP),PPMIDL(MBP),UUMIDL(MBP),VVMIDL(MBP) 773
774 21     REAL RIGRAD(MBP),PIGRAD(MBP),UIGRAD(MBP),VIGRAD(MBP) 774
775 22     REAL RJGRAD(MBP),PJGRAD(MBP),UJGRAD(MBP),VJGRAD(MBP) 775
776 23     REAL RMAX(MBP),PMAX(MBP),UMAX(MBP),VMAX(MBP)      776
777 24     REAL RMIN(MBP),PMIN(MBP),UMIN(MBP),VMIN(MBP)      777
778 25     REAL RLEFTT(MBP),ULEFTT(MBP),VLEFTT(MBP),PLEFTT(MBP) 778
779 26     REAL RRIGHT(MBP),URIGHT(MBP),VRIGHT(MBP),PRIGHT(MBP) 779
780 27     REAL ROR(3),UOR(3),VOR(3),POR(3)                  780
781 28     REAL ROL(3),UOL(3),VOL(3),POL(3)                  781
782 29     REAL AA(3,3),BB(3,4),B(3),INDX(3),ATEMP(3,3,3),BTEMP(3,4,3) 782
783 30     REAL AAO(3,3),BBO(3,4)                             783
784 31     C                                          784
785 32     C-----I-----                            785
786 33     C                                          786
787 34     C --- BEGIN LOOP OVER ALL CELLS IN THE DOMAIN ----- 787
788 35     C                                          788
789 36     NS1 = 1                                           789
790 37     NS2 = NOFVES( 1 )                                  790
791 38     DO 90 INS = 1 , NVEES                              791
792 39     C                                          792
793 40     C --- FETCH HYDRO QUANTITIES -----I----- 793
794 41     C                                          794
795 42     DO 105 IS = NS1 , NS2                              795
796 43     C                                          796
797 44     XSM = XS( 1 , IS )                                  797
798 45     YSM = XS( 2 , IS )                                  798
799 46     XSM2 = XSM * XSM                                    799
800 47     YSM2 = YSM * YSM                                    800
801 48     XYSM = XSM * YSM                                    801
802 49     C                                          802
803 50     AAO( 1 , 1 ) = 1.0                                  803

```

804	51	AAO(1 , 2) = XSM	804
805	52	AAO(1 , 3) = YSM	805
806	53	C	806
807	54	AAO(2 , 1) = XSM	807
808	55	AAO(2 , 2) = XSM2	808
809	56	AAO(2 , 3) = XYSM	809
810	57	C	810
811	58	AAO(3 , 1) = YSM	811
812	59	AAO(3 , 2) = XYSM	812
813	60	AAO(3 , 3) = YSM2	813
814	61	C	814
815	62	BB1 = HYDV(IS , 1)	815
816	63	BB2 = HYDV(IS , 2)	816
817	64	BB3 = HYDV(IS , 3)	817
818	65	BB4 = HYDV(IS , 4)	818
819	66	C	819
820	67	BB1X = BB1 * XSM	820
821	68	BB2X = BB2 * XSM	821
822	69	BB3X = BB3 * XSM	822
823	70	BB4X = BB4 * XSM	823
824	71	C	824
825	72	BB1Y = BB1 * YSM	825
826	73	BB2Y = BB2 * YSM	826
827	74	BB3Y = BB3 * YSM	827
828	75	BB4Y = BB4 * YSM	828
829	76	C	829
830	77	BB0(1 , 1) = BB1	830
831	78	BB0(1 , 2) = BB2	831
832	79	BB0(1 , 3) = BB3	832
833	80	BB0(1 , 4) = BB4	833
834	81	C	834
835	82	BB0(2 , 1) = BB1X	835
836	83	BB0(2 , 2) = BB2X	836
837	84	BB0(2 , 3) = BB3X	837
838	85	BB0(2 , 4) = BB4X	838
839	86	C	839
840	87	BB0(3 , 1) = BB1Y	840
841	88	BB0(3 , 2) = BB2Y	841
842	89	BB0(3 , 3) = BB3Y	842
843	90	BB0(3 , 4) = BB4Y	843
844	91	C	844
845	92	DO 115 IK = 1 , 3	845
846	93	IE = JS(IK + 3 , IS)	846
847	94	IF(IE . GT . 0) THEN	847
848	95	ISS = JE(4 , IE)	848
849	96	ELSE	849
850	97	ISS = JE(3 , - IE)	850
851	98	END IF	851
852	99	C	852
853	100	IF(ISS . NE . 0) THEN	853
854	101	XSS = XS(1 , ISS)	854
855	102	YSS = XS(2 , ISS)	855
856	103	C	856
857	104	HYDVR = HYDV(ISS , 1)	857
858	105	HYDVU = HYDV(ISS , 2)	858
859	106	HYDVV = HYDV(ISS , 3)	859
860	107	HYDVP = HYDV(ISS , 4)	860
861	108	C	861
862	109	ELSE	862
863	110	C	863
864	111	IE = IABS(IE)	864
865	112	HYDVR = BB1	865
866	113	HYDVU = BB2	866
867	114	HYDVV = BB3	867
868	115	HYDVP = BB4	868
869	116	C	869
870	117	XSS = 2. * XMIDL(IE) - XSM	870
871	118	YSS = 2. * YMIDL(IE) - YSM	871
872	119	C	872
873	120	IJES = JE(5 , IE)	873
874	121	IF(IJES . EQ . 6 . OR . IJES . EQ . 5) THEN	874
875	122	UUVV = - (BB2 * XN(IE) + BB3 * YN(IE))	875
876	123	VVUU = - BB2 * YN(IE) + BB3 * XN(IE)	876
877	124	C	877
		HYDVU = UUVV * XN(IE) - VVUU * YN(IE)	

878	125	C	HYDVV = UVV * YN(IE) + VVUU * XN(IE)	878
879	126	C		879
880	127	C	ELSEIF(IJE5 . EQ . 8) THEN	880
881	128	C	HYDVR = RIN	881
882	129	C	HYDVU = UIN	882
883	130	C	HYDVV = VIN	883
884	131	C	HYDVP = PIN	884
885	132	C		885
886	133	C	END IF	886
887	134	C	END IF	887
888	135	C		888
889	136	C	XSS2 = XSS * XSS	889
890	137	C	YSS2 = YSS * YSS	890
891	138	C	XYSS = XSS * YSS	891
892	139	C		892
893	140	C	ATEMP(1 , 1 , IK) = 1.0	893
894	141	C	ATEMP(1 , 2 , IK) = XSS	894
895	142	C	ATEMP(1 , 3 , IK) = YSS	895
896	143	C		896
897	144	C	ATEMP(2 , 1 , IK) = XSS	897
898	145	C	ATEMP(2 , 2 , IK) = XSS2	898
899	146	C	ATEMP(2 , 3 , IK) = XYSS	899
900	147	C		900
901	148	C	ATEMP(3 , 1 , IK) = YSS	901
902	149	C	ATEMP(3 , 2 , IK) = XYSS	902
903	150	C	ATEMP(3 , 3 , IK) = YSS2	903
904	151	C		904
905	152	C	BTEMP(1 , 1 , IK) = HYDVR	905
906	153	C	BTEMP(1 , 2 , IK) = HYDVU	906
907	154	C	BTEMP(1 , 3 , IK) = HYDVV	907
908	155	C	BTEMP(1 , 4 , IK) = HYDVP	908
909	156	C		909
910	157	C	BTEMP(2 , 1 , IK) = HYDVR * XSS	910
911	158	C	BTEMP(2 , 2 , IK) = HYDVU * XSS	911
912	159	C	BTEMP(2 , 3 , IK) = HYDVV * XSS	912
913	160	C	BTEMP(2 , 4 , IK) = HYDVP * XSS	913
914	161	C		914
915	162	C	BTEMP(3 , 1 , IK) = HYDVR * YSS	915
916	163	C	BTEMP(3 , 2 , IK) = HYDVU * YSS	916
917	164	C	BTEMP(3 , 3 , IK) = HYDVV * YSS	917
918	165	C	BTEMP(3 , 4 , IK) = HYDVP * YSS	918
919	166	C		919
920	167	C	115 CONTINUE	920
921	168	C		921
922	169	C	AA(1 , 1) = AAO(1 , 1) * 3. + ATEMP(1 , 1 , 1) +	922
923	170	C	ATEMP(1 , 1 , 2) + ATEMP(1 , 1 , 3)	923
924	171	C	AA(1 , 2) = AAO(1 , 2) * 3. + ATEMP(1 , 2 , 1) +	924
925	172	C	ATEMP(1 , 2 , 2) + ATEMP(1 , 2 , 3)	925
926	173	C	AA(1 , 3) = AAO(1 , 3) * 3. + ATEMP(1 , 3 , 1) +	926
927	174	C	ATEMP(1 , 3 , 2) + ATEMP(1 , 3 , 3)	927
928	175	C		928
929	176	C	AA(2 , 1) = AAO(2 , 1) * 3. + ATEMP(2 , 1 , 1) +	929
930	177	C	ATEMP(2 , 1 , 2) + ATEMP(2 , 1 , 3)	930
931	178	C	AA(2 , 2) = AAO(2 , 2) * 3. + ATEMP(2 , 2 , 1) +	931
932	179	C	ATEMP(2 , 2 , 2) + ATEMP(2 , 2 , 3)	932
933	180	C	AA(2 , 3) = AAO(2 , 3) * 3. + ATEMP(2 , 3 , 1) +	933
934	181	C	ATEMP(2 , 3 , 2) + ATEMP(2 , 3 , 3)	934
935	182	C		935
936	183	C	AA(3 , 1) = AAO(3 , 1) * 3. + ATEMP(3 , 1 , 1) +	936
937	184	C	ATEMP(3 , 1 , 2) + ATEMP(3 , 1 , 3)	937
938	185	C	AA(3 , 2) = AAO(3 , 2) * 3. + ATEMP(3 , 2 , 1) +	938
939	186	C	ATEMP(3 , 2 , 2) + ATEMP(3 , 2 , 3)	939
940	187	C	AA(3 , 3) = AAO(3 , 3) * 3. + ATEMP(3 , 3 , 1) +	940
941	188	C	ATEMP(3 , 3 , 2) + ATEMP(3 , 3 , 3)	941
942	189	C		942
943	190	C	BB(1 , 1) = BBO(1 , 1) * 3. + BTEMP(1 , 1 , 1) +	943
944	191	C	BTEMP(1 , 1 , 2) + BTEMP(1 , 1 , 3)	944
945	192	C	BB(1 , 2) = BBO(1 , 2) * 3. + BTEMP(1 , 2 , 1) +	945
946	193	C	BTEMP(1 , 2 , 2) + BTEMP(1 , 2 , 3)	946
947	194	C	BB(1 , 3) = BBO(1 , 3) * 3. + BTEMP(1 , 3 , 1) +	947
948	195	C	BTEMP(1 , 3 , 2) + BTEMP(1 , 3 , 3)	948
949	196	C	BB(1 , 4) = BBO(1 , 4) * 3. + BTEMP(1 , 4 , 1) +	949
950	197	C	BTEMP(1 , 4 , 2) + BTEMP(1 , 4 , 3)	950
951	198	C		951


```

952 199      BB( 2 , 1 ) = BBO( 2 , 1 ) * 3. + BTEMP( 2 , 1 , 1 ) +
953 200      .           BTEMP( 2 , 1 , 2 ) + BTEMP( 2 , 1 , 3 )
954 201      BB( 2 , 2 ) = BBO( 2 , 2 ) * 3. + BTEMP( 2 , 2 , 1 ) +
955 202      .           BTEMP( 2 , 2 , 2 ) + BTEMP( 2 , 2 , 3 )
956 203      BB( 2 , 3 ) = BBO( 2 , 3 ) * 3. + BTEMP( 2 , 3 , 1 ) +
957 204      .           BTEMP( 2 , 3 , 2 ) + BTEMP( 2 , 3 , 3 )
958 205      BB( 2 , 4 ) = BBO( 2 , 4 ) * 3. + BTEMP( 2 , 4 , 1 ) +
959 206      .           BTEMP( 2 , 4 , 2 ) + BTEMP( 2 , 4 , 3 )
960 207      C
961 208      BB( 3 , 1 ) = BBO( 3 , 1 ) * 3. + BTEMP( 3 , 1 , 1 ) +
962 209      .           BTEMP( 3 , 1 , 2 ) + BTEMP( 3 , 1 , 3 )
963 210      BB( 3 , 2 ) = BBO( 3 , 2 ) * 3. + BTEMP( 3 , 2 , 1 ) +
964 211      .           BTEMP( 3 , 2 , 2 ) + BTEMP( 3 , 2 , 3 )
965 212      BB( 3 , 3 ) = BBO( 3 , 3 ) * 3. + BTEMP( 3 , 3 , 1 ) +
966 213      .           BTEMP( 3 , 3 , 2 ) + BTEMP( 3 , 3 , 3 )
967 214      BB( 3 , 4 ) = BBO( 3 , 4 ) * 3. + BTEMP( 3 , 4 , 1 ) +
968 215      .           BTEMP( 3 , 4 , 2 ) + BTEMP( 3 , 4 , 3 )
969 216      C
970 217      DETERM = AA( 1 , 1 ) * ( AA( 2 , 2 ) * AA( 3 , 3 ) -
971 218      .           AA( 3 , 2 ) * AA( 2 , 3 ) ) +
972 219      .           AA( 2 , 1 ) * ( AA( 1 , 3 ) * AA( 3 , 2 ) -
973 220      .           AA( 3 , 3 ) * AA( 1 , 2 ) ) +
974 221      .           AA( 3 , 1 ) * ( AA( 1 , 2 ) * AA( 2 , 3 ) -
975 222      .           AA( 2 , 2 ) * AA( 1 , 3 ) )
976 223      C
977 224      DTRMIN = 1. / DETERM
978 225      C
979 226      AAA1 = AA( 2 , 3 ) * AA( 3 , 1 ) - AA( 2 , 1 ) * AA( 3 , 3 )
980 227      AAA2 = AA( 3 , 3 ) * AA( 1 , 1 ) - AA( 3 , 1 ) * AA( 1 , 3 )
981 228      AAA3 = AA( 1 , 3 ) * AA( 2 , 1 ) - AA( 1 , 1 ) * AA( 2 , 3 )
982 229      C
983 230      AAA4 = AA( 2 , 1 ) * AA( 3 , 2 ) - AA( 3 , 1 ) * AA( 2 , 2 )
984 231      AAA5 = AA( 3 , 1 ) * AA( 1 , 2 ) - AA( 1 , 1 ) * AA( 3 , 2 )
985 232      AAA6 = AA( 1 , 1 ) * AA( 2 , 2 ) - AA( 2 , 1 ) * AA( 1 , 2 )
986 233      C
987 234      RGRAD( IS , 1 ) = DTRMIN * ( BB( 1 , 1 ) * AAA1 +
988 235      .           BB( 2 , 1 ) * AAA2 +
989 236      .           BB( 3 , 1 ) * AAA3 )
990 237      C
991 238      RGRAD( IS , 2 ) = DTRMIN * ( BB( 1 , 1 ) * AAA4 +
992 239      .           BB( 2 , 1 ) * AAA5 +
993 240      .           BB( 3 , 1 ) * AAA6 )
994 241      C
995 242      UGRAD( IS , 1 ) = DTRMIN * ( BB( 1 , 2 ) * AAA1 +
996 243      .           BB( 2 , 2 ) * AAA2 +
997 244      .           BB( 3 , 2 ) * AAA3 )
998 245      C
999 246      UGRAD( IS , 2 ) = DTRMIN * ( BB( 1 , 2 ) * AAA4 +
1000 247      .           BB( 2 , 2 ) * AAA5 +
1001 248      .           BB( 3 , 2 ) * AAA6 )
1002 249      C
1003 250      VGRAD( IS , 1 ) = DTRMIN * ( BB( 1 , 3 ) * AAA1 +
1004 251      .           BB( 2 , 3 ) * AAA2 +
1005 252      .           BB( 3 , 3 ) * AAA3 )
1006 253      C
1007 254      VGRAD( IS , 2 ) = DTRMIN * ( BB( 1 , 3 ) * AAA4 +
1008 255      .           BB( 2 , 3 ) * AAA5 +
1009 256      .           BB( 3 , 3 ) * AAA6 )
1010 257      C
1011 258      PGRAD( IS , 1 ) = DTRMIN * ( BB( 1 , 4 ) * AAA1 +
1012 259      .           BB( 2 , 4 ) * AAA2 +
1013 260      .           BB( 3 , 4 ) * AAA3 )
1014 261      C
1015 262      PGRAD( IS , 2 ) = DTRMIN * ( BB( 1 , 4 ) * AAA4 +
1016 263      .           BB( 2 , 4 ) * AAA5 +
1017 264      .           BB( 3 , 4 ) * AAA6 )
1018 265      C
1019 266      105 CONTINUE
1020 267      C
1021 268      NS1 = NS2 + 1
1022 269      NS2 = NS2 + NOFVES( INS + 1 )
1023 270      90 CONTINUE
1024 271      C
1025 272      C-----

```

```

1026 273 C
1027 274 C --- CALL THE MONOTONICITY LIMITER -----
1028 275 C
1029 276 CALL MONOTN
1030 277 C
1031 278 C -----
1032 279 C
1033 280 C
1034 281 C --- EXIT POINT FROM SUBROUTINE -----
1035 282 C
1036 283 C -----
1037 284 RETURN
1038 285 C -----
1039 286 C
1040 287 C ---
1041 288 END

```

```

1042 1 SUBROUTINE MONOTN
1043 2 C
1044 3 C -----I
1045 4 C I
1046 5 C MONOTN LIMIT THE GRADIENTS SO THAT NO NEW EXTREMUM ARE I
1047 6 C CREATED ARTIFICIALLY DURING THE PROJECTION PROCESS I
1048 7 C I
1049 8 C -----I
1050 9 C
1051 10 include 'cmsh00.h'
1052 11 include 'chyd00.h'
1053 12 include 'cint00.h'
1054 13 include 'cphs10.h'
1055 14 include 'cphs20.h'
1056 15 C
1057 16 C -----
1058 17 C
1059 18 REAL RRMIDL(MBP),PPMIDL(MBP),UUMIDL(MBP),VVMIDL(MBP)
1060 19 REAL RIGRAD(MBP),PIGRAD(MBP),UIGRAD(MBP),VIGRAD(MBP)
1061 20 REAL RJGRAD(MBP),PJGRAD(MBP),UJGRAD(MBP),VJGRAD(MBP)
1062 21 REAL RMAX(MBP),PMAX(MBP),UMAX(MBP),VMAX(MBP)
1063 22 REAL RMIN(MBP),PMIN(MBP),UMIN(MBP),VMIN(MBP)
1064 23 REAL RLEFTT(MBP),ULEFTT(MBP),VLEFTT(MBP),PLEFTT(MBP)
1065 24 REAL RRIGHT(MBP),URIGHT(MBP),VRIGHT(MBP),PRIGHT(MBP)
1066 25 REAL ROR(3),UOR(3),VOR(3),POR(3)
1067 26 REAL ROL(3),UOL(3),VOL(3),POL(3)
1068 27 REAL AA(3,3),BB(3,4),B(3),INDX(3),ATEMP(3,3,3),BTEMP(3,4,3)
1069 28 REAL AAO(3,3),BBO(3,4)
1070 29 C
1071 30 C -----
1072 31 C
1073 32 C --- LIMITER FOR GRADIENTS BEGINS -----
1074 33 C USED TO PREVENT NEW MINIMA AND MAXIMA
1075 34 C AT PROJECTED INTERFACE VALUES.
1076 35 C
1077 36 NS1 = 1
1078 37 NS2 = NOFVES( 1 )
1079 38 DO 80 IS = 1 , NVEES
1080 39 C
1081 40 DO 150 IS = NS1 , NS2
1082 41 KS = IS - NS1 + 1
1083 42 C
1084 43 C --- FIRST TRIANGLE EDGE -----
1085 44 C
1086 45 IE = IABS( JS( 4 , IS ) )
1087 46 C
1088 47 ISL = JE( 3 , IE )
1089 48 ISR = JE( 4 , IE )
1090 49 C
1091 50 RROL = HYDV( ISL , 1 )
1092 51 UUOL = HYDV( ISL , 2 )
1093 52 VVOL = HYDV( ISL , 3 )
1094 53 PPOL = HYDV( ISL , 4 )
1095 54 C
1096 55 IJE5 = JE( 5 , IE )

```

```

1097 56 IF( IJES . EQ . 0 ) THEN 1097
1098 57 C 1098
1099 58 RROR = HYDV( ISR , 1 ) 1099
1100 59 UUOR = HYDV( ISR , 2 ) 1100
1101 60 VVOR = HYDV( ISR , 3 ) 1101
1102 61 PPOR = HYDV( ISR , 4 ) 1102
1103 62 C 1103
1104 63 ELSE 1104
1105 64 C 1105
1106 65 RROR = RROL 1106
1107 66 UUOR = UUOL 1107
1108 67 VVOR = VVOL 1108
1109 68 PPOR = PPOL 1109
1110 69 C 1110
1111 70 C IF( IJES . EQ . 6 . OR . IJES . EQ . 5 ) THEN 1111
1112 71 C UVVV = - ( UUOL * XN( IE ) + VVOL * YN( IE ) ) 1112
1113 72 C VVUU = - UUOL * YN( IE ) + VVOL * XN( IE ) 1113
1114 73 C UUOR = UVVV * XN( IE ) - VVUU * YN( IE ) 1114
1115 74 C VVOR = UVVV * YN( IE ) + VVUU * XN( IE ) 1115
1116 75 C 1116
1117 76 C ELSE IF( IJES . EQ . 8 ) THEN 1117
1118 77 C RROR = RIN 1118
1119 78 C UUOR = UIN 1119
1120 79 C VVOR = VIN 1120
1121 80 C PPOR = PIN 1121
1122 81 C END IF 1122
1123 82 C 1123
1124 83 END IF 1124
1125 84 C 1125
1126 85 ROL( 1 ) = RROL 1126
1127 86 UOL( 1 ) = UUOL 1127
1128 87 VOL( 1 ) = VVOL 1128
1129 88 POL( 1 ) = PPOL 1129
1130 89 C 1130
1131 90 ROR( 1 ) = RROR 1131
1132 91 UOR( 1 ) = UUOR 1132
1133 92 VOR( 1 ) = VVOR 1133
1134 93 POR( 1 ) = PPOR 1134
1135 94 C 1135
1136 95 C --- SECOND TRIANGLE EDGE ----- 1136
1137 96 C 1137
1138 97 IE = IABS( JS( 5 , IS ) ) 1138
1139 98 C 1139
1140 99 ISL = JE( 3 , IE ) 1140
1141 100 ISR = JE( 4 , IE ) 1141
1142 101 C 1142
1143 102 RROL = HYDV( ISL , 1 ) 1143
1144 103 UUOL = HYDV( ISL , 2 ) 1144
1145 104 VVOL = HYDV( ISL , 3 ) 1145
1146 105 PPOL = HYDV( ISL , 4 ) 1146
1147 106 C 1147
1148 107 IJES = JE( 5 , IE ) 1148
1149 108 IF( IJES . EQ . 0 ) THEN 1149
1150 109 C 1150
1151 110 RROR = HYDV( ISR , 1 ) 1151
1152 111 UUOR = HYDV( ISR , 2 ) 1152
1153 112 VVOR = HYDV( ISR , 3 ) 1153
1154 113 PPOR = HYDV( ISR , 4 ) 1154
1155 114 C 1155
1156 115 ELSE 1156
1157 116 C 1157
1158 117 RROR = RROL 1158
1159 118 UUOR = UUOL 1159
1160 119 VVOR = VVOL 1160
1161 120 PPOR = PPOL 1161
1162 121 C 1162
1163 122 C IF( IJES . EQ . 6 . OR . IJES . EQ . 5 ) THEN 1163
1164 123 C UVVV = - ( UUOL * XN( IE ) + VVOL * YN( IE ) ) 1164
1165 124 C VVUU = - UUOL * YN( IE ) + VVOL * XN( IE ) 1165
1166 125 C UUOR = UVVV * XN( IE ) - VVUU * YN( IE ) 1166
1167 126 C VVOR = UVVV * YN( IE ) + VVUU * XN( IE ) 1167
1168 127 C 1168
1169 128 C ELSE IF( IJES . EQ . 8 ) THEN 1169
1170 129 C RROR = RIN 1170

```

```

1171 130 C      UUOR = UIN
1172 131 C      VVOR = VIN
1173 132 C      PPOR = PIN
1174 133 C      END IF
1175 134 C
1176 135 C      END IF
1177 136 C
1178 137 C      ROL( 2 ) = RROL
1179 138 C      UOL( 2 ) = UUOL
1180 139 C      VOL( 2 ) = VVOL
1181 140 C      POL( 2 ) = PPOL
1182 141 C
1183 142 C      ROR( 2 ) = RROR
1184 143 C      UOR( 2 ) = UUOR
1185 144 C      VOR( 2 ) = VVOR
1186 145 C      POR( 2 ) = PPOR
1187 146 C
1188 147 C --- THIRD TRIANGLE EDGE -----
1189 148 C
1190 149 C      IE = IABS( JS( 6 , IS ) )
1191 150 C
1192 151 C      ISL = JE( 3 , IE )
1193 152 C      ISR = JE( 4 , IE )
1194 153 C      RROL = HYDV( ISL , 1 )
1195 154 C      UUOL = HYDV( ISL , 2 )
1196 155 C      VVOL = HYDV( ISL , 3 )
1197 156 C      PPOL = HYDV( ISL , 4 )
1198 157 C
1199 158 C      IJE5 = JE( 5 , IE )
1200 159 C      IF( IJE5 .EQ. 0 ) THEN
1201 160 C
1202 161 C      RROR = HYDV( ISR , 1 )
1203 162 C      UUOR = HYDV( ISR , 2 )
1204 163 C      VVOR = HYDV( ISR , 3 )
1205 164 C      PPOR = HYDV( ISR , 4 )
1206 165 C
1207 166 C      ELSE
1208 167 C
1209 168 C      RROR = RROL
1210 169 C      UUOR = UUOL
1211 170 C      VVOR = VVOL
1212 171 C      PPOR = PPOL
1213 172 C
1214 173 C      IF( IJE5 .EQ. 6 .OR. IJE5 .EQ. 5 ) THEN
1215 174 C      UUVV = - ( UUOL * XN( IE ) + VVOL * YN( IE ) )
1216 175 C      VVUU = - UUOL * YN( IE ) + VVOL * XN( IE )
1217 176 C      UUOR = UUVV * XN( IE ) - VVUU * YN( IE )
1218 177 C      VVOR = UUVV * YN( IE ) + VVUU * XN( IE )
1219 178 C
1220 179 C      ELSE IF( IJE5 .EQ. 8 ) THEN
1221 180 C      RROR = RIN
1222 181 C      UUOR = UIN
1223 182 C      VVOR = VIN
1224 183 C      PPOR = PIN
1225 184 C      END IF
1226 185 C
1227 186 C      END IF
1228 187 C
1229 188 C      ROL( 3 ) = RROL
1230 189 C      UOL( 3 ) = UUOL
1231 190 C      VOL( 3 ) = VVOL
1232 191 C      POL( 3 ) = PPOL
1233 192 C
1234 193 C      ROR( 3 ) = RROR
1235 194 C      UOR( 3 ) = UUOR
1236 195 C      VOR( 3 ) = VVOR
1237 196 C      POR( 3 ) = PPOR
1238 197 C
1239 198 C --- FIND MAXIMA IN THE NEIGHBORHOOD OF A TRIANGLE -----
1240 199 C
1241 200 C      RMAX( KS ) = AMAX1( ROL( 1 ) , ROL( 2 ) , ROL( 3 ) ,
1242 201 C      ROR( 1 ) , ROR( 2 ) , ROR( 3 ) )
1243 202 C      UMAX( KS ) = AMAX1( UOL( 1 ) , UOL( 2 ) , UOL( 3 ) ,
1244 203 C      UOR( 1 ) , UOR( 2 ) , UOR( 3 ) )

```

```

1245 204      VMAX( KS ) = AMAX1( VOL( 1 ) , VOL( 2 ) , VOL( 3 ) ,
1246 205      .      VOR( 1 ) , VOR( 2 ) , VOR( 3 ) )
1247 206      PMAX( KS ) = AMAX1( POL( 1 ) , POL( 2 ) , POL( 3 ) ,
1248 207      .      POR( 1 ) , POR( 2 ) , POR( 3 ) )
1249 208      C
1250 209      C --- FIND MINIMA IN THE NEIGHBORHOOD OF A TRIANGLE -----
1251 210      C
1252 211      RMIN( KS ) = AMIN1( ROL( 1 ) , ROL( 2 ) , ROL( 3 ) ,
1253 212      .      ROR( 1 ) , ROR( 2 ) , ROR( 3 ) )
1254 213      UMIN( KS ) = AMIN1( UOL( 1 ) , UOL( 2 ) , UOL( 3 ) ,
1255 214      .      UOR( 1 ) , UOR( 2 ) , UOR( 3 ) )
1256 215      VMIN( KS ) = AMIN1( VOL( 1 ) , VOL( 2 ) , VOL( 3 ) ,
1257 216      .      VOR( 1 ) , VOR( 2 ) , VOR( 3 ) )
1258 217      PMIN( KS ) = AMIN1( POL( 1 ) , POL( 2 ) , POL( 3 ) ,
1259 218      .      POR( 1 ) , POR( 2 ) , POR( 3 ) )
1260 219      C
1261 220      150 CONTINUE
1262 221      C
1263 222      C --- FIND DIFFERENCES BETWEEN EXTREMA AND THE TRIANGLE CENTERED -----
1264 223      C      QUANTITIES
1265 224      C
1266 225      DO 180 IS = NS1 , NS2
1267 226      KS = IS - NS1 + 1
1268 227      C
1269 228      RRR( KS ) = RMAX( KS ) - HYDV( IS , 1 )
1270 229      RRL( KS ) = RMIN( KS ) - HYDV( IS , 1 )
1271 230      UUR( KS ) = UMAX( KS ) - HYDV( IS , 2 )
1272 231      UUL( KS ) = UMIN( KS ) - HYDV( IS , 2 )
1273 232      VVR( KS ) = VMAX( KS ) - HYDV( IS , 3 )
1274 233      VVL( KS ) = VMIN( KS ) - HYDV( IS , 3 )
1275 234      PPR( KS ) = PMAX( KS ) - HYDV( IS , 4 )
1276 235      PPL( KS ) = PMIN( KS ) - HYDV( IS , 4 )
1277 236      C
1278 237      180 CONTINUE
1279 238      C
1280 239      C --- FIND THE PROJECTED INCRMENTS FOR INTERFACE BASED QUANTITIES -----
1281 240      C
1282 241      DO 170 IS = NS1 , NS2
1283 242      KS = IS - NS1 + 1
1284 243      C
1285 244      C --- FIRST TRIANGLE EDGE -----
1286 245      C
1287 246      IE = IABS( JS( 4 , IS ) )
1288 247      C
1289 248      ISL = JE( 3 , IE )
1290 249      ISR = JE( 4 , IE )
1291 250      C
1292 251      XML = XMIDL( IE ) - XS( 1 , ISL )
1293 252      YML = YMIDL( IE ) - XS( 2 , ISL )
1294 253      C
1295 254      RROR = 1.E-12 +
1296 255      .      RGRAD( ISL , 1 ) * XML + RGRAD( ISL , 2 ) * YML
1297 256      UUOL = 1.E-12 +
1298 257      .      UGRAD( ISL , 1 ) * XML + UGRAD( ISL , 2 ) * YML
1299 258      VVOL = 1.E-12 +
1300 259      .      VGRAD( ISL , 1 ) * XML + VGRAD( ISL , 2 ) * YML
1301 260      PPOL = 1.E-12 +
1302 261      .      PGRAD( ISL , 1 ) * XML + PGRAD( ISL , 2 ) * YML
1303 262      C
1304 263      IJE5 = JE( 5 , IE )
1305 264      IF( IJE5 .EQ. 0 ) THEN
1306 265      C
1307 266      XMR = XMIDL( IE ) - XS( 1 , ISR )
1308 267      YMR = YMIDL( IE ) - XS( 2 , ISR )
1309 268      C
1310 269      RROR = 1.E-12 +
1311 270      .      RGRAD( ISR , 1 ) * XMR + RGRAD( ISR , 2 ) * YMR
1312 271      UUOR = 1.E-12 +
1313 272      .      UGRAD( ISR , 1 ) * XMR + UGRAD( ISR , 2 ) * YMR
1314 273      VVOR = 1.E-12 +
1315 274      .      VGRAD( ISR , 1 ) * XMR + VGRAD( ISR , 2 ) * YMR
1316 275      PPOR = 1.E-12 +
1317 276      .      PGRAD( ISR , 1 ) * XMR + PGRAD( ISR , 2 ) * YMR
1318 277      C

```

```

1319 278 ELSE 1319
1320 279 C 1320
1321 280 RROR = RROR 1321
1322 281 UUOR = UUOL 1322
1323 282 VVOR = VVOL 1323
1324 283 PPOR = PPOL 1324
1325 284 C 1325
1326 285 END IF 1326
1327 286 C 1327
1328 287 ROL( 1 ) = 1. / RROR 1328
1329 288 UOL( 1 ) = 1. / UUOL 1329
1330 289 VOL( 1 ) = 1. / VVOL 1330
1331 290 POL( 1 ) = 1. / PPOL 1331
1332 291 C 1332
1333 292 ROR( 1 ) = 1. / RROR 1333
1334 293 UOR( 1 ) = 1. / UUOR 1334
1335 294 VOR( 1 ) = 1. / VVOR 1335
1336 295 POR( 1 ) = 1. / PPOR 1336
1337 296 C 1337
1338 297 C --- SECOND TRIANGLE EDGE ----- 1338
1339 298 C 1339
1340 299 IE = IABS( JS( 5 , IS ) ) 1340
1341 300 C 1341
1342 301 ISL = JE( 3 , IE ) 1342
1343 302 ISR = JE( 4 , IE ) 1343
1344 303 C 1344
1345 304 XML = XMIDL( IE ) - XS( 1 , ISL ) 1345
1346 305 YML = YMIDL( IE ) - XS( 2 , ISL ) 1346
1347 306 C 1347
1348 307 RROR = 1.E-12 + 1348
1349 308 . RGRAD( ISL , 1 ) * XML + RGRAD( ISL , 2 ) * YML 1349
1350 309 UUOL = 1.E-12 + 1350
1351 310 . UGRAD( ISL , 1 ) * XML + UGRAD( ISL , 2 ) * YML 1351
1352 311 VVOL = 1.E-12 + 1352
1353 312 . VGRAD( ISL , 1 ) * XML + VGRAD( ISL , 2 ) * YML 1353
1354 313 PPOL = 1.E-12 + 1354
1355 314 . PGRAD( ISL , 1 ) * XML + PGRAD( ISL , 2 ) * YML 1355
1356 315 C 1356
1357 316 IJE5 = JE( 5 , IE ) 1357
1358 317 IF( IJE5 .EQ. 0 ) THEN 1358
1359 318 C 1359
1360 319 XMR = XMIDL( IE ) - XS( 1 , ISR ) 1360
1361 320 YMR = YMIDL( IE ) - XS( 2 , ISR ) 1361
1362 321 C 1362
1363 322 RROR = 1.E-12 + 1363
1364 323 . RGRAD( ISR , 1 ) * XMR + RGRAD( ISR , 2 ) * YMR 1364
1365 324 UUOR = 1.E-12 + 1365
1366 325 . UGRAD( ISR , 1 ) * XMR + UGRAD( ISR , 2 ) * YMR 1366
1367 326 VVOR = 1.E-12 + 1367
1368 327 . VGRAD( ISR , 1 ) * XMR + VGRAD( ISR , 2 ) * YMR 1368
1369 328 PPOR = 1.E-12 + 1369
1370 329 . PGRAD( ISR , 1 ) * XMR + PGRAD( ISR , 2 ) * YMR 1370
1371 330 C 1371
1372 331 ELSE 1372
1373 332 C 1373
1374 333 RROR = RROR 1374
1375 334 UUOR = UUOL 1375
1376 335 VVOR = VVOL 1376
1377 336 PPOR = PPOL 1377
1378 337 C 1378
1379 338 END IF 1379
1380 339 C 1380
1381 340 ROL( 2 ) = 1. / RROR 1381
1382 341 UOL( 2 ) = 1. / UUOL 1382
1383 342 VOL( 2 ) = 1. / VVOL 1383
1384 343 POL( 2 ) = 1. / PPOL 1384
1385 344 C 1385
1386 345 ROR( 2 ) = 1. / RROR 1386
1387 346 UOR( 2 ) = 1. / UUOR 1387
1388 347 VOR( 2 ) = 1. / VVOR 1388
1389 348 POR( 2 ) = 1. / PPOR 1389
1390 349 C 1390
1391 350 C --- THIRD TRIANGLE EDGE ----- 1391
1392 351 C 1392

```

```

1393 352 IE = IABS( JS( 6 , IS ) ) 1393
1394 353 C 1394
1395 354 ISL = JE( 3 , IE ) 1395
1396 355 ISR = JE( 4 , IE ) 1396
1397 356 C 1397
1398 357 XML = XMIDL( IE ) - XS( 1 , ISL ) 1398
1399 358 YML = YMIDL( IE ) - XS( 2 , ISL ) 1399
1400 359 C 1400
1401 360 RROL = 1.E-12 + 1401
1402 361 RGRAD( ISL , 1 ) * XML + RGRAD( ISL , 2 ) * YML 1402
1403 362 UUOL = 1.E-12 + 1403
1404 363 UGRAD( ISL , 1 ) * XML + UGRAD( ISL , 2 ) * YML 1404
1405 364 VVOL = 1.E-12 + 1405
1406 365 VGRAD( ISL , 1 ) * XML + VGRAD( ISL , 2 ) * YML 1406
1407 366 PPOL = 1.E-12 + 1407
1408 367 PGRAD( ISL , 1 ) * XML + PGRAD( ISL , 2 ) * YML 1408
1409 368 C 1409
1410 369 IJE5 = JE( 5 , IE ) 1410
1411 370 IF( IJE5 .EQ. 0 ) THEN 1411
1412 371 C 1412
1413 372 XMR = XMIDL( IE ) - XS( 1 , ISR ) 1413
1414 373 YMR = YMIDL( IE ) - XS( 2 , ISR ) 1414
1415 374 C 1415
1416 375 RROR = 1.E-12 + 1416
1417 376 RGRAD( ISR , 1 ) * XMR + RGRAD( ISR , 2 ) * YMR 1417
1418 377 UUOR = 1.E-12 + 1418
1419 378 UGRAD( ISR , 1 ) * XMR + UGRAD( ISR , 2 ) * YMR 1419
1420 379 VVOR = 1.E-12 + 1420
1421 380 VGRAD( ISR , 1 ) * XMR + VGRAD( ISR , 2 ) * YMR 1421
1422 381 PPOR = 1.E-12 + 1422
1423 382 PGRAD( ISR , 1 ) * XMR + PGRAD( ISR , 2 ) * YMR 1423
1424 383 C 1424
1425 384 ELSE 1425
1426 385 C 1426
1427 386 RROR = RROL 1427
1428 387 UUOR = UUOL 1428
1429 388 VVOR = VVOL 1429
1430 389 PPOR = PPOL 1430
1431 390 C 1431
1432 391 END IF 1432
1433 392 C 1433
1434 393 ROL( 3 ) = 1. / RROL 1434
1435 394 UOL( 3 ) = 1. / UUOL 1435
1436 395 VOL( 3 ) = 1. / VVOL 1436
1437 396 POL( 3 ) = 1. / PPOL 1437
1438 397 C 1438
1439 398 ROR( 3 ) = 1. / RROR 1439
1440 399 UOR( 3 ) = 1. / UUOR 1440
1441 400 VOR( 3 ) = 1. / VVOR 1441
1442 401 POR( 3 ) = 1. / PPOR 1442
1443 402 C 1443
1444 403 ISNR = SIGN( 1. , ROR( 1 ) ) 1444
1445 404 ISNL = SIGN( 1. , ROL( 1 ) ) 1445
1446 405 C 1446
1447 406 C --- PERFORM THE LIMITING ON THE INCREMENTS ----- 1447
1448 407 C 1448
1449 408 TEMPR = ( 1 + ISNR ) * RRR( KS ) + 1449
1450 409 ( 1 - ISNR ) * RRL( KS ) 1450
1451 410 RUVPR1 = 0.5 * TEMPR * ROR( 1 ) 1451
1452 411 C 1452
1453 412 TEMPL = ( 1 + ISNL ) * RRR( KS ) + 1453
1454 413 ( 1 - ISNL ) * RRL( KS ) 1454
1455 414 RUVPL1 = 0.5 * TEMPL * ROL( 1 ) 1455
1456 415 C 1456
1457 416 ISNR = SIGN( 1. , ROR( 2 ) ) 1457
1458 417 ISNL = SIGN( 1. , ROL( 2 ) ) 1458
1459 418 C 1459
1460 419 TEMPR = ( 1 + ISNR ) * RRR( KS ) + 1460
1461 420 ( 1 - ISNR ) * RRL( KS ) 1461
1462 421 RUVPR2 = 0.5 * TEMPR * ROR( 2 ) 1462
1463 422 C 1463
1464 423 TEMPL = ( 1 + ISNL ) * RRR( KS ) + 1464
1465 424 ( 1 - ISNL ) * RRL( KS ) 1465
1466 425 RUVPL2 = 0.5 * TEMPL * ROL( 2 ) 1466

```

1467	426	C		1467
1468	427		ISNR = SIGN(1. , ROR(3))	1468
1469	428		ISNL = SIGN(1. , ROL(3))	1469
1470	429	C		1470
1471	430		TEMPR = (1 + ISNR) * RRR(KS) +	1471
1472	431		(1 - ISNR) * RRL(KS)	1472
1473	432		RUVPR3 = 0.5 * TEMPR * ROR(3)	1473
1474	433	C		1474
1475	434		TEMPL = (1 + ISNL) * RRR(KS) +	1475
1476	435		(1 - ISNL) * RRL(KS)	1476
1477	436		RUVPL3 = 0.5 * TEMPL * ROL(3)	1477
1478	437	C		1478
1479	438		RMIN(KS) = AMIN1(1. , RUVPR1 , RUVPL1 , RUVPR2 , RUVPL2 ,	1479
1480	439		RUVPR3 , RUVPL3)	1480
1481	440	C		1481
1482	441		ISNR = SIGN(1. , UOR(1))	1482
1483	442		ISNL = SIGN(1. , UOL(1))	1483
1484	443	C		1484
1485	444		TEMPR = (1 + ISNR) * UUR(KS) +	1485
1486	445		(1 - ISNR) * UUL(KS)	1486
1487	446		RUVPR1 = 0.5 * TEMPR * UOR(1)	1487
1488	447	C		1488
1489	448		TEMPL = (1 + ISNL) * UUR(KS) +	1489
1490	449		(1 - ISNL) * UUL(KS)	1490
1491	450		RUVPL1 = 0.5 * TEMPL * UOL(1)	1491
1492	451	C		1492
1493	452		ISNR = SIGN(1. , UOR(2))	1493
1494	453		ISNL = SIGN(1. , UOL(2))	1494
1495	454	C		1495
1496	455		TEMPR = (1 + ISNR) * UUR(KS) +	1496
1497	456		(1 - ISNR) * UUL(KS)	1497
1498	457		RUVPR2 = 0.5 * TEMPR * UOR(2)	1498
1499	458	C		1499
1500	459		TEMPL = (1 + ISNL) * UUR(KS) +	1500
1501	460		(1 - ISNL) * UUL(KS)	1501
1502	461		RUVPL2 = 0.5 * TEMPL * UOL(2)	1502
1503	462	C		1503
1504	463		ISNR = SIGN(1. , UOR(3))	1504
1505	464		ISNL = SIGN(1. , UOL(3))	1505
1506	465	C		1506
1507	466		TEMPR = (1 + ISNR) * UUR(KS) +	1507
1508	467		(1 - ISNR) * UUL(KS)	1508
1509	468		RUVPR3 = 0.5 * TEMPR * UOR(3)	1509
1510	469	C		1510
1511	470		TEMPL = (1 + ISNL) * UUR(KS) +	1511
1512	471		(1 - ISNL) * UUL(KS)	1512
1513	472		RUVPL3 = 0.5 * TEMPL * UOL(3)	1513
1514	473	C		1514
1515	474		UMIN(KS) = AMIN1(1. , RUVPR1 , RUVPL1 , RUVPR2 , RUVPL2 ,	1515
1516	475		RUVPR3 , RUVPL3)	1516
1517	476	C		1517
1518	477		ISNR = SIGN(1. , VOR(1))	1518
1519	478		ISNL = SIGN(1. , VOL(1))	1519
1520	479	C		1520
1521	480		TEMPR = (1 + ISNR) * VVR(KS) +	1521
1522	481		(1 - ISNR) * VVL(KS)	1522
1523	482		RUVPR1 = 0.5 * TEMPR * VOR(1)	1523
1524	483	C		1524
1525	484		TEMPL = (1 + ISNL) * VVR(KS) +	1525
1526	485		(1 - ISNL) * VVL(KS)	1526
1527	486		RUVPL1 = 0.5 * TEMPL * VOL(1)	1527
1528	487	C		1528
1529	488		ISNR = SIGN(1. , VOR(2))	1529
1530	489		ISNL = SIGN(1. , VOL(2))	1530
1531	490	C		1531
1532	491		TEMPR = (1 + ISNR) * VVR(KS) +	1532
1533	492		(1 - ISNR) * VVL(KS)	1533
1534	493		RUVPR2 = 0.5 * TEMPR * VOR(2)	1534
1535	494	C		1535
1536	495		TEMPL = (1 + ISNL) * VVR(KS) +	1536
1537	496		(1 - ISNL) * VVL(KS)	1537
1538	497		RUVPL2 = 0.5 * TEMPL * VOL(2)	1538
1539	498	C		1539
1540	499		ISNR = SIGN(1. , VOR(3))	1540


```

1541 500      ISNL = SIGN( 1. , VOL( 3 ) )
1542 501      C
1543 502      TEMPR = ( 1 + ISNR ) * VVR( KS ) +
1544 503      .      ( 1 - ISNR ) * VVL( KS )
1545 504      RUVPR3 = 0.5 * TEMPR * VOR( 3 )
1546 505      C
1547 506      TEMPL = ( 1 + ISNL ) * VVR( KS ) +
1548 507      .      ( 1 - ISNL ) * VVL( KS )
1549 508      RUVPL3 = 0.5 * TEMPL * VOL( 3 )
1550 509      C
1551 510      VMIN( KS ) = AMINI( 1. , RUVPR1 , RUVPL1 , RUVPR2 , RUVPL2 ,
1552 511      .      RUVPR3 , RUVPL3 )
1553 512      C
1554 513      ISNR = SIGN( 1. , POR( 1 ) )
1555 514      ISNL = SIGN( 1. , POL( 1 ) )
1556 515      C
1557 516      TEMPR = ( 1 + ISNR ) * PPR( KS ) +
1558 517      .      ( 1 - ISNR ) * PPL( KS )
1559 518      RUVPR1 = 0.5 * TEMPR * POR( 1 )
1560 519      C
1561 520      TEMPL = ( 1 + ISNL ) * PPR( KS ) +
1562 521      .      ( 1 - ISNL ) * PPL( KS )
1563 522      RUVPL1 = 0.5 * TEMPL * POL( 1 )
1564 523      C
1565 524      ISNR = SIGN( 1. , POR( 2 ) )
1566 525      ISNL = SIGN( 1. , POL( 2 ) )
1567 526      C
1568 527      TEMPR = ( 1 + ISNR ) * PPR( KS ) +
1569 528      .      ( 1 - ISNR ) * PPL( KS )
1570 529      RUVPR2 = 0.5 * TEMPR * POR( 2 )
1571 530      C
1572 531      TEMPL = ( 1 + ISNL ) * PPR( KS ) +
1573 532      .      ( 1 - ISNL ) * PPL( KS )
1574 533      RUVPL2 = 0.5 * TEMPL * POL( 2 )
1575 534      C
1576 535      ISNR = SIGN( 1. , POR( 3 ) )
1577 536      ISNL = SIGN( 1. , POL( 3 ) )
1578 537      C
1579 538      TEMPR = ( 1 + ISNR ) * PPR( KS ) +
1580 539      .      ( 1 - ISNR ) * PPL( KS )
1581 540      RUVPR3 = 0.5 * TEMPR * POR( 3 )
1582 541      C
1583 542      TEMPL = ( 1 + ISNL ) * PPR( KS ) +
1584 543      .      ( 1 - ISNL ) * PPL( KS )
1585 544      RUVPL3 = 0.5 * TEMPL * POL( 3 )
1586 545      C
1587 546      PMIN( KS ) = AMINI( 1. , RUVPR1 , RUVPL1 , RUVPR2 , RUVPL2 ,
1588 547      .      RUVPR3 , RUVPL3 )
1589 548      C
1590 549      170 CONTINUE
1591 550      C
1592 551      C --- LIMIT THE ACTUAL GRADIENTS -----
1593 552      C
1594 553      DO 330 IH = 1 , 2
1595 554      C
1596 555      DO 330 IS = NS1 , NS2
1597 556      .      KS = IS - NS1 + 1
1598 557      C
1599 558      RGRAD( IS , IH ) = RGRAD( IS , IH ) * RMIN( KS ) * FLATDR
1600 559      UGRAD( IS , IH ) = UGRAD( IS , IH ) * UMIN( KS ) * FLATDR
1601 560      VGRAD( IS , IH ) = VGRAD( IS , IH ) * VMIN( KS ) * FLATDR
1602 561      PGRAD( IS , IH ) = PGRAD( IS , IH ) * PMIN( KS ) * FLATDR
1603 562      C
1604 563      330 CONTINUE
1605 564      C
1606 565      NS1 = NS2 + 1
1607 566      NS2 = NS2 + NOFVES( INS + 1 )
1608 567      80 CONTINUE
1609 568      C
1610 569      C -----
1611 570      C
1612 571      C --- CALL THE CHARECTERISTIC LIMITER -----
1613 572      C
1614 573      CALL FCHART

```

```

1615 574 C
1616 575 C -----
1617 576 C
1618 577 C
1619 578 C --- EXIT POINT FROM SUBROUTINE -----
1620 579 C
1621 580 C -----
1622 581 C RETURN
1623 582 C -----
1624 583 C
1625 584 C ---
1626 585 C END

```

```

1627 1 SUBROUTINE GRADNG
1628 2 C
1629 3 C -----|
1630 4 C |
1631 5 C GRADNG COMPUTE THE GRADIENT FOR SECOND ORDER CALCULATION |
1632 6 C USING THE INFORMATION STORED ASSOCIATED WITH THE |
1633 7 C VERTICIES OF THE TRIANGLE TO COMPUTE THE GRADIENT |
1634 8 C |
1635 9 C -----|
1636 10 C
1637 11 C include 'cmsh00.h'
1638 12 C include 'chyd00.h'
1639 13 C include 'cint00.h'
1640 14 C include 'cphs10.h'
1641 15 C include 'cphs20.h'
1642 16 C
1643 17 C -----
1644 18 C
1645 19 C --- BEGIN LOOP OVER ALL CELLS IN THE DOMAIN -----
1646 20 C
1647 21 C NS1 = 1
1648 22 C NS2 = NOFVES( 1 )
1649 23 C DO 90 INS = 1 , NVEES
1650 24 C
1651 25 C --- FETCH HYDRO QUANTITIES -----
1652 26 C
1653 27 C DO 105 IS = NS1 , NS2
1654 28 C KS = IS - NS1 + 1
1655 29 C
1656 30 C IV1 = JS( 1 , IS )
1657 31 C IV2 = JS( 2 , IS )
1658 32 C IV3 = JS( 3 , IS )
1659 33 C XV1 = XV( 1 , IV1 )
1660 34 C XV2 = XV( 1 , IV2 )
1661 35 C XV3 = XV( 1 , IV3 )
1662 36 C YV1 = XV( 2 , IV1 )
1663 37 C YV2 = XV( 2 , IV2 )
1664 38 C YV3 = XV( 2 , IV3 )
1665 39 C C = ( XV2 - XV1 ) * ( YV3 - YV2 ) - ( XV3 - XV2 ) * ( YV2 - YV1 )
1666 40 C CINV = 1. / C
1667 41 C
1668 42 C RRMDL1 = HYDVVV( IV1 , 1 )
1669 43 C UUMDL1 = HYDVVV( IV1 , 2 ) / RRMDL1
1670 44 C VVMDL1 = HYDVVV( IV1 , 3 ) / RRMDL1
1671 45 C PPMDL1 = ( HYDVVV( IV1 , 4 ) - .5 * RRMDL1 * ( UUMDL1 * UUMDL1 +
1672 46 C VVMDL1 * VVMDL1 ) ) * ( HYDVVV( IV1 , 5 ) - 1. )
1673 47 C
1674 48 C RRMDL2 = HYDVVV( IV2 , 1 )
1675 49 C UUMDL2 = HYDVVV( IV2 , 2 ) / RRMDL2
1676 50 C VVMDL2 = HYDVVV( IV2 , 3 ) / RRMDL2
1677 51 C PPMDL2 = ( HYDVVV( IV2 , 4 ) - .5 * RRMDL2 * ( UUMDL2 * UUMDL2 +
1678 52 C VVMDL2 * VVMDL2 ) ) * ( HYDVVV( IV2 , 5 ) - 1. )
1679 53 C
1680 54 C RRMDL3 = HYDVVV( IV3 , 1 )
1681 55 C UUMDL3 = HYDVVV( IV3 , 2 ) / RRMDL3
1682 56 C VVMDL3 = HYDVVV( IV3 , 3 ) / RRMDL3
1683 57 C PPMDL3 = ( HYDVVV( IV3 , 4 ) - .5 * RRMDL3 * ( UUMDL3 * UUMDL3 +
1684 58 C VVMDL3 * VVMDL3 ) ) * ( HYDVVV( IV3 , 5 ) - 1. )
1685 59 C

```

```

1686      60      ZV1 = RRMDL1
1687      61      ZV2 = RRMDL2
1688      62      ZV3 = RRMDL3
1689      63      A = ( YV2 - YV1 ) * ( ZV3 - ZV2 ) - ( YV3 - YV2 ) * ( ZV2 - ZV1 )
1690      64      B = ( ZV2 - ZV1 ) * ( XV3 - XV2 ) - ( ZV3 - ZV2 ) * ( XV2 - XV1 )
1691      65      C
1692      66      RGRAD( IS , 1 ) = - A * CINV
1693      67      RGRAD( IS , 2 ) = - B * CINV
1694      68      C
1695      69      ZV1 = UUMDL1
1696      70      ZV2 = UUMDL2
1697      71      ZV3 = UUMDL3
1698      72      A = ( YV2 - YV1 ) * ( ZV3 - ZV2 ) - ( YV3 - YV2 ) * ( ZV2 - ZV1 )
1699      73      B = ( ZV2 - ZV1 ) * ( XV3 - XV2 ) - ( ZV3 - ZV2 ) * ( XV2 - XV1 )
1700      74      C
1701      75      UGRAD( IS , 1 ) = - A * CINV
1702      76      UGRAD( IS , 2 ) = - B * CINV
1703      77      C
1704      78      ZV1 = VVMDL1
1705      79      ZV2 = VVMDL2
1706      80      ZV3 = VVMDL3
1707      81      A = ( YV2 - YV1 ) * ( ZV3 - ZV2 ) - ( YV3 - YV2 ) * ( ZV2 - ZV1 )
1708      82      B = ( ZV2 - ZV1 ) * ( XV3 - XV2 ) - ( ZV3 - ZV2 ) * ( XV2 - XV1 )
1709      83      C
1710      84      VGRAD( IS , 1 ) = - A * CINV
1711      85      VGRAD( IS , 2 ) = - B * CINV
1712      86      C
1713      87      ZV1 = PPMDL1
1714      88      ZV2 = PPMDL2
1715      89      ZV3 = PPMDL3
1716      90      A = ( YV2 - YV1 ) * ( ZV3 - ZV2 ) - ( YV3 - YV2 ) * ( ZV2 - ZV1 )
1717      91      B = ( ZV2 - ZV1 ) * ( XV3 - XV2 ) - ( ZV3 - ZV2 ) * ( XV2 - XV1 )
1718      92      C
1719      93      PGRAD( IS , 1 ) = - A * CINV
1720      94      PGRAD( IS , 2 ) = - B * CINV
1721      95      C
1722      96      105 CONTINUE
1723      97      C
1724      98      NS1 = NS2 + 1
1725      99      NS2 = NS2 + NOFVES( INS + 1 )
1726     100      90 CONTINUE
1727     101      C
1728     102      C -----
1729     103      C
1730     104      C --- CALL THE MONOTONICITY LIMITER -----
1731     105      C
1732     106      CALL MONOTN
1733     107      C
1734     108      C -----
1735     109      C
1736     110      C
1737     111      C --- EXIT POINT FROM SUBROUTINE -----
1738     112      C
1739     113      C -----
1740     114      RETURN
1741     115      C -----
1742     116      C
1743     117      C ---
1744     118      END

```

```

1745 1 SUBROUTINE GRADNO 1745
1746 2 C 1746
1747 3 C-----I 1747
1748 4 C I 1748
1749 5 C GRADNO COMPUTE THE GRADIENT FOR SECOND ORDER CALCULATION I 1749
1750 6 C USING THE INFORMATION STORED ASSOCIATED WITH THE I 1750
1751 7 C VERTICIES OF THE TRIANGLE TO COMPUTE THE GRADIENT I 1751
1752 8 C APPLYING THE GRADIENT THEOREM I 1752
1753 9 C I 1753
1754 10 C-----I 1754
1755 11 C 1755
1756 12 include 'cmsh00.h' 1756
1757 13 include 'chyd00.h' 1757
1758 14 include 'cint00.h' 1758
1759 15 include 'cphs10.h' 1759
1760 16 include 'cphs20.h' 1760
1761 17 C 1761
1762 18 C----- 1762
1763 19 C 1763
1764 20 REAL RRMIDL(MBP),PPMIDL(MBP),UUMIDL(MBP),VVMIDL(MBP) 1764
1765 21 REAL RIGRAD(MBP),PIGRAD(MBP),UIGRAD(MBP),VIGRAD(MBP) 1765
1766 22 REAL RJGRAD(MBP),PJGRAD(MBP),UJGRAD(MBP),VJGRAD(MBP) 1766
1767 23 REAL RMAX(MBP),PMAX(MBP),UMAX(MBP),VMAX(MBP) 1767
1768 24 REAL RMIN(MBP),PMIN(MBP),UMIN(MBP),VMIN(MBP) 1768
1769 25 REAL RLEFTT(MBP),ULEFTT(MBP),VLEFTT(MBP),PLEFTT(MBP) 1769
1770 26 REAL RRIGHT(MBP),URIGHT(MBP),VRIGHT(MBP),PRIGHT(MBP) 1770
1771 27 REAL ROR(3),UOR(3),VOR(3),POR(3) 1771
1772 28 REAL ROL(3),UOL(3),VOL(3),POL(3) 1772
1773 29 REAL AA(3,3),BB(3,4),B(3),INDX(3),ATEMP(3,3,3),BTEMP(3,4,3) 1773
1774 30 REAL AAO(3,3),BBO(3,4) 1774
1775 31 C 1775
1776 32 C----- 1776
1777 33 C 1777
1778 34 C --- BEGIN LOOP OVER ALL CELLS IN THE DOMAIN ----- 1778
1779 35 C 1779
1780 36 DO 120 IH = 1 , 2 1780
1781 37 DO 120 IS = 1 , NS 1781
1782 38 RGRAD( IS , IH ) = 0. 1782
1783 39 UGRAD( IS , IH ) = 0. 1783
1784 40 VGRAD( IS , IH ) = 0. 1784
1785 41 PGRAD( IS , IH ) = 0. 1785
1786 42 120 CONTINUE 1786
1787 43 C 1787
1788 44 NE1 = 1 1788
1789 45 NE2 = NOFVEE( 1 ) 1789
1790 46 DO 90 INE = 1 , NVEEE 1790
1791 47 C 1791
1792 48 C --- FETCH HYDRO QUANTITIES ----- 1792
1793 49 C 1793
1794 50 DO 105 IE = NE1 , NE2 1794
1795 51 KE = IE - NE1 + 1 1795
1796 52 C 1796
1797 53 IV1 = JE( 1 , IE ) 1797
1798 54 IV2 = JE( 2 , IE ) 1798
1799 55 RRMIDL = ( HYDVVV( IV1 , 1 ) + HYDVVV( IV2 , 1 ) ) * .5 1799
1800 56 UUMIDL = ( HYDVVV( IV1 , 2 ) + HYDVVV( IV2 , 2 ) ) * .5 / RRMIDL 1800
1801 57 VVMIDL = ( HYDVVV( IV1 , 3 ) + HYDVVV( IV2 , 3 ) ) * .5 / RRMIDL 1801
1802 58 PPMIDL = ( HYDVVV( IV1 , 4 ) + HYDVVV( IV2 , 4 ) ) * .5 1802
1803 59 GGMDL = ( HYDVVV( IV1 , 5 ) + HYDVVV( IV2 , 5 ) ) * .5 1803
1804 60 PPMIDL = ( PPMIDL - .5 * RRMIDL * 1804
1805 61 ( UUMIDL * UUMIDL + VVMIDL * VVMIDL ) ) * ( GGMDL - 1. ) 1805
1806 62 C 1806
1807 63 RRMIDL( KE ) = RRMIDL 1807
1808 64 UUMIDL( KE ) = UUMIDL 1808
1809 65 VVMIDL( KE ) = VVMIDL 1809
1810 66 PPMIDL( KE ) = PPMIDL 1810
1811 67 C 1811
1812 68 105 CONTINUE 1812
1813 69 C 1813
1814 70 DO 110 IE = NE1 , NE2 1814
1815 71 KE = IE - NE1 + 1 1815
1816 72 C 1816
1817 73 XEXN = XE( 1 , IE ) * XN( IE ) 1817
1818 74 XEYN = XE( 1 , IE ) * YN( IE ) 1818

```

```

1819 75 C 1819
1820 76 RIGRAD( KE ) = RRMIDL( KE ) * XEXN 1820
1821 77 UIGRAD( KE ) = UUMIDL( KE ) * XEXN 1821
1822 78 VIGRAD( KE ) = VVMIDL( KE ) * XEXN 1822
1823 79 PIGRAD( KE ) = PPMIDL( KE ) * XEXN 1823
1824 80 C 1824
1825 81 RJGRAD( KE ) = RRMIDL( KE ) * XEYN 1825
1826 82 UJGRAD( KE ) = UUMIDL( KE ) * XEYN 1826
1827 83 VJGRAD( KE ) = VVMIDL( KE ) * XEYN 1827
1828 84 PJGRAD( KE ) = PPMIDL( KE ) * XEYN 1828
1829 85 C 1829
1830 86 110 CONTINUE 1830
1831 87 C 1831
1832 88 DO 130 IE = NE1 , NE2 1832
1833 89 KE = IE - NE1 + 1 1833
1834 90 C 1834
1835 91 ISL = JE( 3 , IE ) 1835
1836 92 ISR = JE( 4 , IE ) 1836
1837 93 IJES = JE( 5 , IE ) 1837
1838 94 C 1838
1839 95 IF( IJES . EQ . 0 ) THEN 1839
1840 96 C 1840
1841 97 RGRAD( ISL , 1 ) = RGRAD( ISL , 1 ) + RIGRAD( KE ) 1841
1842 98 RGRAD( ISR , 1 ) = RGRAD( ISR , 1 ) - RIGRAD( KE ) 1842
1843 99 RGRAD( ISL , 2 ) = RGRAD( ISL , 2 ) + RJGRAD( KE ) 1843
1844 100 RGRAD( ISR , 2 ) = RGRAD( ISR , 2 ) - RJGRAD( KE ) 1844
1845 101 UGRAD( ISL , 1 ) = UGRAD( ISL , 1 ) + UIGRAD( KE ) 1845
1846 102 UGRAD( ISR , 1 ) = UGRAD( ISR , 1 ) - UIGRAD( KE ) 1846
1847 103 UGRAD( ISL , 2 ) = UGRAD( ISL , 2 ) + UJGRAD( KE ) 1847
1848 104 UGRAD( ISR , 2 ) = UGRAD( ISR , 2 ) - UJGRAD( KE ) 1848
1849 105 VGRAD( ISL , 1 ) = VGRAD( ISL , 1 ) + VIGRAD( KE ) 1849
1850 106 VGRAD( ISR , 1 ) = VGRAD( ISR , 1 ) - VIGRAD( KE ) 1850
1851 107 VGRAD( ISL , 2 ) = VGRAD( ISL , 2 ) + VJGRAD( KE ) 1851
1852 108 VGRAD( ISR , 2 ) = VGRAD( ISR , 2 ) - VJGRAD( KE ) 1852
1853 109 PGRAD( ISL , 1 ) = PGRAD( ISL , 1 ) + PIGRAD( KE ) 1853
1854 110 PGRAD( ISR , 1 ) = PGRAD( ISR , 1 ) - PIGRAD( KE ) 1854
1855 111 PGRAD( ISL , 2 ) = PGRAD( ISL , 2 ) + PJGRAD( KE ) 1855
1856 112 PGRAD( ISR , 2 ) = PGRAD( ISR , 2 ) - PJGRAD( KE ) 1856
1857 113 C 1857
1858 114 ELSE 1858
1859 115 C 1859
1860 116 RGRAD( ISL , 1 ) = RGRAD( ISL , 1 ) + RIGRAD( KE ) 1860
1861 117 RGRAD( ISL , 2 ) = RGRAD( ISL , 2 ) + RJGRAD( KE ) 1861
1862 118 UGRAD( ISL , 1 ) = UGRAD( ISL , 1 ) + UIGRAD( KE ) 1862
1863 119 UGRAD( ISL , 2 ) = UGRAD( ISL , 2 ) + UJGRAD( KE ) 1863
1864 120 VGRAD( ISL , 1 ) = VGRAD( ISL , 1 ) + VIGRAD( KE ) 1864
1865 121 VGRAD( ISL , 2 ) = VGRAD( ISL , 2 ) + VJGRAD( KE ) 1865
1866 122 PGRAD( ISL , 1 ) = PGRAD( ISL , 1 ) + PIGRAD( KE ) 1866
1867 123 PGRAD( ISL , 2 ) = PGRAD( ISL , 2 ) + PJGRAD( KE ) 1867
1868 124 C 1868
1869 125 END IF 1869
1870 126 C 1870
1871 127 130 CONTINUE 1871
1872 128 NE1 = NE2 + 1 1872
1873 129 NE2 = NE2 + NOFVEE( INE + 1 ) 1873
1874 130 90 CONTINUE 1874
1875 131 C 1875
1876 132 DO 140 IH = 1 , 2 1876
1877 133 DO 140 IS = 1 , NS 1877
1878 134 RGRAD( IS , IH ) = RGRAD( IS , IH ) * SAREA( IS ) 1878
1879 135 UGRAD( IS , IH ) = UGRAD( IS , IH ) * SAREA( IS ) 1879
1880 136 VGRAD( IS , IH ) = VGRAD( IS , IH ) * SAREA( IS ) 1880
1881 137 PGRAD( IS , IH ) = PGRAD( IS , IH ) * SAREA( IS ) 1881
1882 138 140 CONTINUE 1882
1883 139 C 1883
1884 140 C----- 1884
1885 141 C 1885
1886 142 C --- CALL THE MONOTONICITY LIMITER ----- 1886
1887 143 C 1887
1888 144 CALL MONOTN 1888
1889 145 C 1889
1890 146 C----- 1890
1891 147 C 1891
1892 148 C 1892

```

```

1893 149 C --- EXIT POINT FROM SUBROUTINE ----- 1893
1894 150 C 1894
1895 151 C ----- 1895
1896 152 C RETURN 1896
1897 153 C ----- 1897
1898 154 C 1898
1899 155 C --- 1899
1900 156 C END 1900

```

```

1901 1 SUBROUTINE GRADNS 1901
1902 2 C 1902
1903 3 C ----- I 1903
1904 4 C I 1904
1905 5 C GRADNS COMPUTE THE GRADIENT FOR SECOND ORDER CALCULATION I 1905
1906 6 C USING THE INFORMATION ASSOCIATE WITH THE BARICENTER I 1906
1907 7 C OF THE TWO TRIANGLES FROM THE TWO SIDE OF EACH | 1907
1908 8 C EDGE COMPUTING THE VALUE FOR THE EDGE AND APPLYING I 1908
1909 9 C THE GRADIENT THEOREM TO COMPUTE THE GRADIENT I 1909
1910 10 C I 1910
1911 11 C ----- I 1911
1912 12 C 1912
1913 13 C include 'cmsh00.h' 1913
1914 14 C include 'chyd00.h' 1914
1915 15 C include 'cint00.h' 1915
1916 16 C include 'cphs10.h' 1916
1917 17 C include 'cphs20.h' 1917
1918 18 C 1918
1919 19 C ----- 1919
1920 20 C 1920
1921 21 C REAL RRIDL(MBP),PPIDL(MBP),UUMIDL(MBP),VVMIDL(MBP) 1921
1922 22 C REAL RIGRAD(MBP),PIGRAD(MBP),UIGRAD(MBP),VIGRAD(MBP) 1922
1923 23 C REAL RJGRAD(MBP),PJGRAD(MBP),UJGRAD(MBP),VJGRAD(MBP) 1923
1924 24 C REAL RMAX(MBP),PMAX(MBP),UMAX(MBP),VMAX(MBP) 1924
1925 25 C REAL RMIN(MBP),PMIN(MBP),UMIN(MBP),VMIN(MBP) 1925
1926 26 C REAL RLEFTT(MBP),ULEFTT(MBP),VLEFTT(MBP),PLEFTT(MBP) 1926
1927 27 C REAL RRIGHT(MBP),URIGHT(MBP),VRIGHT(MBP),PRIGHT(MBP) 1927
1928 28 C REAL ROR(3),UOR(3),VOR(3),POR(3) 1928
1929 29 C REAL ROL(3),UOL(3),VOL(3),POL(3) 1929
1930 30 C REAL AA(3,3),BB(3,4),B(3),INDX(3),ATEMP(3,3,3),BTEMP(3,4,3) 1930
1931 31 C REAL AAO(3,3),BB0(3,4) 1931
1932 32 C 1932
1933 33 C ----- 1933
1934 34 C 1934
1935 35 C --- BEGIN LOOP OVER ALL CELLS IN THE DOMAIN ----- 1935
1936 36 C 1936
1937 37 C DO 120 IH = 1 , 2 1937
1938 38 C DO 120 IS = 1 , NS 1938
1939 39 C RGRAD( IS , IH ) = 0. 1939
1940 40 C UGRAD( IS , IH ) = 0. 1940
1941 41 C VGRAD( IS , IH ) = 0. 1941
1942 42 C PGRAD( IS , IH ) = 0. 1942
1943 43 120 CONTINUE 1943
1944 44 C 1944
1945 45 C NE1 = 1 1945
1946 46 C NE2 = NOFVEE( 1 ) 1946
1947 47 C DO 90 INE = 1 , NVEEE 1947
1948 48 C 1948
1949 49 C --- FETCH HYDRO QUANTITIES ----- 1949
1950 50 C 1950
1951 51 C DO 105 IE = NE1 , NE2 1951
1952 52 C KE = IE - NE1 + 1 1952
1953 53 C 1953
1954 54 C ISL = JE( 3 , IE ) 1954
1955 55 C ISR = JE( 4 , IE ) 1955
1956 56 C IJE5 = JE( 5 , IE ) 1956
1957 57 C 1957
1958 58 C IF( IJE5 .EQ. 0 ) THEN 1958
1959 59 C 1959
1960 60 C RRMDL = XYMIDL( IE ) * ( HYDV( ISR , 1 ) - 1960
1961 61 C HYDV( ISL , 1 ) ) + HYDV( ISL , 1 ) 1961
1962 62 C UUMDL = XYMIDL( IE ) * ( HYDV( ISR , 2 ) - 1962
1963 63 C HYDV( ISL , 2 ) ) + HYDV( ISL , 2 ) 1963

```

```

1964 64      VVMDL = XYMIDL( IE ) * ( HYDV( ISR , 3 ) -
1965 65      .      HYDV( ISL , 3 ) ) + HYDV( ISL , 3 )
1966 66      PPMDL = XYMIDL( IE ) * ( HYDV( ISR , 4 ) -
1967 67      .      HYDV( ISL , 4 ) ) + HYDV( ISL , 4 )
1968 68      C
1969 69      ELSE
1970 70      C
1971 71      RRMDL = HYDV( ISL , 1 )
1972 72      UUMDL = HYDV( ISL , 2 )
1973 73      VVMDL = HYDV( ISL , 3 )
1974 74      PPMDL = HYDV( ISL , 4 )
1975 75      C
1976 76      END IF
1977 77      C
1978 78      RRMIDL( KE ) = RRMDL
1979 79      UUMIDL( KE ) = UUMDL
1980 80      VVMIDL( KE ) = VVMDL
1981 81      PPMIDL( KE ) = PPMDL
1982 82      C
1983 83 105 CONTINUE
1984 84      C
1985 85      DO 110 IE = NE1 , NE2
1986 86          KE = IE - NE1 + 1
1987 87      C
1988 88      XEXN = XE( 1 , IE ) * XN( IE )
1989 89      XEYN = XE( 1 , IE ) * YN( IE )
1990 90      C
1991 91      RIGRAD( KE ) = RRMIDL( KE ) * XEXN
1992 92      UIGRAD( KE ) = UUMIDL( KE ) * XEXN
1993 93      VIGRAD( KE ) = VVMIDL( KE ) * XEXN
1994 94      PIGRAD( KE ) = PPMIDL( KE ) * XEXN
1995 95      C
1996 96      RJGRAD( KE ) = RRMIDL( KE ) * XEYN
1997 97      UJGRAD( KE ) = UUMIDL( KE ) * XEYN
1998 98      VJGRAD( KE ) = VVMIDL( KE ) * XEYN
1999 99      PJGRAD( KE ) = PPMIDL( KE ) * XEYN
2000 100      C
2001 101 110 CONTINUE
2002 102      C
2003 103      DO 130 IE = NE1 , NE2
2004 104          KE = IE - NE1 + 1
2005 105      C
2006 106          ISL = JE( 3 , IE )
2007 107          ISR = JE( 4 , IE )
2008 108          IJES = JE( 5 , IE )
2009 109      C
2010 110          IF( IJES .EQ. 0 ) THEN
2011 111      C
2012 112          RGRAD( ISL , 1 ) = RGRAD( ISL , 1 ) + RIGRAD( KE )
2013 113          RGRAD( ISR , 1 ) = RGRAD( ISR , 1 ) - RIGRAD( KE )
2014 114          RGRAD( ISL , 2 ) = RGRAD( ISL , 2 ) + RJGRAD( KE )
2015 115          RGRAD( ISR , 2 ) = RGRAD( ISR , 2 ) - RJGRAD( KE )
2016 116          UGRAD( ISL , 1 ) = UGRAD( ISL , 1 ) + UIGRAD( KE )
2017 117          UGRAD( ISR , 1 ) = UGRAD( ISR , 1 ) - UIGRAD( KE )
2018 118          UGRAD( ISL , 2 ) = UGRAD( ISL , 2 ) + UJGRAD( KE )
2019 119          UGRAD( ISR , 2 ) = UGRAD( ISR , 2 ) - UJGRAD( KE )
2020 120          VGRAD( ISL , 1 ) = VGRAD( ISL , 1 ) + VIGRAD( KE )
2021 121          VGRAD( ISR , 1 ) = VGRAD( ISR , 1 ) - VIGRAD( KE )
2022 122          VGRAD( ISL , 2 ) = VGRAD( ISL , 2 ) + VJGRAD( KE )
2023 123          VGRAD( ISR , 2 ) = VGRAD( ISR , 2 ) - VJGRAD( KE )
2024 124          PGRAD( ISL , 1 ) = PGRAD( ISL , 1 ) + PIGRAD( KE )
2025 125          PGRAD( ISR , 1 ) = PGRAD( ISR , 1 ) - PIGRAD( KE )
2026 126          PGRAD( ISL , 2 ) = PGRAD( ISL , 2 ) + PJGRAD( KE )
2027 127          PGRAD( ISR , 2 ) = PGRAD( ISR , 2 ) - PJGRAD( KE )
2028 128      C
2029 129      ELSE
2030 130      C
2031 131          RGRAD( ISL , 1 ) = RGRAD( ISL , 1 ) + RIGRAD( KE )
2032 132          RGRAD( ISL , 2 ) = RGRAD( ISL , 2 ) + RJGRAD( KE )
2033 133          UGRAD( ISL , 1 ) = UGRAD( ISL , 1 ) + UIGRAD( KE )
2034 134          UGRAD( ISL , 2 ) = UGRAD( ISL , 2 ) + UJGRAD( KE )
2035 135          VGRAD( ISL , 1 ) = VGRAD( ISL , 1 ) + VIGRAD( KE )
2036 136          VGRAD( ISL , 2 ) = VGRAD( ISL , 2 ) + VJGRAD( KE )
2037 137          PGRAD( ISL , 1 ) = PGRAD( ISL , 1 ) + PIGRAD( KE )

```

```

2038 138      PGRAD( ISL , 2 ) = PGRAD( ISL , 2 ) + PJGRAD( KE )      2038
2039 139      C                                                              2039
2040 140      END IF                                                              2040
2041 141      C                                                              2041
2042 142      130 CONTINUE                                                       2042
2043 143      NE1 = NE2 + 1                                                       2043
2044 144      NE2 = NE2 + NOFVEE( INE + 1 )                                       2044
2045 145      90 CONTINUE                                                         2045
2046 146      C                                                              2046
2047 147      DO 140 IH = 1 , 2                                                    2047
2048 148      DO 140 IS = 1 , NS                                                    2048
2049 149      RGRAD( IS , IH ) = RGRAD( IS , IH ) * SAREA( IS )                 2049
2050 150      UGRAD( IS , IH ) = UGRAD( IS , IH ) * SAREA( IS )                 2050
2051 151      VGRAD( IS , IH ) = VGRAD( IS , IH ) * SAREA( IS )                 2051
2052 152      PGRAD( IS , IH ) = PGRAD( IS , IH ) * SAREA( IS )                 2052
2053 153      140 CONTINUE                                                         2053
2054 154      C                                                              2054
2055 155      C-----                                                              2055
2056 156      C-----                                                              2056
2057 157      C --- CALL THE MONOTONICITY LIMITER -----                      2057
2058 158      C-----                                                              2058
2059 159      CALL MONOTN                                                         2059
2060 160      C-----                                                              2060
2061 161      C-----                                                              2061
2062 162      C-----                                                              2062
2063 163      C-----                                                              2063
2064 164      C --- EXIT POINT FROM SUBROUTINE -----                      2064
2065 165      C-----                                                              2065
2066 166      C-----                                                              2066
2067 167      RETURN                                                              2067
2068 168      C-----                                                              2068
2069 169      C-----                                                              2069
2070 170      C-----                                                              2070
2071 171      END                                                                2071

```

```

2072 1      SUBROUTINE LUDCMP(A,N,NP,INDX,D)                                       2072
2073 2      C-----                                                              2073
2074 3      C-----                                                              2074
2075 4      C-----                                                              2075
2076 5      C      PERFORM AN L U DECOMPOSITION OF THE A MATRIX                 2076
2077 6      C-----                                                              2077
2078 7      C-----                                                              2078
2079 8      C-----                                                              2079
2080 9      PARAMETER (NMAX=100,TINY=1.0E-20)                                       2080
2081 10     DIMENSION A(NP,NP),INDX(N),VV(NMAX)                                       2081
2082 11     D=1.                                                                      2082
2083 12     DO 12 I=1,N                                                              2083
2084 13     AAMAX=0.                                                                    2084
2085 14     DO 11 J=1,N                                                              2085
2086 15     IF (ABS(A(I,J)).GT.AAMAX) AAMAX=ABS(A(I,J))                             2086
2087 16     11 CONTINUE                                                              2087
2088 17     IF (AAMAX.EQ.0.) PAUSE 'Singular matrix.'                               2088
2089 18     VV(I)=1./AAMAX                                                            2089
2090 19     12 CONTINUE                                                              2090
2091 20     DO 19 J=1,N                                                              2091
2092 21     IF (J.GT.1) THEN                                                         2092
2093 22     DO 14 I=1,J-1                                                            2093
2094 23     SUM=A(I,J)                                                                2094
2095 24     IF (I.GT.1) THEN                                                         2095
2096 25     DO 13 K=1,I-1                                                            2096
2097 26     SUM=SUM-A(I,K)*A(K,J)                                                    2097
2098 27     13 CONTINUE                                                              2098
2099 28     A(I,J)=SUM                                                                2099
2100 29     ENDIF                                                                    2100
2101 30     14 CONTINUE                                                              2101
2102 31     ENDIF                                                                    2102
2103 32     AAMAX=0.                                                                    2103
2104 33     DO 16 I=J,N                                                              2104
2105 34     SUM=A(I,J)                                                                2105
2106 35     IF (J.GT.1) THEN                                                         2106
2107 36     DO 15 K=1,J-1                                                            2107
2108 37     SUM=SUM-A(I,K)*A(K,J)                                                    2108

```


2109	38	15	CONTINUE	2109
2110	39		A(I,J)=SUM	2110
2111	40		ENDIF	2111
2112	41		DUM=VV(I)*ABS(SUM)	2112
2113	42		IF (DUM.GE.AAMAX) THEN	2113
2114	43		IMAX=I	2114
2115	44		AAMAX=DUM	2115
2116	45		ENDIF	2116
2117	46	16	CONTINUE	2117
2118	47		IF (J.NE.IMAX)THEN	2118
2119	48		DO 17 K=1,N	2119
2120	49		DUM=A(IMAX,K)	2120
2121	50		A(IMAX,K)=A(J,K)	2121
2122	51		A(J,K)=DUM	2122
2123	52	17	CONTINUE	2123
2124	53		D=-D	2124
2125	54		VV(IMAX)=VV(J)	2125
2126	55		ENDIF	2126
2127	56		INDX(J)=IMAX	2127
2128	57		IF(J.NE.N)THEN	2128
2129	58		IF(A(J,J).EQ.0.)A(J,J)=TINY	2129
2130	59		DUM=1./A(J,J)	2130
2131	60		DO 18 I=J+1,N	2131
2132	61		A(I,J)=A(I,J)*DUM	2132
2133	62	18	CONTINUE	2133
2134	63		ENDIF	2134
2135	64	19	CONTINUE	2135
2136	65		IF(A(N,N).EQ.0.)A(N,N)=TINY	2136
2137	66		RETURN	2137
2138	67		END	2138
2139	68	c		2139

2140	1		SUBROUTINE LUBKSB(A,N,NP,INDX,B)	2140
2141	2		DIMENSION A(NP,NP),INDX(N),B(N)	2141
2142	3		II=0	2142
2143	4		DO 12 I=1,N	2143
2144	5		LL=INDX(I)	2144
2145	6		SUM=B(LL)	2145
2146	7		B(LL)=B(I)	2146
2147	8		IF (II.NE.0)THEN	2147
2148	9		DO 11 J=II,I-1	2148
2149	10		SUM=SUM-A(I,J)*B(J)	2149
2150	11	11	CONTINUE	2150
2151	12		ELSE IF (SUM.NE.0.) THEN	2151
2152	13		II=I	2152
2153	14		ENDIF	2153
2154	15		B(I)=SUM	2154
2155	16	12	CONTINUE	2155
2156	17		DO 14 I=N,1,-1	2156
2157	18		SUM=B(I)	2157
2158	19		IF(I.LT.N)THEN	2158
2159	20		DO 13 J=I+1,N	2159
2160	21		SUM=SUM-A(I,J)*B(J)	2160
2161	22	13	CONTINUE	2161
2162	23		ENDIF	2162
2163	24		B(I)=SUM/A(I,I)	2163
2164	25	14	CONTINUE	2164
2165	26		RETURN	2165
2166	27		END	2166

```

2167 1 SUBROUTINE FIRST 2167
2168 2 C 2168
2169 3 C-----I 2169
2170 4 C I 2170
2171 5 C FIRST IS USED TO FIND THE LEFT AND RIGHT INTERFACE I 2171
2172 6 C QUANTITIES TO FIRST ORDER WITHOUT USING EITHER THE I 2172
2173 7 C GRADIENT OR THE CHARACTERISTICS. I 2173
2174 8 C I 2174
2175 9 C-----I 2175
2176 10 C 2176
2177 11 include 'cmsh00.h' 2177
2178 12 include 'chyd00.h' 2178
2179 13 include 'cint00.h' 2179
2180 14 include 'cphs10.h' 2180
2181 15 include 'cphs20.h' 2181
2182 16 C 2182
2183 17 C----- 2183
2184 18 C 2184
2185 19 DO 110 IE = 1 , NE 2185
2186 20 ISL = JE( 3 , IE ) 2186
2187 21 ISR = JE( 4 , IE ) 2187
2188 22 IJES = JE( 5 , IE ) 2188
2189 23 RL( IE ) = HYDV( ISL , 1 ) 2189
2190 24 UL( IE ) = HYDV( ISL , 2 ) * XN( IE ) 2190
2191 25 + HYDV( ISL , 3 ) * YN( IE ) 2191
2192 26 VL( IE ) = - HYDV( ISL , 2 ) * YN( IE ) 2192
2193 27 + HYDV( ISL , 3 ) * XN( IE ) 2193
2194 28 PL( IE ) = HYDV( ISL , 4 ) 2194
2195 29 C 2195
2196 30 C --- EDGES IN THE COMPUTATIONAL DOMAIN ----- 2196
2197 31 C 2197
2198 32 IF( IJES .EQ. 0 ) THEN 2198
2199 33 RR( IE ) = HYDV( ISR , 1 ) 2199
2200 34 UR( IE ) = HYDV( ISR , 2 ) * XN( IE ) 2200
2201 35 + HYDV( ISR , 3 ) * YN( IE ) 2201
2202 36 VR( IE ) = - HYDV( ISR , 2 ) * YN( IE ) 2202
2203 37 + HYDV( ISR , 3 ) * XN( IE ) 2203
2204 38 PR( IE ) = HYDV( ISR , 4 ) 2204
2205 39 C 2205
2206 40 C --- EDGES ON THE BOUNDARY WITH ENFORCED CONDITIONS ----- 2206
2207 41 C 2207
2208 42 IJES = 6 A WALL WITH REFLECTING NORMAL COMPONENTS 2208
2209 43 = 7 SUPERSONIC OUTFLOW ZERO NORMAL DERIVATIVE 2209
2210 44 = 8 INFLOW WITH PRESPECIFIED VALUES (RIN,UIN,VIN,PIN) 2210
2211 45 C 2211
2212 46 ELSEIF( IJES .EQ. 8 ) THEN 2212
2213 47 RR( IE ) = RIN 2213
2214 48 UR( IE ) = UIN * XN( IE ) + VIN * YN( IE ) 2214
2215 49 VR( IE ) = - UIN * YN( IE ) + VIN * XN( IE ) 2215
2216 50 PR( IE ) = PIN 2216
2217 51 C 2217
2218 52 ELSEIF( IJES .EQ. 7 ) THEN 2218
2219 53 RR( IE ) = RL( IE ) 2219
2220 54 UR( IE ) = UL( IE ) 2220
2221 55 VR( IE ) = VL( IE ) 2221
2222 56 PR( IE ) = PL( IE ) 2222
2223 57 C 2223
2224 58 ELSEIF( IJES .EQ. 6 .OR. IJES .EQ. 5 ) THEN 2224
2225 59 RR( IE ) = RL( IE ) 2225
2226 60 UR( IE ) = - UL( IE ) 2226
2227 61 VR( IE ) = VL( IE ) 2227
2228 62 PR( IE ) = PL( IE ) 2228
2229 63 C 2229
2230 64 END IF 2230
2231 65 110 CONTINUE 2231
2232 66 C 2232
2233 67 C----- 2233
2234 68 C 2234
2235 69 C --- EXIT POINT FROM SUBROUTINE ----- 2235
2236 70 C 2236
2237 71 C 2237
2238 72 RETURN 2238
2239 73 C 2239
2240 74 C 2240

```

2241 75 C ---
2242 76 END

2241
2242

```

2243 1      SUBROUTINE FCHART                                2243
2244 2      C                                                2244
2245 3      C-----I                                         2245
2246 4      C                                                I                                         2246
2247 5      C      FCHART LIMITS THE PROJECTED INTERFACE VALUES ACCORDING TO I 2247
2248 6      C      CHARACTERISTICS.                               I 2248
2249 7      C-----I                                         2249
2250 8      C-----I                                         2250
2251 9      C                                                2251
2252 10     include 'cmsh00.h'                                2252
2253 11     include 'chyd00.h'                                2253
2254 12     include 'cint00.h'                                2254
2255 13     include 'cphs10.h'                               2255
2256 14     include 'cphs20.h'                               2256
2257 15     C                                                2257
2258 16     C-----I                                         2258
2259 17     C                                                2259
2260 18     REAL ZZLEFT(MBP),ZOLEFT(MBP),ZMLEFT(MBP)         2260
2261 19     REAL ZZRIGHT(MBP),ZORIGT(MBP),ZPRIGT(MBP)        2261
2262 20     REAL UPLEFT(MBP),UNLEFT(MBP),URLEFT(MBP),SQGRTL(MBP) 2262
2263 21     REAL UPRIGT(MBP),UMRIGHT(MBP),URRIGHT(MBP),SQGMTR(MBP) 2263
2264 22     REAL UVLEFT(MBP),UVRIGT(MBP),CNLEFT(MBP),CNRIGT(MBP) 2264
2265 23     REAL RLEFTT(MBP),ULEFTT(MBP),VLEFTT(MBP),PLEFTT(MBP) 2265
2266 24     REAL RRIGHT(MBP),URIGHT(MBP),VRIGHT(MBP),PRIGHT(MBP) 2266
2267 25     C-----I                                         2267
2268 26     C-----I                                         2268
2269 27     C                                                2269
2270 28     NE1 = 1                                           2270
2271 29     NE2 = NOFVEE( 1 )                                2271
2272 30     DO 90 INE = 1 , NVEEE                             2272
2273 31     C                                                2273
2274 32     DO 110 IE = NE1 , NE2                             2274
2275 33         KE = IE - NE1 + 1                             2275
2276 34     C                                                2276
2277 35         ISL = JE( 3 , IE )                             2277
2278 36         ISR = JE( 4 , IE )                             2278
2279 37         GAMAL( KE ) = HYDV( ISL , 5 )                 2279
2280 38     C-----I                                         2280
2281 39         CNLFTS = GAMAL( KE ) * HYDV( ISL , 4 ) / HYDV( ISL , 1 ) 2281
2282 40         CNLFT = SQRT( CNLFTS )                         2282
2283 41         UVLFT = HYDV( ISL , 2 ) * XXN( IE ) +         2283
2284 42             HYDV( ISL , 3 ) * YYN( IE )                 2284
2285 43     C-----I                                         2285
2286 44         IJES = JE( 5 , IE )                             2286
2287 45         IF( IJES .EQ. 0 ) THEN                         2287
2288 46     C-----I                                         2288
2289 47         GAMAR( KE ) = HYDV( ISR , 5 )                 2289
2290 48         CNRGTS = GAMAR( KE ) * HYDV( ISR , 4 ) / HYDV( ISR , 1 ) 2290
2291 49         CNRGT = SQRT( CNRGTS )                         2291
2292 50     C-----I                                         2292
2293 51         UVRGT = HYDV( ISR , 2 ) * XXN( IE ) +         2293
2294 52             HYDV( ISR , 3 ) * YYN( IE )                 2294
2295 53     C-----I                                         2295
2296 54         ELSE                                           2296
2297 55     C-----I                                         2297
2298 56         GAMAR( KE ) = GAMAL( KE )                     2298
2299 57         CNRGT = CNLFT                                   2299
2300 58         UVRGT = UVLFT                                   2300
2301 59     C-----I                                         2301
2302 60         END IF                                          2302
2303 61     C-----I                                         2303
2304 62         CNLEFT( KE ) = CNLFT                            2304
2305 63         CNRIGHT( KE ) = CNRGT                          2305
2306 64     C-----I                                         2306
2307 65         UVLEFT( KE ) = UVLFT                            2307
2308 66         UVRIGT( KE ) = UVRGT                           2308
2309 67     C-----I                                         2309
2310 68     110 CONTINUE                                       2310
2311 69     C-----I                                         2311

```

```

2312 70      DO 130 KE = 1 , NOFVEE( INE )
2313 71      C
2314 72      ZZLEFT( KE ) = .5 * ( UVLEFT( KE ) + CNLEFT( KE ) ) * DTT
2315 73      ZZRIGT( KE ) = -.5 * ( UVRIGT( KE ) - CNRIGT( KE ) ) * DTT
2316 74      C
2317 75      130 CONTINUE
2318 76      C
2319 77      C CHARACTERISTICS LOCATIONS
2320 78      C
2321 79      DO 140 KE = 1 , NOFVEE( INE )
2322 80      C
2323 81      IF( ZZLEFT( KE ) . LT . 0. ) ZZLEFT( KE ) = 0.
2324 82      IF( ZZRIGT( KE ) . LT . 0. ) ZZRIGT( KE ) = 0.
2325 83      C
2326 84      140 CONTINUE
2327 85      C
2328 86      DO 150 KE = 1 , NOFVEE( INE )
2329 87      C
2330 88      ZOLEFT( KE ) = .5 * UVLEFT( KE ) * DTT
2331 89      ZORIGT( KE ) = -.5 * UVRIGT( KE ) * DTT
2332 90      ZPRIGT( KE ) = -.5 * ( UVRIGT( KE ) + CNRIGT( KE ) ) * DTT
2333 91      ZMLEFT( KE ) = .5 * ( UVLEFT( KE ) - CNLEFT( KE ) ) * DTT
2334 92      C
2335 93      C 150 CONTINUE
2336 94      C
2337 95      C FIRST GUESS LEFT AND RIGHT VARIABLES, LINEAR INTERPOLATON
2338 96      C
2339 97      DO 160 IE = NE1 , NE2
2340 98      KE = IE - NE1 + 1
2341 99      C
2342 100     ISL = JE( 3 , IE )
2343 101     ISR = JE( 4 , IE )
2344 102     C
2345 103     XX = XMIDL( IE ) - ZZLEFT( KE ) * XXN( IE ) - XS( 1 , ISL )
2346 104     YY = YMIDL( IE ) - ZZLEFT( KE ) * YYN( IE ) - XS( 2 , ISL )
2347 105     C
2348 106     HRRL = HYDV( ISL , 1 ) +
2349 107     .   RGRAD( ISL , 1 ) * XX + RGRAD( ISL , 2 ) * YY
2350 108     HUUL = HYDV( ISL , 2 ) +
2351 109     .   UGRAD( ISL , 1 ) * XX + UGRAD( ISL , 2 ) * YY
2352 110     HVVL = HYDV( ISL , 3 ) +
2353 111     .   VGRAD( ISL , 1 ) * XX + VGRAD( ISL , 2 ) * YY
2354 112     HPPL = HYDV( ISL , 4 ) +
2355 113     .   PGRAD( ISL , 1 ) * XX + PGRAD( ISL , 2 ) * YY
2356 114     C
2357 115     GMTLFT = GAMAL( KE ) * HRRL * HPPL
2358 116     SQGMTL( KE ) = SQRT( GMTLFT )
2359 117     C
2360 118     UMLFT = 0.
2361 119     C
2362 120     IF( UVLEFT( KE ) - CNLEFT( KE ) . GT . 0. ) THEN
2363 121     C
2364 122     C XX = ( ZMLEFT( KE ) - ZZLEFT( KE ) ) * XXN( IE )
2365 123     C YY = ( ZMLEFT( KE ) - ZZLEFT( KE ) ) * YYN( IE )
2366 124     C UUU = UGRAD( ISL , 1 ) * XX + UGRAD( ISL , 2 ) * YY
2367 125     C VVV = VGRAD( ISL , 1 ) * XX + VGRAD( ISL , 2 ) * YY
2368 126     C UVU = UUU * XXN( IE ) + VVV * YYN( IE )
2369 127     C PPP = PGRAD( ISL , 1 ) * XX + PGRAD( ISL , 2 ) * YY
2370 128     C UMLFT = .5 * ( UVU - PPP / SQGMTL( KE ) ) / SQGMTL( KE )
2371 129     C
2372 130     C UMLFT = 0.
2373 131     C
2374 132     IF( UVLEFT( KE ) . GT . 0. ) THEN
2375 133     C
2376 134     C XX = ( ZOLEFT( KE ) - ZZLEFT( KE ) ) * XXN( IE )
2377 135     C YY = ( ZOLEFT( KE ) - ZZLEFT( KE ) ) * YYN( IE )
2378 136     C PPP = PGRAD( ISL , 1 ) * XX + PGRAD( ISL , 2 ) * YY
2379 137     C RRRR = HYDV( ISL , 1 ) +
2380 138     C .   RGRAD( ISL , 1 ) * XX + RGRAD( ISL , 2 ) * YY
2381 139     C URLFT = PPP / GMTLFT + 1. / HRRL - 1. / RRRR
2382 140     C
2383 141     C END IF
2384 142     C
2385 143     C
2386 144     IJES = JE( 5 , IE )
2387 145     IF( IJES . EQ . 0 ) THEN

```

```

2386 144      XX = XMIDL( IE ) + ZZRIGHT( KE ) * XXN( IE ) - XS( 1 , ISR )
2387 145      YY = YMIDL( IE ) + ZZRIGHT( KE ) * YYN( IE ) - XS( 2 , ISR )
2388 146      C
2389 147      HRRR = HYDV( ISR , 1 ) +
2390 148      . RGRAD( ISR , 1 ) * XX + RGRAD( ISR , 2 ) * YY
2391 149      HUUR = HYDV( ISR , 2 ) +
2392 150      . UGRAD( ISR , 1 ) * XX + UGRAD( ISR , 2 ) * YY
2393 151      HVVR = HYDV( ISR , 3 ) +
2394 152      . VGRAD( ISR , 1 ) * XX + VGRAD( ISR , 2 ) * YY
2395 153      HPPR = HYDV( ISR , 4 ) +
2396 154      . PGRAD( ISR , 1 ) * XX + PGRAD( ISR , 2 ) * YY
2397 155      C
2398 156      GMTRGT = GAMAR( KE ) * HRRR * HPPR
2399 157      SQGMTR( KE ) = SQRT( GMTRGT )
2400 158      C
2401 159      C      UPRGT = 0.
2402 160      C      IF( UVRIGHT( KE ) + CNRIGHT( KE ) . LT . 0. ) THEN
2403 161      C      XX = ( ZZRIGHT( KE ) - ZPRIGHT( KE ) ) * XXN( IE )
2404 162      C      YY = ( ZZRIGHT( KE ) - ZPRIGHT( KE ) ) * YYN( IE )
2405 163      C      UUU = UGRAD( ISR , 1 ) * XX + UGRAD( ISR , 2 ) * YY
2406 164      C      VVV = VGRAD( ISR , 1 ) * XX + VGRAD( ISR , 2 ) * YY
2407 165      C      UVU = UUU * XXN( IE ) + VVV * YYN( IE )
2408 166      C      PPP = PGRAD( ISR , 1 ) * XX + PGRAD( ISR , 2 ) * YY
2409 167      C      UPRGT = - .5 * ( UVU + PPP / SQGMTR( KE ) ) / SQGMTR( KE )
2410 168      C      END IF
2411 169      C
2412 170      C      URRGT = 0.
2413 171      C      IF( UVRIGHT( KE ) . LT . 0. ) THEN
2414 172      C      XX = ( ZZRIGHT( KE ) - ZORIGHT( KE ) ) * XXN( IE )
2415 173      C      YY = ( ZZRIGHT( KE ) - ZORIGHT( KE ) ) * YYN( IE )
2416 174      C      PPP = PGRAD( ISR , 1 ) * XX + PGRAD( ISR , 2 ) * YY
2417 175      C      XX = XMIDL( IE ) + ZORIGHT( KE ) * XXN( IE ) - XS( 1 , ISR )
2418 176      C      YY = YMIDL( IE ) + ZORIGHT( KE ) * YYN( IE ) - XS( 2 , ISR )
2419 177      C      RRRR = HYDV( ISR , 1 ) +
2420 178      . RGRAD( ISR , 1 ) * XX + RGRAD( ISR , 2 ) * YY
2421 179      C      URRGT = PPP / GMTRGT + 1. / HRRR - 1. / RRRR
2422 180      C      END IF
2423 181      C
2424 182      ELSE
2425 183      C
2426 184      HRRR = HRRL
2427 185      HUUR = HUUL
2428 186      HVVR = HVVL
2429 187      HPPR = HPPL
2430 188      C
2431 189      C      UPRGT = UMLFT
2432 190      C      URRGT = URLFT
2433 191      C
2434 192      END IF
2435 193      C
2436 194      RRL( KE ) = HRRL
2437 195      UUL( KE ) = HUUL * XN( IE ) + HVVL * YN( IE )
2438 196      VVL( KE ) = - HUUL * YN( IE ) + HVVL * XN( IE )
2439 197      PPL( KE ) = HPPL
2440 198      C
2441 199      RRR( KE ) = HRRR
2442 200      UUR( KE ) = HUUR * XN( IE ) + HVVR * YN( IE )
2443 201      VVR( KE ) = - HUUR * YN( IE ) + HVVR * XN( IE )
2444 202      PPR( KE ) = HPPR
2445 203      C
2446 204      C      UMLFT( KE ) = UMLFT
2447 205      C      URLFT( KE ) = URLFT
2448 206      C
2449 207      C      UPRIGHT( KE ) = UPRGT
2450 208      C      URRIGHT( KE ) = URRGT
2451 209      C
2452 210      160 CONTINUE
2453 211      C
2454 212      C      FINAL VALUES FOR RIGHT AND LEFT STATES
2455 213      C
2456 214      C      DO 180 KE = 1 , NOFVEE( INE )
2457 215      C
2458 216      C      GMTLFT = SQGMTL( KE ) * SQGMTL( KE )
2459 217      C      GMTRGT = SQGMTR( KE ) * SQGMTR( KE )

```

```

2460 218 C
2461 219 C      RRL( KE ) = 1. / ( 1. / RRL( KE ) - ( UMLEFT( KE ) +
2462 220 C          URLEFT( KE ) ) )
2463 221 C      UUL( KE ) = UUL( KE ) - SQGML( KE ) * UMLEFT( KE )
2464 222 C      PPL( KE ) = PPL( KE ) + GMTLFT * UMLEFT( KE )
2465 223 C
2466 224 C      RRR( KE ) = 1. / ( 1. / RRR( KE ) - ( UPRIGT( KE ) +
2467 225 C          URRIGT( KE ) ) )
2468 226 C      UUR( KE ) = UUR( KE ) + SQGMTR( KE ) * UPRIGT( KE )
2469 227 C      PPR( KE ) = PPR( KE ) + GMTRGT * UPRIGT( KE )
2470 228 C
2471 229 C 180 CONTINUE
2472 230 C
2473 231 C      DO 200 IE = NE1 , NE2
2474 232 C          KE = IE - NE1 + 1
2475 233 C
2476 234 C          ISL = JE( 3 , IE )
2477 235 C          ISR = JE( 4 , IE )
2478 236 C
2479 237 C          IJES = JE( 5 , IE )
2480 238 C
2481 239 C --- PROJECTED VALUES ON THE LEFT SIDE OF THE INTERFACE -----
2482 240 C
2483 241 C          RL( IE ) = RRL( KE )
2484 242 C          UL( IE ) = UUL( KE )
2485 243 C          VL( IE ) = VVL( KE )
2486 244 C          PL( IE ) = PPL( KE )
2487 245 C
2488 246 C --- PROJECTED VALUES ON THE RIGHT SIDE OF THE INTERFACE -----
2489 247 C
2490 248 C --- EDGES IN THE COMPUTATIONAL DOMAIN -----
2491 249 C
2492 250 C      IF( IJES .EQ. 0 ) THEN
2493 251 C          RR( IE ) = RRR( KE )
2494 252 C          UR( IE ) = UUR( KE )
2495 253 C          VR( IE ) = VVR( KE )
2496 254 C          PR( IE ) = PPR( KE )
2497 255 C
2498 256 C --- EDGES ON THE BOUNDARY -----
2499 257 C
2500 258 C      ELSEIF( IJES .EQ. 8 ) THEN
2501 259 C          RR( IE ) = RIN
2502 260 C          UR( IE ) = UIN * XN( IE ) + VIN * YN( IE )
2503 261 C          VR( IE ) = - UIN * YN( IE ) + VIN * XN( IE )
2504 262 C          PR( IE ) = PIN
2505 263 C
2506 264 C      ELSEIF( IJES .EQ. 7 ) THEN
2507 265 C          RR( IE ) = RL( IE )
2508 266 C          UR( IE ) = UL( IE )
2509 267 C          VR( IE ) = VL( IE )
2510 268 C          PR( IE ) = PL( IE )
2511 269 C
2512 270 C      ELSEIF( IJES .EQ. 6 .OR. IJES .EQ. 5 ) THEN
2513 271 C          RR( IE ) = RL( IE )
2514 272 C          UR( IE ) = - UL( IE )
2515 273 C          VR( IE ) = VL( IE )
2516 274 C          PR( IE ) = PL( IE )
2517 275 C
2518 276 C      END IF
2519 277 C 200 CONTINUE
2520 278 C
2521 279 C      NE1 = NE2 + 1
2522 280 C      NE2 = NE2 + NOFVEE( INE + 1 )
2523 281 C 90 CONTINUE
2524 282 C
2525 283 C -----
2526 284 C
2527 285 C --- EXIT POINT FROM SUBROUTINE -----
2528 286 C
2529 287 C -----
2530 288 C      RETURN
2531 289 C -----
2532 290 C
2533 291 C -----

```

2534 292 END

2534

```

2535      1      SUBROUTINE PRLCTN                                2535
2536      2      C                                                2536
2537      3      C-----I                                        2537
2538      4      C                                                I 2538
2539      5      C      PRLCTN INITIALIZE PARTICLES LOCATION IN THE COMPUTATION I 2539
2540      6      C      DOMAIN FOR THE FIRST TIME                    I 2540
2541      7      C                                                I 2541
2542      8      C-----I                                        I 2542
2543      9      C                                                I 2543
2544     10      include      'cmsh00.h'                            2544
2545     11      include      'chyd00.h'                            2545
2546     12      include      'cint00.h'                            2546
2547     13      include      'cphs10.h'                            2547
2548     14      include      'cphs20.h'                            2548
2549     15      C                                                2549
2550     16      C-----I                                        2550
2551     17      C                                                2551
2552     18      IPT = 0                                            2552
2553     19      DO 110 IPRTCL = 1 , NPT                            2553
2554     20      C                                                2554
2555     21      IDUM = 0                                           2555
2556     22      DO 130 IS = 1 , NS                                2556
2557     23      IF( IDUM .EQ. 0 ) THEN                            2557
2558     24      C                                                2558
2559     25      IV1 = JS( 1 , IS )                                2559
2560     26      IV2 = JS( 2 , IS )                                2560
2561     27      IV3 = JS( 3 , IS )                                2561
2562     28      C                                                2562
2563     29      X1 = XV( 1 , IV1 )                                2563
2564     30      Y1 = XV( 2 , IV1 )                                2564
2565     31      X2 = XV( 1 , IV2 )                                2565
2566     32      Y2 = XV( 2 , IV2 )                                2566
2567     33      C                                                2567
2568     34      XX = ( X2 - X1 )                                    2568
2569     35      XXP = ( XPRTCL( 1 , IPRTCL ) - X1 )              2569
2570     36      C                                                2570
2571     37      YY = ( Y2 - Y1 )                                    2571
2572     38      YYP = ( XPRTCL( 2 , IPRTCL ) - Y1 )              2572
2573     39      C                                                2573
2574     40      A1 = XX * YYP - YY * XXP                          2574
2575     41      C                                                2575
2576     42      X1 = XV( 1 , IV2 )                                2576
2577     43      Y1 = XV( 2 , IV2 )                                2577
2578     44      X2 = XV( 1 , IV3 )                                2578
2579     45      Y2 = XV( 2 , IV3 )                                2579
2580     46      C                                                2580
2581     47      XX = ( X2 - X1 )                                    2581
2582     48      XXP = ( XPRTCL( 1 , IPRTCL ) - X1 )              2582
2583     49      C                                                2583
2584     50      YY = ( Y2 - Y1 )                                    2584
2585     51      YYP = ( XPRTCL( 2 , IPRTCL ) - Y1 )              2585
2586     52      C                                                2586
2587     53      A2 = XX * YYP - YY * XXP                          2587
2588     54      C                                                2588
2589     55      X1 = XV( 1 , IV3 )                                2589
2590     56      Y1 = XV( 2 , IV3 )                                2590
2591     57      X2 = XV( 1 , IV1 )                                2591
2592     58      Y2 = XV( 2 , IV1 )                                2592
2593     59      C                                                2593
2594     60      XX = ( X2 - X1 )                                    2594
2595     61      XXP = ( XPRTCL( 1 , IPRTCL ) - X1 )              2595
2596     62      C                                                2596
2597     63      YY = ( Y2 - Y1 )                                    2597
2598     64      YYP = ( XPRTCL( 2 , IPRTCL ) - Y1 )              2598
2599     65      C                                                2599
2600     66      A3 = XX * YYP - YY * XXP                          2600
2601     67      C                                                2601
2602     68      IA1 = INT( SIGN( 1.1 , A1 ) )                      2602
2603     69      IA2 = INT( SIGN( 1.1 , A2 ) )                      2603
2604     70      IA3 = INT( SIGN( 1.1 , A3 ) )                      2604

```

```

2605 71      IAJ = IA1 + IA2 + IA3                                2605
2606 72      C                                                    2606
2607 73      IF( IAJ .EQ. 3 ) THEN                                2607
2608 74      IPT = IPT + 1                                        2608
2609 75      IJKPRT( IPT ) = IS                                  2609
2610 76      XPRTCL( 1 , IPT ) = XPRTCL( 1 , IPRTCL )           2610
2611 77      XPRTCL( 2 , IPT ) = XPRTCL( 2 , IPRTCL )           2611
2612 78      C      PRINT *, XPRTCL(1,IPT),XPRTCL(2,IPT),IJKPRT(IPT) 2612
2613 79      IDUM = 1                                            2613
2614 80      END IF                                              2614
2615 81      END IF                                              2615
2616 82      C                                                    2616
2617 83      130 CONTINUE                                         2617
2618 84      110 CONTINUE                                         2618
2619 85      NPT = IPT                                           2619
2620 86      C      PRINT *,      NPT,(XPRTCL(1,IPT),XPRTCL(2,IPT),IPT-1,NPT) 2620
2621 87      C      WRITE (10,*) NPT,(XPRTCL(1,IPT),XPRTCL(2,IPT),IPT-1,NPT) 2621
2622 88      C                                                    2622
2623 89      C                                                    2623
2624 90      C --- EXIT POINT FROM SUBROUTINE -----            2624
2625 91      C                                                    2625
2626 92      C      -----            2626
2627 93      RETURN                                              2627
2628 94      C      -----            2628
2629 95      C                                                    2629
2630 96      C      ---            2630
2631 97      END                                                2631

```

```

2632 1      SUBROUTINE PRPTHC                                2632
2633 2      C                                                    2633
2634 3      C-----I                                           2634
2635 4      C                                                    2635
2636 5      C      PRPTH TRACE PARTICLES PATH IN THE COMPUTATION DOMAIN I 2636
2637 6      C                                                    2637
2638 7      C-----I                                           2638
2639 8      C                                                    2639
2640 9      include 'cmsh00.h'                                    2640
2641 10     include 'chyd00.h'                                    2641
2642 11     include 'cint00.h'                                    2642
2643 12     include 'cphs10.h'                                    2643
2644 13     include 'cphs20.h'                                    2644
2645 14      C                                                    2645
2646 15     DO 110 IPRTCL = 1 , NPT                                2646
2647 16     KFINDD = 0                                            2647
2648 17      C                                                    2648
2649 18     DO 110 IK = 1 , 3                                       2649
2650 19     KFINDD = 0                                            2650
2651 20     IJES = 0                                              2651
2652 21     IS = IJKPRT( IPRTCL )                                  2652
2653 22     XP = XPRTCL( 1 , IPRTCL )                              2653
2654 23     YP = XPRTCL( 2 , IPRTCL )                              2654
2655 24      C                                                    2655
2656 25     DO 120 IJ = 1 , 3                                       2656
2657 26     IE = JS( IJ + 3 , IS )                                  2657
2658 27      C                                                    2658
2659 28     IF( IE .GT. 0 ) THEN                                    2659
2660 29      C                                                    2660
2661 30     IV1 = JE( 1 , IE )                                       2661
2662 31     IV2 = JE( 2 , IE )                                       2662
2663 32      C                                                    2663
2664 33     X1 = XV( 1 , IV1 )                                       2664
2665 34     Y1 = XV( 2 , IV1 )                                       2665
2666 35     X2 = XV( 1 , IV2 )                                       2666
2667 36     Y2 = XV( 2 , IV2 )                                       2667
2668 37      C                                                    2668
2669 38     XX = ( X2 - X1 )                                       2669
2670 39     XXP = ( XP - X1 )                                       2670
2671 40      C                                                    2671
2672 41     YY = ( Y2 - Y1 )                                       2672
2673 42     YYP = ( YP - Y1 )                                       2673
2674 43      C                                                    2674
2675 44     A = XX * YYP - YY * XXP                                2675

```



```

2676 45 IF( A . LT . 0. ) THEN 2676
2677 46 IJKPRT( IPRTCL ) = JE( 4 , IE ) 2677
2678 47 IJE5 = JE( 5 , IE ) 2678
2679 48 XREV = 1. / XE( 1 , IE ) 2679
2680 49 KFINN = KFINN + 1 2680
2681 50 END IF 2681
2682 51 C 2682
2683 52 ELSE 2683
2684 53 C 2684
2685 54 IV1 = JE( 2 , - IE ) 2685
2686 55 IV2 = JE( 1 , - IE ) 2686
2687 56 C 2687
2688 57 X1 = XV( 1 , IV1 ) 2688
2689 58 Y1 = XV( 2 , IV1 ) 2689
2690 59 X2 = XV( 1 , IV2 ) 2690
2691 60 Y2 = XV( 2 , IV2 ) 2691
2692 61 C 2692
2693 62 XX = ( X2 - X1 ) 2693
2694 63 XXP = ( XP - X1 ) 2694
2695 64 C 2695
2696 65 YY = ( Y2 - Y1 ) 2696
2697 66 YYP = ( YP - Y1 ) 2697
2698 67 C 2698
2699 68 A = XX * YYP - YY * XXP 2699
2700 69 IF( A . LT . 0. ) THEN 2700
2701 70 IJKPRT( IPRTCL ) = JE( 3 , - IE ) 2701
2702 71 IJE5 = JE( 5 , - IE ) 2702
2703 72 XREV = 1. / XE( 1 , - IE ) 2703
2704 73 KFINN = KFINN + 1 2704
2705 74 END IF 2705
2706 75 END IF 2706
2707 76 C 2707
2708 77 120 CONTINUE 2708
2709 78 C 2709
2710 79 IF( KFINN . GT . 0 . AND . IJE5 . NE . 0 ) THEN 2710
2711 80 IJKPRT( IPRTCL ) = IS 2711
2712 81 C 2712
2713 82 AA = X2 - X1 2713
2714 83 BB = Y2 - Y1 2714
2715 84 CC = XP - X1 2715
2716 85 DD = YP - Y1 2716
2717 86 TREV = ( CC * AA + DD * BB ) * XREV * XREV 2717
2718 87 IF( BB . NE . 0. ) THEN 2718
2719 88 XPRTCP = X1 + TREV * AA 2719
2720 89 XPRTCL( 1 , IPRTCL ) = XP + 1.1 * ( XPRTCP - XP ) 2720
2721 90 END IF 2721
2722 91 IF( AA . NE . 0. ) THEN 2722
2723 92 YPRTCP = Y1 + TREV * BB 2723
2724 93 XPRTCL( 2 , IPRTCL ) = YP + 1.1 * ( YPRTCP - YP ) 2724
2725 94 END IF 2725
2726 95 C 2726
2727 96 END IF 2727
2728 97 110 CONTINUE 2728
2729 98 C 2729
2730 99 DO 180 IPRTCL = 1 , NPT 2730
2731 100 C 2731
2732 101 IS = IJKPRT( IPRTCL ) 2732
2733 102 UPRTCL = HYDV( IS , 2 ) 2733
2734 103 VPRTCL = HYDV( IS , 3 ) 2734
2735 104 C 2735
2736 105 XPRTCL( 1 , IPRTCL ) = XPRTCL( 1 , IPRTCL ) + UPRTCL * DTT 2736
2737 106 XPRTCL( 2 , IPRTCL ) = XPRTCL( 2 , IPRTCL ) + VPRTCL * DTT 2737
2738 107 WPRTCL( 1 , IPRTCL ) = UPRTCL 2738
2739 108 WPRTCL( 2 , IPRTCL ) = VPRTCL 2739
2740 109 C 2740
2741 110 DO 180 IK = 1 , 3 2741
2742 111 IS = IJKPRT( IPRTCL ) 2742
2743 112 XP = XPRTCL( 1 , IPRTCL ) 2743
2744 113 YP = XPRTCL( 2 , IPRTCL ) 2744
2745 114 KFINN = 0 2745
2746 115 IJE5 = 0 2746
2747 116 C 2747
2748 117 DO 170 IJ = 1 , 3 2748
2749 118 IE = JS( IJ + 3 , IS ) 2749

```

```

2750 119 C
2751 120 IF( IE . GT . 0 ) THEN
2752 121 C
2753 122 IV1 = JE( 1 , IE )
2754 123 IV2 = JE( 2 , IE )
2755 124 C
2756 125 X1 = XV( 1 , IV1 )
2757 126 Y1 = XV( 2 , IV1 )
2758 127 X2 = XV( 1 , IV2 )
2759 128 Y2 = XV( 2 , IV2 )
2760 129 C
2761 130 XX = ( X2 - X1 )
2762 131 XXP = ( XP - X1 )
2763 132 C
2764 133 YY = ( Y2 - Y1 )
2765 134 YYP = ( YP - Y1 )
2766 135 C
2767 136 A = XX * YYP - YY * XXP
2768 137 IF( A . LT . 0. ) THEN
2769 138 IJKPRT( IPRTCL ) = JE( 4 , IE )
2770 139 IJES = JE( 5 , IE )
2771 140 XREV = 1. / XE( 1 , IE )
2772 141 KFINO = KFINO + 1
2773 142 END IF
2774 143 C
2775 144 ELSE
2776 145 C
2777 146 IV1 = JE( 2 , - IE )
2778 147 IV2 = JE( 1 , - IE )
2779 148 C
2780 149 X1 = XV( 1 , IV1 )
2781 150 Y1 = XV( 2 , IV1 )
2782 151 X2 = XV( 1 , IV2 )
2783 152 Y2 = XV( 2 , IV2 )
2784 153 C
2785 154 XX = ( X2 - X1 )
2786 155 XXP = ( XP - X1 )
2787 156 C
2788 157 YY = ( Y2 - Y1 )
2789 158 YYP = ( YP - Y1 )
2790 159 C
2791 160 A = XX * YYP - YY * XXP
2792 161 IF( A . LT . 0. ) THEN
2793 162 IJKPRT( IPRTCL ) = JE( 3 , - IE )
2794 163 IJES = JE( 5 , - IE )
2795 164 XREV = 1. / XE( 1 , - IE )
2796 165 KFINO = KFINO + 1
2797 166 END IF
2798 167 C
2799 168 END IF
2800 169 170 CONTINUE
2801 170 C
2802 171 IF( KFINO . GT . 0 . AND . IJES . NE . 0 ) THEN
2803 172 IJKPRT( IPRTCL ) = IS
2804 173 C
2805 174 AA = X2 - X1
2806 175 BB = Y2 - Y1
2807 176 CC = XP - X1
2808 177 DD = YP - Y1
2809 178 TREV = ( CC * AA + DD * BB ) * XREV * XREV
2810 179 IF( BB . NE . 0. ) THEN
2811 180 XPRTCP = X1 + TREV * AA
2812 181 XPRTCL( 1 , IPRTCL ) = XP + 1.1 * ( XPRTCP - XP )
2813 182 END IF
2814 183 IF( AA . NE . 0. ) THEN
2815 184 YPRTCP = Y1 + TREV * BB
2816 185 XPRTCL( 2 , IPRTCL ) = YP + 1.1 * ( YPRTCP - YP )
2817 186 END IF
2818 187 C
2819 188 END IF
2820 189 180 CONTINUE
2821 190 C
2822 191 C
2823 192 C --- EXIT POINT FROM SUBROUTINE -----

```

```
2824 193 C  
2825 194 C  
2826 195 C  
2827 196 C  
2828 197 C  
2829 198 C  
2830 199 C  
-----  
RETURN  
-----  
---  
END
```

```
2824  
2825  
2826  
2827  
2828  
2829  
2830
```

Thu Jul 1 14:16:08 1993

adaphd.f

Module List - order of occurrence

page i

#	routine	page
1	VERCEN	1
2	DISECT	4
3	DYNPTN	12
4	OYYPTN	21
5	INTPTN	30
6	DELPTNT	40
7	RELAXY	42
8	LAPLAC	47
9	RECNC	49
10	EOS	53
11	LIFTDR	56

Thu Jul 1 14:16:08 1993

adaphd.f

Module List - alphabetical order

#	routine	page
1	DELPTNT	40
2	DISECT	4
3	DYNPTN	12
4	OYYPTN	21
5	EOS	53
6	INTPTN	30
7	LAPLAC	47
8	LIFTDR	56
9	RECNC	49
10	RELAXY	42
11	VERCEN	1

```

1      1      SUBROUTINE VERGEN( IT )
2      2      C
3      3      C-----I
4      4      C
5      5      C      VERGEN ADD A VERTEX IN THE IT TRIANGLE. THE VERTEX
6      6      C      IS ADDED IN THE CENTROID OF THE TRIANGLE.
7      7      C-----I
8      8      C
9      9      C
10     10     C      IMPLICIT REAL (A-H,O-Z)
11     11     C
12     12     C      include      'cmsh00.h'
13     13     C      include      'chyd00.h'
14     14     C      include      'cint00.h'
15     15     C      include      'cphs10.h'
16     16     C      include      'cphs20.h'
17     17     C
18     18     C      SET UP THE NEW TRIANGLE BOOKKEEPING.
19     19     C
20     20     C      IV1 = JS( 1 , IT )
21     21     C      IV2 = JS( 2 , IT )
22     22     C      IV3 = JS( 3 , IT )
23     23     C
24     24     C      IE1 = JS( 4 , IT )
25     25     C      IE2 = JS( 5 , IT )
26     26     C      IE3 = JS( 6 , IT )
27     27     C      IE1A = IABS( IE1 )
28     28     C      IE2A = IABS( IE2 )
29     29     C      IE3A = IABS( IE3 )
30     30     C
31     31     C      PUT IN NEW TRIANGLES
32     32     C
33     33     C      NV = NV + 1
34     34     C      XV( 1 , NV ) = ( XV( 1 , IV1 ) + XV( 1 , IV2 ) +
35     35     C      .      XV( 1 , IV3 ) ) * THIRD
36     36     C      XV( 2 , NV ) = ( XV( 2 , IV1 ) + XV( 2 , IV2 ) +
37     37     C      .      XV( 2 , IV3 ) ) * THIRD
38     38     C      JV( 1 , NV ) = 0
39     39     C
40     40     C      DO 110 IR = 1 , MHQ
41     41     C      HYDVVV( NV , IR ) = ( HYDVVV( IV1 , IR ) +
42     42     C      .      HYDVVV( IV2 , IR ) +
43     43     C      .      HYDVVV( IV3 , IR ) ) * THIRD
44     44     110 CONTINUE
45     45     C
46     46     C      NE = NE + 1
47     47     C      JE( 1 , NE ) = NV
48     48     C      JE( 2 , NE ) = IV1
49     49     C      JE( 5 , NE ) = 0
50     50     C      NE = NE + 1
51     51     C      JE( 1 , NE ) = NV
52     52     C      JE( 2 , NE ) = IV2
53     53     C      JE( 5 , NE ) = 0
54     54     C      NE = NE + 1
55     55     C      JE( 1 , NE ) = NV
56     56     C      JE( 2 , NE ) = IV3
57     57     C      JE( 5 , NE ) = 0
58     58     C      NEM1 = NE - 1
59     59     C      NEM2 = NE - 2
60     60     C
61     61     C      TRIANGLE ONE, THE ORIGINAL IT.
62     62     C
63     63     C      JS( 3 , IT ) = NV
64     64     C      JS( 5 , IT ) = - NEM1
65     65     C      JS( 6 , IT ) = - NEM2
66     66     C
67     67     C      TRIANGLE TWO.
68     68     C
69     69     C      NS = NS + 1
70     70     C      JS( 1 , NS ) = IV2
71     71     C      JS( 2 , NS ) = IV3
72     72     C      JS( 3 , NS ) = NV
73     73     C      JS( 4 , NS ) = IE2

```

```

74 74 JS( 5 , NS ) = - NE
75 75 JS( 6 , NS ) = NEM1
76 76 C
77 77 C TRIANGLE THREE.
78 78 C
79 79 NS = NS + 1
80 80 JS( 1 , NS ) = IV3
81 81 JS( 2 , NS ) = IV1
82 82 JS( 3 , NS ) = NV
83 83 JS( 4 , NS ) = IE3
84 84 JS( 5 , NS ) = - NEM2
85 85 JS( 6 , NS ) = NE
86 86 C
87 87 C NOW FIX THE LEFT AND RIGHT FOR EDGES.
88 88 C
89 89 NSM1 = NS - 1
90 90 IF( JE( 4 , IE2A ) .EQ. IT ) JE( 4 , IE2A ) = NSM1
91 91 IF( JE( 3 , IE2A ) .EQ. IT ) JE( 3 , IE2A ) = NSM1
92 92 IF( JE( 4 , IE3A ) .EQ. IT ) JE( 4 , IE3A ) = NS
93 93 IF( JE( 3 , IE3A ) .EQ. IT ) JE( 3 , IE3A ) = NS
94 94 JE( 4 , NEM2 ) = NS
95 95 JE( 3 , NEM2 ) = IT
96 96 JE( 4 , NEM1 ) = IT
97 97 JE( 3 , NEM1 ) = NSM1
98 98 JE( 4 , NE ) = NSM1
99 99 JE( 3 , NE ) = NS
100 100 C
101 101 JV( 2 , NV ) = NE
102 102 C
103 103 XSAREA = XS( 3 , IT ) * THIRD
104 104 XS( 3 , IT ) = XSAREA
105 105 XS( 3 , NSM1 ) = XSAREA
106 106 XS( 3 , NS ) = XSAREA
107 107 C
108 108 XS( 1 , IT ) = ( XV( 1 , IV1 ) + XV( 1 , IV2 ) +
109 109 . XV( 1 , NV ) ) * THIRD
110 110 XS( 1 , NSM1 ) = ( XV( 1 , IV2 ) + XV( 1 , IV3 ) +
111 111 . XV( 1 , NV ) ) * THIRD
112 112 XS( 1 , NS ) = ( XV( 1 , IV3 ) + XV( 1 , IV1 ) +
113 113 . XV( 1 , NV ) ) * THIRD
114 114 XS( 2 , IT ) = ( XV( 2 , IV1 ) + XV( 2 , IV2 ) +
115 115 . XV( 2 , NV ) ) * THIRD
116 116 XS( 2 , NSM1 ) = ( XV( 2 , IV2 ) + XV( 2 , IV3 ) +
117 117 . XV( 2 , NV ) ) * THIRD
118 118 XS( 2 , NS ) = ( XV( 2 , IV3 ) + XV( 2 , IV1 ) +
119 119 . XV( 2 , NV ) ) * THIRD
120 120 C
121 121 XSAREA = 1. / XS( 3 , IT )
122 122 SAREA( IT ) = XSAREA
123 123 SAREA( NS ) = XSAREA
124 124 SAREA( NSM1 ) = XSAREA
125 125 C
126 126 DO 630 IR = 1 , MHQ
127 127 HYDV( IT , IR ) = ( HYDVVV( IV1 , IR ) +
128 128 . HYDVVV( IV2 , IR ) +
129 129 . HYDVVV( NV , IR ) ) * THIRD
130 130 HYDV( NS , IR ) = ( HYDVVV( IV3 , IR ) +
131 131 . HYDVVV( IV1 , IR ) +
132 132 . HYDVVV( NV , IR ) ) * THIRD
133 133 HYDV( NSM1 , IR ) = ( HYDVVV( IV2 , IR ) +
134 134 . HYDVVV( IV3 , IR ) +
135 135 . HYDVVV( NV , IR ) ) * THIRD
136 136 630 CONTINUE
137 137 C
138 138 HDUM = 1. / ( HYDV( IT , 1 ) + 1.E-12 )
139 139 HYDV( IT , 2 ) = HYDV( IT , 2 ) * HDUM
140 140 HYDV( IT , 3 ) = HYDV( IT , 3 ) * HDUM
141 141 HYDV( IT , 4 ) = ( HYDV( IT , 4 ) -
142 142 .5 * HYDV( IT , 1 ) ) *
143 143 . ( HYDV( IT , 2 ) * HYDV( IT , 2 ) +
144 144 . HYDV( IT , 3 ) * HYDV( IT , 3 ) ) *
145 145 . ( HYDV( IT , 5 ) - 1. )
146 146 C
147 147 HDUM = 1. / ( HYDV( NS , 1 ) + 1.E-12 )

```

```

148 148          HYDV( NS , 2 ) = HYDV( NS , 2 ) * HDUM          148
149 149          HYDV( NS , 3 ) = HYDV( NS , 3 ) * HDUM          149
150 150          HYDV( NS , 4 ) = ( HYDV( NS , 4 ) -            150
151 151          .5 * HYDV( NS , 1 ) *                            151
152 152          ( HYDV( NS , 2 ) * HYDV( NS , 2 ) +              152
153 153          HYDV( NS , 3 ) * HYDV( NS , 3 ) ) *              153
154 154          ( HYDV( NS , 5 ) - 1. )                          154
155 155          C                                                155
156 156          HDUM      = 1. / ( HYDV( NSM1 , 1 ) + 1.E-12 )    156
157 157          HYDV( NSM1 , 2 ) = HYDV( NSM1 , 2 ) * HDUM        157
158 158          HYDV( NSM1 , 3 ) = HYDV( NSM1 , 3 ) * HDUM        158
159 159          HYDV( NSM1 , 4 ) = ( HYDV( NSM1 , 4 ) -            159
160 160          .5 * HYDV( NSM1 , 1 ) *                            160
161 161          ( HYDV( NSM1 , 2 ) * HYDV( NSM1 , 2 ) +              161
162 162          HYDV( NSM1 , 3 ) * HYDV( NSM1 , 3 ) ) *              162
163 163          ( HYDV( NSM1 , 5 ) - 1. )                          163
164 164          C                                                164
165 165          DO 114 IR = 1 , 2                                    165
166 166          RGRAD( NS , IR ) = RGRAD( IT , IR )                166
167 167          RGRAD( NSM1 , IR ) = RGRAD( IT , IR )              167
168 168          C                                                168
169 169          UGRAD( NS , IR ) = UGRAD( IT , IR )                169
170 170          UGRAD( NSM1 , IR ) = UGRAD( IT , IR )              170
171 171          C                                                171
172 172          VGRAD( NS , IR ) = VGRAD( IT , IR )                172
173 173          VGRAD( NSM1 , IR ) = VGRAD( IT , IR )              173
174 174          C                                                174
175 175          PGRAD( NS , IR ) = PGRAD( IT , IR )                175
176 176          PGRAD( NSM1 , IR ) = PGRAD( IT , IR )              176
177 177          114 CONTINUE                                       177
178 178          C                                                178
179 179          JEN( 1 ) = IE1A                                       179
180 180          JEN( 2 ) = IE2A                                       180
181 181          JEN( 3 ) = IE3A                                       181
182 182          JEN( 4 ) = NEM2                                       182
183 183          JEN( 5 ) = NEM1                                       183
184 184          JEN( 6 ) = NE                                           184
185 185          C                                                185
186 186          DO 30 IENN = 1 , 6                                       186
187 187          IEN = JEN( IENN )                                       187
188 188          JV1 = JE( 1 , IEN )                                       188
189 189          JV2 = JE( 2 , IEN )                                       189
190 190          AX = XV( 1 , JV2 ) - XV( 1 , JV1 )                       190
191 191          AY = XV( 2 , JV2 ) - XV( 2 , JV1 )                       191
192 192          XE( 1 , IEN ) = SQRT( AX * AX + AY * AY )               192
193 193          XEREV = 1. / XE( 1 , IEN )                               193
194 194          XN( IEN ) = AY * XEREV                                       194
195 195          YN( IEN ) = - AX * XEREV                                       195
196 196          ISSR = JE( 4 , IEN )                                       196
197 197          ISSL = JE( 3 , IEN )                                       197
198 198          C                                                198
199 199          IJES = JE( 5 , IEN )                                       199
200 200          IF( IJES . NE . 0 ) THEN                                   200
201 201          C                                                201
202 202          AA = XV( 1 , JV2 ) - XV( 1 , JV1 )                       202
203 203          BB = XV( 2 , JV2 ) - XV( 2 , JV1 )                       203
204 204          XEL = XS( 1 , ISSL )                                       204
205 205          YEL = XS( 2 , ISSL )                                       205
206 206          CC = XEL - XV( 1 , JV1 )                                       206
207 207          DD = YEL - XV( 2 , JV1 )                                       207
208 208          EE = ( AA * CC + BB * DD ) * XEREV * XEREV             208
209 209          XER = XV( 1 , JV1 ) + AA * EE                                       209
210 210          YER = XV( 2 , JV1 ) + BB * EE                                       210
211 211          AX = XER - XEL                                       211
212 212          AY = YER - YEL                                       212
213 213          XE( 2 , IEN ) = SQRT( AX * AX + AY * AY )               213
214 214          XEREV = 1. / XE( 2 , IEN )                                       214
215 215          XXN( IEN ) = AX * XEREV                                       215
216 216          YYN( IEN ) = AY * XEREV                                       216
217 217          XE( 2 , IEN ) = 2. * XE( 2 , IEN )                       217
218 218          XMIDL( IEN ) = .5                                           218
219 219          XMIDL( IEN ) = XER                                       219
220 220          YMIDL( IEN ) = YER                                       220
221 221          C                                                221

```

```

222 222 ELSE
223 223 C
224 224 XER = XS( 1 , ISSR )
225 225 YER = XS( 2 , ISSR )
226 226 XEL = XS( 1 , ISSL )
227 227 YEL = XS( 2 , ISSL )
228 228 C
229 229 AA = XV( 1 , JV2 ) - XV( 1 , JVI )
230 230 BB = XV( 2 , JV2 ) - XV( 2 , JVI )
231 231 CC = XEL - XER
232 232 DD = YEL - YER
233 233 ACA = XER - XV( 1 , JVI )
234 234 DBD = YER - XV( 2 , JVI )
235 235 EE = ( ACA * DD - DBD * CC ) / ( AA * DD - BB * CC )
236 236 XMIDL( IEN ) = XV( 1 , JVI ) + AA * EE
237 237 YMIDL( IEN ) = XV( 2 , JVI ) + BB * EE
238 238 C
239 239 XEMID = XMIDL( IEN ) - XEL
240 240 YEMID = YMIDL( IEN ) - YEL
241 241 C
242 242 AX = XER - XEL
243 243 AY = YER - YEL
244 244 XE( 2 , IEN ) = SQRT( AX * AX + AY * AY )
245 245 XEREV = 1. / XE( 2 , IEN )
246 246 XXN( IEN ) = AX * XEREV
247 247 YYN( IEN ) = AY * XEREV
248 248 C
249 249 XYMIDL( IEN ) = SQRT( XEMID * XEMID + YEMID * YEMID ) * XEREV
250 250 C
251 251 END IF
252 252 C
253 253 30 CONTINUE
254 254 C
255 255 C --- EXIT POINT FROM SUBROUTINE -----
256 256 C
257 257 C
258 258 C ---TURN
259 259 C
260 260 C
261 261 C ---
262 262 C END

```

```

263 1 SUBROUTINE DISECT ( N , IDONE , IDUMP )
264 2 C
265 3 C -----I
266 4 C I
267 5 C DISECT DISECTS THE LINE N TO CREATE TWO NEW TRIANGLES AND I
268 6 C A NEW VERTEX. IF THE LINE N IS ON A SOLID BOUNDARY, ONLY I
269 7 C ONE NEW TRIANGLE IS CREATED. I
270 8 C I
271 9 C DISECT CANNOT BE USED FOR PERIODIC SIDES. HOWEVER, A I
272 10 C CALL TO TSHIFT CAN BE USED TO MAKE THOSE SIDES INTERNAL I
273 11 C BEFORE CALLING DISECT. I
274 12 C I
275 13 C INPUT: N - THE SIDE TO BE DISECTED. I
276 14 C I
277 15 C OUTPUT: N3 - THE SECOND HALF OF WHAT WAS N. WHEN N IS I
278 16 C DISECTED, THE INDEX N IS RETAINED FOR ONE OF I
279 17 C THE NEW SIDES, THE OTHER IS N3. I
280 18 C I
281 19 C I1 - THE STARTING VERTEX OF THE INPUT LINE N; I
282 20 C I2 - THE ENDING VERTEX OF THE INPUT LINE N; I
283 21 C I3 - THE THIRD VERTEX IN THE TRIANGLE TO THE RIGHT; I
284 22 C I4 - THE THIRD VERTEX IN THE TRIANGLE TO THE LEFT; I
285 23 C I5 - THE NEW VERTEX. I
286 24 C I
287 25 C -----I
288 26 C
289 27 C IMPLICIT REAL (A-H,O-Z)
290 28 C
291 29 C include 'cmsh00.h'
292 30 C include 'chyd00.h'

```



```

293 31      include 'cint00.h'      293
294 32      include 'cphs10.h'     294
295 33      include 'cphs20.h'     295
296 34      C      296
297 35      INTEGER IS(2), IVS(2)  297
298 36      C      298
299 37      ITRING = 0              299
300 38      IDONE = 0              300
301 39      IJES = JE( 5 , N )     301
302 40      IF( IJES . NE . 0 ) ITRING = 1  302
303 41      IS = 0                  303
304 42      IEROR = 0              304
305 43      EROR = 1.0E-3          305
306 44      C      306
307 45      C      FIND THE VERTICES OF THE LINE N.  307
308 46      C      308
309 47      I1 = JE( 1 , N )        309
310 48      I2 = JE( 2 , N )        310
311 49      IT1 = JE( 4 , N )       311
312 50      IT2 = JE( 3 , N )       312
313 51      C      313
314 52      C      FIND THE TWO VERTICES TO WHICH THE NEW LINES WILL BE DRAWN.  314
315 53      C      THESE ARE THE VERTICES OTHER THAN I1 AND I2 IN THE  315
316 54      C      TRIANGLES TO EITHER SIDE OF N. IVS STORES THE INDEX OF  316
317 55      C      THESE VERTICES AND IS STORES WHETHER THEY ARE VERTEX 1, 2  317
318 56      C      OR 3 IN THE TRIANGLE IT.  318
319 57      C      319
320 58      DO 10 I = 1 , 2        320
321 59      IVS(I) = 0              321
322 60      IT = JE( 5-I , N )      322
323 61      IF( IT . NE . 0 ) THEN  323
324 62      DO 20 J = 1 , 3        324
325 63      IV = JS( J , IT )      325
326 64      IF( IV . NE . I1 .AND. IV . NE . I2 ) THEN  326
327 65      IVS( I ) = IV          327
328 66      IS( I ) = J            328
329 67      END IF                329
330 68      20      CONTINUE        330
331 69      END IF                331
332 70      10      CONTINUE        332
333 71      I3 = IVS(1)            333
334 72      I4 = IVS(2)            334
335 73      IS1 = IS(1)            335
336 74      IS2 = IS(2)            336
337 75      C      337
338 76      C      COMPARE OPPOSING ANGLE PAIRS IN THE QUADRILATERAL  338
339 77      C      339
340 78      IF( ITRING . EQ . 0 ) THEN  340
341 79      AX = XV( 1 , I3 ) - XV( 1 , I1 )  341
342 80      AY = XV( 2 , I3 ) - XV( 2 , I1 )  342
343 81      BX = XV( 1 , I4 ) - XV( 1 , I1 )  343
344 82      BY = XV( 2 , I4 ) - XV( 2 , I1 )  344
345 83      CX = XV( 1 , I4 ) - XV( 1 , I2 )  345
346 84      CY = XV( 2 , I4 ) - XV( 2 , I2 )  346
347 85      DX = XV( 1 , I3 ) - XV( 1 , I2 )  347
348 86      DY = XV( 2 , I3 ) - XV( 2 , I2 )  348
349 87      AI2 = AX * BY - AY * BX  349
350 88      AI1 = CX * DY - CY * DX  350
351 89      XLN = XE( 1 , N )      351
352 90      ROUND F = EROR * XLN * XLN  352
353 91      IF( AI2 . LT . ROUND F . OR . AI1 . LT . ROUND F ) RETURN  353
354 92      END IF                354
355 93      C      355
356 94      C      CREATE A NEW VERTEX MIDWAY ON LINE N.  356
357 95      C      357
358 96      IDONE = 1              358
359 97      NV = NV + 1            359
360 98      IS = NV                360
361 99      C      361
362 100     C      CHANGE THE LINE N SO THAT IT STARTS AT THE SAME VERTEX,  362
363 101     C      BUT NOW ENDS AT IS.  363
364 102     C      364
365 103     JE( 2 , N ) = IS      365
366 104     C      366

```

```

367 105 C DRAW THE THREE NEW LINES, ALL ENDING AT 15. 367
368 106 C 368
369 107 DO 30 I = 1, 2 369
370 108 IF( JE( 5 - I, N ) . NE . 0 ) THEN 370
371 109 NE = NE + 1 371
372 110 JE( 1, NE ) = IVS( I ) 372
373 111 JE( 2, NE ) = 15 373
374 112 C 374
375 113 JE( 5, NE ) = 0 375
376 114 IF( I . EQ . 1 ) N1 = NE 376
377 115 IF( I . EQ . 2 ) N2 = NE 377
378 116 END IF 378
379 117 30 CONTINUE 379
380 118 C 380
381 119 C WE NEED TO HANDLE THE LINE FROM 12 TO 15 SEPARATELY, 381
382 120 C SINCE WE ARE NOT ADDING A LINE TO 12, BUT REPLACING 382
383 121 C THE OLD ONE. 383
384 122 C 384
385 123 NE = NE + 1 385
386 124 C 386
387 125 JE( 5, NE ) = JE( 5, N ) 387
388 126 N3 = NE 388
389 127 JE( 3, N3 ) = 0 389
390 128 JE( 4, N3 ) = 0 390
391 129 C 391
392 130 C RESET THE OLD TRIANGLES AND SET UP THE NEW TRIANGLES. 392
393 131 C 393
394 132 C N WAS ORIGINALLY DRAWN FROM 11 TO 12 -- NOW FROM 11 TO 15. 394
395 133 C N1 IS THE NEW LINE FROM 13 TO 15. 395
396 134 C N2 IS THE NEW LINE FROM 14 TO 15. 396
397 135 C N3 IS THE NEW LINE FROM 12 TO 15. 397
398 136 C NAA IS THE OLD LINE FROM 14 TO 11. 398
399 137 C NBB IS THE OLD LINE FROM 11 TO 13. 399
400 138 C NCC IS THE OLD LINE FROM 13 TO 12. 400
401 139 C NDD IS THE OLD LINE FROM 12 TO 14. 401
402 140 C THE DIRECTIONS OF LINES NAA THROUGH NDD ARE NOT 402
403 141 C EXPLICITLY USED. 403
404 142 C 404
405 143 IF( IT1 . NE . 0 ) THEN 405
406 144 NCC = JS( IS1 + 3, IT1 ) 406
407 145 JS( IS1 + 3, IT1 ) = N1 407
408 146 J = MOD( IS1, 3 ) + 1 408
409 147 IF( JS( J, IT1 ) . NE . 12 ) THEN 409
410 148 IEROR = 2 410
411 149 J1 = J 411
412 150 END IF 412
413 151 JS( J, IT1 ) = 15 413
414 152 C 414
415 153 JJ = MOD( IS1 + 1, 3 ) + 1 415
416 154 NBB = IABS( JS( JJ + 3, IT1 ) ) 416
417 155 C 417
418 156 NS = NS + 1 418
419 157 JS( 1, NS ) = 12 419
420 158 JS( 2, NS ) = 15 420
421 159 JS( 3, NS ) = 13 421
422 160 JS( 4, NS ) = N3 422
423 161 JS( 5, NS ) = - N1 423
424 162 JS( 6, NS ) = NCC 424
425 163 JE( 3, N3 ) = NS 425
426 164 JE( 4, N1 ) = NS 426
427 165 JE( 3, N1 ) = IT1 427
428 166 NCC = IABS( NCC ) 428
429 167 IF( JE( 4, NCC ) . EQ . IT1 ) JE( 4, NCC ) = NS 429
430 168 IF( JE( 3, NCC ) . EQ . IT1 ) JE( 3, NCC ) = NS 430
431 169 C 431
432 170 END IF 432
433 171 C 433
434 172 IF( IT2 . NE . 0 ) THEN 434
435 173 J = MOD( IS2 + 1, 3 ) + 1 435
436 174 NDD = JS( J + 3, IT2 ) 436
437 175 JS( J + 3, IT2 ) = - N2 437
438 176 IF( JS( J, IT2 ) . NE . 12 ) THEN 438
439 177 IEROR = 3 439
440 178 J2 = J 440

```

```

441 179          END IF
442 180          JS( J , IT2 ) = I5
443 181 C
444 182          NAA = IABS( JS( IS2 + 3 , IT2 ) )
445 183 C
446 184          NS          = NS + 1
447 185          JS( 1 , NS ) = I2
448 186          JS( 2 , NS ) = I4
449 187          JS( 3 , NS ) = I5
450 188          JS( 4 , NS ) = NDD
451 189          JS( 5 , NS ) = N2
452 190 C
453 191          IF( ITRING . EQ . 0 ) THEN
454 192 C
455 193          JE( 1 , N3 ) = I2
456 194          JE( 2 , N3 ) = I5
457 195          JS( 6 , NS ) = - N3
458 196          JE( 4 , N3 ) = NS
459 197 C
460 198          ELSE
461 199 C
462 200          JE( 1 , N3 ) = I5
463 201          JE( 2 , N3 ) = I2
464 202          JS( 6 , NS ) = N3
465 203          JE( 3 , N3 ) = NS
466 204 C
467 205          END i
468 206 C
469 207          JE( 3 , N2 ) = NS
470 208          JE( 4 , N2 ) = IT2
471 209          NDD          = IABS( NDD )
472 210          IF( JE( 4 , NDD ) . EQ . IT2 ) JE( 4 , NDD ) = NS
473 211          IF( JE( 3 , NDD ) . EQ . IT2 ) JE( 3 , NDD ) = NS
474 212 C
475 213          END IF
476 214 C
477 215          NSM1 = NS - 1
478 216          NEM1 = NE - 1
479 217          NEM2 = NE - 2
480 218 C
481 219          IF( !TRING . EQ . 0 ) THEN
482 220          XV( 1 , I5 ) = 0.25 * ( XV( 1 , I1 ) + XV( 1 , I2 ) +
483 221                  .          XV( 1 , I3 ) + XV( 1 , I4 ) )
484 222          XV( 2 , I5 ) = 0.25 * ( XV( 2 , I1 ) + XV( 2 , I2 ) +
485 223                  .          XV( 2 , I3 ) + XV( 2 , I4 ) )
486 224          JV( 1 , I5 ) = 0
487 225 C
488 226          DO 85 IR = 1 , MHQ
489 227          HYDVVV( I5 , IR ) = 0.25 * ( HYDVVV( I1 , IR ) +
490 228                  .          HYDVVV( I2 , IR ) +
491 229                  .          HYDVVV( I3 , IR ) +
492 230                  .          HYDVVV( I4 , IR ) )
493 231 85          CONTINUE
494 232 C
495 233          JV( 2 , NV ) = N
496 234          IF( JV( 2 , I2 ) . GT . 0 ) JV( 2 , I2 ) = N3
497 235 C
498 236          DX = XV( 1 , I3 ) - XV( 1 , I1 )
499 237          DXX = XV( 1 , I5 ) - XV( 1 , I1 )
500 238          DY = XV( 2 , I3 ) - XV( 2 , I1 )
501 239          DYY = XV( 2 , I5 ) - XV( 2 , I1 )
502 240          XS( 3 , IT1 ) = .5 * ( DX * DYY - DXX * DY )
503 241          DX = XV( 1 , I2 ) - XV( 1 , I3 )
504 242          DXX = XV( 1 , I5 ) - XV( 1 , I3 )
505 243          DY = XV( 2 , I2 ) - XV( 2 , I3 )
506 244          DYY = XV( 2 , I5 ) - XV( 2 , I3 )
507 245          XS( 3 , NSM1 ) = .5 * ( DX * DYY - DXX * DY )
508 246          DX = XV( 1 , I4 ) - XV( 1 , I2 )
509 247          DXX = XV( 1 , I5 ) - XV( 1 , I2 )
510 248          DY = XV( 2 , I4 ) - XV( 2 , I2 )
511 249          DYY = XV( 2 , I5 ) - XV( 2 , I2 )
512 250          XS( 3 , NS ) = .5 * ( DX * DYY - DXX * DY )
513 251          DX = XV( 1 , I1 ) - XV( 1 , I4 )
514 252          DXX = XV( 1 , I5 ) - XV( 1 , I4 )

```

```

515 253      DY = XV( 2 , 11 ) - XV( 2 , 14 )      515
516 254      DYY = XV( 2 , 15 ) - XV( 2 , 14 )    516
517 255      XS( 3 , IT2 ) = .5 * ( DX * DYY - DXX * DY ) 517
518 256      C                                          518
519 257      XS( 1 , IT1 ) = ( XV( 1 , 11 ) + XV( 1 , 13 ) + 519
520 258          XV( 1 , NV ) ) * THIRD            520
521 259      XS( 1 , NSM1 ) = ( XV( 1 , 13 ) + XV( 1 , 12 ) + 521
522 260          XV( 1 , NV ) ) * THIRD            522
523 261      XS( 2 , IT1 ) = ( XV( 2 , 11 ) + XV( 2 , 13 ) + 523
524 262          XV( 2 , NV ) ) * THIRD            524
525 263      XS( 2 , NSM1 ) = ( XV( 2 , 13 ) + XV( 2 , 12 ) + 525
526 264          XV( 2 , NV ) ) * THIRD            526
527 265      C                                          527
528 266      XS( 1 , NS ) = ( XV( 1 , 12 ) + XV( 1 , 14 ) + 528
529 267          XV( 1 , NV ) ) * THIRD            529
530 268      XS( 1 , IT2 ) = ( XV( 1 , 14 ) + XV( 1 , 11 ) + 530
531 269          XV( 1 , NV ) ) * THIRD            531
532 270      XS( 2 , NS ) = ( XV( 2 , 12 ) + XV( 2 , 14 ) + 532
533 271          XV( 2 , NV ) ) * THIRD            533
534 272      XS( 2 , IT2 ) = ( XV( 2 , 14 ) + XV( 2 , 11 ) + 534
535 273          XV( 2 , NV ) ) * THIRD            535
536 274      C                                          536
537 275      DO 94 IR = 1 , MHQ                          537
538 276      HYDV( IT1 , IR ) = ( HYDVVV( I1 , IR ) + 538
539 277          HYDVVV( I3 , IR ) + 539
540 278          HYDVVV( NV , IR ) ) * THIRD      540
541 279      HYDV( NSM1 , IR ) = ( HYDVVV( I3 , IR ) + 541
542 280          HYDVVV( I2 , IR ) + 542
543 281          HYDVVV( NV , IR ) ) * THIRD      543
544 282      HYDV( IT2 , IR ) = ( HYDVVV( I4 , IR ) + 544
545 283          HYDVVV( I1 , IR ) + 545
546 284          HYDVVV( NV , IR ) ) * THIRD      546
547 285      HYDV( NS , IR ) = ( HYDVVV( I2 , IR ) + 547
548 286          HYDVVV( I4 , IR ) + 548
549 287          HYDVVV( NV , IR ) ) * THIRD      549
550 288      94  CONTINUE                               550
551 289      C                                          551
552 290      HDUM = 1. / ( HYDV( IT1 , 1 ) + 1.E-12 ) 552
553 291      HYDV( IT1 , 2 ) = HYDV( IT1 , 2 ) * HDUM 553
554 292      HYDV( IT1 , 3 ) = HYDV( IT1 , 3 ) * HDUM 554
555 293      HYDV( IT1 , 4 ) = ( HYDV( IT1 , 4 ) - 555
556 294          .5 * HYDV( IT1 , 1 ) ) * 556
557 295          ( HYDV( IT1 , 2 ) * HYDV( IT1 , 2 ) + 557
558 296          HYDV( IT1 , 3 ) * HYDV( IT1 , 3 ) ) * 558
559 297          ( HYDV( IT1 , 5 ) - 1. ) 559
560 298      C                                          560
561 299      HDUM = 1. / ( HYDV( NSM1 , 1 ) + 1.E-12 ) 561
562 300      HYDV( NSM1 , 2 ) = HYDV( NSM1 , 2 ) * HDUM 562
563 301      HYDV( NSM1 , 3 ) = HYDV( NSM1 , 3 ) * HDUM 563
564 302      HYDV( NSM1 , 4 ) = ( HYDV( NSM1 , 4 ) - 564
565 303          .5 * HYDV( NSM1 , 1 ) ) * 565
566 304          ( HYDV( NSM1 , 2 ) * HYDV( NSM1 , 2 ) + 566
567 305          HYDV( NSM1 , 3 ) * HYDV( NSM1 , 3 ) ) * 567
568 306          ( HYDV( NSM1 , 5 ) - 1. ) 568
569 307      C                                          569
570 308      HDUM = 1. / ( HYDV( IT2 , 1 ) + 1.E-12 ) 570
571 309      HYDV( IT2 , 2 ) = HYDV( IT2 , 2 ) * HDUM 571
572 310      HYDV( IT2 , 3 ) = HYDV( IT2 , 3 ) * HDUM 572
573 311      HYDV( IT2 , 4 ) = ( HYDV( IT2 , 4 ) - 573
574 312          .5 * HYDV( IT2 , 1 ) ) * 574
575 313          ( HYDV( IT2 , 2 ) * HYDV( IT2 , 2 ) + 575
576 314          HYDV( IT2 , 3 ) * HYDV( IT2 , 3 ) ) * 576
577 315          ( HYDV( IT2 , 5 ) - 1. ) 577
578 316      C                                          578
579 317      HDUM = 1. / ( HYDV( NS , 1 ) + 1.E-12 ) 579
580 318      HYDV( NS , 2 ) = HYDV( NS , 2 ) * HDUM 580
581 319      HYDV( NS , 3 ) = HYDV( NS , 3 ) * HDUM 581
582 320      HYDV( NS , 4 ) = ( HYDV( NS , 4 ) - 582
583 321          .5 * HYDV( NS , 1 ) ) * 583
584 322          ( HYDV( NS , 2 ) * HYDV( NS , 2 ) + 584
585 323          HYDV( NS , 3 ) * HYDV( NS , 3 ) ) * 585
586 324          ( HYDV( NS , 5 ) - 1. ) 586
587 325      C                                          587
588 326      SAREA( IT1 ) = 1. / XS( 3 , IT1 ) 588

```

```

589 327          SAREA( NSM1 ) = 1. / XS( 3 , NSM1 )          589
590 328          SAREA( IT2 ) = 1. / XS( 3 , IT2 )          590
591 329          SAREA( NS ) = 1. / XS( 3 , NS )            591
592 330          C                                          592
593 331          DO 112 IR = 1 , 2                            593
594 332          RGRAD( NS , IR ) = RGRAD( IT2 , IR )        594
595 333          RGRAD( NSM1 , IR ) = RGRAD( IT1 , IR )      595
596 334          UGRAD( NS , IR ) = UGRAD( IT2 , IR )        596
597 335          UGRAD( NSM1 , IR ) = UGRAD( IT1 , IR )      597
598 336          VGRAD( NS , IR ) = VGRAD( IT2 , IR )        598
599 337          VGRAD( NSM1 , IR ) = VGRAD( IT1 , IR )      599
600 338          PGRAD( NS , IR ) = PGRAD( IT2 , IR )        600
601 339          PGRAD( NSM1 , IR ) = PGRAD( IT1 , IR )      601
602 340          112          CONTINUE                        602
603 341          C                                          603
604 342          KSDEL( NS ) = IDUMP                          604
605 343          KSDEL( NSM1 ) = IDUMP                        605
606 344          KSDEL( IT1 ) = IDUMP                        606
607 345          KSDEL( IT2 ) = IDUMP                        607
608 346          C                                          608
609 347          JEN( 1 ) = NAA                               609
610 348          JEN( 2 ) = NBB                               610
611 349          JEN( 3 ) = NCC                               611
612 350          JEN( 4 ) = NDD                               612
613 351          JEN( 5 ) = N                                613
614 352          JEN( 6 ) = N1                               614
615 353          JEN( 7 ) = N2                               615
616 354          JEN( 8 ) = N3                               616
617 355          JENN = 8                                     617
618 356          C                                          618
619 357          ELSE                                         619
620 358          C                                          620
621 359          XV( 1 , I5 ) = 0.5 * ( XV( 1 , I1 ) + XV( 1 , I2 ) ) 621
622 360          XV( 2 , I5 ) = 0.5 * ( XV( 2 , I1 ) + XV( 2 , I2 ) ) 622
623 361          JV( 1 , I5 ) = 0                             623
624 362          C                                          624
625 363          IF( IOSPCL .EQ. 1 .AND. IJE5 .EQ. 6 ) THEN 625
626 364          ANGL = 1.570796327                          626
627 365          DX = XV( 1 , I5 ) - 1.5                     627
628 366          IF( DX .NE. 0. ) ANGL = ATAN2( XV( 2 , I5 ) , DX ) 628
629 367          XV( 1 , I5 ) = COS( ANGL ) + 1.5            629
630 368          XV( 2 , I5 ) = SIN( ANGL )                  630
631 369          END IF                                       631
632 370          C                                          632
633 371          DO 80 IR = 1 , MHQ                            633
634 372          HYDVVV( I5 , IR ) = 0.5 * ( HYDVVV( I1 , IR ) + 634
635 373          HYDVVV( I2 , IR ) )                          635
636 374          80          CONTINUE                          636
637 375          C                                          637
638 376          JV( 2 , I1 ) = - N                            638
639 377          JV( 2 , NV ) = - N3                          639
640 378          C                                          640
641 379          XSAREA = .5 * XS( 3 , IT2 )                 641
642 380          XS( 3 , IT2 ) = XSAREA                       642
643 381          XS( 3 , NS ) = XSAREA                       643
644 382          C                                          644
645 383          XS( 1 , NS ) = ( XV( 1 , I2 ) + XV( 1 , I4 ) + 645
646 384          XV( 1 , NV ) ) * THIRD                       646
647 385          XS( 1 , IT2 ) = ( XV( 1 , I4 ) + XV( 1 , I1 ) + 647
648 386          XV( 1 , NV ) ) * THIRD                       648
649 387          XS( 2 , NS ) = ( XV( 2 , I2 ) + XV( 2 , I4 ) + 649
650 388          XV( 2 , NV ) ) * THIRD                       650
651 389          XS( 2 , IT2 ) = ( XV( 2 , I4 ) + XV( 2 , I1 ) + 651
652 390          XV( 2 , NV ) ) * THIRD                       652
653 391          C                                          653
654 392          DO 92 IR = 1 , MHQ                            654
655 393          HYDV( IT2 , IR ) = ( HYDVVV( I4 , IR ) + 655
656 394          HYDVVV( I1 , IR ) + 656
657 395          HYDVVV( NV , IR ) ) * THIRD                   657
658 396          HYDV( NS , IR ) = ( HYDVVV( I2 , IR ) + 658
659 397          HYDVVV( I4 , IR ) + 659
660 398          HYDVVV( NV , IR ) ) * THIRD                   660
661 399          92          CONTINUE                          661
662 400          C                                          662

```

663	401	HDUM = 1. / (HYDV(IT2 , 1) + 1.E-12)	663
664	402	HYDV(IT2 , 2) = HYDV(IT2 , 2) * HDUM	664
665	403	HYDV(IT2 , 3) = HYDV(IT2 , 3) * HDUM	665
666	404	HYDV(IT2 , 4) = (HYDV(IT2 , 4) -	666
667	405	.5 * HYDV(IT2 , 1) *	667
668	406	(HYDV(IT2 , 2) * HYDV(IT2 , 2) +	668
669	407	HYDV(IT2 , 3) * HYDV(IT2 , 3)) *	669
670	408	(HYDV(IT2 , 5) - 1.)	670
671	409	C	671
672	410	HDUM = 1. / (HYDV(NS , 1) + 1.E-12)	672
673	411	HYDV(NS , 2) = HYDV(NS , 2) * HDUM	673
674	412	HYDV(NS , 3) = HYDV(NS , 3) * HDUM	674
675	413	HYDV(NS , 4) = (HYDV(NS , 4) -	675
676	414	.5 * HYDV(NS , 1) *	676
677	415	(HYDV(NS , 2) * HYDV(NS , 2) +	677
678	416	HYDV(NS , 3) * HYDV(NS , 3)) *	678
679	417	(HYDV(NS , 5) - 1.)	679
680	418	C	680
681	419	XSYREA = 1. / XSAREA	681
682	420	SAREA(IT2) = XSYREA	682
683	421	SAREA(NS) = XSYREA	683
684	422	C	684
685	423	DO 122 IR = 1 , 2	685
686	424	RGRAD(NS , IR) = RGRAD(IT2 , IR)	686
687	425	UGRAD(NS , IR) = UGRAD(IT2 , IR)	687
688	426	VGRAD(NS , IR) = VGRAD(IT2 , IR)	688
689	427	PGRAD(NS , IR) = PGRAD(IT2 , IR)	689
690	428	122 CONTINUE	690
691	429	C	691
692	430	KSDDEL(NS) = IDUMP	692
693	431	KSDDEL(IT2) = IDUMP	693
694	432	C	694
695	433	JEN(1) = NAA	695
696	434	JEN(2) = NDD	696
697	435	JEN(3) = N2	697
698	436	JEN(4) = N3	698
699	437	JEN(5) = N	699
700	438	JENN = 5	700
701	439	C	701
702	440	END IF	702
703	441	C	703
704	442	DO 90 IENN = 1 , JENN	704
705	443	IEN = JEN(IENN)	705
706	444	JV1 = JE(1 , IEN)	706
707	445	JV2 = JE(2 , IEN)	707
708	446	AX = XV(1 , JV2) - XV(1 , JV1)	708
709	447	AY = XV(2 , JV2) - XV(2 , JV1)	709
710	448	XE(1 , IEN) = SQRT(AX * AX + AY * AY)	710
711	449	XEREV = 1. / XE(1 , IEN)	711
712	450	XN(IEN) = AY * XEREV	712
713	451	YN(IEN) = - AX * XEREV	713
714	452	ISSR = JE(4 , IEN)	714
715	453	ISSL = JE(3 , IEN)	715
716	454	C	716
717	455	IJE5 = JE(5 , IEN)	717
718	456	IF(IJE5 . NE . 0) THEN	718
719	457	C	719
720	458	AA = XV(1 , JV2) - XV(1 , JV1)	720
721	459	BB = XV(2 , JV2) - XV(2 , JV1)	721
722	460	XEL = XS(1 , ISSL)	722
723	461	YEL = XS(2 , ISSL)	723
724	462	CC = XEL - XV(1 , JV1)	724
725	463	DD = YEL - XV(2 , JV1)	725
726	464	EE = (AA * CC + BB * DD) * XEREV * XEREV	726
727	465	XER = XV(1 , JV1) + AA * EE	727
728	466	YER = XV(2 , JV1) + BB * EE	728
729	467	AX = XER - XEL	729
730	468	AY = YER - YEL	730
731	469	XE(2 , IEN) = SQRT(AX * AX + AY * AY)	731
732	470	XEREV = 1. / XE(2 , IEN)	732
733	471	XXN(IEN) = AX * XEREV	733
734	472	YYN(IEN) = AY * XEREV	734
735	473	XE(2 , IEN) = 2. * XE(2 , IEN)	735
736	474	XYMIDL(IEN) = .5	736

```

737 475      XMIDL( IEN ) = XER
738 476      YMIDL( IEN ) = YER
739 477      C
740 478      ELSE
741 479      C
742 480      XER = XS( 1 , ISSR )
743 481      YER = XS( 2 , ISSR )
744 482      XEL = XS( 1 , ISSL )
745 483      YEL = XS( 2 , ISSL )
746 484      C
747 485      AA = XV( 1 , JV2 ) - XV( 1 , JVI )
748 486      BB = XV( 2 , JV2 ) - XV( 2 , JVI )
749 487      CC = XEL - XER
750 488      DD = YEL - YER
751 489      ACA = XER - XV( 1 , JVI )
752 490      DBD = YER - XV( 2 , JVI )
753 491      EE = ( ACA * DD - DBD * CC ) / ( AA * DD - BB * CC )
754 492      XMIDL( IEN ) = XV( 1 , JVI ) + AA * EE
755 493      YMIDL( IEN ) = XV( 2 , JVI ) + BB * EE
756 494      C
757 495      XEMID = XMIDL( IEN ) - XEL
758 496      YEMID = YMIDL( IEN ) - YEL
759 497      C
760 498      AX = XER - XEL
761 499      AY = YER - YEL
762 500      XE( 2 , IEN ) = SQRT( AX * AX + AY * AY )
763 501      XEREV = 1. / XE( 2 , IEN )
764 502      XXN( IEN ) = AX * XEREV
765 503      YYN( IEN ) = AY * XEREV
766 504      C
767 505      XYMIDL( IEN ) = SQRT( XEMID * XEMID + YEMID * YEMID ) * XEREV
768 506      C
769 507      END IF
770 508      C
771 509      90 CONTINUE
772 510      C
773 511      IF( IEROR.NE.0 ) THEN
774 512          WRITE (6,1000) N
775 513          IF( IEROR.EQ.2 ) WRITE (6,1002) I2, J1, IT1, I5
776 514          IF( IEROR.EQ.3 ) WRITE (6,1003) I2, J2, IT2, I5
777 515          STOP
778 516      END IF
779 517      C
780 518      C --- EXIT POINT FROM SUBROUTINE -----
781 519      C
782 520      C -----
783 521      RETURN
784 522      C -----
785 523      C
786 524      C --- FORMATS -----
787 525      C
788 526      1000  FORMAT(//O TROUBLE WITH BOOKKEEPING DATA FOUND IN DISECT ',
789 527          'FOR LINE ', I5//)
790 528      1002  FORMAT(' DISECT I2= ',I5,' J= ',I5,' IT1= ',I5,' I5= ',I5)
791 529      1003  FORMAT(' DISECT I2= ',I5,' J= ',I5,' IT2= ',I5,' I5= ',I5)
792 530      C
793 531      C ---
794 532      END

```

```

795      1      SUBROUTINE DYNPTN( DAREA , NOFDIV , IDUMP , LTRIG )
796      2      C
797      3      C-----I
798      4      C
799      5      C      DYNPTN ADAPT THE GRID DYNAMICALLY, ADD VERTECES
800      6      C
801      7      C-----I
802      8      C
803      9      C      IMPLICIT REAL (A-H,O-Z)
804     10      C
805     11      include 'cmsh00.h'
806     12      include 'chyd00.h'
807     13      include 'cint00.h'
808     14      include 'cphs10.h'
809     15      include 'cphs20.h'
810     16      C
811     17      INTEGER JTRIG(MEM),KTRIG(MEM),IRECNC(MEM)
812     18      INTEGER JSE(MEM),JEE(MEM),IOFDVS(10),NOFDVS(10)
813     19      C
814     20      EQUIVALENCE (UL,JTRIG)
815     21      EQUIVALENCE (VR,KTRIG)
816     22      EQUIVALENCE (VL,IRECNC)
817     23      EQUIVALENCE (PR,JSE)
818     24      EQUIVALENCE (PL,JEE)
819     25      C
820     26      SMINVG = SAREVG * DAREA
821     27      RMINVG = .7 * SMINVG
822     28      DO 115 IS = 1 , NS
823     29      JEE( IS ) = 0
824     30      115 CONTINUE
825     31      C
826     32      NSS = 0
827     33      FLUXPP = .00001 * HYDMOM( 4 )
828     34      FLUXUU = .00001 * HYDMOM( 2 )
829     35      FLUXRR = .00001 * HYDMOM( 1 )
830     36      DO 120 IS = 1 , NS
831     37      PCRTRY = HYDFLX( IS , 4 ) - FLUXPP
832     38      IPCRTR = SIGN( 1. , PCRTRY )
833     39      UCRTRY = HYDFLX( IS , 2 ) - FLUXUU
834     40      IUCRTR = SIGN( 1. , UCRTRY )
835     41      RCRTRY = HYDFLX( IS , 1 ) - FLUXRR
836     42      IRCRTR = SIGN( 1. , RCRTRY )
837     43      IF( (
838     44      .   IPCRTR . EQ . 1 . OR .
839     45      .   IUCRTR . EQ . 1 . OR .
840     46      .   IRCRTR . EQ . 1 ) . AND .
841     47      .   KSDEL( IS ) . LT . IDUMP ) THEN
842     48      KSDEL( IS ) = IDUMP
843     49      JEE( IS ) = 1
844     50      NSS = NSS + 1
845     51      JTRIG( NSS ) = IS
846     52      END IF
847     53      120 CONTINUE
848     54      C
849     55      DO 130 IS = 1 , NSS
850     56      JSE( IS ) = JTRIG( IS )
851     57      130 CONTINUE
852     58      C
853     59      MSS = NSS
854     60      DO 140 KDIV = 1 , NOFDIV
855     61      ITRIG = 0
856     62      DO 150 KS = 1 , MSS
857     63      C
858     64      ISS = JSE( KS )
859     65      C
860     66      DO 160 KR = 1 , 3
861     67      IVV = JS( KR , ISS )
862     68      C
863     69      IE = JV( 2 , IVV )
864     70      IF( IE . GT . 0 ) THEN
865     71      C
866     72      IV1 = JE( 1 , IE )
867     73      IF( IV1 . EQ . IVV ) THEN
868     74      ISI = JE( 3 , IE )

```


869	75	ELSE	869
870	76	ISI = JE(4 , IE)	870
871	77	END IF	871
872	78	IS = ISI	872
873	79	C	873
874	80	750 CONTINUE	874
875	81	C	875
876	82	JES = JEE(IS)	876
877	83	XAS = XS(3 , IS)	877
878	84	IF(JES . EQ . 0 . AND . XAS . LT . SAREVG) THEN	878
879	85	ITRIG = ITRIG + 1	879
880	86	KTRIG(ITRIG) = IS	880
881	87	KSDEL(IS) = IDUMP	881
882	88	JEE(IS) = 1	882
883	89	END IF	883
884	90	C	884
885	91	DO 760 IR = 1 , 3	885
886	92	JR = MOD(IR , 3) + 1	886
887	93	IEA = IABS(JS(JR + 3 , IS))	887
888	94	IF(IEA . EQ . IE) THEN	888
889	95	JJR = MOD(JR + 1 , 3) + 4	889
890	96	IER = IABS(JS(JJR , IS))	890
891	97	C	891
892	98	IV1 = JE(1 , IER)	892
893	99	IF(IV1 . EQ . IVV) THEN	893
894	100	ISR = JE(3 , IER)	894
895	101	ELSE	895
896	102	ISR = JE(4 , IER)	896
897	103	END IF	897
898	104	END IF	898
899	105	C	899
900	106	760 CONTINUE	900
901	107	C	901
902	108	IF(ISR . NE . ISI) THEN	902
903	109	IS = ISR	903
904	110	IE = IER	904
905	111	GO TO 750	905
906	112	END IF	906
907	113	C	907
908	114	ELSE	908
909	115	C	909
910	116	IE = - IE	910
911	117	IV1 = JE(1 , IE)	911
912	118	IF(IV1 . EQ . IVV) THEN	912
913	119	ISI = JE(3 , IE)	913
914	120	ELSE	914
915	121	ISI = JE(4 , IE)	915
916	122	END IF	916
917	123	IS = ISI	917
918	124	ISI = 0	918
919	125	IEE = IE	919
920	126	C	920
921	127	650 CONTINUE	921
922	128	C	922
923	129	JES = JEE(IS)	923
924	130	XAS = XS(3 , IS)	924
925	131	IF(JES . EQ . 0 . AND . XAS . LT . SAREVG) THEN	925
926	132	ITRIG = ITRIG + 1	926
927	133	KTRIG(ITRIG) = IS	927
928	134	KSDEL(IS) = IDUMP	928
929	135	JEE(IS) = 1	929
930	136	END IF	930
931	137	C	931
932	138	DO 660 IR = 1 , 3	932
933	139	JR = MOD(IR , 3) + 1	933
934	140	IEA = IABS(JS(JR + 3 , IS))	934
935	141	IF(IEA . EQ . IE) THEN	935
936	142	JJR = MOD(JR + 1 , 3) + 4	936
937	143	IER = IABS(JS(JJR , IS))	937
938	144	C	938
939	145	IV1 = JE(1 , IER)	939
940	146	IF(IV1 . EQ . IVV) THEN	940
941	147	ISR = JE(3 , IER)	941
942	148	ELSE	942

```

943 149          ISR = 'E( 4 , IER )                      943
944 150          END IF                                    944
945 151          END IF                                    945
946 152  C                                               946
947 153 660      CONTINUE                                  947
948 154  C                                               948
949 155          IF( ISR . NE . ISI ) THEN                949
950 156          IS = ISR                                  950
951 157          IE = IER                                  951
952 158          GO TO 650                                  952
953 159          END IF                                    953
954 160  C                                               954
955 161          END IF                                    955
956 162 160      CONTINUE                                  956
957 163  C                                               957
958 164 150      CONTINUE                                  958
959 165  C                                               959
960 166          DO 170 IS = 1 , ITRIG                    960
961 167          JTRIG( IS + MSS ) = KTRIG( IS )          961
962 168          JSE( IS ) = KTRIG( IS )                  962
963 169 170      CONTINUE                                  963
964 170          NSS = ITRIG                               964
965 171          MSS = MSS + ITRIG                         965
966 172  C                                               966
967 173 140      CONTINUE                                  967
968 174          NSS = MSS                                 968
969 175  C                                               969
970 176          DO 300 KDIV = 1 , 1                       970
971 177          LTRIG = NSS                               971
972 178  C                                               972
973 179          DO 310 IS = 1 , NSS                       973
974 180          ISS = JTRIG( IS )                         974
975 181          XSAREA = XS( 3 , ISS )                   975
976 182          IF( XSAREA . GE . RMINVG ) THEN          976
977 183  C                                               977
978 184          DO 335 IR = 4 , 6                          978
979 185          IE = IABS( JS( IR , ISS ) )               979
980 186          IJE5 = JE( 5 , IE )                      980
981 187          IF( IJE5 . NE . 0 ) THEN                  981
982 188          JR2 = MOD( IR - 3 , 3 ) + 4              982
983 189          IE2 = IABS( JS( JR2 , ISS ) )            983
984 190          JR3 = MOD( IR - 2 , 3 ) + 4              984
985 191          IE3 = IABS( JS( JR3 , ISS ) )            985
986 192          XE1 = XE( 1 , IE )                      986
987 193          XE2 = XE( 1 , IE2 )                     987
988 194          XE3 = XE( 1 , IE3 )                     988
989 195          XEDIST = 1. / XE1                         989
990 196          YE2 = XE2 * XEDIST                       990
991 197          YE3 = XE3 * XEDIST                       991
992 198          ZE2 = ( YE2 - 1.5 ) * ( YE2 - .1 )        992
993 199          ZE3 = ( YE3 - 1.5 ) * ( YE3 - . )        993
994 200          YY2 = XE1 * XE1 + XE2 * XE2 + .35 * XE1 * XE2 - XE3 * XE3 994
995 201          YY3 = XE1 * XE1 + XE3 * XE3 + .35 * XE1 * XE3 - XE2 * XE2 995
996 202          IF( ZE2 . LT . 0 . AND . ZE3 . LT . 0 . AND . 996
997 203          . YY2 . GT . 0 . AND . YY3 . GT . 0 . ) THEN 997
998 204          CALL DISECT ( IE , IDONE , IDUMP )         998
999 205  C                                               999
1000 206          LTRIG = LTRIG + 1                        1000
1001 207          JTRIG( LTRIG ) = NS                      1001
1002 208          KSDDEL( NS ) = IDUMP                    1002
1003 209  C                                               1003
1004 210          END IF                                    1004
1005 211          END IF                                    1005
1006 212 335      CONTINUE                                  1006
1007 213          END IF                                    1007
1008 214 310      CONTINUE                                  1008
1009 215  C                                               1009
1010 216          NSS = LTRIG                               1010
1011 217          IEDGE = 0                                  1011
1012 218          NCOLOR = 0                                1012
1013 219  C                                               1013
1014 220          DO 295 IE = 1 , NE                        1014
1015 221          JSE( IE ) = 0                            1015
1016 222 295      CONTINUE                                  1016

```

```

1017 223 C
1018 224      DO 320 IS = 1 , NSS
1019 225      ISS = JTRIG( IS )
1020 226      XSAREA = XS( 3 , ISS )
1021 227 C
1022 228      XXS = XS( 1 , ISS )
1023 229      YYS = XS( 2 , ISS )
1024 230      IZZ = 1
1025 231      IF( IWINDOW .EQ. 1 ) THEN
1026 232      XXSS = - XXS * XXS + XXS + .75
1027 233      YYSS = - YYS * YYS + 1.
1028 234      IZZ = INT( SIGN( 1. , XXSS * YYSS ) )
1029 235      END IF
1030 236 C
1031 237      IF( XSAREA .GT. RMINVG .AND. IZZ .EQ. 1 ) THEN
1032 238 C
1033 239      DO 735 IR = 4 , 6
1034 240      IE = IABS( JS( IR , ISS ) )
1035 241      IF( JSE( IE ) .EQ. 0 ) THEN
1036 242      IEDGE = IEDGE + 1
1037 243      IRECNC( IEDGE ) = IE
1038 244      NCOLOR = NCOLOR + 1
1039 245      JEE( NCOLOR ) = IE
1040 246      JSE( IE ) = 1
1041 247      END IF
1042 248 735 CONTINUE
1043 249 C
1044 250      AREAXS = SAREA( ISS )
1045 251      IE1 = IABS( JS( 4 , ISS ) )
1046 252      XE1 = XE( 1 , IE1 )
1047 253      HD1 = AREAXS * XE1 * XE1
1048 254      IJE5 = JE( 5 , IE1 )
1049 255      IE2 = IABS( JS( 5 , ISS ) )
1050 256      XE2 = XE( 1 , IE2 )
1051 257      HD2 = AREAXS * XE2 * XE2
1052 258      IJE5 = IJE5 + JE( 5 , IE2 )
1053 259      IE3 = IABS( JS( 6 , ISS ) )
1054 260      XE3 = XE( 1 , IE3 )
1055 261      HD3 = AREAXS * XE3 * XE3
1056 262      IJE5 = IJE5 + JE( 5 , IE3 )
1057 263      RATIO = AMAX1( HD1 , HD2 , HD3 )
1058 264      IRATIO = 0
1059 265      IF( RATIO .LE. 7. .AND. IJE5 .EQ. 0 .AND.
1060 266      XSAREA .GT. SMINVG ) IRATIO = 1
1061 267      IF( IJE5 .GT. 0 ) IRATIO = 2
1062 268 C
1063 269      IF( IRATIO .EQ. 2 ) THEN
1064 270      IJE51 = JE( 5 , IE1 )
1065 271      IJE52 = JE( 5 , IE2 )
1066 272      IJE53 = JE( 5 , IE3 )
1067 273      IF( IJE51 .NE. 0 ) THEN
1068 274      IEDIST = IE1
1069 275      XE1 = XE( 1 , IE1 )
1070 276      XE2 = XE( 1 , IE2 )
1071 277      XE3 = XE( 1 , IE3 )
1072 278      END IF
1073 279      IF( IJE52 .NE. 0 ) THEN
1074 280      IEDIST = IE2
1075 281      XE1 = XE( 1 , IE2 )
1076 282      XE2 = XE( 1 , IE1 )
1077 283      XE3 = XE( 1 , IE3 )
1078 284      END IF
1079 285      IF( IJE53 .NE. 0 ) THEN
1080 286      IEDIST = IE3
1081 287      XE1 = XE( 1 , IE3 )
1082 288      XE2 = XE( 1 , IE2 )
1083 289      XE3 = XE( 1 , IE1 )
1084 290      END IF
1085 291      XEDIST = 1. / XE( 1 , IEDIST )
1086 292      YE2 = XE2 * XEDIST
1087 293      YE3 = XE3 * XEDIST
1088 294      ZE2 = ( YE2 - 1.5 ) * ( YE2 - .1 )
1089 295      ZE3 = ( YE3 - 1.5 ) * ( YE3 - .1 )
1090 296      YY2 = XE1 * XE1 + XE2 * XE2 + .35 * XE1 * XE2 - XE3 * XE3

```

```

1091 297      YY3 = XE1 * XE1 + XE3 * XE3 + .35 * XE1 * XE2 - XE2 * XE2      1091
1092 298      IF( ZE2 . LT . 0 . AND . ZE3 . LT . 0 . AND .      1092
1093 299      . YZ2 . GT . 0 . AND . YZ3 . GT . 0 . ) THEN      1093
1094 300      CALL DISECT ( IEDIT , IDONE , IDUMP )      1094
1095 301      C      1095
1096 302      LTRIG = LTRIG + 1      1096
1097 303      JTRIG( LTRIG ) = NS      1097
1098 304      KSDEL( NS ) = IDUMP      1098
1099 305      C      1099
1100 306      IEDGE = IEDGE + 1      1100
1101 307      IRECNC( IEDGE ) = NE      1101
1102 308      NCOLOR = NCOLOR + 1      1102
1103 309      JEE( NCOLOR ) = NE      1103
1104 310      JSE( NE ) = 1      1104
1105 311      IEDGE = IEDGE + 1      1105
1106 312      IRECNC( IEDGE ) = NE - 1      1106
1107 313      NCOLOR = NCOLOR + 1      1107
1108 314      JEE( NCOLOR ) = NE - 1      1108
1109 315      JSE( NE - 1 ) = 1      1109
1110 316      C      1110
1111 317      END IF      1111
1112 318      END IF      1112
1113 319      C      1113
1114 320      IF( IRATIO . EQ . 1 ) THEN      1114
1115 321      C      1115
1116 322      CALL VERCEN( ISS )      1116
1117 323      KSDEL( ISS ) = IDUMP      1117
1118 324      LTRIG = LTRIG + 1      1118
1119 325      JTRIG( LTRIG ) = NS - 1      1119
1120 326      KSDEL( NS - 1 ) = IDUMP      1120
1121 327      C      1121
1122 328      LTRIG = LTRIG + 1      1122
1123 329      JTRIG( LTRIG ) = NS      1123
1124 330      KSDEL( NS ) = IDUMP      1124
1125 331      C      1125
1126 332      IEDGE = IEDGE + 1      1126
1127 333      IRECNC( IEDGE ) = NE      1127
1128 334      NCOLOR = NCOLOR + 1      1128
1129 335      JEE( NCOLOR ) = NE      1129
1130 336      JSE( NE ) = 1      1130
1131 337      IEDGE = IEDGE + 1      1131
1132 338      IRECNC( IEDGE ) = NE - 1      1132
1133 339      NCOLOR = NCOLOR + 1      1133
1134 340      JEE( NCOLOR ) = NE - 1      1134
1135 341      JSE( NE - 1 ) = 1      1135
1136 342      IEDGE = IEDGE + 1      1136
1137 343      IRECNC( IEDGE ) = NE - 2      1137
1138 344      NCOLOR = NCOLOR + 1      1138
1139 345      JEE( NCOLOR ) = NE - 2      1139
1140 346      JSE( NE - 2 ) = 1      1140
1141 347      C      1141
1142 348      ELSE      1142
1143 349      C      1143
1144 350      IDISCT = 0      1144
1145 351      DO 545 KK = 4 , 6      1145
1146 352      IEE = JS( KK , ISS )      1146
1147 353      IEF = IABS( IEE )      1147
1148 354      IJES5 = JE( 5 , IEF )      1148
1149 355      IF( IJES5 . EQ . 0 ) THEN      1149
1150 356      IF( IEE . GT . 0 ) THEN      1150
1151 357      ISI = JE( 4 , IEE )      1151
1152 358      ELSE      1152
1153 359      ISI = JE( 3 , IEF )      1153
1154 360      END IF      1154
1155 361      AREAXS = SAREA( ISI )      1155
1156 362      IE1 = IABS( JS( 4 , ISI ) )      1156
1157 363      XE1 = XE( 1 , IE1 )      1157
1158 364      IJES5 = JE( 5 , IE1 )      1158
1159 365      HD1 = AREAXS * XE1 * XE1      1159
1160 366      E2 = IABS( JS( 5 , ISI ) )      1160
1161 367      XE2 = XE( 1 , IE2 )      1161
1162 368      IJES5 = IJES5 + JE( 5 , IE2 )      1162
1163 369      HD2 = AREAXS * XE2 * XE2      1163
1164 370      IE3 = IABS( JS( 6 , ISI ) )      1164

```

```

1165 371      XE3 = XE( 1 , IE3 )
1166 372      IJE55 = IJE55 + JE( 5 , IE3 )
1167 373      HD3 = AREAXS * XE3 * XE3
1168 374      RATIO = AMAX1( HD1 , HD2 , HD3 )
1169 375      YSAREA = XS( 3 , ISI )
1170 376      IF( RATIO .LT. 7. .AND. YSAREA .GT. SMINVG .AND.
1171 377          IJE55 .EQ. 0 ) THEN
1172 378          IDISCT = 1
1173 379          DO 435 IR = 4 , 6
1174 380          IE = IABS( JS( IR , ISI ) )
1175 381          IF( JSE( IE ) .EQ. 0 ) THEN
1176 382          IEDGE = IEDGE + 1
1177 383          IRECNC( IEDGE ) = IE
1178 384          NCOLOR = NCOLOR + 1
1179 385          JEE( NCOLOR ) = IE
1180 386          JSE( IE ) = 1
1181 387          END IF
1182 388      435 CONTINUE
1183 389          CALL VERCEN( ISI )
1184 390          KSDEL( ISI ) = IDUMP
1185 391          LTRIG = LTRIG + 1
1186 392          JTRIG( LTRIG ) = NS - 1
1187 393          KSDEL( NS - 1 ) = IDUMP
1188 394      C
1189 395          LTRIG = LTRIG + 1
1190 396          JTRIG( LTRIG ) = NS
1191 397          KSDEL( NS ) = IDUMP
1192 398      C
1193 399          IEDGE = IEDGE + 1
1194 400          IRECNC( IEDGE ) = NE
1195 401          NCOLOR = NCOLOR + 1
1196 402          JEE( NCOLOR ) = NE
1197 403          JSE( NE ) = 1
1198 404          IEDGE = IEDGE + 1
1199 405          IRECNC( IEDGE ) = NE - 1
1200 406          NCOLOR = NCOLOR + 1
1201 407          JEE( NCOLOR ) = NE - 1
1202 408          JSE( NE - 1 ) = 1
1203 409          IEDGE = IEDGE + 1
1204 410          IRECNC( IEDGE ) = NE - 2
1205 411          NCOLOR = NCOLOR + 1
1206 412          JEE( NCOLOR ) = NE - 2
1207 413          JSE( NE - 2 ) = 1
1208 414          END IF
1209 415          END IF
1210 416      545 CONTINUE
1211 417      C
1212 418          IF( IDISCT .EQ. 0 ) THEN
1213 419          IE1 = IABS( JS( 4 , ISS ) )
1214 420          XE1 = XE( 1 , IE1 )
1215 421          IE2 = IABS( JS( 5 , ISS ) )
1216 422          XE2 = XE( 1 , IE2 )
1217 423          IE3 = IABS( JS( 6 , ISS ) )
1218 424          XE3 = XE( 1 , IE3 )
1219 425          IEDIST = IE1
1220 426          XEDIST = XE1
1221 427          IF( XE2 .GT. XEDIST ) THEN
1222 428          XEDIST = XE2
1223 429          IEDIST = IE2
1224 430          END IF
1225 431          IF( XE3 .GT. XEDIST ) THEN
1226 432          XEDIST = XE3
1227 433          IEDIST = IE3
1228 434          END IF
1229 435          ISL = JE( 3 , IEDIST )
1230 436          ISR = JE( 4 , IEDIST )
1231 437          XSISL = XS( 3 , ISL )
1232 438          XSISR = XS( 3 , ISR )
1233 439          IJE5 = JE( 5 , IEDIST )
1234 440          IF( XSISL .GT. RMINVG .AND. XSISR .GT. RMINVG .AND.
1235 441          IJE5 .EQ. 0 .AND. IRATIO .NE. 2 ) THEN
1236 442          IF( ISS .NE. ISL ) THEN
1237 443          DO 345 IR = 4 , 6
1238 444          IE = IABS( JS( IR , ISL ) )

```

```

1239 445      IF( JSE( IE ) . EQ . 0 ) THEN
1240 446      IEDGE = IEDGE + 1
1241 447      IRECNC( IEDGE ) = IE
1242 448      NCOLOR = NCOLOR + 1
1243 449      JEE( NCOLOR ) = IE
1244 450      JSE( IE ) = 1
1245 451      END IF
1246 452 345     CONTINUE
1247 453      END IF
1248 454 C
1249 455      IF( ISS . NE . ISR ) THEN
1250 456      DO 355 IR = 4 , 6
1251 457      IE = IABS( JS( IR , ISR ) )
1252 458      IF( JSE( IE ) . EQ . 0 ) THEN
1253 459      IEDGE = IEDGE + 1
1254 460      IRECNC( IEDGE ) = IE
1255 461      NCOLOR = NCOLOR + 1
1256 462      JEE( NCOLOR ) = IE
1257 463      JSE( IE ) = 1
1258 464      END IF
1259 465 355     CONTINUE
1260 466      END IF
1261 467 C
1262 468      IDONE = 0
1263 469      CALL DISECT ( IEDIST , IDONE , IDUMP )
1264 470      IF( IDONE . EQ . 1 ) THEN
1265 471 C
1266 472      LTRIG = LTRIG + 1
1267 473      JTRIG( LTRIG ) = NS
1268 474      KSDDEL( NS ) = IDUMP
1269 475      LTRIG = LTRIG + 1
1270 476      JTRIG( LTRIG ) = NS - 1
1271 477      KSDDEL( NS - 1 ) = IDUMP
1272 478 C
1273 479      IEDGE = IEDGE + 1
1274 480      IRECNC( IEDGE ) = NE
1275 481      NCOLOR = NCOLOR + 1
1276 482      JEE( NCOLOR ) = NE
1277 483      JSE( NE ) = 1
1278 484      IEDGE = IEDGE + 1
1279 485      IRECNC( IEDGE ) = NE - 1
1280 486      NCOLOR = NCOLOR + 1
1281 487      JEE( NCOLOR ) = NE - 1
1282 488      JSE( NE - 1 ) = 1
1283 489      IEDGE = IEDGE + 1
1284 490      IRECNC( IEDGE ) = NE - 2
1285 491      NCOLOR = NCOLOR + 1
1286 492      JEE( NCOLOR ) = NE - 2
1287 493      JSE( NE - 2 ) = 1
1288 494      END IF
1289 495 C
1290 496      END IF
1291 497      END IF
1292 498      END IF
1293 499      END IF
1294 500 C
1295 501 320     CONTINUE
1296 502 C
1297 503      DO 340 IEM = 1 , NCOLOR
1298 504      IE = JEE( IEM )
1299 505 C
1300 506      ISL = JE( 3 , IE )
1301 507      YSAREA = XS( 3 , ISL )
1302 508      IJE5 = JE( 5 , IE )
1303 509      IF( YSAREA . GE . RMINVG . AND . IJE5 . NE . 0 ) THEN
1304 510      IE1 = IABS( JS( 4 , ISL ) )
1305 511      IE2 = IABS( JS( 5 , ISL ) )
1306 512      IE3 = IABS( JS( 6 , ISL ) )
1307 513      IJE51 = JE( 5 , IE1 )
1308 514      IJE52 = JE( 5 , IE2 )
1309 515      IJE53 = JE( 5 , IE3 )
1310 516      IF( IJE51 . NE . 0 ) THEN
1311 517      IEDIST = IE1
1312 518      XE1 = XE( 1 , IE1 )

```

```

1313 519      XE2 = XE( 1 , IE2 )      1313
1314 520      XE3 = XE( 1 , IE3 )      1314
1315 521      END IF                    1315
1316 522      IF( IJES2 . NE . 0 ) THEN  1316
1317 523      IEDIST = IE2              1317
1318 524      XE1 = XE( 1 , IE2 )      1318
1319 525      XE2 = XE( 1 , IE1 )      1319
1320 526      XE3 = XE( 1 , IE3 )      1320
1321 527      END IF                    1321
1322 528      IF( IJES3 . NE . 0 ) THEN  1322
1323 529      IEDIST = IE3              1323
1324 530      XE1 = XE( 1 , IE3 )      1324
1325 531      XE2 = XE( 1 , IE2 )      1325
1326 532      XE3 = XE( 1 , IE1 )      1326
1327 533      END IF                    1327
1328 534      XEDIST = 1. / XE( 1 , IEDIST ) 1328
1329 535      YE2 = XE2 * XEDIST         1329
1330 536      YE3 = XE3 * XEDIST         1330
1331 537      ZE2 = ( YE2 - 1.5 ) * ( YE2 - .1 ) 1331
1332 538      ZE3 = ( YE3 - 1.5 ) * ( YE3 - .1 ) 1332
1333 539      YY2 = XE1 * XE1 + XE2 * XE2 + .35 * XE1 * XE2 - XE3 * XE3 1333
1334 540      YY3 = XE1 * XE1 + XE3 * XE3 + .35 * XE1 * XE3 - XE2 * XE2 1334
1335 541      IF( ZE2 . LT . .0 . AND . ZE3 . LT . 0 . . AND . 1335
1336 542      . YY2 . GT . 0 . . AND . YY3 . GT . 0 . ) THEN 1336
1337 543      CALL DISECT ( IEDIST , IDONE , IDUMP ) 1337
1338 544      C                            1338
1339 545      LTRIG = LTRIG + 1            1339
1340 546      JTRIG( LTRIG ) = NS        1340
1341 547      KSDDEL( NS ) = IDUMP       1341
1342 548      C                            1342
1343 549      IEDGE = IEDGE + 1          1343
1344 550      IRECNC( IEDGE ) = NE       1344
1345 551      NCOLOR = NCOLOR + 1       1345
1346 552      JEE( NCOLOR ) = NE        1346
1347 553      JSE( NE ) = 1              1347
1348 554      IEDGE = IEDGE + 1          1348
1349 555      IRECNC( IEDGE ) = NE - 1   1349
1350 556      NCOLOR = NCOLOR + 1       1350
1351 557      JEE( NCOLOR ) = NE - 1    1351
1352 558      JSE( NE - 1 ) = 1          1352
1353 559      C                            1353
1354 560      ELSE                        1354
1355 561      C                            1355
1356 562      IEDIST = IE1              1356
1357 563      XEDIST = XE1              1357
1358 564      IF( XE2 . GT . XEDIST ) THEN 1358
1359 565      XEDIST = XE2              1359
1360 566      IEDIST = IE2              1360
1361 567      END IF                    1361
1362 568      IF( XE3 . GT . XEDIST ) THEN 1362
1363 569      XEDIST = XE3              1363
1364 570      IEDIST = IE3              1364
1365 571      END IF                    1365
1366 572      ISL = JE( 3 , IEDIST )     1366
1367 573      ISR = JE( 4 , IEDIST )     1367
1368 574      XSISL = XS( 3 , ISL )      1368
1369 575      XSISR = XS( 3 , ISR )      1369
1370 576      IJES = JE( 5 , IEDIST )   1370
1371 577      IF( XSISL . GT . RMINVG . AND . XSISR . GT . RMINVG . AND . 1371
1372 578      . IJES . EQ . 0 ) THEN      1372
1373 579      DO 645 IR = 4 , 6           1373
1374 580      IE = IABS( JS( IR , ISL ) ) 1374
1375 581      IF( JSE( IE ) . EQ . 0 ) THEN 1375
1376 582      IEDGE = IEDGE + 1          1376
1377 583      IRECNC( IEDGE ) = IE       1377
1378 584      NCOLOR = NCOLOR + 1       1378
1379 585      JEE( NCOLOR ) = IE         1379
1380 586      JSE( IE ) = 1              1380
1381 587      END IF                    1381
1382 588      645 CONTINUE              1382
1383 589      DO 655 IR = 4 , 6           1383
1384 590      IE = IABS( JS( IR , ISR ) ) 1384
1385 591      IF( JSE( IE ) . EQ . 0 ) THEN 1385
1386 592      IEDGE = IEDGE + 1          1386

```

1387	593		IRECNC(IEDGE) = IE	1387
1388	594		NCOLOR = NCOLOR + 1	1388
1389	595		JEE(NCOLOR) = IE	1389
1390	596		JSE(IE) = 1	1390
1391	597		END IF	1391
1392	598	655	CONTINUE	1392
1393	599	C		1393
1394	600		IDONE = 0	1394
1395	601		CALL DISECT (IEDIST , IDONE , IDUMP)	1395
1396	602		IF(IDONE .EQ. 1) THEN	1396
1397	603	C		1397
1398	604		LTRIG = LTRIG + 1	1398
1399	605		JTRIG(LTRIG) = NS	1399
1400	606		KDELTA(NS) = IDUMP	1400
1401	607		LTRIG = LTRIG + 1	1401
1402	608		JTRIG(LTRIG) = NS - 1	1402
1403	609		KDELTA(NS - 1) = IDUMP	1403
1404	610	C		1404
1405	611		IEDGE = IEDGE + 1	1405
1406	612		IRECNC(IEDGE) = NE	1406
1407	613		NCOLOR = NCOLOR + 1	1407
1408	614		JEE(NCOLOR) = NE	1408
1409	615		JSE(NE) = 1	1409
1410	616		IEDGE = IEDGE + 1	1410
1411	617		IRECNC(IEDGE) = NE - 1	1411
1412	618		NCOLOR = NCOLOR + 1	1412
1413	619		JEE(NCOLOR) = NE - 1	1413
1414	620		JSE(NE - 1) = 1	1414
1415	621		IEDGE = IEDGE + 1	1415
1416	622		IRECNC(IEDGE) = NE - 2	1416
1417	623		NCOLOR = NCOLOR + 1	1417
1418	624		JEE(NCOLOR) = NE - 2	1418
1419	625		JSE(NE - 2) = 1	1419
1420	626		END IF	1420
1421	627	C		1421
1422	628		END IF	1422
1423	629		END IF	1423
1424	630		END IF	1424
1425	631	340	CONTINUE	1425
1426	632	C		1426
1427	633		NSS = LTRIG	1427
1428	634	C		1428
1429	635		DO 370 IEM = 1 , NCOLOR	1429
1430	636		IE = JEE(IEM)	1430
1431	637		CALL RECNC(IE , IDONE , ITL , ITR , JA , JB , JC , JD)	1431
1432	638		CALL RECNC(JA , JADONE , ITL , ITR , JAA , JAB , JAC , JAD)	1432
1433	639		CALL RECNC(JB , JBDONE , ITL , ITR , JBA , JBB , JBC , JBD)	1433
1434	640		CALL RECNC(JC , JCDONE , ITL , ITR , JCA , JCB , JCC , JCD)	1434
1435	641		CALL RECNC(JD , JDDONE , ITL , ITR , JDA , JDB , JDC , JDD)	1435
1436	642	370	CONTINUE	1436
1437	643	C		1437
1438	644	300	CONTINUE	1438
1439	645	C		1439
1440	646		NVECE = NE / MBL	1440
1441	647		NREME = NE - NVECE * MBL	1441
1442	648		NVECS = NS / MBL	1442
1443	649		NREMS = NS - NVECS * MBL	1443
1444	650		NVECV = NV / MBL	1444
1445	651		NREMV = NV - NVECV * MBL	1445
1446	652	C		1446
1447	653		DO 400 INE = 1 , NVECE	1447
1448	654		NOFVEE(INE) = MBL	1448
1449	655	400	CONTINUE	1449
1450	656		NVEEE = NVECE	1450
1451	657		IF(NREME .GT. 0) THEN	1451
1452	658		NVEEE = NVECE + 1	1452
1453	659		NOFVEE(NVEEE) = NREME	1453
1454	660		END IF	1454
1455	661	C		1455
1456	662		DO 410 INS = 1 , NVECS	1456
1457	663		NOFVES(INS) = MBL	1457
1458	664	410	CONTINUE	1458
1459	665		NVEES = NVECS	1459
1460	666		IF(NREMS .GT. 0) THEN	1460


```

1461 667 NVEES = NVECS + 1 1461
1462 668 NOFVES( NVEES ) = NREMS 1462
1463 669 END IF 1463
1464 670 C 1464
1465 671 DO 420 INV = 1 , NVECV 1465
1466 672 NOFVEV( INV ) = MBL 1466
1467 673 420 CONTINUE 1467
1468 674 NVEEV = NVECV 1468
1469 675 IF( NREMV . GT . 0 ) THEN 1469
1470 676 NVEEV = NVECV + 1 1470
1471 677 NOFVEV( NVEEV ) = NREMV 1471
1472 678 END IF 1472
1473 679 C 1473
1474 680 PRINT*,NV,NE,NS 1474
1475 681 C 1475
1476 682 C --- EXIT POINT FROM SUBROUTINE ----- 1476
1477 683 C 1477
1478 684 C 1478
1479 685 RETURN 1479
1480 686 C ----- 1480
1481 687 C 1481
1482 688 C --- 1482
1483 689 END 1483

```

```

1484 1 SUBROUTINE DYYPTN( DAREA , NOFDIV , IOUMP , LTRIG ) 1484
1485 2 C 1485
1486 3 C-----I 1486
1487 4 C I 1487
1488 5 C DYYPTN ADAPT THE GRID DYNAMICALLY, ADD VERTECES I 1488
1489 6 C SUB DIVIDE THE TRIANGLE THAT WERE FLAGED IN DYNPTN I 1489
1490 7 C I 1490
1491 8 C-----I 1491
1492 9 C 1492
1493 10 IMPLICIT REAL (A-H,O-Z) 1493
1494 11 C 1494
1495 12 include 'cmsh00.h' 1495
1496 13 include 'chyd00.h' 1496
1497 14 include 'cint00.h' 1497
1498 15 include 'cphs10.h' 1498
1499 16 include 'cphs20.h' 1499
1500 17 C 1500
1501 18 INTEGER JTRIG(MEM),KTRIG(MEM),IRECNC(MEM) 1501
1502 19 INTEGER JSE(MEM),JEE(MEM),IOFDVS(10),NOFDVS(10) 1502
1503 20 C 1503
1504 21 EQUIVALENCE (UL,JTRIG) 1504
1505 22 EQUIVALENCE (VR,KTRIG) 1505
1506 23 EQUIVALENCE (VL,IRECNC) 1506
1507 24 EQUIVALENCE (PR,JSE) 1507
1508 25 EQUIVALENCE (PL,JEE) 1508
1509 26 C 1509
1510 27 SMINVG = SAREVG * DAREA 1510
1511 28 AMINVG = SAREVG * THIRO 1511
1512 29 RMINVG = .7 * SMINVG 1512
1513 30 DO 115 IS = 1 , NS 1513
1514 31 JEE( IS ) = 0 1514
1515 32 115 CONTINUE 1515
1516 33 MSS = 0 1516
1517 34 NSS = LTRIG 1517
1518 35 C 1518
1519 36 DO 140 KDIV = 1 , NOFDIV 1519
1520 37 ITRIG = 0 1520
1521 38 DO 150 KS = 1 , NSS 1521
1522 39 C 1522
1523 40 ISS = JTRIG( KS ) 1523
1524 41 IF( ISS . NE . 0 ) THEN 1524
1525 42 C 1525
1526 43 DO 160 KR = 1 , 3 1526
1527 44 IVV = JS( KR , ISS ) 1527
1528 45 C 1528
1529 46 IE = JV( 2 , IVV ) 1529
1530 47 IF( IE . GT . 0 ) THEN 1530
1531 48 C 1531

```

1532	49	IV1 = JE(1 , IE)	1532
1533	50	IF(IV1 . EQ . IVV) THEN	1533
1534	51	ISI = JE(3 , IE)	1534
1535	52	ELSE	1535
1536	53	ISI = JE(4 , IE)	1536
1537	54	END IF	1537
1538	55	IS = ISI	1538
1539	56	C	1539
1540	57	750 CONTINUE	1540
1541	58	C	1541
1542	59	JES = JEE(IS)	1542
1543	60	XAS = XS(3 , IS)	1543
1544	61	IF(JES . EQ . 0 . AND . XAS . GT . RMINVG) THEN	1544
1545	62	ITRIG = ITRIG + 1	1545
1546	63	KTRIG(ITRIG) = IS	1546
1547	64	KSDLT(IS) = IDUMP	1547
1548	65	JEE(IS) = 1	1548
1549	66	END IF	1549
1550	67	C	1550
1551	68	DO 760 IR = 1 , 3	1551
1552	69	JR = MOD(IR , 3) + 1	1552
1553	70	IEA = IABS(JS(JR + 3 , IS))	1553
1554	71	IF(IEA . EQ . IE) THEN	1554
1555	72	JJR = MOD(JR + 1 , 3) + 4	1555
1556	73	IER = IABS(JS(JJR , IS))	1556
1557	74	C	1557
1558	75	IV1 = JE(1 , IER)	1558
1559	76	IF(IV1 . EQ . IVV) THEN	1559
1560	77	ISR = JE(3 , IER)	1560
1561	78	ELSE	1561
1562	79	ISR = JE(4 , IER)	1562
1563	80	END IF	1563
1564	81	END IF	1564
1565	82	C	1565
1566	83	760 CONTINUE	1566
1567	84	C	1567
1568	85	IF(ISR . NE . ISI) THEN	1568
1569	86	IS = ISR	1569
1570	87	IE = IER	1570
1571	88	GO TO 750	1571
1572	89	END IF	1572
1573	90	C	1573
1574	91	ELSE	1574
1575	92	C	1575
1576	93	IE = - IE	1576
1577	94	IV1 = JE(1 , IE)	1577
1578	95	IF(IV1 . EQ . IVV) THEN	1578
1579	96	ISI = JE(3 , IE)	1579
1580	97	ELSE	1580
1581	98	ISI = JE(4 , IE)	1581
1582	99	END IF	1582
1583	100	IS = ISI	1583
1584	101	ISI = 0	1584
1585	102	IIE = IE	1585
1586	103	C	1586
1587	104	650 CONTINUE	1587
1588	105	C	1588
1589	106	JES = JEE(IS)	1589
1590	107	XAS = XS(3 , IS)	1590
1591	108	IF(JES . EQ . 0 . AND . XAS . GT . RMINVG) THEN	1591
1592	109	ITRIG = ITRIG + 1	1592
1593	110	KTRIG(ITRIG) = IS	1593
1594	111	KSDLT(IS) = IDUMP	1594
1595	112	JEE(IS) = 1	1595
1596	113	END IF	1596
1597	114	C	1597
1598	115	DO 660 IR = 1 , 3	1598
1599	116	JR = MOD(IR , 3) + 1	1599
1600	117	IEA = IABS(JS(JR + 3 , IS))	1600
1601	118	IF(IEA . EQ . IE) THEN	1601
1602	119	JJR = MOD(JR + 1 , 3) + 4	1602
1603	120	IER = IABS(JS(JJR , IS))	1603
1604	121	C	1604
1605	122	IV1 = JE(1 , IER)	1605

```

1606 123      IF( IV1 . EQ . IVV ) THEN      1606
1607 124      ISR = JE( 3 , IER )          1607
1608 125      ELSE                          1608
1609 126      ISR = JE( 4 , IER )          1609
1610 127      END IF                        1610
1611 128      END IF                        1611
1612 129      C                             1612
1613 130      660      CONTINUE              1613
1614 131      C                             1614
1615 132      IF( ISR . NE . ISI ) THEN     1615
1616 133      IS = ISR                       1616
1617 134      IE = IER                     1617
1618 135      GO TO 650                     1618
1619 136      END IF                        1619
1620 137      C                             1620
1621 138      END IF                        1621
1622 139      160      CONTINUE              1622
1623 140      C                             1623
1624 141      END IF                        1624
1625 142      150      CONTINUE              1625
1626 143      C                             1626
1627 144      DO 170 IS = 1 , ITRIG         1627
1628 145      JTRIG( IS + MSS ) = KTRIG( IS ) 1628
1629 146      170      CONTINUE              1629
1630 147      NSS = ITRIG                   1630
1631 148      MSS = MSS + ITRIG             1631
1632 149      C                             1632
1633 150      140      CONTINUE              1633
1634 151      NSS = MSS                     1634
1635 152      C                             1635
1636 153      DO 300 KDIV = 1 , 1           1636
1637 154      LTRIG = NSS                     1637
1638 155      C                             1638
1639 156      DO 310 IS = 1 , NSS           1639
1640 157      ISS = JTRIG( IS )              1640
1641 158      XSAREA = XS( 3 , ISS )        1641
1642 159      IF( XSAREA . GE . RMINVG ) THEN 1642
1643 160      C                             1643
1644 161      DO 335 IR = 4 , 6             1644
1645 162      IE = IABS( JS( IR , ISS ) )    1645
1646 163      IJE5 = JE( 5 , IE )           1646
1647 164      IF( IJE5 . NE . 0 ) THEN     1647
1648 165      JR2 = MOD( IR - 3 , 3 ) + 4    1648
1649 166      IE2 = IABS( JS( JR2 , ISS ) ) 1649
1650 167      JR3 = MOD( IR - 2 , 3 ) + 4    1650
1651 168      IE3 = IABS( JS( JR3 , ISS ) ) 1651
1652 169      XE1 = XE( 1 , IE )            1652
1653 170      XE2 = XE( 1 , IE2 )           1653
1654 171      XE3 = XE( 1 , IE3 )           1654
1655 172      XEDIST = 1. / XE1             1655
1656 173      YE2 = XE2 * XEDIST             1656
1657 174      YE3 = XE3 * XEDIST             1657
1658 175      ZE2 = ( YE2 - 1.5 ) * ( YE2 - .1 ) 1658
1659 176      ZE3 = ( YE3 - 1.5 ) * ( YE3 - .1 ) 1659
1660 177      YY2 = XE1 * XE1 + XE2 * XE2 + .35 * XE1 * XE2 - XE3 * XE3 1660
1661 178      YY3 = XE1 * XE1 + XE3 * XE3 + .35 * XE1 * XE3 - XE2 * XE2 1661
1662 179      IF( ZE2 . LT . 0 . AND . ZE3 . LT . 0 . AND . 1662
1663 180      YY2 . GT . 0 . AND . YY3 . GT . 0 . ) THEN 1663
1664 181      CALL DISECT ( IE , IDONE , IDUMP ) 1664
1665 182      C                             1665
1666 183      LTRIG = LTRIG + 1              1666
1667 184      JTRIG( LTRIG ) = NS           1667
1668 185      KSDELTA( NS ) = IDUMP         1668
1669 186      C                             1669
1670 187      END IF                        1670
1671 188      END IF                        1671
1672 189      335      CONTINUE              1672
1673 190      END IF                        1673
1674 191      310      CONTINUE              1674
1675 192      C                             1675
1676 193      NSS = LTRIG                     1676
1677 194      IEDGE = 0                       1677
1678 195      NCOLOR = 0                     1678
1679 196      C                             1679

```

```

1680 197      DO 295 IE = 1 , NE
1681 198      JSE( IE ) = 0
1682 199      295 CONTINUE
1683 200      C
1684 201      DO 320 IS = 1 , NSS
1685 202      ISS = JTRIG( IS )
1686 203      XSAREA = XS( 3 , ISS )
1687 204      C
1688 205      IF( XSAREA . GT . RMINVG ) THEN
1689 206      C
1690 207      DO 735 IR = 4 , 6
1691 208      IE = IABS( JS( IR , ISS ) )
1692 209      IF( JSE( IE ) . EQ . 0 ) THEN
1693 210      IEDGE = IEDGE + 1
1694 211      IRECNC( IEDGE ) = IE
1695 212      NCOLOR = NCOLOR + 1
1696 213      JEE( NCOLOR ) = IE
1697 214      JSE( IE ) = 1
1698 215      END IF
1699 216      735 CONTINUE
1700 217      C
1701 218      AREAXS = SAREA( ISS )
1702 219      IE1 = IABS( JS( 4 , ISS ) )
1703 220      XE1 = XE( 1 , IE1 )
1704 221      HD1 = AREAXS * XE1 * XE1
1705 222      IJE5 = JE( 5 , IE1 )
1706 223      IE2 = IABS( JS( 5 , ISS ) )
1707 224      XE2 = XE( 1 , IE2 )
1708 225      HD2 = AREAXS * XE2 * XE2
1709 226      IJE5 = IJE5 + JE( 5 , IE2 )
1710 227      IE3 = IABS( JS( 6 , ISS ) )
1711 228      XE3 = XE( 1 , IE3 )
1712 229      HD3 = AREAXS * XE3 * XE3
1713 230      IJE5 = IJE5 + JE( 5 , IE3 )
1714 231      RATIO = AMAX1( HD1 , HD2 , HD3 )
1715 232      IRATIO = 0
1716 233      IF( RATIO . LE . 7. . AND . IJE5 . EQ . 0 . AND .
1717 234      XSAREA . GT . SMINVG ) IRATIO = 1
1718 235      IF( IJE5 . GT . 0 ) IRATIO = 2
1719 236      C
1720 237      IF( IRATIO . EQ . 2 ) THEN
1721 238      IJE51 = JE( 5 , IE1 )
1722 239      IJE52 = JE( 5 , IE2 )
1723 240      IJE53 = JE( 5 , IE3 )
1724 241      IF( IJE51 . NE . 0 ) THEN
1725 242      IEDIST = IE1
1726 243      XE1 = XE( 1 , IE1 )
1727 244      XE2 = XE( 1 , IE2 )
1728 245      XE3 = XE( 1 , IE3 )
1729 246      END IF
1730 247      IF( IJE52 . NE . 0 ) THEN
1731 248      IEDIST = IE2
1732 249      XE1 = XE( 1 , IE2 )
1733 250      XE2 = XE( 1 , IE1 )
1734 251      XE3 = XE( 1 , IE3 )
1735 252      END IF
1736 253      IF( IJE53 . NE . 0 ) THEN
1737 254      IEDIST = IE3
1738 255      XE1 = XE( 1 , IE3 )
1739 256      XE2 = XE( 1 , IE2 )
1740 257      XE3 = XE( 1 , IE1 )
1741 258      END IF
1742 259      XEDIST = 1. / XE( 1 , IEDIST )
1743 260      YE2 = XE2 * XEDIST
1744 261      YE3 = XE3 * XEDIST
1745 262      ZE2 = ( YE2 - 1.5 ) * ( YE2 - .1 )
1746 263      ZE3 = ( YE3 - 1.5 ) * ( YE3 - .1 )
1747 264      YY2 = XE1 * XE1 + XE2 * XE2 + .35 * XE1 * XE2 - XE3 * XE3
1748 265      YY3 = XE1 * XE1 + XE3 * XE3 + .35 * XE1 * XE3 - XE2 * XE2
1749 266      IF( ZE2 . LT . 0 . AND . ZE3 . LT . 0 . AND .
1750 267      YY2 . GT . 0 . AND . YY3 . GT . 0 ) THEN
1751 268      CALL DISECT ( IEDIST , IDONE , IDUMP )
1752 269      C
1753 270      LTRIG = LTRIG + 1

```

1754	271	JTRIG(LTRIG) = NS	1754
1755	272	KSDEL(NS) = IDUMP	1755
1756	273	C	1756
1757	274	IEDGE = IEDGE + 1	1757
1758	275	IRECNC(IEDGE) = NE	1758
1759	276	NCOLOR = NCOLOR + 1	1759
1760	277	JEE(NCOLOR) = NE	1760
1761	278	JSE(NE) = 1	1761
1762	279	IEDGE = IEDGE + 1	1762
1763	280	IRECNC(IEDGE) = NE - 1	1763
1764	281	NCOLOR = NCOLOR + 1	1764
1765	282	JEE(NCOLOR) = NE - 1	1765
1766	283	JSE(NE - 1) = 1	1766
1767	284	C	1767
1768	285	END IF	1768
1769	286	END IF	1769
1770	287	C	1770
1771	288	IF(IRATIO . EQ . 1) THEN	1771
1772	289	C	1772
1773	290	CALL VERCEN(ISS)	1773
1774	291	KSDEL(ISS) = IDUMP	1774
1775	292	LTRIG = LTRIG + 1	1775
1776	293	JTRIG(LTRIG) = NS - 1	1776
1777	294	KSDEL(NS - 1) = IDUMP	1777
1778	295	C	1778
1779	296	LTRIG = LTRIG + 1	1779
1780	297	JTRIG(LTRIG) = NS	1780
1781	298	KSDEL(NS) = IDUMP	1781
1782	299	C	1782
1783	300	IEDGE = IEDGE + 1	1783
1784	301	IRECNC(IEDGE) = NE	1784
1785	302	NCOLOR = NCOLOR + 1	1785
1786	303	JEE(NCOLOR) = NE	1786
1787	304	JSE(NE) = 1	1787
1788	305	IEDGE = IEDGE + 1	1788
1789	306	IRECNC(IEDGE) = NE - 1	1789
1790	307	NCOLOR = NCOLOR + 1	1790
1791	308	JEE(NCOLOR) = NE - 1	1791
1792	309	JSE(NE - 1) = 1	1792
1793	310	IEDGE = IEDGE + 1	1793
1794	311	IRECNC(IEDGE) = NE - 2	1794
1795	312	NCOLOR = NCOLOR + 1	1795
1796	313	JEE(NCOLOR) = NE - 2	1796
1797	314	JSE(NE - 2) = 1	1797
1798	315	C	1798
1799	316	ELSE	1799
1800	317	C	1800
1801	318	IDISCT = 0	1801
1802	319	DO 545 KK = 4 , 6	1802
1803	320	IEE = JS(KK , ISS)	1803
1804	321	IEF = IABS(IEE)	1804
1805	322	IJE55 = JE(5 , IEF)	1805
1806	323	IF(IJE55 . EQ . 0) THEN	1806
1807	324	IF(IEE . GT . 0) THEN	1807
1808	325	ISI = JE(4 , IEE)	1808
1809	326	ELSE	1809
1810	327	ISI = JE(3 , IEF)	1810
1811	328	END IF	1811
1812	329	AREAXS = SAREA(ISI)	1812
1813	330	IE1 = IABS(JS(4 , ISI))	1813
1814	331	XE1 = XE(1 , IE1)	1814
1815	332	IJE55 = JE(5 , IE1)	1815
1816	333	HD1 = AREAXS * XE1 * XE1	1816
1817	334	IE2 = IABS(JS(5 , ISI))	1817
1818	335	XE2 = XE(1 , IE2)	1818
1819	336	IJE55 = IJE55 + JE(5 , IE2)	1819
1820	337	HD2 = AREAXS * XE2 * XE2	1820
1821	338	IE3 = IABS(JS(6 , ISI))	1821
1822	339	XE3 = XE(1 , IE3)	1822
1823	340	IJE55 = IJE55 + JE(5 , IE3)	1823
1824	341	HD3 = AREAXS * XE3 * XE3	1824
1825	342	RATIO = AMAX1(HD1 , HD2 , HD3)	1825
1826	343	YSAREA = XS(3 , ISI)	1826
1827	344	IF(RATIO . LT . 7 . . AND . YSAREA . GT . SMINVG . AND .	1827

```

1828 345      .      IJE55 . EQ . 0 ) THEN      1828
1829 346      IDISCT = 1                        1829
1830 347      DO 435 IR = 4 , 6                  1830
1831 348      IE = IABS( JS( IR , ISI ) )        1831
1832 349      IF( JSE( IE ) . EQ . 0 ) THEN      1832
1833 350      IEDGE = IEDGE + 1                  1833
1834 351      IRECNC( IEDGE ) = IE              1834
1835 352      NCOLOR = NCOLOR + 1               1835
1836 353      JEE( NCOLOR ) = IE                1836
1837 354      JSE( IE ) = 1                      1837
1838 355      END IF                              1838
1839 356      435  CONTINUE                       1839
1840 357      CALL VERCEN( ISI )                  1840
1841 358      KSDDEL( ISI ) = IDUMP               1841
1842 359      LTRIG = LTRIG + 1                  1842
1843 360      JTRIG( LTRIG ) = NS - 1           1843
1844 361      KSDDEL( NS - 1 ) = IDUMP           1844
1845 362      C                                     1845
1846 363      LTRIG = LTRIG + 1                  1846
1847 364      JTRIG( LTRIG ) = NS              1847
1848 365      KSDDEL( NS ) = IDUMP              1848
1849 366      C                                     1849
1850 367      IEDGE = IEDGE + 1                  1850
1851 368      IRECNC( IEDGE ) = NE              1851
1852 369      NCOLOR = NCOLOR + 1               1852
1853 370      JEE( NCOLOR ) = NE                1853
1854 371      JSE( NE ) = 1                      1854
1855 372      IEDGE = IEDGE + 1                  1855
1856 373      IRECNC( IEDGE ) = NE - 1         1856
1857 374      NCOLOR = NCOLOR + 1               1857
1858 375      JEE( NCOLOR ) = NE - 1           1858
1859 376      JSE( NE - 1 ) = 1                 1859
1860 377      IEDGE = IEDGE + 1                  1860
1861 378      IRECNC( IEDGE ) = NE - 2         1861
1862 379      NCOLOR = NCOLOR + 1               1862
1863 380      JEE( NCOLOR ) = NE - 2           1863
1864 381      JSE( NE - 2 ) = 1                 1864
1865 382      END IF                              1865
1866 383      END IF                              1866
1867 384      545  CONTINUE                       1867
1868 385      C                                     1868
1869 386      IF( IDISCT . EQ . 0 ) THEN          1869
1870 387      IE1 = IABS( JS( 4 , ISS ) )        1870
1871 388      XE1 = XE( 1 , IE1 )                1871
1872 389      IE2 = IABS( JS( 5 , ISS ) )        1872
1873 390      XE2 = XE( 1 , IE2 )                1873
1874 391      IE3 = IABS( JS( 6 , ISS ) )        1874
1875 392      XE3 = XE( 1 , IE3 )                1875
1876 393      IEDIST = IE1                      1876
1877 394      XEDIST = XE1                       1877
1878 395      IF( XE2 . GT . XEDIST ) THEN        1878
1879 396      XEDIST = XE2                       1879
1880 397      IEDIST = IE2                       1880
1881 398      END IF                              1881
1882 399      IF( XE3 . GT . XEDIST ) THEN        1882
1883 400      XEDIST = XE3                       1883
1884 401      IEDIST = IE3                       1884
1885 402      END IF                              1885
1886 403      ISL = JE( 3 , IEDIST )              1886
1887 404      ISR = JE( 4 , IEDIST )              1887
1888 405      XSISL = XS( 3 , ISL )               1888
1889 406      XSISR = XS( 3 , ISR )               1889
1890 407      IJE5 = JE( 5 , IEDIST )            1890
1891 408      IF( XSISL . GT . RMINVG . AND . XSISR . GT . RMINVG . AND .  1891
1892 409      . IJE5 . EQ . 0 . AND . IRATIO . NE . 2 ) THEN 1892
1893 410      IF( ISS . NE . ISL ) THEN            1893
1894 411      DO 345 IR = 4 , 6                  1894
1895 412      IE = IABS( JS( IR , ISL ) )        1895
1896 413      IF( JSE( IE ) . EQ . 0 ) THEN      1896
1897 414      IEDGE = IEDGE + 1                  1897
1898 415      IRECNC( IEDGE ) = IE              1898
1899 416      NCOLOR = NCOLOR + 1               1899
1900 417      JEE( NCOLOR ) = IE                1900
1901 418      JSE( IE ) = 1                      1901

```

```

1902 419      END IF
1903 420      345  CONTINUE
1904 421      END IF
1905 422      C
1906 423          IF( ISS . NE . ISR ) THEN
1907 424          DO 355 IR = 4 , 6
1908 425          IE = IABS( JS( IR , ISR ) )
1909 426          IF( JSE( IE ) . EQ . 0 ) THEN
1910 427          IEDGE = IEDGE + 1
1911 428          IRECNC( IEDGE ) = IE
1912 429          NCOLOR = NCOLOR + 1
1913 430          JEE( NCOLOR ) = IE
1914 431          JSE( IE ) = 1
1915 432          END IF
1916 433      355  CONTINUE
1917 434          END IF
1918 435      C
1919 436          IDONE = 0
1920 437          CALL DISECT ( IEDIST , IDONE , IDUMP )
1921 438          IF( IDONE . EQ . 1 ) THEN
1922 439      C
1923 440          LTRIG = LTRIG + 1
1924 441          JTRIG( LTRIG ) = NS
1925 442          KSDDEL( NS ) = IDUMP
1926 443          LTRIG = LTRIG + 1
1927 444          JTRIG( LTRIG ) = NS - 1
1928 445          KSDDEL( NS - 1 ) = IDUMP
1929 446      C
1930 447          IEDGE = IEDGE + 1
1931 448          IRECNC( IEDGE ) = NE
1932 449          NCOLOR = NCOLOR + 1
1933 450          JEE( NCOLOR ) = NE
1934 451          JSE( NE ) = 1
1935 452          IEDGE = IEDGE + 1
1936 453          IRECNC( IEDGE ) = NE - 1
1937 454          NCOLOR = NCOLOR + 1
1938 455          JEE( NCOLOR ) = NE - 1
1939 456          JSE( NE - 1 ) = 1
1940 457          IEDGE = IEDGE + 1
1941 458          IRECNC( IEDGE ) = NE - 2
1942 459          NCOLOR = NCOLOR + 1
1943 460          JEE( NCOLOR ) = NE - 2
1944 461          JSE( NE - 2 ) = 1
1945 462          END IF
1946 463      C
1947 464          END IF
1948 465          END IF
1949 466          END IF
1950 467          END IF
1951 468      C
1952 469      320  CONTINUE
1953 470      C
1954 471          DO 340 IEM = 1 , NCOLOR
1955 472          IE = JEE( IEM )
1956 473      C
1957 474          ISL = JE( 3 , IE )
1958 475          YSAREA = XS( 3 , ISL )
1959 476          IJE5 = JE( 5 , IE )
1960 477          IF( YSAREA . GE . RMINVG . AND . IJE5 . NE . 0 ) THEN
1961 478          IE1 = IABS( JS( 4 , ISL ) )
1962 479          IE2 = IABS( JS( 5 , ISL ) )
1963 480          IE3 = IABS( JS( 6 , ISL ) )
1964 481          IJE51 = JE( 5 , IE1 )
1965 482          IJE52 = JE( 5 , IE2 )
1966 483          IJE53 = JE( 5 , IE3 )
1967 484          IF( IJE51 . NE . 0 ) THEN
1968 485          IEDIST = IE1
1969 486          XE1 = XE( 1 , IE1 )
1970 487          XE2 = XE( 1 , IE2 )
1971 488          XE3 = XE( 1 , IE3 )
1972 489          END IF
1973 490          IF( IJE52 . NE . 0 ) THEN
1974 491          IEDIST = IE2
1975 492          XE1 = XE( 1 , IE2 )

```

```

1976 493      XE2 = XE( 1 , IE1 )      1976
1977 494      XE3 = XF( 1 , IE3 )      1977
1978 495      END IF      1978
1979 496      IF( IJE53 . NE . 0 ) THEN      1979
1980 497      IEDIST = IE3      1980
1981 498      XE1 = XE( 1 , IE3 )      1981
1982 499      XE2 = XE( 1 , IE2 )      1982
1983 500      XE3 = XE( 1 , IE1 )      1983
1984 501      END IF      1984
1985 502      XEDIST = 1. / XE( 1 , IEDIST )      1985
1986 503      YE2 = XE2 * XEDIST      1986
1987 504      YE3 = XE3 * XEDIST      1987
1988 505      ZE2 = ( YE2 - 1.5 ) * ( YE2 - .1 )      1988
1989 506      ZE3 = ( YE3 - 1.5 ) * ( YE3 - .1 )      1989
1990 507      YY2 = XE1 * XE1 + XE2 * XE2 + .35 * XE1 * XE2 - XE3 * XE3      1990
1991 508      YY3 = XE1 * XE1 + XE2 * XE2 + .35 * XE1 * XE3 - XE2 * XE2      1991
1992 509      IF( ZE2 . LT . 0 . AND . ZE3 . LT . 0 . AND .      1992
1993 510      . YY2 . GT . 0 . AND . YY3 . GT . 0 . ) THEN      1993
1994 511      CALL DISECT ( IEDIST , IDONE , IDUMP )      1994
1995 512      C      1995
1996 513      LTRIG = LTRIG + 1      1996
1997 514      JTRIG( LTRIG ) = NS      1997
1998 515      KSDEL( NS ) = IDUMP      1998
1999 516      C      1999
2000 517      IEDGE = IEDGE + 1      2000
2001 518      IRECNC( IEDGE ) = NE      2001
2002 519      NCOLOR = NCOLOR + 1      2002
2003 520      JEE( NCOLOR ) = NE      2003
2004 521      JSE( NE ) = 1      2004
2005 522      IEDGE = IEDGE + 1      2005
2006 523      IRECNC( IEDGE ) = NE - 1      2006
2007 524      NCOLOR = NCOLOR + 1      2007
2008 525      JEE( NCOLOR ) = NE - 1      2008
2009 526      JSE( NE - 1 ) = 1      2009
2010 527      C      2010
2011 528      ELSE      2011
2012 529      C      2012
2013 530      IEDIST = IE1      2013
2014 531      XEDIST = XE1      2014
2015 532      IF( XE2 . GT . XEDIST ) THEN      2015
2016 533      XEDIST = XE2      2016
2017 534      IEDIST = IE2      2017
2018 535      END IF      2018
2019 536      IF( XE3 . GT . XEDIST ) THEN      2019
2020 537      XEDIST = XE3      2020
2021 538      IEDIST = IE3      2021
2022 539      END IF      2022
2023 540      ISL = JE( 3 , IEDIST )      2023
2024 541      ISR = JE( 4 , IEDIST )      2024
2025 542      XSISL = XS( 3 , ISL )      2025
2026 543      XSISR = XS( 3 , ISR )      2026
2027 544      IJE5 = JE( 5 , IEDIST )      2027
2028 545      IF( XSISL . GT . RMINVG . AND . XSISR . GT . RMINVG . AND .      2028
2029 546      . IJE5 . EQ . 0 ) THEN      2029
2030 547      DO 645 IR = 4 , 6      2030
2031 548      IE = IABS( JS( IR , ISL ) )      2031
2032 549      IF( JSE( IE ) . EQ . 0 ) THEN      2032
2033 550      IEDGE = IEDGE + 1      2033
2034 551      IRECNC( IEDGE ) = IE      2034
2035 552      NCOLOR = NCOLOR + 1      2035
2036 553      JEE( NCOLOR ) = IE      2036
2037 554      JSE( IE ) = 1      2037
2038 555      END IF      2038
2039 556      645 CONTINUE      2039
2040 557      DO 655 IR = 4 , 6      2040
2041 558      IE = IABS( JS( IR , ISR ) )      2041
2042 559      IF( JSE( IE ) . EQ . 0 ) THEN      2042
2043 560      IEDGE = IEDGE + 1      2043
2044 561      IRECNC( IEDGE ) = IE      2044
2045 562      NCOLOR = NCOLOR + 1      2045
2046 563      JEE( NCOLOR ) = IE      2046
2047 564      JSE( IE ) = 1      2047
2048 565      END IF      2048
2049 566      655 CONTINUE      2049

```


2050	567	C		2050
2051	568		IDONE = 0	2051
2052	569		CALL DISECT (IEDIST , IDONE , IDUMP)	2052
2053	570		IF(IDONE . EQ . 1) THEN	2053
2054	571	C		2054
2055	572		LTRIG = LTRIG + 1	2055
2056	573		JTRIG(LTRIG) = NS	2056
2057	574		KSDEL(NS) = IDUMP	2057
2058	575		LTRIG = LTRIG + 1	2058
2059	576		JTRIG(LTRIG) = NS - 1	2059
2060	577		KSDEL(NS - 1) = IDUMP	2060
2061	578	C		2061
2062	579		IEDGE = IEDGE + 1	2062
2063	580		IRECNC(IEDGE) = NE	2063
2064	581		NCOLOR = NCOLOR + 1	2064
2065	582		JEE(NCOLOR) = NE	2065
2066	583		JSE(NE) = 1	2066
2067	584		IEDGE = IEDGE + 1	2067
2068	585		IRECNC(IEDGE) = NE - 1	2068
2069	586		NCOLOR = NCOLOR + 1	2069
2070	587		JEE(NCOLOR) = NE - 1	2070
2071	588		JSE(NE - 1) = 1	2071
2072	589		IEDGE = IEDGE + 1	2072
2073	590		IRECNC(IEDGE) = NE - 2	2073
2074	591		NCOLOR = NCOLOR + 1	2074
2075	592		JEE(NCOLOR) = NE - 2	2075
2076	593		JSE(NE - 2) = 1	2076
2077	594		END IF	2077
2078	595	C		2078
2079	596		END IF	2079
2080	597		END IF	2080
2081	598		END IF	2081
2082	599	340	CONTINUE	2082
2083	600	C		2083
2084	601		NSS = LTRIG	2084
2085	602	C		2085
2086	603		DO 370 IEM = 1 , NCOLOR	2086
2087	604		IE = JEE(IEM)	2087
2088	605		CALL RECNC(IE , IDONE , ITL , ITR , JA , JB , JC , JD)	2088
2089	606		CALL RECNC(JA , JADONE , ITL , ITR , JAA , JAB , JAC , JAD)	2089
2090	607		CALL RECNC(JB , JBDONE , ITL , ITR , JBA , JBB , JBC , JBD)	2090
2091	608		CALL RECNC(JC , JCDONE , ITL , ITR , JCA , JCB , JCC , JCD)	2091
2092	609		CALL RECNC(JD , JDDONE , ITL , ITR , JDA , JDB , JDC , JDD)	2092
2093	610	370	CONTINUE	2093
2094	611	C		2094
2095	612	300	CONTINUE	2095
2096	613	C		2096
2097	614		NVECE = NE / MBL	2097
2098	615		NREME = NE - NVECE * MBL	2098
2099	616		NVECS = NS / MBL	2099
2100	617		NREMS = NS - NVECS * MBL	2100
2101	618		NVECV = NV / MBL	2101
2102	619		NREMV = NV - NVECV * MBL	2102
2103	620	C		2103
2104	621		DO 400 INE = 1 , NVECE	2104
2105	622		NOFVEE(INE) = MBL	2105
2106	623	400	CONTINUE	2106
2107	624		NVEEE = NVECE	2107
2108	625		IF(NREME . GT . 0) THEN	2108
2109	626		NVEEE = NVECE + 1	2109
2110	627		NOFVEE(NVEEE) = NREME	2110
2111	628		END IF	2111
2112	629	C		2112
2113	630		DO 410 INS = 1 , NVECS	2113
2114	631		NOFVES(INS) = MBL	2114
2115	632	410	CONTINUE	2115
2116	633		NVEES = NVECS	2116
2117	634		IF(NREMS . GT . 0) THEN	2117
2118	635		NVEES = NVECS + 1	2118
2119	636		NOFVES(NVEES) = NREMS	2119
2120	637		END IF	2120
2121	638	C		2121
2122	639		DO 420 INV = 1 , NVECV	2122
2123	640		NOFVEV(INV) = MBL	2123

```

2124 541 420 CONTINUE 2124
2125 542 NVEEV = NVECV 2125
2126 543 IF( NREMV . GT . 0 ) THEN 2126
2127 544 NVEEV = NVECV + 1 2127
2128 545 NOFVEV( NVEEV ) = NREMV 2128
2129 546 END IF 2129
2130 547 C 2130
2131 548 PRINT*,NV,NE,NS 2131
2132 549 C 2132
2133 550 C --- EXIT POINT FROM SUBROUTINE ----- 2133
2134 551 C 2134
2135 552 C ----- 2135
2136 553 RETURN 2136
2137 554 C ----- 2137
2138 555 C 2138
2139 556 C --- 2139
2140 557 END 2140

```

```

2141 1 SUBROUTINE INTPTN( DAREA , NOFDIV , IDUMP , LTRIG ) 2141
2142 2 C 2142
2143 3 C -----| 2143
2144 4 C | 2144
2145 5 C INTPTN ADAPT THE GRID DYNAMICALLY, ADD VERTECES | 2145
2146 6 C SUB DIVIDE TO REFINE AT THE INITIAL STAGE OF THE SIMULATION | 2146
2147 7 C | 2147
2148 8 C -----| 2148
2149 9 C | 2149
2150 10 IMPLICIT REAL (A-H,O-Z) 2150
2151 11 C 2151
2152 12 include 'cmsh00.h' 2152
2153 13 include 'chyd00.h' 2153
2154 14 include 'cint00.h' 2154
2155 15 include 'cphs10.h' 2155
2156 16 include 'cphs20.h' 2156
2157 17 C 2157
2158 18 INTEGER JTRIG(MEM),KTRIG(MEM),IRECNC(MEM) 2158
2159 19 INTEGER JSE(MEM),JEE(MEM),IOFDVS(10),NOFDVS(10) 2159
2160 20 C 2160
2161 21 EQUIVALENCE (UL,JTRIG) 2161
2162 22 EQUIVALENCE (VR,KTRIG) 2162
2163 23 EQUIVALENCE (VL,IRECNC) 2163
2164 24 EQUIVALENCE (PR,JSE) 2164
2165 25 EQUIVALENCE (PL,JEE) 2165
2166 26 C 2166
2167 27 SMINVG = SAREVG * DAREA 2167
2168 28 RMINVG = .7 * SMINVG 2168
2169 29 C 2169
2170 30 DO 115 IS = 1 , NS 2170
2171 31 JEE( IS ) = 0 2171
2172 32 115 CONTINUE 2172
2173 33 C 2173
2174 34 NSS = 0 2174
2175 35 DO 120 IS = 1 , NS 2175
2176 36 DO 120 IR = 4 , 6 2176
2177 37 C IE = IABS( JS( IR , IS ) ) 2177
2178 38 C IJE5 = JE( 5 , IE ) 2178
2179 39 XSS = XS( 1 , IS ) 2179
2180 40 IF( XSS . GT . -.05 . AND . XSS . LT . .05 . AND . 2180
2181 41 KSDDEL( IS ) . LT . IDUMP ) THEN 2181
2182 42 C IF( IJE5 . EQ . 8 ) THEN 2182
2183 43 KSDDEL( IS ) = IDUMP 2183
2184 44 JEE( IS ) = 1 2184
2185 45 NSS = NSS + 1 2185
2186 46 JTRIG( NSS ) = IS 2186
2187 47 END IF 2187
2188 48 120 CONTINUE 2188
2189 49 C 2189
2190 50 DO 130 IS = 1 , NSS 2190
2191 51 JSE( IS ) = JTRIG( IS ) 2191
2192 52 130 CONTINUE 2192
2193 53 C 2193
2194 54 MSS = NSS 2194

```

```

2195      55      DO 140 KDIV = 1 , NOFDIV      2195
2196      56      ITRIG = 0      2196
2197      57      DO 150 KS = 1 , NSS      2197
2198      58      C      2198
2199      59      ISS = JSE( KS )      2199
2200      60      C      2200
2201      61      DO 160 KR = 1 , 3      2201
2202      62      IVV = JS( KR , ISS )      2202
2203      63      C      2203
2204      64      IE = JV( 2 , IVV )      2204
2205      65      IF( IE . GT . 0 ) THEN      2205
2206      66      C      2206
2207      67      IV1 = JE( 1 , IE )      2207
2208      68      IF( IV1 . EQ . IVV ) THEN      2208
2209      69      ISI = JE( 3 , IE )      2209
2210      70      ELSE      2210
2211      71      ISI = JE( 4 , IE )      2211
2212      72      END IF      2212
2213      73      IS = ISI      2213
2214      74      C      2214
2215      75      750      CONTINUE      2215
2216      76      C      2216
2217      77      JES = JEE( IS )      2217
2218      78      XAS = XS( 3 , S )      2218
2219      79      IF( JES . EQ . 0 . AND . XAS . LT . SAREVG ) THEN      2219
2220      80      ITRIG = ITRIG + 1      2220
2221      81      KTRIG( ITRIG ) = IS      2221
2222      82      KSDDEL( IS ) = IDUMP      2222
2223      83      JEE( IS ) = 1      2223
2224      84      END IF      2224
2225      85      C      2225
2226      86      DO 760 IR = 1 , 3      2226
2227      87      JR = MOD( IR , 3 ) + 1      2227
2228      88      IEA = IABS( JS( JR + 3 , IS ) )      2228
2229      89      IF( IEA . EQ . IE ) THEN      2229
2230      90      JJR = MOD( JR + 1 , 3 ) + 4      2230
2231      91      IER = IABS( JS( JJR , IS ) )      2231
2232      92      C      2232
2233      93      IV1 = JE( 1 , IER )      2233
2234      94      IF( IV1 . EQ . IVV ) THEN      2234
2235      95      ISR = JE( 3 , IER )      2235
2236      96      ELSE      2236
2237      97      ISR = JE( 4 , IER )      2237
2238      98      END IF      2238
2239      99      END IF      2239
2240      100      760      CONTINUE      2240
2241      101      C      2241
2242      102      IF( ISR . NE . ISI ) THEN      2242
2243      103      IS = ISR      2243
2244      104      IE = IER      2244
2245      105      C      2245
2246      106      GO TO 750      2246
2247      107      END IF      2247
2248      108      C      2248
2249      109      ELSE      2249
2250      110      C      2250
2251      111      IE = - IE      2251
2252      112      IV1 = JE( 1 , IE )      2252
2253      113      IF( IV1 . EQ . IVV ) THEN      2253
2254      114      ISI = JE( 3 , IE )      2254
2255      115      ELSE      2255
2256      116      ISI = JE( 4 , IE )      2256
2257      117      END IF      2257
2258      118      IS = ISI      2258
2259      119      ISI = 0      2259
2260      120      C      2260
2261      121      650      CONTINUE      2261
2262      122      C      2262
2263      123      JES = JEE( IS )      2263
2264      124      XAS = XS( 3 , IS )      2264
2265      125      IF( JES . EQ . 0 . AND . XAS . LT . SAREVG ) THEN      2265
2266      126      ITRIG = ITRIG + 1      2266
2267      127      KTRIG( ITRIG ) = IS      2267
2268      128      KSDDEL( IS ) = IDUMP      2268

```

2269	129		JEE(IS) = 1	2269
2270	130		END IF	2270
2271	131	C		2271
2272	132		DO 660 IR = 1 , 3	2272
2273	133		JR = MOD(IR , 3) + 1	2273
2274	134		IEA = IABS(JS(JR + 3 , IS))	2274
2275	135		IF(IEA . EQ . IE) THEN	2275
2276	136		JJR = MOD(JR + 1 , 3) + 4	2276
2277	137		IER = IABS(JS(JJR , IS))	2277
2278	138	C		2278
2279	139		IV1 = JE(1 , IER)	2279
2280	140		IF(IV1 . EQ . IVV) THEN	2280
2281	141		ISR = JE(3 , IER)	2281
2282	142		ELSE	2282
2283	143		ISR = JE(4 , IER)	2283
2284	144		END IF	2284
2285	145		END IF	2285
2286	146	C		2286
2287	147	660	CONTINUE	2287
2288	148	C		2288
2289	149		IF(ISR . NE . ISI) THEN	2289
2290	150		IS = ISR	2290
2291	151		IE = IER	2291
2292	152		GO TO 650	2292
2293	153		END IF	2293
2294	154		END IF	2294
2295	155	160	CONTINUE	2295
2296	156	C		2296
2297	157	150	CONTINUE	2297
2298	158	C		2298
2299	159		DO 170 IS = 1 , ITRIG	2299
2300	160		JTRIG(IS + MSS) = KTRIG(IS)	2300
2301	161		JSE(IS) = KTRIG(IS)	2301
2302	162	170	CONTINUE	2302
2303	163		NSS = ITRIG	2303
2304	164		MSS = MSS + ITRIG	2304
2305	165	C		2305
2306	166	140	CONTINUE	2306
2307	167		NSS = MSS	2307
2308	168	C		2308
2309	169		DO 300 KDIV = 1 , 1	2309
2310	170		LTRIG = NSS	2310
2311	171		IEDGE = 0	2311
2312	172		NCOLOR = 0	2312
2313	173	C		2313
2314	174		DO 290 IE = 1 , NE	2314
2315	175		JSE(IE) = 0	2315
2316	176	290	CONTINUE	2316
2317	177	C		2317
2318	178		DO 310 IS = 1 , NSS	2318
2319	179		ISS = JTRIG(IS)	2319
2320	180		XSAREA = XS(3 , ISS)	2320
2321	181		IF(XSAREA . GE . RMINVG) THEN	2321
2322	182	C		2322
2323	183		DO 335 IR = 4 , 6	2323
2324	184		IE = IABS(JS(IR , ISS))	2324
2325	185		IJE5 = JE(5 , IE)	2325
2326	186		IF(IJE5 . NE . 0) THEN	2326
2327	187		JR2 = MOD(IR - 3 , 3) + 4	2327
2328	188		IE2 = IABS(JS(JR2 , ISS))	2328
2329	189		JR3 = MOD(IR - 2 , 3) + 4	2329
2330	190		IE3 = IABS(JS(JR3 , ISS))	2330
2331	191		XE1 = XE(1 , IE)	2331
2332	192		XE2 = XE(1 , IE2)	2332
2333	193		XE3 = XE(1 , IE3)	2333
2334	194		XEDIST = 1. / XE1	2334
2335	195		YE2 = XE2 * XEDIST	2335
2336	196		YE3 = XE3 * XEDIST	2336
2337	197		ZE2 = (YE2 - 1.5) * (YE2 - .1)	2337
2338	198		ZE3 = (YE3 - 1.5) * (YE3 - .1)	2338
2339	199		YY2 = XE1 * XE1 + XE2 * XE2 + .35 * XE1 * XE2 - XE3 * XE3	2339
2340	200		YY3 = XE1 * XE1 + XE3 * XE3 + .35 * XE1 * XE3 - XE2 * XE2	2340
2341	201		IF(ZE2 . LT . .0 . AND . ZE3 . LT . 0 . AND .	2341
2342	202		YY2 . GT . 0 . AND . YY3 . GT . 0 .) THEN	2342

2343	203		CALL DISECT (IE , IDONE , IOUMP)	2343
2344	204	C		2344
2345	205		LTRIG = LTRIG + 1	2345
2346	206		JTRIG(LTRIG) = NS	2346
2347	207		KSOELT(NS) = IOUMP	2347
2348	208	C		2348
2349	209		IEDGE = IEDGE + 1	2349
2350	210		IRECNC(IEDGE) = NE	2350
2351	211		NCOLOR = NCOLOR + 1	2351
2352	212		JEE(NCOLOR) = NE	2352
2353	213		JSE(NE) = 1	2353
2354	214		IEDGE = IEDGE + 1	2354
2355	215		IRECNC(IEDGE) = NE - 1	2355
2356	216		NCOLOR = NCOLOR + 1	2356
2357	217		JEE(NCOLOR) = NE - 1	2357
2358	218		JSE(NE - 1) = 1	2358
2359	219	C		2359
2360	220		END IF	2360
2361	221		END IF	2361
2362	222	335	CONTINUE	2362
2363	223		END IF	2363
2364	224	310	CONTINUE	2364
2365	225	C		2365
2366	226		NSS = LTRIG	2366
2367	227		IEDGE = 0	2367
2368	228		NCOLOR = 0	2368
2369	229	C		2369
2370	230		DO 295 IE = 1 , NE	2370
2371	231		JSE(IE) = 0	2371
2372	232	295	CONTINUE	2372
2373	233	C		2373
2374	234		DO 320 IS = 1 , NSS	2374
2375	235		ISS = JTRIG(IS)	2375
2376	236		XSAREA = XS(3 , ISS)	2376
2377	237	C		2377
2378	238		DO 735 IR = 4 , 6	2378
2379	239		IE = IABS(JS(IR , ISS))	2379
2380	240		IF(JSE(IE) . EQ . 0) THEN	2380
2381	241		IEDGE = IEDGE + 1	2381
2382	242		IRECNC(IEDGE) = IE	2382
2383	243		NCOLOR = NCOLOR + 1	2383
2384	244		JEE(NCOLOR) = IE	2384
2385	245		JSE(IE) = 1	2385
2386	246		END IF	2386
2387	247	735	CONTINUE	2387
2388	248	C		2388
2389	249		IF(XSAREA . GT . RMINVG) THEN	2389
2390	250	C		2390
2391	251		AREAXS = SAREA(ISS)	2391
2392	252		IE1 = IABS(JS(4 , ISS))	2392
2393	253		XE1 = XE(1 , IE1)	2393
2394	254		HD1 = AREAXS * XE1 * XE1	2394
2395	255		IJE5 = JE(5 , IE1)	2395
2396	256		IE2 = IABS(JS(5 , ISS))	2396
2397	257		XE2 = XE(1 , IE2)	2397
2398	258		HD2 = AREAXS * XE2 * XE2	2398
2399	259		IJE5 = IJE5 + JE(5 , IE2)	2399
2400	260		IE3 = IABS(JS(6 , ISS))	2400
2401	261		XE3 = XE(1 , IE3)	2401
2402	262		HD3 = AREAXS * XE3 * XE3	2402
2403	263		IJE5 = IJE5 + JE(5 , IE3)	2403
2404	264		RATIO = AMAX1(HD1 , HD2 , HD3)	2404
2405	265		IRATIO = 0	2405
2406	266		IF(RATIO . LE . 7 . . AND . IJE5 . EQ . 0 . AND .	2406
2407	267		XSAREA . GT . SMINVG) IRATIO = 1	2407
2408	268		IF(IJE5 . GT . 0) IRATIO = 2	2408
2409	269	C		2409
2410	270		IF(IRATIO . EQ . 2) THEN	2410
2411	271		IJE51 = JE(5 , IE1)	2411
2412	272		IJE52 = JE(5 , IE2)	2412
2413	273		IJE53 = JE(5 , IE3)	2413
2414	274		IF(IJE51 . NE . 0) THEN	2414
2415	275		IEDIST = IE1	2415
2416	276		XE1 = XE(1 , IE1)	2416

```

2417 277      XE2 = XE( 1 , IE2 )
2418 278      XE3 = XE( 1 , IE3 )
2419 279      END IF
2420 280      IF( IJES2 . NE . 0 ) THEN
2421 281      IEDIST = IE2
2422 282      XE1 = XE( 1 , IE2 )
2423 283      XE2 = XE( 1 , IE1 )
2424 284      XE3 = XE( 1 , IE3 )
2425 285      END IF
2426 286      IF( IJES3 . NE . 0 ) THEN
2427 287      IEDIST = IE3
2428 288      XE1 = XE( 1 , IE3 )
2429 289      XE2 = XE( 1 , IE2 )
2430 290      XE3 = XE( 1 , IE1 )
2431 291      END IF
2432 292      XEDIST = 1. / XE( 1 , IEDIST )
2433 293      YE2 = XE2 * XEDIST
2434 294      YE3 = XE3 * XEDIST
2435 295      ZE2 = ( YE2 - 1.5 ) * ( YE2 - .1 )
2436 296      ZE3 = ( YE3 - 1.5 ) * ( YE3 - .1 )
2437 297      YY2 = XE1 * XE1 + XE2 * XE2 + .35 * XE1 * XE2 - XF3 * XE3
2438 298      YY3 = XE1 * XE1 + XE3 * XE3 + .35 * XE1 * XE3 - XE2 * XE2
2439 299      YY3 = XE1 * XE1 + XE3 * XE3 - XE2 * XE2
2440 300      IF( ZE2 . LT . .0 . AND . ZE3 . LT . 0. . AND .
2441 301      . YY2 . GT . 0. . AND . YY3 . GT . 0. ) THEN
2442 302      CALL DISECT ( IEDIST , IDONE , IDUMP )
2443 303      C
2444 304      LTRIG = LTRIG + 1
2445 305      JTRIG( LTRIG ) = NS
2446 306      KSDEL( NS ) = IDUMP
2447 307      C
2448 308      IEDGE = IEDGE + 1
2449 309      IRECNC( IEDGE ) = NE
2450 310      NCOLOR = NCOLOR + 1
2451 311      JEE( NCOLOR ) = NE
2452 312      JSE( NE ) = 1
2453 313      IEDGE = IEDGE + 1
2454 314      IRECNC( IEDGE ) = NE - 1
2455 315      NCOLOR = NCOLOR + 1
2456 316      JEE( NCOLOR ) = NE - 1
2457 317      JSE( NE - 1 ) = 1
2458 318      C
2459 319      END IF
2460 320      END IF
2461 321      C
2462 322      IF( IRATIO . EQ . 1 ) THEN
2463 323      C
2464 324      CALL VERCEN( ISS )
2465 325      KSDEL( ISS ) = IDUMP
2466 326      LTRIG = LTRIG + 1
2467 327      JTRIG( LTRIG ) = NS - 1
2468 328      KSDEL( NS - 1 ) = IDUMP
2469 329      C
2470 330      LTRIG = LTRIG + 1
2471 331      JTRIG( LTRIG ) = NS
2472 332      KSDEL( NS ) = IDUMP
2473 333      C
2474 334      IEDGE = IEDGE + 1
2475 335      IRECNC( IEDGE ) = NE
2476 336      NCOLOR = NCOLOR + 1
2477 337      JEE( NCOLOR ) = NE
2478 338      JSE( NE ) = 1
2479 339      IEDGE = IEDGE + 1
2480 340      IRECNC( IEDGE ) = NE - 1
2481 341      NCOLOR = NCOLOR + 1
2482 342      JEE( NCOLOR ) = NE - 1
2483 343      JSE( NE - 1 ) = 1
2484 344      IEDGE = IEDGE + 1
2485 345      IRECNC( IEDGE ) = NE - 2
2486 346      NCOLOR = NCOLOR + 1
2487 347      JEE( NCOLOR ) = NE - 2
2488 348      JSE( NE - 2 ) = 1
2489 349      C
2490 350      ELSE

```

```

2491 351 C
2492 352 IDISCT = 0
2493 353 DO 545 KK = 4 , 6
2494 354 IEE = JS( KK , ISS )
2495 355 IEF = IABS( IEE )
2496 356 IJE55 = JE( 5 , IEF )
2497 357 IF( IJE55 .EQ. 0 ) THEN
2498 358 IF( IEE .GT. 0 ) THEN
2499 359 ISI = JE( 4 , IEE )
2500 360 ELSE
2501 361 ISI = JE( 3 , IEF )
2502 362 END IF
2503 363 AREAXS = SAREA( ISI )
2504 364 IE1 = IABS( JS( 4 , ISI ) )
2505 365 XE1 = XE( 1 , IE1 )
2506 366 IJE55 = JE( 5 , IE1 )
2507 367 HD1 = AREAXS * XE1 * XE1
2508 368 IE2 = IABS( JS( 5 , ISI ) )
2509 369 XE2 = XE( 1 , IE2 )
2510 370 IJE55 = IJE55 + JE( 5 , IE2 )
2511 371 HD2 = AREAXS * XE2 * XE2
2512 372 IE3 = IABS( JS( 6 , ISI ) )
2513 373 XE3 = XE( 1 , IE3 )
2514 374 IJE55 = IJE55 + JE( 5 , IE3 )
2515 375 HD3 = AREAXS * XE3 * XE3
2516 376 RATIO = AMAX1( HD1 , HD2 , HD3 )
2517 377 YSAREA = XS( 3 , ISI )
2518 378 IF( RATIO .LT. 7. .AND. YSAREA .GT. SMINVG .AND.
2519 379 IJE55 .EQ. 0 ) THEN
2520 380 IDISCT = 1
2521 381 DO 435 IR = 4 , 6
2522 382 IE = IABS( JS( IR , ISI ) )
2523 383 IF( JSE( IE ) .EQ. 0 ) THEN
2524 384 IEDGE = IEDGE + 1
2525 385 IRECNC( IEDGE ) = IE
2526 386 NCOLOR = NCOLOR + 1
2527 387 JEE( NCOLOR ) = IE
2528 388 JSE( IE ) = 1
2529 389 END IF
2530 390 435 CONTINUE
2531 391 CALL VERCEN( ISI )
2532 392 KSDEL( ISI ) = IDUMP
2533 393 LTRIG = LTRIG + 1
2534 394 JTRIG( LTRIG ) = NS - 1
2535 395 KSDEL( NS - 1 ) = IDUMP
2536 396 C
2537 397 LTRIG = LTRIG + 1
2538 398 JTRIG( LTRIG ) = NS
2539 399 KSDEL( NS ) = IDUMP
2540 400 C
2541 401 IEDGE = IEDGE + 1
2542 402 IRECNC( IEDGE ) = NE
2543 403 NCOLOR = NCOLOR + 1
2544 404 JEE( NCOLOR ) = NE
2545 405 JSE( NE ) = 1
2546 406 IEDGE = IEDGE + 1
2547 407 IRECNC( IEDGE ) = NE - 1
2548 408 NCOLOR = NCOLOR + 1
2549 409 JEE( NCOLOR ) = NE - 1
2550 410 JSE( NE - 1 ) = 1
2551 411 IEDGE = IEDGE + 1
2552 412 IRECNC( IEDGE ) = NE - 2
2553 413 NCOLOR = NCOLOR + 1
2554 414 JEE( NCOLOR ) = NE - 2
2555 415 JSE( NE - 2 ) = 1
2556 416 END IF
2557 417 END IF
2558 418 545 CONTINUE
2559 419 C
2560 420 IF( IDISCT .EQ. 0 ) THEN
2561 421 IE1 = IABS( JS( 4 , ISS ) )
2562 422 XE1 = XE( 1 , IE1 )
2563 423 IE2 = IABS( JS( 5 , ISS ) )
2564 424 XE2 = XE( 1 , IE2 )

```

```

2565 425      IE3 = IABS( JS( 6 , ISS ) )
2566 426      XE3 = XE( 1 , IE3 )
2567 427      IEDIST = IE1
2568 428      XEDIST = XE1
2569 429      IF( XE2 . GT . XEDIST ) THEN
2570 430      XEDIST = XE2
2571 431      IEDIST = IE2
2572 432      END IF
2573 433      IF( XE3 . GT . XEDIST ) THEN
2574 434      XEDIST = XE3
2575 435      IEDIST = IE3
2576 436      END IF
2577 437      ISL = JE( 3 , IEDIST )
2578 438      ISR = JE( 4 , IEDIST )
2579 439      XSISL = XS( 3 , ISL )
2580 440      XSISR = XS( 3 , ISR )
2581 441      IJES = JE( 5 , IEDIST )
2582 442      IF( XSISL . GT . RMINVG . AND . XSISR . GT . RMINVG . AND .
2583 443      . IJES . EQ . 0 . AND . IRATIO . NE . 2 ) THEN
2584 444      IF( ISS . NE . ISL ) THEN
2585 445      DO 345 IR = 4 , 6
2586 446      IE = IABS( JS( IR , ISL ) )
2587 447      IF( JSE( IE ) . EQ . 0 ) THEN
2588 448      IEDGE = IEDGE + 1
2589 449      IRECNC( IEDGE ) = IE
2590 450      NCOLOR = NCOLOR + 1
2591 451      JEE( NCOLOR ) = IE
2592 452      JSE( IE ) = 1
2593 453      END IF
2594 454      345 CONTINUE
2595 455      END IF
2596 456      C
2597 457      IF( ISS . NE . ISR ) THEN
2598 458      DO 355 IR = 4 , 6
2599 459      IE = IABS( JS( IR , ISR ) )
2600 460      IF( JSE( IE ) . EQ . 0 ) THEN
2601 461      IEDGE = IEDGE + 1
2602 462      IRECNC( IEDGE ) = IE
2603 463      NCOLOR = NCOLOR + 1
2604 464      JEE( NCOLOR ) = IE
2605 465      JSE( IE ) = 1
2606 466      END IF
2607 467      355 CONTINUE
2608 468      END IF
2609 469      C
2610 470      IDONE = 0
2611 471      CALL DISECT ( IEDIST , IDONE , IDUMP )
2612 472      IF( IDONE . EQ . 1 ) THEN
2613 473      C
2614 474      LTRIG = LTRIG + 1
2615 475      JTRIG( LTRIG ) = NS
2616 476      KSDFLT( NS ) = IDUMP
2617 477      LTRIG = LTRIG + 1
2618 478      JTRIG( LTRIG ) = NS - 1
2619 479      KSDFLT( NS - 1 ) = IDUMP
2620 480      C
2621 481      IEDGE = IEDGE + 1
2622 482      IRECNC( IEDGE ) = NE
2623 483      NCOLOR = NCOLOR + 1
2624 484      JEE( NCOLOR ) = NE
2625 485      JSE( NE ) = 1
2626 486      IEDGE = IEDGE + 1
2627 487      IRECNC( IEDGE ) = NE - 1
2628 488      NCOLOR = NCOLOR + 1
2629 489      JEE( NCOLOR ) = NE - 1
2630 490      JSE( NE - 1 ) = 1
2631 491      IEDGE = IEDGE + 1
2632 492      IRECNC( IEDGE ) = NE - 2
2633 493      NCOLOR = NCOLOR + 1
2634 494      JEE( NCOLOR ) = NE - 2
2635 495      JSE( NE - 2 ) = 1
2636 496      END IF
2637 497      C
2638 498      END IF

```



```

2639 499      END IF      2639
2640 500      END IF      2640
2641 501      END IF      2641
2642 502      C          2642
2643 503      320 CONTINUE  2643
2644 504      C          2644
2645 505      DO 340 IEM = 1 , NCOLOR 2645
2646 506      IE = JEE( IEM ) 2646
2647 507      C          2647
2648 508      ISL = JE( 3 , IE ) 2648
2649 509      YSAREA = XS( 3 , ISL ) 2649
2650 510      IJE5 = JE( 5 , IE ) 2650
2651 511      IF( YSAREA . GE . RMINVG . AND . IJE5 . NE . 0 ) THEN 2651
2652 512      IE1 = IABS( JS( 4 , ISL ) ) 2652
2653 513      IE2 = IABS( JS( 5 , ISL ) ) 2653
2654 514      IE3 = IABS( JS( 6 , ISL ) ) 2654
2655 515      IJE51 = JE( 5 , IE1 ) 2655
2656 516      IJE52 = JE( 5 , IE2 ) 2656
2657 517      IJE53 = JE( 5 , IE3 ) 2657
2658 518      IF( IJE51 . NE . 0 ) THEN 2658
2659 519      IEDIST = IE1 2659
2660 520      XE1 = XE( 1 , IE1 ) 2660
2661 521      XE2 = XE( 1 , IE2 ) 2661
2662 522      XE3 = XE( 1 , IE3 ) 2662
2663 523      END IF 2663
2664 524      IF( IJE52 . NE . 0 ) THEN 2664
2665 525      IEDIST = IE2 2665
2666 526      XE1 = XE( 1 , IE2 ) 2666
2667 527      XE2 = XE( 1 , IE1 ) 2667
2668 528      XE3 = XE( 1 , IE3 ) 2668
2669 529      END IF 2669
2670 530      IF( IJE53 . NE . 0 ) THEN 2670
2671 531      IEDIST = IE3 2671
2672 532      XE1 = XE( 1 , IE3 ) 2672
2673 533      XE2 = XE( 1 , IE2 ) 2673
2674 534      XE3 = XE( 1 , IE1 ) 2674
2675 535      END IF 2675
2676 536      XEDIST = 1. / XE( 1 , IEDIST ) 2676
2677 537      YE2 = XE2 * XEDIST 2677
2678 538      YE3 = XE3 * XEDIST 2678
2679 539      ZE2 = ( YE2 - 1.5 ) * ( YE2 - .1 ) 2679
2680 540      ZE3 = ( YE3 - 1.5 ) * ( YE3 - .1 ) 2680
2681 541      YY2 = XE1 * XE1 + XE2 * XE2 + .35 * XE1 * XE2 - XE3 * XE3 2681
2682 542      YY3 = XE1 * XE1 + XE3 * XE3 + .35 * XE1 * XE3 - XE2 * XE2 2682
2683 543      IF( ZE2 . LT . .0 . AND . ZE3 . LT . 0 . AND . 2683
2684 544      YY2 . GT . 0 . AND . YY3 . GT . 0 . ) THEN 2684
2685 545      CALL DISECT ( IEDIST , IDONE , IDUMP ) 2685
2686 546      C          2686
2687 547      LTRIG = LTRIG + 1 2687
2688 548      JTRIG( LTRIG ) = NS 2688
2689 549      KSDDEL( NS ) = IDUMP 2689
2690 550      C          2690
2691 551      IEDGE = IEDGE + 1 2691
2692 552      IRECNC( IEDGE ) = NE 2692
2693 553      NCOLOR = NCOLOR + 1 2693
2694 554      JEE( NCOLOR ) = NE 2694
2695 555      JSE( NE ) = 1 2695
2696 556      IEDGE = IEDGE + 1 2696
2697 557      IRECNC( IEDGE ) = NE - 1 2697
2698 558      NCOLOR = NCOLOR + 1 2698
2699 559      JEE( NCOLOR ) = NE - 1 2699
2700 560      JSE( NE - 1 ) = 1 2700
2701 561      C          2701
2702 562      ELSE 2702
2703 563      C          2703
2704 564      IEDIST = IE1 2704
2705 565      XEDIST = XE1 2705
2706 566      IF( XE2 . GT . XEDIST ) THEN 2706
2707 567      XEDIST = XE2 2707
2708 568      IEDIST = IE2 2708
2709 569      END IF 2709
2710 570      IF( XE3 . GT . XEDIST ) THEN 2710
2711 571      XEDIST = XE3 2711
2712 572      IEDIST = IE3 2712

```

```

2713 573      END IF                                2713
2714 574      ISL = JE( 3 , IEDIST )                2714
2715 575      ISR = JE( 4 , IEDIST )                2715
2716 576      XSISL = XS( 3 , ISL )                 2716
2717 577      XSISR = XS( 3 , ISR )                 2717
2718 578      IJE5 = JE( 5 , IEDIST )               2718
2719 579      IF( XSISL . GT . RMINVG . AND . XSISR . GT . RMINVG . AND .
2720 580      . IJE5 . EQ . 0 ) THEN                 2720
2721 581      DO 645 IR = 4 , 6                       2721
2722 582      IE = IABS( JS( IR , ISL ) )             2722
2723 583      IF( JSE( IE ) . EQ . 0 ) THEN          2723
2724 584      IEDGE = IEDGE + 1                       2724
2725 585      IRECNC( IEDGE ) = IE                   2725
2726 586      NCOLOR = NCOLOR + 1                   2726
2727 587      JEE( NCOLOR ) = IE                     2727
2728 588      JSE( IE ) = 1                          2728
2729 589      END IF                                2729
2730 590      645 CONTINUE                          2730
2731 591      DO 655 IR = 4 , 6                       2731
2732 592      IE = IABS( JS( IR , ISR ) )             2732
2733 593      IF( JSE( IE ) . EQ . 0 ) THEN          2733
2734 594      IEDGE = IEDGE + 1                       2734
2735 595      IRECNC( IEDGE ) = IE                   2735
2736 596      NCOLOR = NCOLOR + 1                   2736
2737 597      JEE( NCOLOR ) = IE                     2737
2738 598      JSE( IE ) = 1                          2738
2739 599      END IF                                2739
2740 600      655 CONTINUE                          2740
2741 601      C                                     2741
2742 602      IDONE = 0                               2742
2743 603      CALL DISECT ( IEDIST , IDONE , IDUMP ) 2743
2744 604      IF( IDONE . EQ . 1 ) THEN             2744
2745 605      C                                     2745
2746 606      LTRIG = LTRIG + 1                       2746
2747 607      JTRIG( LTRIG ) = NS                     2747
2748 608      KSDEL( NS ) = IDUMP                     2748
2749 609      LTRIG = LTRIG + 1                       2749
2750 610      JTRIG( LTRIG ) = NS - 1                 2750
2751 611      KSDEL( NS - 1 ) = IDUMP                 2751
2752 612      C                                     2752
2753 613      IEDGE = IEDGE + 1                       2753
2754 614      IRECNC( IEDGE ) = NE                   2754
2755 615      NCOLOR = NCOLOR + 1                   2755
2756 616      JEE( NCOLOR ) = NE                     2756
2757 617      JSE( NE ) = 1                          2757
2758 618      IEDGE = IEDGE + 1                       2758
2759 619      IRECNC( IEDGE ) = NE - 1               2759
2760 620      NCOLOR = NCOLOR + 1                   2760
2761 621      JEE( NCOLOR ) = NE - 1                 2761
2762 622      JSE( NE - 1 ) = 1                      2762
2763 623      IEDGE = IEDGE + 1                       2763
2764 624      IRECNC( IEDGE ) = NE - 2               2764
2765 625      NCOLOR = NCOLOR + 1                   2765
2766 626      JEE( NCOLOR ) = NE - 2                 2766
2767 627      JSE( NE - 2 ) = 1                      2767
2768 628      C                                     2768
2769 629      END IF                                2769
2770 630      END IF                                2770
2771 631      END IF                                2771
2772 632      END IF                                2772
2773 633      340 CONTINUE                          2773
2774 634      C                                     2774
2775 635      NSS = LTRIG                            2775
2776 636      C                                     2776
2777 637      DO 370 IEM = 1 , NCOLOR                 2777
2778 638      IE = JEE( IEM )                          2778
2779 639      CALL RECNC( IE , IDONE , ITL , ITR , JA , JB , JC , JD ) 2779
2780 640      CALL RECNC( JA , JADONE , ITL , ITR , JAA , JAB , JAC , JAD ) 2780
2781 641      CALL RECNC( JB , JBDONE , ITL , ITR , JBA , JBB , JBC , JBD ) 2781
2782 642      CALL RECNC( JC , JCDONE , ITL , ITR , JCA , JCB , JCC , JCD ) 2782
2783 643      CALL RECNC( JD , JDDONE , ITL , ITR , JDA , JDB , JDC , JDD ) 2783
2784 644      370 CONTINUE                          2784
2785 645      C                                     2785
2786 646      300 CONTINUE                          2786

```

2787	647	C		2787
2788	648		NVECE = NE / MBL	2788
2789	649		NREME = NE - NVECE * MBL	2789
2790	650		NVECS = NS / MBL	2790
2791	651		NREMS = NS - NVECS * MBL	2791
2792	652		NVECV = NV / MBL	2792
2793	653		NREMV = NV - NVECV * MBL	2793
2794	654	C		2794
2795	655		DO 400 INE = 1 , NVECE	2795
2796	656		NOFVEE(INE) = MBL	2796
2797	657	400	CONTINUE	2797
2798	658		NVEEE = NVECE	2798
2799	659		IF(NREME . GT . 0) THEN	2799
2800	660		NVEEE = NVECE + 1	2800
2801	661		NOFVEE(NVEEE) = NREME	2801
2802	662		END IF	2802
2803	663	C		2803
2804	664		DO 410 INS = 1 , NVECS	2804
2805	665		NOFVES(INS) = MBL	2805
2806	666	410	CONTINUE	2806
2807	667		NVEES = NVECS	2807
2808	668		IF(NREMS . GT . 0) THEN	2808
2809	669		NVEES = NVECS + 1	2809
2810	670		NOFVES(NVEES) = NREMS	2810
2811	671		END IF	2811
2812	672	C		2812
2813	673		DO 420 INV = 1 , NVECV	2813
2814	674		NOFVEV(INV) = MBL	2814
2815	675	420	CONTINUE	2815
2816	676		NVEEV = NVECV	2816
2817	677		IF(NREMV . GT . 0) THEN	2817
2818	678		NVEEV = NVECV + 1	2818
2819	679		NOFVEV(NVEEV) = NREMV	2819
2820	680		END IF	2820
2821	681	C		2821
2822	682		PRINT*,NV,NE,NS	2822
2823	683	C		2823
2824	684	C	--- EXIT POINT FROM SUBROUTINE -----	2824
2825	685	C		2825
2826	686	C	-----	2826
2827	687		RETURN	2827
2828	688	C	-----	2828
2829	689	C		2829
2830	690	C	---	2830
2831	691		END	2831

```

2832 1 SUBROUTINE DELPTNT( DAREA , IDUMP )
2833 2 C
2834 3 C -----I
2835 4 C I
2836 5 C DELPTN ADAPT THE GRID DYNAMICALLY, DELETE VERTECES I
2837 6 C WILL FLAGED TRIANGLES FOR DELETION I
2838 7 C I
2839 8 C -----I
2840 9 C
2841 10 IMPLICIT REAL (A-H,O-Z)
2842 11 C
2843 12 include 'cmsh00.h'
2844 13 include 'chyd00.h'
2845 14 include 'cint00.h'
2846 15 include 'cphs10.h'
2847 16 include 'cphs20.h'
2848 17 C
2849 18 INTEGER JTRIG(MEM),KTRIG(MEM),IRECNC(MEM)
2850 19 INTEGER JSE(MEM),JEE(MEM),IOFDVS(10),NOFDVS(10)
2851 20 INTEGER IITRIG(200)
2852 21 REAL ADFCTR(8),DLFCTR(8)
2853 22 C
2854 23 EQUIVALENCE (UL,JTRIG)
2855 24 EQUIVALENCE (VR,KTRIG)
2856 25 EQUIVALENCE (VL,IRECNC)
2857 26 EQUIVALENCE (PR,JSE)
2858 27 EQUIVALENCE (PL,JEE)
2859 28 C
2860 29 DLFCTR( 1 ) = DAREA
2861 30 DLFCTR( 2 ) = .4
2862 31 DLFCTR( 3 ) = .5
2863 32 DLFCTR( 4 ) = .65
2864 33 DLFCTR( 5 ) = .8
2865 34 C
2866 35 SMINVG = SAREVG * DAREA
2867 36 DO 112 IS = 1 , NS
2868 37 JSDEL( IS ) = 0
2869 38 112 CONTINUE
2870 39 ISDEL = 0
2871 40 C
2872 41 NSS = 0
2873 42 FLUXPP = .00001 * HYDOMM( 4 )
2874 43 FLUXUU = .00001 * HYDOMM( 2 )
2875 44 FLUXRR = .00001 * HYDOMM( 1 )
2876 45 DO 120 IS = 1 , NS
2877 46 PCRTRY = HYDFLX( IS , 4 ) - FLUXPP
2878 47 IPCRTR = SIGN( 1. , PCRTRY )
2879 48 UCRTRY = HYDFLX( IS , 2 ) - FLUXUU
2880 49 IUCRTR = SIGN( 1. , UCRTRY )
2881 50 RCRTRY = HYDFLX( IS , 1 ) - FLUXRR
2882 51 IRCRTR = SIGN( 1. , RCRTRY )
2883 52 NIDUMP = IDUMP - NAREAD
2884 53 IF(
2885 54 . IPCRTR . EQ . - 1 . AND .
2886 55 . IUCRTR . EQ . - 1 . AND .
2887 56 . IRCRTR . EQ . - 1 . AND .
2888 57 . KSDDEL( IS ) . LE . NIDUMP . AND .
2889 58 . KSDDEL( IS ) . NE . 0 ) THEN
2890 59 NSS = NSS + 1
2891 60 JTRIG( NSS ) = IS
2892 61 END IF
2893 62 120 CONTINUE
2894 63 C
2895 64 PRINT*,NV,NE,NS,NSS
2896 65 C
2897 66 ISDEL = NSS
2898 67 DO 210 IS = 1 , NSS
2899 68 JSDEL( IS ) = JTRIG( IS )
2900 69 210 CONTINUE
2901 70 C
2902 71 DO 300 KDIV = 1 , 1
2903 72 ILOOP = 1
2904 73 310 CONTINUE
2905 74 ISS = JSDEL( ILOOP )

```

```

2906 75 XSAREA = XS( 3 , ISS )
2907 76 INDCTR = 0
2908 77 IF( XSAREA . LT . SMINVG ) THEN
2909 78 C
2910 79 CALL VERDEL( ISS , INDCTR , NIDUMP , JJTRIG , IITRIG )
2911 80 IF( INDCTR . EQ . 1 ) THEN
2912 81 ILOOP = 1
2913 82 ELSE
2914 83 JSDEL( ILOOP ) = 0
2915 84 ILOOP = ILOOP + 1
2916 85 END IF
2917 86 ELSE
2918 87 JSDEL( ILOOP ) = 0
2919 88 ILOOP = ILOOP + 1
2920 89 END IF
2921 90 C
2922 91 IF( ISDEL . GT . ILOOP ) GO TO 310
2923 92 PRINT *,KDIV,NV,NE,NS,DLFCTR(KDIV)
2924 93 300 CONTINUE
2925 94 C
2926 95 NVECE = NE / MBL
2927 96 NREME = NE - NVECE * MBL
2928 97 NVECS = NS / MBL
2929 98 NREMS = NS - NVECS * MBL
2930 99 NVECV = NV / MBL
2931 100 NREMV = NV - NVECV * MBL
2932 101 C
2933 102 DO 400 INE = 1 , NVECE
2934 103 NOFVEE( INE ) = MBL
2935 104 400 CONTINUE
2936 105 NVEEE = NVECE
2937 106 IF( NREME . GT . 0 ) THEN
2938 107 NVEEE = NVECE + 1
2939 108 NOFVEE( NVEEE ) = NREME
2940 109 END IF
2941 110 C
2942 111 DO 410 INS = 1 , NVECS
2943 112 NOFVES( INS ) = MBL
2944 113 410 CONTINUE
2945 114 NVEES = NVECS
2946 115 IF( NREMS . GT . 0 ) THEN
2947 116 NVEES = NVECS + 1
2948 117 NOFVES( NVEES ) = NREMS
2949 118 END IF
2950 119 C
2951 120 DO 420 INV = 1 , NVECV
2952 121 NOFVEV( INV ) = MBL
2953 122 420 CONTINUE
2954 123 NVEEV = NVECV
2955 124 IF( NREMV . GT . 0 ) THEN
2956 125 NVEEV = NVECV + 1
2957 126 NOFVEV( NVEEV ) = NREMV
2958 127 END IF
2959 128 C
2960 129 PRINT*,NV,NE,NS
2961 130 C
2962 131 C --- EXIT POINT FROM SUBROUTINE -----
2963 132 C
2964 133 C
2965 134 RETURN
2966 135 C
2967 136 C
2968 137 C
2969 138 END

```

```

2970      1      SUBROUTINE RELAXY( IV )
2971      2      IMPLICIT REAL (A-H,O-Z)
2972      3      C
2973      4      C-----I
2974      5      C-----I
2975      6      C      THIS ROUTINE RELAX THE GRID AFTER DELETION
2976      7      C-----I
2977      8      C-----I
2978      9      C
2979     10      include 'cmsh00.h'
2980     11      include 'chyd00.h'
2981     12      include 'cint00.h'
2982     13      include 'cphs10.h'
2983     14      include 'cphs20.h'
2984     15      ITRIG = 0
2985     16      IETRIG = 0
2986     17      IE = JV( 2 , IV )
2987     18      IF( IE . GT . 0 ) THEN
2988     19      C
2989     20      IV1 = JE( 1 , IE )
2990     21      IV2 = JE( 2 , IE )
2991     22      IF( IV1 . EQ . IV ) THEN
2992     23      ISI = JE( 3 , IE )
2993     24      ELSE
2994     25      ISI = JE( 4 , IE )
2995     26      END IF
2996     27      IS = ISI
2997     28      C
2998     29      75      CONTINUE
2999     30      C
3000     31      DO 65 IR = 1 , 3
3001     32      JR = MOD( IR , 3 ) + 1
3002     33      IEA = IABS( JS( JR + 3 , IS ) )
3003     34      IF( IEA . EQ . IE ) THEN
3004     35      IIR = MOD( JR , 3 ) + 4
3005     36      IEI = JS( IIR , IS )
3006     37      IEII = IABS( IEI )
3007     38      IETRIG = IETRIG + 1
3008     39      JECRSS( IETRIG ) = IEII
3009     40      JJR = MOD( JR + 1 , 3 ) + 4
3010     41      IEM = JS( JJR , IS )
3011     42      IER = IABS( IEM )
3012     43      IETRIG = IETRIG + 1
3013     44      JECRSS( IETRIG ) = IER
3014     45      C
3015     46      IV1 = JE( 1 , IER )
3016     47      IV2 = JE( 2 , IER )
3017     48      IF( IV1 . EQ . IV ) THEN
3018     49      ISR = JE( 3 , IER )
3019     50      ITRIG = ITRIG + 1
3020     51      IICOLR( ITRIG ) = IV2
3021     52      JSCRSS( ITRIG ) = ISR
3022     53      ELSE
3023     54      ISR = JE( 4 , IER )
3024     55      ITRIG = ITRIG + 1
3025     56      IICOLR( ITRIG ) = IV1
3026     57      JSCRSS( ITRIG ) = ISR
3027     58      END IF
3028     59      END IF
3029     60      65      CONTINUE
3030     61      C
3031     62      IF( ISR . NE . ISI ) THEN
3032     63      IS = ISR
3033     64      IE = IER
3034     65      GO TO 75
3035     66      END IF
3036     67      C
3037     68      DO 510 IE = 1 , ITRIG
3038     69      C
3039     70      IEM = MOD( IE - 1 , ITRIG ) + 1
3040     71      IEP = MOD( IE , ITRIG ) + 1
3041     72      IEI = MOD( IE + 1 , ITRIG ) + 1
3042     73      C
3043     74      IV1 = IICOLR( IEM )

```

```

3044 75      IV2 = IICOLR( IEP )
3045 76      IV3 = IICOLR( IEI )
3046 77      C
3047 78      X1 = XV( 1 , IV1 ) - XV( 1 , IV2 )
3048 79      Y1 = XV( 2 , IV1 ) - XV( 2 , IV2 )
3049 80      X2 = XV( 1 , IV3 ) - XV( 1 , IV2 )
3050 81      Y2 = XV( 2 , IV3 ) - XV( 2 , IV2 )
3051 82      XSIN = ( X2 * Y1 - X1 * Y2 )
3052 83      XCOS = ( X1 * X2 + Y1 * Y2 )
3053 84      ANGLE( IE ) = XSIN / ( ABS( XCOS ) + 1.E-7 )
3054 85      IF( ANGLE( IE ) .LT. 0. ) RETURN
3055 86      C
3056 87      510 CONTINUE
3057 88      C
3058 89      XSUM = 0.
3059 90      YSUM = 0.
3060 91      HSUMR = 0.
3061 92      HSUMU = 0.
3062 93      HSUMV = 0.
3063 94      HSUMP = 0.
3064 95      HSUMG = 0.
3065 96      C
3066 97      DO 110 IT = 1 , ITRIG
3067 98      IVV = IICOLR( IT )
3068 99      C
3069 100     XSUM = XSUM + XV( 1 , IVV )
3070 101     YSUM = YSUM + XV( 2 , IVV )
3071 102     C
3072 103     HSUMR = HSUMR + HYDVVV( IVV , 1 )
3073 104     HSUMU = HSUMU + HYDVVV( IVV , 2 )
3074 105     HSUMV = HSUMV + HYDVVV( IVV , 3 )
3075 106     HSUMP = HSUMP + HYDVVV( IVV , 4 )
3076 107     HSUMG = HSUMG + HYDVVV( IVV , 5 )
3077 108     110 CONTINUE
3078 109     C
3079 110     XINVRG = 1. / ITRIG
3080 111     XV( 1 , IV ) = XSUM * XINVRG
3081 112     XV( 2 , IV ) = YSUM * XINVRG
3082 113     HYDVVV( IV , 1 ) = HSUMR * XINVRG
3083 114     HYDVVV( IV , 2 ) = HSUMU * XINVRG
3084 115     HYDVVV( IV , 3 ) = HSUMV * XINVRG
3085 116     HYDVVV( IV , 4 ) = HSUMP * XINVRG
3086 117     HYDVVV( IV , 5 ) = HSUMG * XINVRG
3087 118     C
3088 119     ELSE
3089 120     C
3090 121     IE = - IE
3091 122     IV1 = JE( 1 , IE )
3092 123     IV2 = JE( 2 , IE )
3093 124     IF( IV1 .EQ. IV ) THEN
3094 125     ISI = JE( 3 , IE )
3095 126     ITRIG = ITRIG + 1
3096 127     JSCRSS( ITRIG ) = ISI
3097 128     IICOLR( ITRIG ) = IV2
3098 129     ELSE
3099 130     ISI = JE( 4 , IE )
3100 131     ITRIG = ITRIG + 1
3101 132     JSCRSS( ITRIG ) = ISI
3102 133     IICOLR( ITRIG ) = IV1
3103 134     END IF
3104 135     C
3105 136     IS = ISI
3106 137     ISI = 0
3107 138     IIE = IE
3108 139     IETRIG = IETRIG + 1
3109 140     JECRSS( IETRIG ) = IE
3110 141     C
3111 142     670 CONTINUE
3112 143     C
3113 144     DO 680 IR = 1 , 3
3114 145     JR = MOD( IR , 3 ) + 1
3115 146     IEA = IABS( JS( JR + 3 , IS ) )
3116 147     IF( IEA .EQ. IE ) THEN
3117 148     IIR = MOD( JR , 3 ) + 4

```

```

3118 149      IEI = JS( IIR , IS )
3119 150      IEII = IABS( IEI )
3120 151      IETRIG = IETRIG + 1
3121 152      JECRSS( IETRIG ) = IEII
3122 153      JJR = MOD( JR + 1 , 3 ) + 4
3123 154      IEM = JS( JJR , IS )
3124 155      IER = IABS( IEM )
3125 156      IETRIG = IETRIG + 1
3126 157      JECRSS( IETRIG ) = IER
3127 158      C
3128 159      IV1 = JE( 1 , IER )
3129 160      IV2 = JE( 2 , IER )
3130 161      IF( IV1 .EQ. IV ) THEN
3131 162      ISR = JE( 3 , IER )
3132 163      ITRIG = ITRIG + 1
3133 164      IICOLR( ITRIG ) = IV2
3134 165      JSCRSS( ITRIG ) = ISR
3135 166      ELSE
3136 167      ISR = JE( 4 , IER )
3137 168      ITRIG = ITRIG + 1
3138 169      IICOLR( ITRIG ) = IV1
3139 170      JSCRSS( ITRIG ) = ISR
3140 171      END IF
3141 172      END IF
3142 173      C
3143 174      680 CONTINUE
3144 175      C
3145 176      IF( ISR .NE. ISI ) THEN
3146 177      IS = ISR
3147 178      IE = IER
3148 179      GO TO 670
3149 180      END IF
3150 181      ITRIG = ITRIG - 1
3151 182      C
3152 183      IV1 = JE( 1 , IIE )
3153 184      IV2 = JE( 2 , IIE )
3154 185      C
3155 186      IV3 = JE( 1 , IER )
3156 187      IV4 = JE( 2 , IER )
3157 188      C
3158 189      X1 = XV( 1 , IV1 ) - XV( 1 , IV2 )
3159 190      Y1 = XV( 2 , IV1 ) - XV( 2 , IV2 )
3160 191      X2 = XV( 1 , IV4 ) - XV( 1 , IV3 )
3161 192      Y2 = XV( 2 , IV4 ) - XV( 2 , IV3 )
3162 193      XSIN = ( X2 * Y1 - X1 * Y2 )
3163 194      XCOS = ( X1 * X2 + Y1 * Y2 )
3164 195      XANGLE = XSIN / ( ABS( XCOS ) + 1.E-7 )
3165 196      C
3166 197      IF( ABS( XANGLE ) .GT. 1.E-3 ) RETURN
3167 198      C
3168 199      IVI = IV1
3169 200      IF( IV .EQ. IVI ) IVI = IV2
3170 201      IVL = IV3
3171 202      IF( IV .EQ. IV3 ) IVL = IV4
3172 203      IVTRIG = ITRIG + 1
3173 204      IICOLR( IVTRIG ) = IVL
3174 205      C
3175 206      DO 512 IE = 1 , IVTRIG
3176 207      C
3177 208      IEM = MOD( IE - 1 , IVTRIG ) + 1
3178 209      IEP = MOD( IE , IVTRIG ) + 1
3179 210      IEI = MOD( IE + 1 , IVTRIG ) + 1
3180 211      C
3181 212      IV1 = IICOLR( IEM )
3182 213      IV2 = IICOLR( IEP )
3183 214      IV3 = IICOLR( IEI )
3184 215      C
3185 216      X1 = XV( 1 , IV1 ) - XV( 1 , IV2 )
3186 217      Y1 = XV( 2 , IV1 ) - XV( 2 , IV2 )
3187 218      X2 = XV( 1 , IV3 ) - XV( 1 , IV2 )
3188 219      Y2 = XV( 2 , IV3 ) - XV( 2 , IV2 )
3189 220      XSIN = ( X2 * Y1 - X1 * Y2 )
3190 221      XCOS = ( X1 * X2 + Y1 * Y2 )
3191 222      ANGLE( IE ) = XSIN / ( ABS( XCOS ) + 1.E-7 )

```



```

3192 223      IF( ANGLE( IE ) . LT . 0. ) RETURN
3193 224      C
3194 225      512 CONTINUE
3195 226      C
3196 227      XV( 1 , IV ) = .5 * ( XV( 1 , IV1 ) + XV( 1 , IVL ) )
3197 228      XV( 2 , IV ) = .5 * ( XV( 2 , IV1 ) + XV( 2 , IVL ) )
3198 229      HYDVVV( IV , 1 ) = .5 * ( HYDVVV( IV1 , 1 ) +
3199 230      .           HYDVVV( IVL , 1 ) )
3200 231      HYDVVV( IV , 2 ) = .5 * ( HYDVVV( IV1 , 2 ) +
3201 232      .           HYDVVV( IVL , 2 ) )
3202 233      HYDVVV( IV , 3 ) = .5 * ( HYDVVV( IV1 , 3 ) +
3203 234      .           HYDVVV( IVL , 3 ) )
3204 235      HYDVVV( IV , 4 ) = .5 * ( HYDVVV( IV1 , 4 ) +
3205 236      .           HYDVVV( IVL , 4 ) )
3206 237      HYDVVV( IV , 5 ) = .5 * ( HYDVVV( IV1 , 5 ) +
3207 238      .           HYDVVV( IVL , 5 ) )
3208 239      C
3209 240      END IF
3210 241      C
3211 242      DO 120 ISNN = 1 , ITRIG
3212 243      IJS = JSCRSS( ISNN )
3213 244      C
3214 245      IV1 = JS( 1 , INS )
3215 246      IV2 = JS( 2 , INS )
3216 247      IV3 = JS( 3 , INS )
3217 248      AX = XV( 1 , IV2 ) - XV( 1 , IV1 )
3218 249      AY = XV( 2 , IV2 ) - XV( 2 , IV1 )
3219 250      BX = XV( 1 , IV3 ) - XV( 1 , IV1 )
3220 251      BY = XV( 2 , IV3 ) - XV( 2 , IV1 )
3221 252      XS( 3 , INS ) = 0.5 * ( AX * BY - AY * BX )
3222 253      C
3223 254      SAREA( INS ) = 1. / XS( 3 , INS )
3224 255      HYDFLX( INS , 4 ) = 0.
3225 256      HYDFLX( INS , 1 ) = 0.
3226 257      HYDFLX( INS , 2 ) = 0.
3227 258      KSDELTA( INS ) = 1
3228 259      C
3229 260      XXC = ( XV( 1 , IV1 ) + XV( 1 , IV2 ) + XV( 1 , IV3 ) ) *
3230 261      .           THIRD
3231 262      YYC = ( XV( 2 , IV1 ) + XV( 2 , IV2 ) + XV( 2 , IV3 ) ) *
3232 263      .           THIRD
3233 264      XS( 1 , INS ) = XXC
3234 265      XS( 2 , INS ) = YYC
3235 266      C
3236 267      DO 130 IR = 1 , MHQ
3237 268      HYDV( INS , IR ) = ( HYDVVV( IV1 , IR ) +
3238 269      .           HYDVVV( IV2 , IR ) +
3239 270      .           HYDVVV( IV3 , IR ) ) * THIRD
3240 271      130 CONTINUE
3241 272      C
3242 273      HDUM = 1. / ( HYDV( INS , 1 ) + 1.E-12 )
3243 274      HYDV( INS , 2 ) = HYDV( INS , 2 ) * HDUM
3244 275      HYDV( INS , 3 ) = HYDV( INS , 3 ) * HDUM
3245 276      HYDV( INS , 4 ) = ( HYDV( INS , 4 ) -
3246 277      .           .5 * HYDV( INS , 1 ) ) *
3247 278      .           ( HYDV( INS , 2 ) * HYDV( INS , 2 ) +
3248 279      .           HYDV( INS , 3 ) * HYDV( INS , 3 ) ) *
3249 280      .           ( HYDV( INS , 5 ) - 1. )
3250 281      C
3251 282      120 CONTINUE
3252 283      C
3253 284      DO 140 IENN = 1 , IETRIG
3254 285      IEN = JECRSS( IENN )
3255 286      C
3256 287      JV1 = JE( 1 , IEN )
3257 288      JV2 = JE( 2 , IEN )
3258 289      AX = XV( 1 , JV2 ) - XV( 1 , JV1 )
3259 290      AY = XV( 2 , JV2 ) - XV( 2 , JV1 )
3260 291      XE( 1 , IEN ) = SQRT( AX * AX + AY * AY )
3261 292      XEREV = 1. / XE( 1 , IEN )
3262 293      XN( IEN ) = AY * XEREV
3263 294      YN( IEN ) = - AX * XEREV
3264 295      ISSR = JE( 4 , IEN )
3265 296      ISSL = JE( 3 , IEN )

```

```

3266 297 C
3267 298 IF( JE( 5 , IEN ) . NE . 0 ) THEN
3268 299 C
3269 300 AA = XV( 1 , JV2 ) - XV( 1 , JV1 )
3270 301 BB = XV( 2 , JV2 ) - XV( 2 , JV1 )
3271 302 XEL = XS( 1 , ISSL )
3272 303 YEL = XS( 2 , ISSL )
3273 304 CC = XEL - XV( 1 , JV1 )
3274 305 DD = YEL - XV( 2 , JV1 )
3275 306 EE = ( AA * CC + BB * DD ) * XEREV * XEREV
3276 307 XER = XV( 1 , JV1 ) + AA * EE
3277 308 YER = XV( 2 , JV1 ) + BB * EE
3278 309 AX = XER - XEL
3279 310 AY = YER - YEL
3280 311 XE( 2 , IEN ) = SQRT( AX * AX + AY * AY )
3281 312 XEREV = 1. / XE( 2 , IEN )
3282 313 XXN( IEN ) = AX * XEREV
3283 314 YYN( IEN ) = AY * XEREV
3284 315 XE( 2 , IEN ) = 2. * XE( 2 , IEN )
3285 316 XYMIDL( IEN ) = .5
3286 317 XMIDL( IEN ) = XER
3287 318 YMIDL( IEN ) = YER
3288 319 C
3289 320 ELSE
3290 321 C
3291 322 XER = XS( 1 , ISSR )
3292 323 YER = XS( 2 , ISSR )
3293 324 XEL = XS( 1 , ISSL )
3294 325 YEL = XS( 2 , ISSL )
3295 326 C
3296 327 AA = XV( 1 , JV2 ) - XV( 1 , JV1 )
3297 328 BB = XV( 2 , JV2 ) - XV( 2 , JV1 )
3298 329 CC = XEL - XER
3299 330 DD = YEL - YER
3300 331 ACA = XER - XV( 1 , JV1 )
3301 332 OBD = YER - XV( 2 , JV1 )
3302 333 EE = ( ACA * DD - OBD * CC ) / ( AA * DD - BB * CC )
3303 334 XMIDL( IEN ) = XV( 1 , JV1 ) + AA * EE
3304 335 YMIDL( IEN ) = XV( 2 , JV1 ) + BB * EE
3305 336 C
3306 337 XEMID = XMIDL( IEN ) - XEL
3307 338 YEMID = YMIDL( IEN ) - YEL
3308 339 C
3309 340 AX = XER - XEL
3310 341 AY = YER - YEL
3311 342 XE( 2 , IEN ) = SQRT( AX * AX + AY * AY )
3312 343 XEREV = 1. / XE( 2 , IEN )
3313 344 XXN( IEN ) = AX * XEREV
3314 345 YYN( IEN ) = AY * XEREV
3315 346 C
3316 347 XYMIDL( IEN ) = SQRT( XEMID * XEMID + YEMID * YEMID ) * XEREV
3317 348 C
3318 349 END IF
3319 350 C
3320 351 140 CONTINUE
3321 352 C
3322 353 DO 142 IENN = 1 , IETRIG
3323 354 IE = JECRSS( IENN )
3324 355 CALL RECNC( IE , IDONE , ITL , ITR , JA , JB , JC , JD )
3325 356 CALL RECNC( JA , JADONE , ITL , ITR , JAA , JAB , JAC , JAD )
3326 357 CALL RECNC( JB , JBDONE , ITL , ITR , JBA , JBB , JBC , JBD )
3327 358 CALL RECNC( JC , JCDONE , ITL , ITR , JCA , JCB , JCC , JCD )
3328 359 CALL RECNC( JD , JDDONE , ITL , ITR , JDA , JDB , JDC , JDD )
3329 360 142 CONTINUE
3330 361 C
3331 362 C --- EXIT POINT FROM SUBROUTINE -----
3332 363 C
3333 364 C
3334 365 RETURN
3335 366 C
3336 367 C
3337 368 C
3338 369 END

```

```

3339      1      SUBROUTINE LAPLAC                                3339
3340      2      C                                                3340
3341      3      C-----|                                         3341
3342      4      C                                                3342
3343      5      C          LAPLAC COMPUTE THE LAPLACIAN FOR GRID ADAPTATION | 3343
3344      6      C-----|                                         3344
3345      7      C                                                3345
3346      8      C                                                3346
3347      9      include      'cmsh00.h'                            3347
3348     10      include      'chyd00.h'                            3348
3349     11      include      'cint00.h'                            3349
3350     12      include      'cphs10.h'                           3350
3351     13      include      'cphs20.h'                           3351
3352     14      C                                                3352
3353     15      REAL RRMIDL(MBP),PPMIDL(MBP)                       3353
3354     16      REAL ROR(3),UOR(3),VOR(3),POR(3)                 3354
3355     17      REAL ROL(3),UOL(3),VOL(3),POL(3)                 3355
3356     18      C                                                3356
3357     19      EPSLON = .025                                       3357
3358     20      C                                                3358
3359     21      DO 120 IS = 1 , NS                                  3359
3360     22          RR( IS ) = 0.                                     3360
3361     23          RL( IS ) = 0.                                     3361
3362     24      120 CONTINUE                                       3362
3363     25      C                                                3363
3364     26      C --- BEGIN LOOP OVER ALL EDGES IN THE DOMAIN ----- 3364
3365     27      C                                                3365
3366     28          NE1 = 1                                          3366
3367     29          NE2 = NOFVEE( 1 )                               3367
3368     30          DO 90 INE = 1 , NVEEE                           3368
3369     31      C                                                3369
3370     32      C --- FETCH HYDRO QUANTITIES -----                3370
3371     33      C                                                3371
3372     34          DO 105 IE = NE1 , NE2                           3372
3373     35          KE = IE - NE1 + 1                               3373
3374     36      C                                                3374
3375     37          ISL = JE( 3 , IE )                              3375
3376     38          ISR = JE( 4 , IE )                              3376
3377     39      C                                                3377
3378     40          IF( JE ( 5 , IE ) . EQ . 0 ) THEN              3378
3379     41      C                                                3379
3380     42          RRMIDL = XYMIDL( IE ) * ( RGRAD( ISR , 1 ) -    3380
3381     43          . RGRAD( ISL , 1 ) ) + RGRAD( ISL , 1 )        3381
3382     44          RLMIDL = XYMIDL( IE ) * ( RGRAD( ISR , 2 ) -    3382
3383     45          . RGRAD( ISL , 2 ) ) + RGRAD( ISL , 2 )        3383
3384     46          PRMIDL = XYMIDL( IE ) * ( PGRAD( ISR , 1 ) -    3384
3385     47          . PGRAD( ISL , 1 ) ) + PGRAD( ISL , 1 )        3385
3386     48          PLMIDL = XYMIDL( IE ) * ( PGRAD( ISR , 2 ) -    3386
3387     49          . PGRAD( ISL , 2 ) ) + PGRAD( ISL , 2 )        3387
3388     50      C                                                3388
3389     51          ELSE                                           3389
3390     52      C                                                3390
3391     53          RRMIDL = RGRAD( ISL , 1 )                       3391
3392     54          RLMIDL = RGRAD( ISL , 2 )                       3392
3393     55          PRMIDL = PGRAD( ISL , 1 )                       3393
3394     56          PLMIDL = PGRAD( ISL , 2 )                       3394
3395     57      C                                                3395
3396     58          END IF                                          3396
3397     59      C                                                3397
3398     60          RRMIDL( KE ) = ( RRMIDL * XN( IE ) + RLMIDL * YN( IE ) ) * 3398
3399     61          . XE( 1 , IE )                                  3399
3400     62          PPMIDL( KE ) = ( PRMIDL * XN( IE ) + PLMIDL * YN( IE ) ) * 3400
3401     63          . XE( 1 , IE )                                  3401
3402     64      C                                                3402
3403     65      105 CONTINUE                                       3403
3404     66      C                                                3404
3405     67          DO 130 IE = NE1 , NE2                           3405
3406     68          KE = IE - NE1 + 1                               3406
3407     69      C                                                3407
3408     70          ISL = JE( 3 , IE )                              3408
3409     71          ISR = JE( 4 , IE )                              3409
3410     72      C                                                3410
3411     73          IF( JE( 5 , IE ) . EQ . 0 ) THEN              3411
3412     74      C                                                3412

```

```

3413 75      RR( ISL ) = RR( ISL ) + RRMIDL( KE )      3413
3414 76      RR( ISR ) = RR( ISR ) - RRMIDL( KE )      3414
3415 77      RL( ISL ) = RL( ISL ) + PPMIDL( KE )      3415
3416 78      RL( ISR ) = RL( ISR ) - PPMIDL( KE )      3416
3417 79      C                                          3417
3418 80      ELSE                                          3418
3419 81      C                                          3419
3420 82      RR( ISL ) = RR( ISL ) + RRMIDL( KE )      3420
3421 83      RL( ISL ) = RL( ISL ) + PPMIDL( KE )      3421
3422 84      C                                          3422
3423 85      END IF                                        3423
3424 86      C                                          3424
3425 87      130 CONTINUE                                3425
3426 88      C                                          3426
3427 89      NE1 = NE2 + 1                                3427
3428 90      NE2 = NE2 + NOFVEE( INE + 1 )              3428
3429 91      90 CONTINUE                                3429
3430 92      C                                          3430
3431 93      DO 135 IS = 1 , NS                            3431
3432 94      ZRR = ABS( RR( IS ) ) * SAREA( IS )          3432
3433 95      ZPR = ABS( RL( IS ) ) * SAREA( IS )          3433
3434 96      RR( IS ) = ZRR * SAREVG                      3434
3435 97      RL( IS ) = ZPR * SAREVG                      3435
3436 98      135 CONTINUE                                3436
3437 99      C                                          3437
3438 100     DO 140 IS = 1 , NS                            3438
3439 101     ZRL = ( RGRAD( IS , 1 ) * RGRAD( IS , 1 ) +  3439
3440 102     .      RGRAD( IS , 2 ) * RGRAD( IS , 2 ) ) * SAREVG 3440
3441 103     ZPL = ( PGRAD( IS , 1 ) * PGRAD( IS , 1 ) +  3441
3442 104     .      PGRAD( IS , 2 ) * PGRAD( IS , 2 ) ) * SAREVG 3442
3443 105     ZRR = ABS( HYDV( IS , 1 ) ) * EPSLON          3443
3444 106     ZPP = ABS( HYDV( IS , 4 ) ) * EPSLON          3444
3445 107     RR( IS ) = RR( IS ) / ( ZRL + ZRR )          3445
3446 108     RL( IS ) = RL( IS ) / ( ZPL + ZPP )          3446
3447 109     140 CONTINUE                                3447
3448 110     C                                          3448
3449 111     C --- EXIT POINT FROM SUBROUTINE ----- 3449
3450 112     C                                          3450
3451 113     C                                          3451
3452 114     RETURN                                        3452
3453 115     C                                          3453
3454 116     C                                          3454
3455 117     C                                          3455
3456 118     ---                                        3456
3456 118     END                                          3456

```

```

3457 1      SUBROUTINE RECNC( IE , IDONE , ITL , ITR , JA , JB , JC , JD )      3457
3458 2      IMPLICIT REAL (A-H,O-Z)                                           3458
3459 3      C                                                                    3459
3460 4      C-----I                                                         3460
3461 5      C                                                                    I   3461
3462 6      C      THIS ROUTINE CHECKS FOR RECONNECTION OF EDGE NUMBER IE      I   3462
3463 7      C      TO GET A BETTER CONNECTIVITY BETWEEN ADJACENT TRIANGLES      I   3463
3464 8      C      USED AFTER ADDITION AND DELETION                            I   3464
3465 9      C                                                                    I   3465
3466 10     C-----I                                                         3466
3467 11     C                                                                    3467
3468 12     include      'cmsh00.h'                                           3468
3469 13     include      'chyd00.h'                                           3469
3470 14     include      'cint00.h'                                           3470
3471 15     include      'cphs10.h'                                           3471
3472 16     include      'cphs20.h'                                           3472
3473 17     C                                                                    3473
3474 18     EROR = 1.0E-3                                                       3474
3475 19     C                                                                    3475
3476 20     IDONE = 0                                                           3476
3477 21     IF( IE .EQ. 0 ) RETURN                                             3477
3478 22     IF( JE( 5 , IE ) .NE. 0 ) RETURN                                    3478
3479 23     ITR = JE( 4 , IE )                                                 3479
3480 24     ITL = JE( 3 , IE )                                                 3480
3481 25     C                                                                    3481
3482 26     C      IDENTIFY VERTICES                                           3482
3483 27     C                                                                    3483
3484 28     I1 = JE( 1 , IE )                                                  3484
3485 29     I2 = JE( 2 , IE )                                                  3485
3486 30     DO 1 IV = 1 , 3                                                    3486
3487 31     ID = JS( IV , ITL )                                               3487
3488 32     IF( ID .NE. I1 .AND. ID .NE. I2 ) THEN                            3488
3489 33     I4 = ID                                                            3489
3490 34     IV4 = IV                                                           3490
3491 35     END IF                                                            3491
3492 36     1 CONTINUE                                                         3492
3493 37     C                                                                    3493
3494 38     DO 3 IV = 1 , 3                                                    3494
3495 39     ID = JS( IV , ITR )                                               3495
3496 40     IF( ID .NE. I1 .AND. ID .NE. I2 ) THEN                            3496
3497 41     I3 = ID                                                            3497
3498 42     IV3 = IV                                                           3498
3499 43     END IF                                                            3499
3500 44     3 CONTINUE                                                         3500
3501 45     C      IT MAY HAPPEN THAT I3 IS I4.                                3501
3502 46     IF( I3 .EQ. I4 ) GO TO 999                                         3502
3503 47     C                                                                    3503
3504 48     C      COMPARE OPPOSING ANGLE PAIRS IN THE QUADRILATERAL AND RECONNECT TO 3504
3505 49     C      PRESERVE DIAGONAL DOMINANCE OF THE POISSON SOLVER.         3505
3506 50     C                                                                    3506
3507 51     AX = XV( 1 , I3 ) - XV( 1 , I1 )                                   3507
3508 52     AY = XV( 2 , I3 ) - XV( 2 , I1 )                                   3508
3509 53     BX = XV( 1 , I4 ) - XV( 1 , I1 )                                   3509
3510 54     BY = XV( 2 , I4 ) - XV( 2 , I1 )                                   3510
3511 55     CX = XV( 1 , I4 ) - XV( 1 , I2 )                                   3511
3512 56     CY = XV( 2 , I4 ) - XV( 2 , I2 )                                   3512
3513 57     DX = XV( 1 , I3 ) - XV( 1 , I2 )                                   3513
3514 58     DY = XV( 2 , I3 ) - XV( 2 , I2 )                                   3514
3515 59     AI2 = AX * BY - AY * BX                                           3515
3516 60     AI1 = CX * DY - CY * DX                                           3516
3517 61     XLN = XE( 1 , IE )                                                 3517
3518 62     ROUND F = EROR * XLN * XLN                                         3518
3519 63     C                                                                    3519
3520 64     C      IA IS BETWEEN I1 AND I3                                       3520
3521 65     C      IB IS BETWEEN I1 AND I4                                       3521
3522 66     C      IC IS BETWEEN I2 AND I4                                       3522
3523 67     C      ID IS BETWEEN I2 AND I3                                       3523
3524 68     C                                                                    3524
3525 69     IB = JS( IV4 + 3 , ITL )                                           3525
3526 70     ID = JS( IV3 + 3 , ITR )                                           3526
3527 71     IV4 = MOD( IV4 + 1 , 3 ) + 1                                       3527
3528 72     IV3 = MOD( IV3 + 1 , 3 ) + 1                                       3528
3529 73     IC = JS( IV4 + 3 , ITL )                                           3529
3530 74     IA = JS( IV3 + 3 , ITR )                                           3530

```

```

3531 75 C
3532 76 JB = IABS( IB )
3533 77 JD = IABS( ID )
3534 78 JA = IABS( IA )
3535 79 JC = IABS( IC )
3536 80 IF( AI2 . LT . ROUND( . OR . AI1 . LT . ROUND ) RETURN
3537 81 C
3538 82 XL1 = XE( 1 , JA )
3539 83 XL2 = XE( 1 , JB )
3540 84 XL3 = XE( 1 , JC )
3541 85 XL4 = XE( 1 , JD )
3542 86 C
3543 87 XX = XV( 1 , I3 ) - XV( 1 , I4 )
3544 88 YY = XV( 2 , I3 ) - XV( 2 , I4 )
3545 89 XLL = SQRT( XX * XX + YY * YY )
3546 90 C
3547 91 AREATL = SAREA( ITL )
3548 92 AREATR = SAREA( ITR )
3549 93 ASP2 = AREATL * XL2 * XL2
3550 94 ASP3 = AREATL * XL3 * XL3
3551 95 ASPTL = AREATL * XLN * XLN
3552 96 ASP1 = AREATR * XL1 * XL1
3553 97 ASP4 = AREATR * XL4 * XL4
3554 98 ASPTR = AREATR * XLN * XLN
3555 99 ASPN = AMAX1( ASPTL , ASPTR , ASP1 , ASP2 , ASP3 , ASP4 )
3556 100 C
3557 101 XSISR = 0.5 * AI2
3558 102 XSINSR = 1. / XSISR
3559 103 C
3560 104 XSISL = 0.5 * AI1
3561 105 XSINSL = 1. / XSISL
3562 106 C
3563 107 ASP2 = XSINSR * XL2 * XL2
3564 108 ASP1 = XSINSR * XL1 * XL1
3565 109 ASPSR = XSINSR * XLL * XLL
3566 110 ASP3 = XSINSL * XL3 * XL3
3567 111 ASP4 = XSINSL * XL4 * XL4
3568 112 ASPSL = XSINSL * XLL * XLL
3569 113 ASPL = AMAX1( ASPSL , ASPSR , ASP1 , ASP2 , ASP3 , ASP4 )
3570 114 C
3571 115 IF( ASPN . LT . ASPL ) RETURN
3572 116 C YES, REDRAW LINE- THE OLD CONNECTION VIOLATES DIAGONAL DOMINANCE.
3573 117 C DRAW LINE DIRECTED FROM I4 TO I3
3574 118 C WE HAVE LEFT JE( 3 , IE ) THE SAME SINCE IE IS STILL INTERNAL.
3575 119 IDONE = 1
3576 120 JE( 1 , IE ) = I4
3577 121 JE( 2 , IE ) = I3
3578 122 XE( 1 , IE ) = XLL
3579 123 C
3580 124 C ITR IS STILL TO THE RIGHT, ITL TO THE LEFT OF THE NEW LINE IE .
3581 125 C FIND THE OTHER DIRECTED LINE SEGMENTS
3582 126 C
3583 127 DO 30 I = 1 , 2
3584 128 IM5 = 5 - I
3585 129 IF( JE( IM5 , JB ) . NE . ITL ) GO TO 26
3586 130 JE( IM5 , JB ) = ITR
3587 131 26 CONTINUE
3588 132 IF( JE( IM5 , JD ) . NE . ITR ) GO TO 28
3589 133 JE( IM5 , JD ) = ITL
3590 134 28 CONTINUE
3591 135 30 CONTINUE
3592 136 C
3593 137 C RESET JS( 1 - 6 , ITL AND ITR )
3594 138 C START BOTH TRIANGLES AT I4 WITH ( AND PUT IN COUNTERCLOCKWISE
3595 139 C MANNER)
3596 140 JS( 4 , ITR ) = IB
3597 141 JS( 5 , ITR ) = IA
3598 142 JS( 6 , ITR ) = - IE
3599 143 JS( 1 , ITR ) = I4
3600 144 JS( 2 , ITR ) = I1
3601 145 JS( 3 , ITR ) = I3
3602 146 JS( 4 , ITL ) = IE
3603 147 JS( 5 , ITL ) = IO
3604 148 JS( 6 , ITL ) = IC

```

```

3605 149      JS( 1 , ITL ) = I4      3605
3606 150      JS( 2 , ITL ) = I3      3606
3607 151      JS( 3 , ITL ) = I2      3607
3608 152      C      3608
3609 153      IF( JV( 2 , I1 ) . GT . 0 ) JV( 2 , I1 ) = JA      3609
3610 154      IF( JV( 2 , I2 ) . GT . 0 ) JV( 2 , I2 ) = JC      3610
3611 155      C      3611
3612 156      XEL = ( XV( 1 , I3 ) + XV( 1 , I2 ) + XV( 1 , I4 ) ) * THIRD      3612
3613 157      YEL = ( XV( 2 , I3 ) + XV( 2 , I2 ) + XV( 2 , I4 ) ) * THIRD      3613
3614 158      XER = ( XV( 1 , I3 ) + XV( 1 , I1 ) + XV( 1 , I4 ) ) * THIRD      3614
3615 159      YER = ( XV( 2 , I3 ) + XV( 2 , I1 ) + XV( 2 , I4 ) ) * THIRD      3615
3616 160      C      3616
3617 161      DO 92 IR = 1 , MHQ      3617
3618 162      HYDV( ITL , IR ) = ( HYDVVV( I3 , IR ) +      3618
3619 163      .      HYDVVV( I2 , IR ) +      3619
3620 164      .      HYDVVV( I4 , IR ) ) * THIRD      3620
3621 165      C      3621
3622 166      HYDV( ITR , IR ) = ( HYDVVV( I3 , IR ) +      3622
3623 167      .      HYDVVV( I1 , IR ) +      3623
3624 168      .      HYDVVV( I4 , IR ) ) * THIRD      3624
3625 169      92      CONTINUE      3625
3626 170      C      3626
3627 171      HDUM      = 1. / ( HYDV( ITL , 1 ) + 1.E-12 )      3627
3628 172      HYDV( ITL , 2 ) = HYDV( ITL , 2 ) * HDUM      3628
3629 173      HYDV( ITL , 3 ) = HYDV( ITL , 3 ) * HDUM      3629
3630 174      HYDV( ITL , 4 ) = ( HYDV( ITL , 4 ) -      3630
3631 175      .      .5 * HYDV( ITL , 1 ) ) *      3631
3632 176      .      ( HYDV( ITL , 2 ) * HYDV( ITL , 2 ) +      3632
3633 177      .      HYDV( ITL , 3 ) * HYDV( ITL , 3 ) ) *      3633
3634 178      .      ( HYDV( ITL , 5 ) - 1. )      3634
3635 179      C      3635
3636 180      HDUM      = 1. / ( HYDV( ITR , 1 ) + 1.E-12 )      3636
3637 181      HYDV( ITR , 2 ) = HYDV( ITR , 2 ) * HDUM      3637
3638 182      HYDV( ITR , 3 ) = HYDV( ITR , 3 ) * HDUM      3638
3639 183      HYDV( ITR , 4 ) = ( HYDV( ITR , 4 ) -      3639
3640 184      .      .5 * HYDV( ITR , 1 ) ) *      3640
3641 185      .      ( HYDV( ITR , 2 ) * HYDV( ITR , 2 ) +      3641
3642 186      .      HYDV( ITR , 3 ) * HYDV( ITR , 3 ) ) *      3642
3643 187      .      ( HYDV( ITR , 5 ) - 1. )      3643
3644 188      C      3644
3645 189      RGRAD1 = RGRAD( ITL , 1 ) + RGRAD( ITR , 1 )      3645
3646 190      RGRAD2 = RGRAD( ITL , 2 ) + RGRAD( ITR , 2 )      3646
3647 191      RGRAD( ITL , 1 ) = .5 * RGRAD1      3647
3648 192      RGRAD( ITR , 1 ) = .5 * RGRAD1      3648
3649 193      RGRAD( ITL , 2 ) = .5 * RGRAD2      3649
3650 194      RGRAD( ITR , 2 ) = .5 * RGRAD2      3650
3651 195      C      3651
3652 196      UGRAD1 = UGRAD( ITL , 1 ) + UGRAD( ITR , 1 )      3652
3653 197      UGRAD2 = UGRAD( ITL , 2 ) + UGRAD( ITR , 2 )      3653
3654 198      UGRAD( ITL , 1 ) = .5 * UGRAD1      3654
3655 199      UGRAD( ITR , 1 ) = .5 * UGRAD1      3655
3656 200      UGRAD( ITL , 2 ) = .5 * UGRAD2      3656
3657 201      UGRAD( ITR , 2 ) = .5 * UGRAD2      3657
3658 202      C      3658
3659 203      VGRAD1 = VGRAD( ITL , 1 ) + VGRAD( ITR , 1 )      3659
3660 204      VGRAD2 = VGRAD( ITL , 2 ) + VGRAD( ITR , 2 )      3660
3661 205      VGRAD( ITL , 1 ) = .5 * VGRAD1      3661
3662 206      VGRAD( ITR , 1 ) = .5 * VGRAD1      3662
3663 207      VGRAD( ITL , 2 ) = .5 * VGRAD2      3663
3664 208      VGRAD( ITR , 2 ) = .5 * VGRAD2      3664
3665 209      C      3665
3666 210      PGRAD1 = PGRAD( ITL , 1 ) + PGRAD( ITR , 1 )      3666
3667 211      PGRAD2 = PGRAD( ITL , 2 ) + PGRAD( ITR , 2 )      3667
3668 212      PGRAD( ITL , 1 ) = .5 * PGRAD1      3668
3669 213      PGRAD( ITR , 1 ) = .5 * PGRAD1      3669
3670 214      PGRAD( ITL , 2 ) = .5 * PGRAD2      3670
3671 215      PGRAD( ITR , 2 ) = .5 * PGRAD2      3671
3672 216      C      3672
3673 217      XS( 1 , ITL ) = XEL      3673
3674 218      XS( 2 , ITL ) = YEL      3674
3675 219      XS( 1 , ITR ) = XER      3675
3676 220      XS( 2 , ITR ) = YER      3676
3677 221      C      3677
3678 222      XS( 3 , ITR ) = XSISR      3678

```

3679	223		XS(3 , ITL) = XSISL	3679
3680	224	C		3680
3681	225		SAREA(ITL) = XSINSL	3681
3682	226		SAREA(ITR) = XSINSR	3682
3683	227	C		3683
3684	228		JEN(1) = JA	3684
3685	229		JEN(2) = JB	3685
3686	230		JEN(3) = JC	3686
3687	231		JEN(4) = JD	3687
3688	232		JEN(5) = IE	3688
3689	233	C		3689
3690	234		DO 80 IENN = 1 , 5	3690
3691	235		IEN = JEN(IENN)	3691
3692	236		JV1 = JE(1 , IEN)	3692
3693	237		JV2 = JE(2 , IEN)	3693
3694	238		AX = XV(1 , JV2) - XV(1 , JV1)	3694
3695	239		AY = XV(2 , JV2) - XV(2 , JV1)	3695
3696	240		XEREV = 1. / XE(1 , IEN)	3696
3697	241		XN(IEN) = AY * XEREV	3697
3698	242		YN(IEN) = - AX * XEREV	3698
3699	243		ISSR = JE(4 , IEN)	3699
3700	244		ISSL = JE(3 , IEN)	3700
3701	245		IJE5 = JE(5 , IEN)	3701
3702	246		IF(IJE5 . NE . 0) THEN	3702
3703	247	C		3703
3704	248		AA = XV(1 , JV2) - XV(1 , JV1)	3704
3705	249		BB = XV(2 , JV2) - XV(2 , JV1)	3705
3706	250		XEL = XS(1 , ISSL)	3706
3707	251		YEL = XS(2 , ISSL)	3707
3708	252		CC = XEL - XV(1 , JV1)	3708
3709	253		DD = YEL - XV(2 , JV1)	3709
3710	254		EE = (AA * CC + BB * DD) * XEREV * XEREV	3710
3711	255		XER = XV(1 , JV1) + AA * EE	3711
3712	256		YER = XV(2 , JV1) + BB * EE	3712
3713	257		AX = XER - XEL	3713
3714	258		AY = YER - YEL	3714
3715	259		XE(2 , IEN) = SQRT(AX * AX + AY * AY)	3715
3716	260		XEREV = 1. / XE(2 , IEN)	3716
3717	261		XXN(IEN) = AX * XEREV	3717
3718	262		YYN(IEN) = AY * XEREV	3718
3719	263		XE(2 , IEN) = 2. * XE(2 , IEN)	3719
3720	264		XYMIDL(IEN) = .5	3720
3721	265		XMIDL(IEN) = XER	3721
3722	266		YMIDL(IEN) = YER	3722
3723	267	C		3723
3724	268		ELSE	3724
3725	269	C		3725
3726	270		XER = XS(1 , ISSR)	3726
3727	271		YER = XS(2 , ISSR)	3727
3728	272		XEL = XS(1 , ISSL)	3728
3729	273		YEL = XS(2 , ISSL)	3729
3730	274	C		3730
3731	275		AA = XV(1 , JV2) - XV(1 , JV1)	3731
3732	276		BB = XV(2 , JV2) - XV(2 , JV1)	3732
3733	277		CC = XEL - XER	3733
3734	278		DD = YEL - YER	3734
3735	279		ACA = XER - XV(1 , JV1)	3735
3736	280		OBD = YER - XV(2 , JV1)	3736
3737	281		EE = (ACA * DD - OBD * CC) / (AA * DD - BB * CC)	3737
3738	282		XMIDL(IEN) = XV(1 , JV1) + AA * EE	3738
3739	283		YMIDL(IEN) = XV(2 , JV1) + BB * EE	3739
3740	284	C		3740
3741	285		XEMID = XMIDL(IEN) - XEL	3741
3742	286		YEMID = YMIDL(IEN) - YEL	3742
3743	287	C		3743
3744	288		AX = XER - XEL	3744
3745	289		AY = YER - YEL	3745
3746	290		XE(2 , IEN) = SQRT(AX * AX + AY * AY)	3746
3747	291		XEREV = 1. / XE(2 , IEN)	3747
3748	292		XXN(IEN) = AX * XEREV	3748
3749	293		YYN(IEN) = AY * XEREV	3749
3750	294	C		3750
3751	295		XYMIDL(IEN) = SQRT(XEMID * XEMID + YEMID * YEMID) * XEREV	3751
3752	296	C		3752


```

3753 297      END IF                                3753
3754 298      C                                      3754
3755 299      80  CONTINUE                          3755
3756 300      C                                      3756
3757 301      RETURN                                3757
3758 302      C                                      3758
3759 303      999  WRITE (6,1000) IE                3759
3760 304      C                                      3760
3761 305      C --- EXIT POINT FROM SUBROUTINE ----- 3761
3762 306      C                                      3762
3763 307      C -----                              3763
3764 308      RETURN                                3764
3765 309      C -----                              3765
3766 310      C                                      3766
3767 311      C --- FORMATS -----                  3767
3768 312      C                                      3768
3769 313      1000 FORMAT('OITS ABOUT TO BOMB--RECNC ON EDGE ',I5) 3769
3770 314      C                                      3770
3771 315      C ---                                  3771
3772 316      END                                    3772

```

```

3773 1      SUBROUTINE EOS (RRR,EEE,N,GAMMA)        3773
3774 2      C                                      3774
3775 3      C -----|                              3775
3776 4      C |                                       3776
3777 5      C | AIR IS ASSUMED TO BE CALORICALLY IMPERFECT, THERMALLY 3777
3778 6      C | PERFECT. THEREFORE, INCLUDE IMPERFECTIONS VIA A VARIABLE 3778
3779 7      C | GAMMA DEPENDENT ON DENSITY AND INTERNAL ENERGY.      3779
3780 8      C | THIS ROUTINE PERFORMS A TABLE LOOK UP FOR GAMMA.      3780
3781 9      C |                                       3781
3782 10     C -----|                              3782
3783 11     C                                       3783
3784 12     C INPUT VARIABLE DEFINITIONS.                3784
3785 13     C RRR = MASS DENSITY                          3785
3786 14     C EEE = INTERNAL ENERGY PER UNIT VOLUME     3786
3787 15     C (CONVERTED FOR INTERNAL *CALL TO ENERGY PER UNIT MASS) 3787
3788 16     C N = NUMBER OF ENTRIES IN ARRAYS RRR & EEE 3788
3789 17     C                                       3789
3790 18     C PARAMETER (M = 64 )                          3790
3791 19     C                                       3791
3792 20     C DIMENSION RRR(N), EEE(N), GAMMA(N)          3792
3793 21     C DIMENSION T11(M), T12(M), T21(M), T22(M), RHO(M), E(M)    3793
3794 22     C DIMENSION OMP(M), Q(M), I(M), J(M)          3794
3795 23     C DIMENSION G1(168),G2(112),G3(112),G4(112),G5(112),      3795
3796 24     C | G6(112),G7(112),GF(840)                   3796
3797 25     C |                                       3797
3798 26     C NOTE: THE TABLE LOOK UP TREATS ARRAY GF AS THOUGH IT    3798
3799 27     C WERE DIMENSIONED (8,105).                    3799
3800 28     C                                       3800
3801 29     C EQUIVALENCE (G1(1),GF( 1)), (G2(1),GF(169)), (G3(1),GF(281)), 3801
3802 30     C | (G4(1),GF(393)), (G5(1),GF(505)), (G6(1),GF(617)),      3802
3803 31     C | (G7(1),GF(729))                             3803
3804 32     C |                                       3804
3805 33     C DATA XL16E /2.7725887222397744835689081810414791107177734375/ 3805
3806 34     C -----|                              3806
3807 35     C G = GAMMA - 1.0 IS STORED FOR 32 BIT WORD MACHINES IN POWERS OF 3807
3808 36     C 16 ACROSS FOR MASS DENSITY VARIATION AND INTERMEDIATE VALUES 3808
3809 37     C 1 - 16 FOR POWERS OF 16 VERTICALLY WHICH REPRESENT THE INTERNAL 3809
3810 38     C ENERGY VARIATION.                            3810
3811 39     C |                                       3811
3812 40     C 16**(2) .GE. RHO .GE. 16**(-6)                3812
3813 41     C 16**(15) .GE. E .GE. 16**(8)                  3813
3814 42     C -----|                              3814
3815 43     C DATA G1 /8*.4222,8*.4152,8*.4110,8*.4081,8*.4058,8*.4040, 3815
3816 44     C | 8*.4024,8*.4011,8*.3998,8*.3988,8*.3978,8*.3969,      3816
3817 45     C | 8*.3961,8*.3953,8*.3935,8*.3918,              3817
3818 46     C | .3723,.3715,.3707,.3699,.3690,.3680,.3663,.3637,      3818
3819 47     C | .3555,.3538,.3522,.3502,.3476,.3430,.3344,.3238,      3819
3820 48     C | .3370,.3370,.3370,.3364,.3347,.3277,.3099,.2885,      3820
3821 49     C | .3257,.3227,.3201,.3134,.3062,.3014,.2884,.2591,      3821
3822 50     C | .3166,.3110,.3063,.2946,.2831,.2783,.2677,.2358/      3822
3823 51     C DATA G2/.3111,.3006,.2940,.2787,.2635,.2588,.2502,.2236, 3823

```

3824	52		.3075, .2906, .2810, .2665, .2466, .2418, .2350, .2131,	3824
3825	53		.3043, .2819, .2695, .2554, .2317, .2269, .2216, .2038,	3825
3826	54		.2929, .2740, .2593, .2455, .2206, .2136, .2097, .1955,	3826
3827	55		.2840, .2672, .2500, .2366, .2166, .2015, .1988, .1879,	3827
3828	56		.2764, .2611, .2429, .2285, .2125, .1890, .1890, .1811,	3828
3829	57		.2714, .2555, .2384, .2210, .2079, .1818, .1799, .1747,	3829
3830	58		.2669, .2504, .2343, .2141, .2037, .1822, .1709, .1689,	3830
3831	59		.2624, .2473, .2304, .2096, .1998, .1828, .1684, .1639,	3831
3832	60		.2599, .2446, .2268, .2087, .1961, .1834, .1673, .1601,	3832
3833	61		.2401, .2191, .1972, .1775, .1592, .1444, .1358, .1203,	3833
3834	62		.2002, .1960, .1749, .1536, .1376, .1252, .1107, .1044,	3834
3835	63		.1911, .1829, .1633, .1420, .1266, .1101, .1012, .0933,	3835
3836	64		.1950, .1781, .1566, .1415, .1241, .1118, .1009, .0948/	3836
3837	65		DATA G3/.2001, .1789, .1594, .1443, .1306, .1189, .1095, .1013,	3837
3838	66		.2040, .1826, .1657, .1494, .1338, .1177, .1081, .0980,	3838
3839	67		.2034, .1854, .1683, .1497, .1322, .1169, .1051, .0946,	3839
3840	68		.1969, .1855, .1685, .1487, .1304, .1149, .1024, .0916,	3840
3841	69		.1899, .1837, .1677, .1475, .1287, .1126, .1002, .0900,	3841
3842	70		.1841, .1817, .1667, .1464, .1272, .1109, .0983, .0888,	3842
3843	71		.1800, .1800, .1659, .1455, .1262, .1097, .0965, .0878,	3843
3844	72		.1779, .1787, .1657, .1450, .1254, .1087, .0949, .0868,	3844
3845	73		.1773, .1778, .1656, .1447, .1250, .1080, .0939, .0859,	3845
3846	74		.1783, .1778, .1658, .1448, .1248, .1076, .0933, .0851,	3846
3847	75		.1808, .1781, .1667, .1451, .1248, .1074, .0930, .0843,	3847
3848	76		.2134, .2040, .1978, .1782, .1565, .1368, .1206, .1074,	3848
3849	77		.2210, .2072, .1957, .1739, .1516, .1312, .1137, .1000,	3849
3850	78		.2245, .2109, .1989, .1772, .1563, .1390, .1247, .1133/	3850
3851	79		DATA G4/.2299, .2132, .2017, .1795, .1579, .1384, .1221, .1090,	3851
3852	80		.2350, .2157, .2023, .1798, .1575, .1370, .1197, .1057,	3852
3853	81		.2397, .2194, .2034, .1796, .1572, .1372, .1205, .1070,	3853
3854	82		.2452, .2227, .2050, .1805, .1576, .1379, .1236, .1118,	3854
3855	83		.2510, .2256, .2069, .1814, .1581, .1383, .1231, .1103,	3855
3856	84		.2560, .2282, .2091, .1822, .1585, .1385, .1226, .1083,	3856
3857	85		.2605, .2312, .2111, .1829, .1588, .1386, .1222, .1070,	3857
3858	86		.2677, .2358, .2129, .1836, .1592, .1386, .1218, .1071,	3858
3859	87		.2759, .2403, .2145, .1857, .1598, .1389, .1219, .1078,	3859
3860	88		.2834, .2445, .2160, .1878, .1603, .1394, .1223, .1084,	3860
3861	89		.2905, .2484, .2175, .1898, .1613, .1399, .1226, .1090,	3861
3862	90		.2963, .2531, .2199, .1918, .1625, .1407, .1230, .1096,	3862
3863	91		.4323, .3582, .3109, .2889, .2803, .2706, .2410, .2224,	3863
3864	92		.4610, .4026, .3624, .3212, .2926, .2551, .2375, .2015/	3864
3865	93		DATA G5/.4199, .3837, .3401, .2979, .2623, .2318, .2108, .1854,	3865
3866	94		.3924, .3642, .3194, .2760, .2427, .2157, .1902, .1721,	3866
3867	95		.3794, .3479, .3025, .2673, .2311, .2019, .1842, .1613,	3867
3868	96		.3674, .3448, .2961, .2593, .2255, .1994, .1785, .1594,	3868
3869	97		.3573, .3443, .2910, .2517, .2293, .2006, .1843, .1679,	3869
3870	98		.3661, .3438, .2935, .2597, .2336, .2225, .2143, .2116,	3870
3871	99		.3674, .3435, .3080, .2728, .2606, .2577, .2573, .2573,	3871
3872	100		.3685, .3453, .3210, .3014, .2942, .2933, .2932, .2932,	3872
3873	101		.3814, .3612, .3341, .3276, .3257, .3253, .3252, .3252,	3873
3874	102		.3903, .3752, .3570, .3522, .3513, .3510, .3506, .3496,	3874
3875	103		.4012, .3899, .3782, .3751, .3743, .3741, .3734, .3713,	3875
3876	104		.4155, .4057, .3956, .3930, .3920, .3913, .3907, .3890,	3876
3877	105		.4290, .4205, .4118, .4092, .4077, .4065, .4059, .4047,	3877
3878	106		.5411, .5385, .5359, .5353, .5351, .5350, .5350, .5350/	3878
3879	107		DATA G6/.5823, .5812, .5801, .5797, .5796, .5797, .5797, .5797,	3879
3880	108		.6096, .6090, .6085, .6082, .6082, .6083, .6083, .6083,	3880
3881	109		.6308, .6306, .6305, .6303, .6303, .6305, .6305, .6305,	3881
3882	110		.6481, .6483, .6485, .6483, .6484, .6486, .6487, .6487,	3882
3883	111		.6627, .6632, .6637, .6636, .6637, .6640, .6640, .6640,	3883
3884	112		.6754, .6761, .6769, .6768, .6770, .6773, .6773, .6773,	3884
3885	113		.6866, .6875, .6885, .6884, .6886, .6890, .6890, .6890,	3885
3886	114		.6966, .6977, .6989, .6989, .6991, .6995, .6995, .6995,	3886
3887	115		.7056, .7070, .7083, .7083, .7085, .7090, .7090, .7090,	3887
3888	116		.7139, .7154, .7169, .7169, .7172, .7176, .7177, .7177,	3888
3889	117		.7214, .7231, .7248, .7248, .7251, .7256, .7256, .7256,	3889
3890	118		.7285, .7303, .7321, .7321, .7325, .7330, .7330, .7330,	3890
3891	119		.7350, .7370, .7390, .7390, .7393, .7398, .7399, .7399,	3891
3892	120		.7411, .7432, .7453, .7454, .7457, .7463, .7463, .7463/	3892
3893	121		DATA G7/.8069, .8103, .8138, .8139, .8145, .8152, .8153, .8153,	3893
3894	122		.8454, .8496, .8538, .8540, .8547, .8556, .8557, .8557,	3894
3895	123		.8727, .8774, .8822, .8825, .8832, .8842, .8843, .8843,	3895
3896	124		.8938, .8990, .9042, .9046, .9054, .9064, .9065, .9065,	3896
3897	125		.9111, .9166, .9222, .9226, .9235, .9246, .9247, .9247,	3897

```

3898 126 ! .9258,.9316,.9374,.9379,.9387,.9399,.9400,.9400, 3898
3899 127 ! .9384,.9445,.9506,.9511,.9520,.9532,.9533,.9533, 3899
3900 128 ! .9496,.9559,.9622,.9627,.9637,.9649,.9650,.9650, 3900
3901 129 ! .9596,.9661,.9727,.9731,.9741,.9754,.9755,.9755, 3901
3902 130 ! .9686,.9753,.9821,.9826,.9836,.9849,.9850,.9850, 3902
3903 131 ! .9769,.9837,.9906,.9912,.9922,.9936,.9937,.9937, 3903
3904 132 ! .9845,.9915,.9986,.9991,.9999,.9999,.9999,.9999, 3904
3905 133 ! .9915,.9987,.9999,.9999,.9999,.9999,.9999,.9999, 3905
3906 134 ! .9981,.9999,.9999,.9999,.9999,.9999,.9999,.9999/ 3906
3907 135 C ----- 3907
3908 136 C REAL AIR EOS, TABLE LOOKUP ON GILMORE DATA. (NO TEMP. MODEL) 3908
3909 137 C TO AVOID COSTLY LOGARITHMIC FUNCTIONS THE TABLE "G" IS STORED IN A 3909
3910 138 C FORM SO THAT THE HEXADECIMAL WORD STRUCTURE OF A 32 BIT MACHINE 3910
3911 139 C MAY BE EXPLOITED. 3911
3912 140 C THIS LOGIC MAY BE TRANSFERRED TO OTHER MACHINES BY RECALCULATING 3912
3913 141 C THE TABLE "G" APPROPRIATE TO THE WORD ARCHITECTURE OF THAT MACHINE. 3913
3914 142 C MACHINE DEPENDENT FUNCTIONS AND KEY NUMBERS MUST ALSO BE CHANGED. 3914
3915 143 C ----- 3915
3916 144 RL16E = 1./XL16E 3916
3917 145 IST = 0 3917
3918 146 NR = N 3918
3919 147 C 3919
3920 148 10 CONTINUE 3920
3921 149 NST = MINO(NR,M) 3921
3922 150 C 3922
3923 151 DO 20 IRE=1,NST 3923
3924 152 RHO(IRE) = .774413*RRR(IST+IRE) 3924
3925 153 E(IRE) = AMAX1(3.e8,10000.*EEE(IST+IRE)/RRR(IST+IRE)) 3925
3926 154 C 3926
3927 155 C CALCULATE MASS DENSITY VARIATION INDEX "I". 3927
3928 156 C 3928
3929 157 TEM = ALOG(RHO(IRE))*RL16E + 500.0 3929
3930 158 I(IRE) = AINT(TEM) 3930
3931 159 OMP(IRE) = TEM - FLOAT(I(IRE)) 3931
3932 160 I(IRE) = 502 - I(IRE) 3932
3933 161 I(IRE) = MAXO(I(IRE),1) 3933
3934 162 C 3934
3935 163 C CALCULATE INTERNAL ENERGY VARIATION INDEX "J". 3935
3936 164 C 3936
3937 165 TEM = ALOG(E(IRE))*RL16E 3937
3938 166 JCY = AINT(TEM) 3938
3939 167 TEM = TEM - FLOAT(JCY) 3939
3940 168 TEM = EXP(XL16E*TEM) 3940
3941 169 JCY = JCY - 7 3941
3942 170 JS = AINT(TEM) 3942
3943 171 Q(IRE) = TEM - FLOAT(JS) 3943
3944 172 J(IRE) = JS + 15*JCY 3944
3945 173 J(IRE) = MINO(J(IRE),104) 3945
3946 174 J(IRE) = I(IRE) + 8*J(IRE) 3946
3947 175 I(IRE) = J(IRE) - 8 3947
3948 176 20 CONTINUE 3948
3949 177 C 3949
3950 178 DO 30 IRE=1,NST 3950
3951 179 T11(IRE) = GF(I(IRE)) 3951
3952 180 T21(IRE) = GF(I(IRE)+1) 3952
3953 181 T12(IRE) = GF(J(IRE)) 3953
3954 182 T22(IRE) = GF(J(IRE)+1) 3954
3955 183 30 CONTINUE 3955
3956 184 C 3956
3957 185 C CALCULATE GAMMA BY LINEAR INTERPOLATION. 3957
3958 186 C 3958
3959 187 DO 40 IRE=1,NST 3959
3960 188 T12(IRE) = T12(IRE) - T11(IRE) 3960
3961 189 T22(IRE) = T22(IRE) - T21(IRE) 3961
3962 190 GAMMA(IST+IRE) = OMP(IRE)*(T11(IRE) + Q(IRE)*T12(IRE)) 3962
3963 191 ! + (1. - OMP(IRE))*(T21(IRE) + Q(IRE)*T22(IRE)) 3963
3964 192 ! + 1. 3964
3965 193 40 CONTINUE 3965
3966 194 C 3966
3967 195 NR = NR - NST 3967
3968 196 IST = IST + NST 3968
3969 197 IF(NR.GT.0) GO TO 10 3969
3970 198 C 3970
3971 199 C --- EXIT POINT FROM SUBROUTINE ----- 3971

```

3972	200	C		3972
3973	201	C		3973
3974	202		-----	3974
3975	203	C	RETURN	3975
3976	204	C	-----	3976
3977	205	C	---	3977
3978	206		END	3978

3979	1		SUBROUTINE LIFTDR	3979
3980	2	C		3980
3981	3		include 'cmsh00.h'	3981
3982	4		include 'chyd00.h'	3982
3983	5		include 'cint00.h'	3983
3984	6		include 'cphs10.h'	3984
3985	7		include 'cphs20.h'	3985
3986	8		REAL PRESS(1000),DYNPRS(1000),XLOCAT(1000),YLOCAT(1000)	3986
3987	9	C		3987
3988	10		XLIFT = 0.	3988
3989	11		XDRAG = 0.	3989
3990	12		XMOMN = 0.	3990
3991	13		UINVR = 2. / UVIN / UVIN / RIN	3991
3992	14		XYU = COS(ALPHA)	3992
3993	15		XYV = SIN(ALPHA)	3993
3994	16		NBB = 0	3994
3995	17		DO 210 IE = 1 , NE	3995
3996	18		IJE5 = JE(5 , IE)	3996
3997	19		IF(IJE5 . EQ . 5) THEN	3997
3998	20		NBB = NBB + 1	3998
3999	21		IV1 = JE(1 , IE)	3999
4000	22		IV2 = JE(2 , IE)	4000
4001	23		ISL = JE(3 , IE)	4001
4002	24		PRES = HYDV(ISL , 4) - PINL	4002
4003	25		PRESS(NBB) = PRES	4003
4004	26		XLIFT = XLIFT + PRES * XE(1 , IE) *	4004
4005	27		(- XN(IE) * XYV + YN(IE) * XYU)	4005
4006	28		XDRAG = XDRAG + PRES * XE(1 , IE) *	4006
4007	29		(XN(IE) * XYU + YN(IE) * XYV)	4007
4008	30		XLOCAT(NBB) = .5 * (XV(1 , IV1) + XV(1 , IV2))	4008
4009	31		XXV = XLOCAT(NBB)	4009
4010	32		YLOCAT(NBB) = .5 * (XV(2 , IV1) + XV(2 , IV2))	4010
4011	33		YYV = YLOCAT(NBB)	4011
4012	34		XMOMN = XMOMN + PRES * XE(1 , IE) *	4012
4013	35		(XN(IE) * XXV - YN(IE) * YYV)	4013
4014	36	C		4014
4015	37		END IF	4015
4016	38	C		4016
4017	39	210	CONTINUE	4017
4018	40	C		4018
4019	41		XLIFT = XLIFT * UINVR	4019
4020	42		XDRAG = XDRAG * UINVR	4020
4021	43		XMOMN = XMOMN * UINVR	4021
4022	44		WRITE (4) NBB, (XLOCAT(KK), YLOCAT(KK), PRESS(KK), KK=1, NBB)	4022
4023	45		WRITE (9) XLIFT, XDRAG, XMOMN, XMCHIN, ALFA	4023
4024	46		PRINT *, XLIFT, XDRAG, XMOMN, XMCHIN, ALFA	4024
4025	47	C		4025
4026	48	C	--- EXIT POINT FROM SUBROUTINE -----	4026
4027	49	C		4027
4028	50	C	-----	4028
4029	51		RETURN	4029
4030	52	C	-----	4030
4031	53	C		4031
4032	54	C	---	4032
4033	55		END	4033

```

1      1      SUBROUTINE VERDEL( KSD , INDCTR , NIDUMP , JJTRIG , IITRIG )
2      2      C
3      3      C-----I
4      4      C I
5      5      C VERDEL FORCE DELETION OF CELL NUMBER KSD I
6      6      C I
7      7      C-----I
8      8      C
9      9      C IMPLICIT REAL (A-H,O-Z)
10     10     C
11     11     include 'cms00.h'
12     12     include 'chyd00.h'
13     13     include 'cint00.h'
14     14     include 'cphs10.h'
15     15     include 'cphs20.h'
16     16     C
17     17     INTEGER JUV(MEM),JUE(MEM),JUS(MEM)
18     18     INTEGER IUV(MEM),IUE(MEM),IUS(MEM)
19     19     INTEGER IITRIG(200)
20     20     C
21     21     EQUIVALENCE (PR,JUV)
22     22     EQUIVALENCE (UR,JUE)
23     23     EQUIVALENCE (VR,JUS)
24     24     EQUIVALENCE (PL,IUV)
25     25     EQUIVALENCE (UL,IUE)
26     26     EQUIVALENCE (VL,IUS)
27     27     C
28     28     C JUV( IVV ) >> NV
29     29     C IUV( NV ) >> IVV
30     30     C
31     31     C DO *** KI = 1 , JVDDEL
32     32     C IVM = NVDEL( KI )
33     33     C JVM = IVDEL( KI )
34     34     C JUV( IVM ) = JVM
35     35     C IUV( JVM ) = IVM
36     36     C
37     37     C DO *** KI= 1 , IETRIG
38     38     C IEM = NECRSS( KI )
39     39     C JEM = IECRSS( KI )
40     40     C JUE( IEM ) = JEM
41     41     C IUE( JEM ) = IEM
42     42     C
43     43     C DO *** KI = 1 , ITRIG
44     44     C ISM = NSCRSS( KI )
45     45     C JSM = ISCRSS( KI )
46     46     C JUS( ISM ) = JSM
47     47     C IUS( JSM ) = ISM
48     48     C
49     49     FLUXPP = .00001 * HYDMOM( 4 )
50     50     FLUXUU = .00001 * HYDMOM( 2 )
51     51     FLUXRR = .00001 * HYDMOM( 1 )
52     52     AREVGG = AREDEL * SAREVG
53     53     XYLONG = 0.
54     54     XYSHRT = 10000000.
55     55     XYLNGT = 0.
56     56     C
57     57     KV1 = JS( 1 , KSD )
58     58     KV2 = JS( 2 , KSD )
59     59     KV3 = JS( 3 , KSD )
60     60     JKV1 = JV( 2 , KV1 )
61     61     JKV2 = JV( 2 , KV2 )
62     62     JKV3 = JV( 2 , KV3 )
63     63     KE1 = JS( 4 , KSD )
64     64     KE2 = JS( 5 , KSD )
65     65     KE3 = JS( 6 , KSD )
66     66     IKE1 = IABS( KE1 )
67     67     IKE2 = IABS( KE2 )
68     68     IKE3 = IABS( KE3 )
69     69     IJE51 = JE( 5 , IKE1 )
70     70     IJE52 = JE( 5 , IKE2 )
71     71     IJE53 = JE( 5 , IKE3 )
72     72     IKKE = 0
73     73     IF( IJE53 . NE . 0 . AND . JKV2 . LT . 0 ) THEN

```

74	74	IKKE = 4	74
75	75	IEIN1 = - JKV1	75
76	76	IEIN2 = - JKV2	76
77	77	IKKE1 = IKE3	77
78	78	KKE1 = KE3	78
79	79	KKV1 = KV3	79
80	80	IKKE2 = IKE1	80
81	81	KKE2 = KE1	81
82	82	KKV2 = KV1	82
83	83	IKKE3 = IKE2	83
84	84	KKE3 = KE2	84
85	85	KKV3 = KV2	85
86	86	C	86
87	87	ELSE IF(IJE52 . NE . 0 . AND . JKV1 . LT . 0) THEN	87
88	88	IKKE = 4	88
89	89	IEIN1 = - JKV3	89
90	90	IEIN2 = - JKV1	90
91	91	IKKE1 = IKE2	91
92	92	KKE1 = KE2	92
93	93	KKV1 = KV2	93
94	94	IKKE2 = IKE3	94
95	95	KKE2 = KE3	95
96	96	KKV2 = KV3	96
97	97	IKKE3 = IKE1	97
98	98	KKE3 = KE1	98
99	99	KKV3 = KV1	99
100	100	C	100
101	101	ELSE IF(IJE51 . NE . 0 . AND . JKV3 . LT . 0) THEN	101
102	102	IKKE = 4	102
103	103	IEIN1 = - JKV2	103
104	104	IEIN2 = - JKV3	104
105	105	IKKE1 = IKE1	105
106	106	KKE1 = KE1	106
107	107	KKV1 = KV1	107
108	108	IKKE2 = IKE2	108
109	109	KKE2 = KE2	109
110	110	KKV2 = KV2	110
111	111	IKKE3 = IKE3	111
112	112	KKE3 = KE3	112
113	113	KKV3 = KV3	113
114	114	C	114
115	115	ELSE IF(IJE53 . EQ . 0 . AND . JKV3 . LT . 0 . AND .	115
116	116	JKV1 . LT . 0) THEN	116
117	117	IKKE = 3	117
118	118	IEIN1 = - JKV3	118
119	119	IEIN2 = - JKV1	119
120	120	IKKE1 = IKE3	120
121	121	KKE1 = KE3	121
122	122	KKV1 = KV3	122
123	123	IKKE2 = IKE1	123
124	124	KKE2 = KE1	124
125	125	KKV2 = KV1	125
126	126	IKKE3 = IKE2	126
127	127	KKE3 = KE2	127
128	128	KKV3 = KV2	128
129	129	C	129
130	130	ELSE IF(IJE52 . EQ . 0 . AND . JKV2 . LT . 0 . AND .	130
131	131	JKV3 . LT . 0) THEN	131
132	132	IKKE = 3	132
133	133	IEIN1 = - JKV2	133
134	134	IEIN2 = - JKV3	134
135	135	IKKE1 = IKE2	135
136	136	KKE1 = KE2	136
137	137	KKV1 = KV2	137
138	138	IKKE2 = IKE3	138
139	139	KKE2 = KE3	139
140	140	KKV2 = KV3	140
141	141	IKKE3 = IKE1	141
142	142	KKE3 = KE1	142
143	143	KKV3 = KV1	143
144	144	C	144
145	145	ELSE IF(IJE51 . EQ . 0 . AND . JKV1 . LT . 0 . AND .	145
146	146	JKV2 . LT . 0) THEN	146
147	147	IKKE = 3	147

148	148	IEIN1 = - JKV1	148
149	149	IEIN2 = - JKV2	149
150	150	IKKE1 = IKE1	150
151	151	KKV1 = KE1	151
152	152	KKV1 = KV1	152
153	153	IKKE2 = IKE2	153
154	154	KKE2 = KE2	154
155	155	KKV2 = KV2	155
156	156	IKKE3 = IKE3	156
157	157	KKE3 = KE3	157
158	158	KKV3 = KV3	158
159	159	C	159
160	160	ELSE IF(IJES3 . NE . 0) THEN	160
161	161	IKKE = 1	161
162	162	IEIN = - JKV1	162
163	163	IKKE1 = IKE3	163
164	164	KKE1 = KE3	164
165	165	KKV1 = KV3	165
166	166	IKKE2 = IKE1	166
167	167	KKE2 = KE1	167
168	168	KKV2 = KV1	168
169	169	KKE3 = KE2	169
170	170	IKKE3 = IKE2	170
171	171	KKV3 = KV2	171
172	172	C	172
173	173	ELSE IF(IJES2 . NE . 0) THEN	173
174	174	IKKE = 1	174
175	175	IEIN = - JKV3	175
176	176	IKKE1 = IKE2	176
177	177	KKE1 = KE2	177
178	178	KKV1 = KV2	178
179	179	IKKE2 = IKE3	179
180	180	KKE2 = KE3	180
181	181	KKV2 = KV3	181
182	182	IKKE3 = IKE1	182
183	183	KKE3 = KE1	183
184	184	KKV3 = KV1	184
185	185	C	185
186	186	ELSE IF(IJES1 . NE . 0) THEN	186
187	187	IKKE = 1	187
188	188	IEIN = - JKV2	188
189	189	IKKE1 = IKE1	189
190	190	KKE1 = KE1	190
191	191	KKV1 = KV1	191
192	192	IKKE2 = IKE2	192
193	193	KKE2 = KE2	193
194	194	KKV2 = KV2	194
195	195	IKKE3 = IKE3	195
196	196	KKE3 = KE3	196
197	197	KKV3 = KV3	197
198	198	C	198
199	199	ELSE IF(JKV3 . LT . 0) THEN	199
200	200	IKKE = 2	200
201	201	IEIN = - JKV3	201
202	202	IKKE1 = IKE3	202
203	203	KKE1 = KE3	203
204	204	KKV1 = KV3	204
205	205	IKKE2 = IKE1	205
206	206	KKE2 = KE1	206
207	207	KKV2 = KV1	207
208	208	IKKE3 = IKE2	208
209	209	KKE3 = KE2	209
210	210	KKV3 = KV2	210
211	211	C	211
212	212	ELSE IF(JKV2 . LT . 0) THEN	212
213	213	IKKE = 2	213
214	214	IEIN = - JKV2	214
215	215	IKKE1 = IKE2	215
216	216	KKE1 = KE2	216
217	217	KKV1 = KV2	217
218	218	IKKE2 = IKE3	218
219	219	KKE2 = KE3	219
220	220	KKV2 = KV3	220
221	221	IKKE3 = IKE1	221

```

222 222 KKE3 = KE1 222
223 223 KKV3 = KV1 223
224 224 C 224
225 225 ELSE IF( JKV1 . LT . 0 ) THEN 225
226 226 IKKE = 2 226
227 227 IEIN = - JKV1 227
228 228 IKKE1 = IKE1 228
229 229 KKE1 = KE1 229
230 230 KKV1 = KV1 230
231 231 IKKE2 = IKE2 231
232 232 KKE2 = KE2 232
233 233 KKV2 = KV2 233
234 234 IKKE3 = IKE3 234
235 235 KKE3 = KE3 235
236 236 KKV3 = KV3 236
237 237 END IF 237
238 238 C 238
239 239 IF( IKKE . EQ . 4 ) THEN 239
240 240 JV1 = JE( 1 , IEIN2 ) 240
241 241 JV2 = JE( 2 , IEIN2 ) 241
242 242 JJV3 = JE( 1 , IKKE3 ) 242
243 243 JJV4 = JE( 2 , IKKE3 ) 243
244 244 IF( JJV3 . EQ . JV1 ) THEN 244
245 245 JV3 = JJV3 245
246 246 JV4 = JJV4 246
247 247 ELSE 247
248 248 JV3 = JJV4 248
249 249 JV4 = JJV3 249
250 250 END IF 250
251 251 XA = XV( 1 , JV2 ) - XV( 1 , JV1 ) 251
252 252 YA = XV( 2 , JV2 ) - XV( 2 , JV1 ) 252
253 253 XB = XV( 1 , JV4 ) - XV( 1 , JV3 ) 253
254 254 YB = XV( 2 , JV4 ) - XV( 2 , JV3 ) 254
255 255 AB = XA * XB + YA * YB 255
256 256 IF( AB . GT . 0. ) IKKE = 5 256
257 257 END IF 257
258 258 C 258
259 259 C IJTRIG NUMBER OF CIRCUMFERENCE EDGES AROUND VOID 259
260 260 C ITRIG NUMBER OF TRIANGLES TO BE DELETED 260
261 261 C IETRIG NUMBER OF EDGES TO BE DELETED 261
262 262 C JVDDEL NUMBER OF VERTICES TO BE DELETED 262
263 263 C 263
264 264 C IVDDEL(*) SEQUENCE OF VERTICES TO BE DELETED 264
265 265 C ISCRSS(*) SEQUENCE OF TRIANGLES TO BE DELETED 265
266 266 C IECRSS(*) SEQUENCE OF EDGES TO BE DELETED 266
267 267 C 267
268 268 IF( JV( 1 , KV1 ) . EQ . 3 ) RETURN 268
269 269 IF( JV( 1 , KV2 ) . EQ . 3 ) RETURN 269
270 270 IF( JV( 1 , KV3 ) . EQ . 3 ) RETURN 270
271 271 IJTRIG = 0 271
272 272 ITRIG = 0 272
273 273 IETRIG = 0 273
274 274 JVDDEL = 0 274
275 275 JLOOP = 0 275
276 276 C 276
277 277 IF( IKKE . EQ . 0 ) THEN 277
278 278 C 278
279 279 C THE TRIANGLE TO BE DELETED IS INTIRELY IN THE DOMAIN OF COMPUTATION . 279
280 280 C THE FIRST LOOP IS AROUND VERTEX KV1 . 280
281 281 C 281
282 282 IVV = KV1 282
283 283 IE = IKE3 283
284 284 IV1 = JE( 1 , IE ) 284
285 285 IF( IV1 . EQ . IVV ) THEN 285
286 286 ISI = JE( 3 , IE ) 286
287 287 ELSE 287
288 288 ISI = JE( 4 , IE ) 288
289 289 END IF 289
290 290 IS = ISI 290
291 291 C 291
292 292 110 CONTINUE 292
293 293 C 293
294 294 ITRIG = ITRIG + 1 294
295 295 ISCRSS( ITRIG ) = IS 295

```



```

296 296 C
297 297 IF( ITRIG . EQ . 2 ) THEN
298 298 IJTRIG = 0
299 299 IETRIG = IETRIG + 1
300 300 IECRSS( IETRIG ) = IEIB
301 301 END IF
302 302 C
303 303 IETRIG = IETRIG + 1
304 304 IECRSS( IETRIG ) = IE
305 305 C
306 306 IF(
307 307 . HYDFLX( IS , 4 ) . GT . FLUXPP . OR .
308 308 . HYDFLX( IS , 2 ) . GT . FLUXUU . OR .
309 309 . HYDFLX( IS , 1 ) . GT . FLUXRR . OR .
310 310 . KSDEL( IS ) . GT . NIDUMP . OR .
311 311 . XS( 3 , IS ) . GT . AREVGG ) THEN
312 312 INDCTR = 3
313 313 RETURN
314 314 END IF
315 315 C
316 316 DO 120 IR = 1 , 3
317 317 JR = MOD( IR , 3 ) + 1
318 318 IEA = IABS( JS( JR + 3 , IS ) )
319 319 IF( IEA . EQ . IE ) THEN
320 320 IIR = MOD( JR , 3 ) + 4
321 321 IEI = JS( IIR , IS )
322 322 IEIB = IABS( IEI )
323 323 XEIEB = XE( 1 , IEIB )
324 324 XYLNGT = XYLNGT + XEIEB
325 325 IF( XYLONG . LT . XEIEB ) XYLONG = XEIEB
326 326 IF( XYSHRT . GT . XEIEB ) XYSHRT = XEIEB
327 327 IJTRIG = IJTRIG + 1
328 328 IICOLR( IJTRIG ) = IEI
329 329 JJR = MOD( JR + 1 , 3 ) + 4
330 330 IER = IABS( JS( JJR , IS ) )
331 331 C
332 332 IV1 = JE( 1 , IER )
333 333 IF( IV1 . EQ . IVV ) THEN
334 334 ISR = JE( 3 , IER )
335 335 ELSE
336 336 ISR = JE( 4 , IER )
337 337 END IF
338 338 END IF
339 339 C
340 340 120 CONTINUE
341 341 C
342 342 IF( ISR . NE . ISI ) THEN
343 343 IS = ISR
344 344 IE = IER
345 345 GO TO 110
346 346 END IF
347 347 C
348 348 IETRIG = IETRIG + 1
349 349 IECRSS( IETRIG ) = IKE2
350 350 IJTRIG = IJTRIG - 2
351 351 C
352 352 C FIRST LOOP SURROUNDING KV1 IS DONE, SECOND LOOP OVER KV2 START .
353 353 C
354 354 IVV = KV2
355 355 IE = IABS( IICOLR( IJTRIG + 1 ) )
356 356 IV1 = JE( 1 , IE )
357 357 IF( IV1 . EQ . IVV ) THEN
358 358 ISI = JE( 3 , IE )
359 359 ELSE
360 360 ISI = JE( 4 , IE )
361 361 END IF
362 362 IS = ISI
363 363 C
364 364 ILOOP = 0
365 365 130 CONTINUE
366 366 JDOUBL = IABS( IICOLR( IJTRIG ) )
367 367 C
368 368 ITRIG = ITRIG + 1
369 369 ISCRSS( ITRIG ) = IS

```

```

370 370 IETRIG = IETRIG + 1 370
371 371 IECRSS( IETRIG ) = IE 371
372 372 C 372
373 373 IF( 373
374 374 . HYDFLX( IS , 4 ) . GT . FLUXPP . OR . 374
375 375 . HYDFLX( IS , 2 ) . GT . FLUXUU . OR . 375
376 376 . HYDFLX( IS , 1 ) . GT . FLUXRR . OR . 376
377 377 . KSDDEL( IS ) . GT . NIDUMP . OR . 377
378 378 . XS( 3 , IS ) . GT . AREVGG ) THEN 378
379 379 INDCTR = 3 379
380 380 RETURN 380
381 381 END IF 381
382 382 C 382
383 383 DO 140 IR = 1 , 3 383
384 384 JR = MOD( IR , 3 ) + 1 384
385 385 IEA = IABS( JS( JR + 3 , IS ) ) 385
386 386 IF( IEA . EQ . IE ) THEN 386
387 387 IIR = MOD( JR , 3 ) + 4 387
388 388 IEI = JS( IIR , IS ) 388
389 389 IEIB = IABS( IEI ) 389
390 390 XEIEB = XE( 1 , IEIB ) 390
391 391 XYLNGT = XYLNGT + XEIEB 391
392 392 IF( XYLONG . LT . XEIEB ) XYLONG = XEIEB 392
393 393 IF( XYSHRT . GT . XEIEB ) XYSHRT = XEIEB 393
394 394 ILOOP = ILOOP + 1 394
395 395 IF( ILOOP . EQ . 1 . AND . JDOUBL . EQ . IEIB ) THEN 395
396 396 JLOOP = 1 396
397 397 IETRIG = IETRIG + 1 397
398 398 IECRSS( IETRIG ) = JDOUBL 398
399 399 IJTRIG = IJTRIG - 1 399
400 400 IF( IEI . GT . 0 ) THEN 400
401 401 JKVV = JE( 1 , IEIB ) 401
402 402 ELSE 402
403 403 JKVV = JE( 2 , IEIB ) 403
404 404 END IF 404
405 405 JVDDELT = JVDDELT + 1 405
406 406 IVDEL( JVDDELT ) = JKVV 406
407 407 ILOOP = 0 407
408 408 ELSE 408
409 409 IJTRIG = IJTRIG + 1 409
410 410 IICOLR( IJTRIG ) = IEI 410
411 411 END IF 411
412 412 JJR = MOD( JR + 1 , 3 ) + 4 412
413 413 IER = IABS( JS( JJR , IS ) ) 413
414 414 C 414
415 415 IV1 = JE( 1 , IER ) 415
416 416 IF( IV1 . EQ . IVV ) THEN 416
417 417 ISR = JE( 3 , IER ) 417
418 418 ELSE 418
419 419 ISR = JE( 4 , IER ) 419
420 420 END IF 420
421 421 END IF 421
422 422 C 422
423 423 140 CONTINUE 423
424 424 C 424
425 425 IF( IER . NE . IKE2 ) THEN 425
426 426 IS = ISR 426
427 427 IE = IER 427
428 428 GO TO 130 428
429 429 END IF 429
430 430 IJTRIG = IJTRIG - 1 430
431 431 C 431
432 432 C SECOND LOOP SURROUNDING KV2 IS DONE, THIRD LOOP OVER KV3 START . 432
433 433 C 433
434 434 KET = IECRSS( 2 ) 434
435 435 IVV = KV3 435
436 436 IE = IABS( IICOLR( IJTRIG + 1 ) ) 436
437 437 IF( IE . EQ . KET ) THEN 437
438 438 JLOOP = 2 438
439 439 C 439
440 440 150 CONTINUE 440
441 441 IKET = IICOLR( 1 ) 441
442 442 KKET = IABS( IKET ) 442
443 443 JKET = IABS( IICOLR( IJTRIG ) ) 443

```

```

444 444 IF( JKET . EQ . KKET ) THEN 444
445 445 JLOOP = 3 445
446 446 IF( IKET . GT . 0 ) THEN 446
447 447 JKVV = JE( 1 , KKET ) 447
448 448 ELSE 448
449 449 JKVV = JE( 2 , KKET ) 449
450 450 END IF 450
451 451 JVDELTA = JVDelta + 1 451
452 452 IVDelta( JVDelta ) = JKVV 452
453 453 DO 160 KK = 2 , IJTRIG 453
454 454 IICOLR( KK - 1 ) = IICOLR( KK ) 454
455 455 160 CONTINUE 455
456 456 IJTRIG = IJTRIG - 2 456
457 457 IETRIG = IETRIG + 1 457
458 458 IECRSS( IETRIG ) = KKET 458
459 459 GO TO 150 459
460 460 END IF 460
461 461 GO TO 170 461
462 462 END IF 462
463 463 IV1 = JE( 1 , IE ) 463
464 464 IF( IV1 . EQ . IVV ) THEN 464
465 465 ISI = JE( 3 , IE ) 465
466 466 ELSE 466
467 467 ISI = JE( 4 , IE ) 467
468 468 END IF 468
469 469 IS = ISI 469
470 470 C 470
471 471 ILOOP = 0 471
472 472 180 CONTINUE 472
473 473 KDOUBL = IABS( IICOLR( IJTRIG ) ) 473
474 474 C 474
475 475 ITRIG = ITRIG + 1 475
476 476 ISCRSS( ITRIG ) = IS 476
477 477 IETRIG = IETRIG + 1 477
478 478 IECRSS( IETRIG ) = IE 478
479 479 C 479
480 480 IF( 480
481 481 . HYDFLX( IS , 4 ) . GT . FLUXPP . OR . 481
482 482 . HYDFLX( IS , 2 ) . GT . FLUXUU . OR . 482
483 483 . HYDFLX( IS , 1 ) . GT . FLUXRR . OR . 483
484 484 . KSDelta( IS ) . GT . NIDUMP . OR . 484
485 485 . XS( 3 , IS ) . GT . AREVGG ) THEN 485
486 486 INDOCTR = 3 486
487 487 RETURN 487
488 488 END IF 488
489 489 C 489
490 490 DO 190 IR = 1 , 3 490
491 491 JR = MOD( IR , 3 ) + 1 491
492 492 IEA = IABS( JS( JR + 3 , IS ) ) 492
493 493 IF( IEA . EQ . IE ) THEN 493
494 494 IIR = MOD( JR , 3 ) + 4 494
495 495 IEI = JS( IIR , IS ) 495
496 496 IEIB = IABS( IEI ) 496
497 497 XEIEB = XE( 1 , IEIB ) 497
498 498 XYLNgt = XYLNgt + XEIEB 498
499 499 IF( XYLONG . LT . XEIEB ) XYLONG = XEIEB 499
500 500 IF( XYSHRT . GT . XEIEB ) XYSHRT = XEIEB 500
501 501 ILOOP = ILOOP + 1 501
502 502 IF( ILOOP . EQ . 1 . AND . KDOUBL . EQ . IEIB ) THEN 502
503 503 JLOOP = 4 503
504 504 IETRIG = IETRIG + 1 504
505 505 IECRSS( IETRIG ) = KDOUBL 505
506 506 IJTRIG = IJTRIG - 1 506
507 507 IF( IEI . GT . 0 ) THEN 507
508 508 JKVV = JE( 1 , IEIB ) 508
509 509 ELSE 509
510 510 JKVV = JE( 2 , IEIB ) 510
511 511 END IF 511
512 512 JVDelta = JVDelta + 1 512
513 513 IVDelta( JVDelta ) = JKVV 513
514 514 ILOOP = 0 514
515 515 ELSE 515
516 516 IJTRIG = IJTRIG + 1 516
517 517 IICOLR( IJTRIG ) = IEI 517

```

```

518 518      END IF
519 519      JJR = MOD( JR + 1 , 3 ) + 4
520 520      IER = IABS( JS( JJR , IS ) )
521 521      C
522 522      IVI = JE( 1 , IER )
523 523      IF( IVI . EQ . IVV ) THEN
524 524      ISR = JE( 3 , IER )
525 525      ELSE
526 526      ISR = JE( 4 , IER )
527 527      END IF
528 528      END IF
529 529      C
530 530      190 CONTINUE
531 531      C
532 532      IF( IER . NE . KET ) THEN
533 533      IS = ISR
534 534      IE = IER
535 535      GO TO 180
536 536      END IF
537 537      C
538 538      200 CONTINUE
539 539      IKET = IICOLR( 1 )
540 540      KKET = IABS( IKET )
541 541      JKET = IABS( IICOLR( IJTRIG ) )
542 542      IF( JKET . EQ . KKET ) THEN
543 543      JLOOP = 5
544 544      IF( IKET . GT . 0 ) THEN
545 545      JKVV = JE( 1 , KKET )
546 546      ELSE
547 547      JKVV = JE( 2 , KKET )
548 548      END IF
549 549      JVDELT = JVDELT + 1
550 550      IVDELT( JVDELT ) = JKVV
551 551      DO 210 KK = 2 , IJTRIG
552 552      IICOLR( KK - 1 ) = IICOLR( KK )
553 553      210 CONTINUE
554 554      IJTRIG = IJTRIG - 2
555 555      IETRIG = IETRIG + 1
556 556      IECRSS( IETRIG ) = KKET
557 557      GO TO 200
558 558      END IF
559 559      C
560 560      170 CONTINUE
561 561      C
562 562      INOCTR = 2
563 563      C      IF( XYLONG / XYSHRT . GT . 10. . AND . JLOOP . EQ . 0 ) RETURN
564 564      C
565 565      ELSE IF( IKKE . EQ . 1 ) THEN
566 566      C
567 567      C      BEGINING THE DELETION PROCESS IF KSD HAS AN EDGE ON THE BOUNDARY
568 568      C      THE FIRST LOOP IS AROUND VERTEX KKV2 .
569 569      C
570 570      IVV = KKV2
571 571      IE = IEIN
572 572      IVIN = JE( 2 , IE )
573 573      XXYIIB = XE( 1 , IE ) + XE( 1 , IKKE1 )
574 574      IVI = JE( 1 , IE )
575 575      IF( IVI . EQ . IVV ) THEN
576 576      ISI = JE( 3 , IE )
577 577      ELSE
578 578      ISI = JE( 4 , IE )
579 579      END IF
580 580      IS = ISI
581 581      C
582 582      220 CONTINUE
583 583      C
584 584      ITRIG = ITRIG + 1
585 585      ISCRSS( ITRIG ) = IS
586 586      C
587 587      IETRIG = IETRIG + 1
588 588      IECRSS( IETRIG ) = IE
589 589      C
590 590      IF(
591 591      .      HYDFLX( IS , 4 ) . GT . FLUXPP . OR .

```

```

592 592 . HYDFLX( IS , 2 ) . GT . FLUXUU . OR . 592
593 593 . HYDFLX( IS , 1 ) . GT . FLUXRR . OR . 593
594 594 . KSDEL( IS ) . GT . NIDUMP . OR . 594
595 595 . XS( 3 , IS ) . GT . AREVGG ) THEN 595
596 596 INDCTR = 3 596
597 597 RETURN 597
598 598 END IF 598
599 599 C 599
600 600 DO 230 IR = 1 , 3 600
601 601 JR = MOD( IR , 3 ) + 1 601
602 602 IEA = IABS( JS( JR + 3 , IS ) ) 602
603 603 IF( IEA . EQ . IE ) THEN 603
604 604 IIR = MOD( JR , 3 ) + 4 604
605 605 IEI = JS( IIR , IS ) 605
606 606 IEIB = IABS( IEI ) 606
607 607 XEIEB = XE( 1 , IEIB ) 607
608 608 XYLNGT = XYLNGT + XEIEB 608
609 609 IF( XYLONG . LT . XEIEB ) XYLONG = XEIEB 609
610 610 IF( XYSHRT . GT . XEIEB ) XYSHRT = XEIEB 610
611 611 IJTRIG = IJTRIG + 1 611
612 612 IICOLR( IJTRIG ) = IEI 612
613 613 JJR = MOD( JR + 1 , 3 ) + 4 613
614 614 IER = IABS( JS( JJR , IS ) ) 614
615 615 C 615
616 616 IV1 = JE( 1 , IER ) 616
617 617 IF( IV1 . EQ . IVV ) THEN 617
618 618 ISR = JE( 3 , IER ) 618
619 619 ELSE 619
620 620 ISR = JE( 4 , IER ) 620
621 621 END IF 621
622 622 END IF 622
623 623 C 623
624 624 230 CONTINUE 624
625 625 C 625
626 626 IF( IER . NE . IKKE1 ) THEN 626
627 627 IS = ISR 627
628 628 IE = IER 628
629 629 GO TO 220 629
630 630 END IF 630
631 631 C 631
632 632 IETRIG = IETRIG + 1 632
633 633 IECRSS( IETRIG ) = IKKE1 633
634 634 IJTRIG = IJTRIG - 2 634
635 635 C 635
636 636 C FIRST LOOP SURROUNDING KKV2 IS DONE, SECOND LOOP OVER KKV3 START . 636
637 637 C 637
638 638 IVV = KKV3 638
639 639 IE = IABS( IICOLR( IJTRIG + 1 ) ) 639
640 640 IV1 = JE( 1 , IE ) 640
641 641 IF( IV1 . EQ . IVV ) THEN 641
642 642 ISI = JE( 3 , IE ) 642
643 643 ELSE 643
644 644 ISI = JE( 4 , IE ) 644
645 645 END IF 645
646 646 IS = ISI 646
647 647 C 647
648 648 ILOOP = 0 648
649 649 240 CONTINUE 649
650 650 JDOUBL = IABS( IICOLR( IJTRIG ) ) 650
651 651 C 651
652 652 ITRIG = ITRIG + 1 652
653 653 ISCRSS( ITRIG ) = IS 653
654 654 IETRIG = IETRIG + 1 654
655 655 IECRSS( IETRIG ) = IE 655
656 656 C 656
657 657 IF( 657
658 658 . HYDFLX( IS , 4 ) . GT . FLUXPP . OR . 658
659 659 . HYDFLX( IS , 2 ) . GT . FLUXUU . OR . 659
660 660 . HYDFLX( IS , 1 ) . GT . FLUXRR . OR . 660
661 661 . KSDEL( IS ) . GT . NIDUMP . OR . 661
662 662 . XS( 3 , IS ) . GT . AREVGG ) THEN 662
663 663 INDCTR = 3 663
664 664 RETURN 664
665 665 END IF 665

```

```

666 666 C
667 667 DO 250 IR = 1 , 3
668 668 JR = MOD( IR , 3 ) + 1
669 669 IEA = IABS( JS( JR + 3 , IS ) )
670 670 IF( IEA . EQ . IE ) THEN
671 671 IIR = MOD( JR , 3 ) + 4
672 672 IEI = JS( IIR , IS )
673 673 IEIB = IABS( IEI )
674 674 XEIEB = XE( 1 , IEIB )
675 675 XYLNGT = XYLNGT + XEIEB
676 676 IF( XYLONG . LT . XEIEB ) XYLONG = XEIEB
677 677 IF( XYSHRT . GT . XEIEB ) XYSHRT = XEIEB
678 678 ILOOP = ILOOP + 1
679 679 IF( ILOOP . EQ . 1 . AND . JDOUBL . EQ . IEIB ) THEN
680 680 JLOOP = 1
681 681 IETRIG = IETRIG + 1
682 682 IECRSS( IETRIG ) = JDOUBL
683 683 IJTRIG = IJTRIG - 1
684 684 IF( IEI . GT . 0 ) THEN
685 685 JKVV = JE( 1 , IEIB )
686 686 ELSE
687 687 JKVV = JE( 2 , IEIB )
688 688 END IF
689 689 JVDDELT = JVDDELT + 1
690 690 IVDELT( JVDDELT ) = JKVV
691 691 ILOOP = 0
692 692 ELSE
693 693 IJTRIG = IJTRIG + 1
694 694 IICOLR( IJTRIG ) = IEI
695 695 END IF
696 696 JJR = MOD( JR + 1 , 3 ) + 4
697 697 IER = IABS( JS( JJR , IS ) )
698 698 C
699 699 IV1 = JE( 1 , IER )
700 700 IF( IV1 . EQ . IVV ) THEN
701 701 ISR = JE( 3 , IER )
702 702 ELSE
703 703 ISR = JE( 4 , IER )
704 704 END IF
705 705 END IF
706 706 C
707 707 250 CONTINUE
708 708 C
709 709 IF( IER . NE . IKKE3 ) THEN
710 710 IS = ISR
711 711 IE = IER
712 712 GO TO 240
713 713 END IF
714 714 C
715 715 IETRIG = IETRIG + 1
716 716 IECRSS( IETRIG ) = IKKE3
717 717 IJTRIG = IJTRIG - 1
718 718 C
719 719 C SECOND LOOP SURROUNDING KKV3 IS DONE, THIRD LOOP OVER KKV1 START .
720 720 C
721 721 IVV = KKV1
722 722 IE = IABS( IICOLR( IJTRIG + 1 ) )
723 723 IF( JE( 5 , IE ) . NE . 0 ) THEN
724 724 IER = IE
725 725 GO TO 260
726 726 END IF
727 727 IV1 = JE( 1 , IE )
728 728 IF( IV1 . EQ . IVV ) THEN
729 729 ISI = JE( 3 , IE )
730 730 ELSE
731 731 ISI = JE( 4 , IE )
732 732 END IF
733 733 IS = ISI
734 734 ISI = 0
735 735 C
736 736 ILOOP = 0
737 737 270 CONTINUE
738 738 KDOUBL = IABS( IICOLR( IJTRIG ) )
739 739 C

```

```

740 740 ITRIG = ITRIG + 1
741 741 ISCRSS( ITRIG ) = IS
742 742 IETRIG = IETRIG + 1
743 743 IECRSS( IETRIG ) = IE
744 744 C
745 745 IF(
746 746 . HYDFLX( IS , 4 ) . GT . FLUXPP . OR .
747 747 . HYDFLX( IS , 2 ) . GT . FLUXUU . OR .
748 748 . HYDFLX( IS , 1 ) . GT . FLUXRR . OR .
749 749 . KSDDEL( IS ) . GT . NIDUMP . OR .
750 750 . XS( 3 , IS ) . GT . AREVGG ) THEN
751 751 INDCR = 3
752 752 RETURN
753 753 END IF
754 754 C
755 755 DO 280 IR = 1 , 3
756 756 JR = MOD( IR , 3 ) + 1
757 757 IEA = IABS( JS( JR + 3 , IS ) )
758 758 IF( IEA . EQ . IE ) THEN
759 759 IIR = MOD( JR , 3 ) + 4
760 760 IEI = JS( IIR , IS )
761 761 IEIB = IABS( IEI )
762 762 XEIEB = XE( 1 , IEIB )
763 763 XYLNGT = XYLNGT + XEIEB
764 764 IF( XYLONG . LT . XEIEB ) XYLONG = XEIEB
765 765 IF( XYSHRT . GT . XEIEB ) XYSHRT = XEIEB
766 766 ILOOP = ILOOP + 1
767 767 IF( ILOOP . EQ . 1 . AND . KDOUBL . EQ . IEIB ) THEN
768 768 JLOOP = 2
769 769 IETRIG = IETRIG + 1
770 770 IECRSS( IETRIG ) = KDOUBL
771 771 IJTRIG = IJTRIG - 1
772 772 IF( IEI . GT . 0 ) THEN
773 773 JKVV = JE( 1 , IEIB )
774 774 ELSE
775 775 JKVV = JE( 2 , IEIB )
776 776 END IF
777 777 JVDDEL = JVDDEL + 1
778 778 IVDEL( JVDDEL ) = JKVV
779 779 ILOOP = 0
780 780 ELSE
781 781 IJTRIG = IJTRIG + 1
782 782 IICOLR( IJTRIG ) = IEI
783 783 END IF
784 784 JJR = MOD( JR + 1 , 3 ) + 4
785 785 IER = IABS( JS( JJR , IS ) )
786 786 C
787 787 IV1 = JE( 1 , IER )
788 788 IF( IV1 . EQ . IVV ) THEN
789 789 ISR = JE( 3 , IER )
790 790 ELSE
791 791 ISR = JE( 4 , IER )
792 792 END IF
793 793 END IF
794 794 C
795 795 280 CONTINUE
796 796 C
797 797 IF( ISR . NE . ISI ) THEN
798 798 IS = ISR
799 799 IE = IER
800 800 GO TO 270
801 801 END IF
802 802 C
803 803 260 CONTINUE
804 804 C
805 805 IETRIG = IETRIG + 1
806 806 IECRSS( IETRIG ) = IER
807 807 C
808 808 ITYPE = JE( 5 , IER )
809 809 C
810 810 XEIEB = XE( 1 , IER )
811 811 XEIEB = XYYIB + XEIEB
812 812 XYLNGT = XYLNGT + XEIEB
813 813 IF( XYLONG . LT . XEIEB ) XYLONG = XEIEB

```

```

814 814          IF( XYSHRT . GT . XEIEB ) XYSHRT = XEIEB          814
815 815          C                                                815
816 816          INDCTR = 2                                          816
817 817          C          IF( XYLONG / XYSHRT . GT . 10. . AND . JLOOP . EQ . 0 ) RETURN 817
818 818          C                                                818
819 819          IV1 = IVIN                                           819
820 820          IE1 = IICOLR( IJTRIG )                               820
821 821          IF( IE1 . GT . 0 ) THEN                              821
822 822          IV2 = JE( 2 , IE1 )                                  822
823 823          ELSE                                                 823
824 824          IV2 = JE( 1 , - IE1 )                                824
825 825          END IF                                             825
826 826          C                                                826
827 827          NEC = IECRSS( IETRIG )                               827
828 828          IETRIG = IETRIG - 1                                  828
829 829          C                                                829
830 830          JV( 2 , IV2 ) = - NEC                                830
831 831          JE( 1 , NEC ) = IV2                                  831
832 832          JE( 2 , NEC ) = IV1                                  832
833 833          JE( 4 , NEC ) = 0                                    833
834 834          JE( 5 , NEC ) = ITYPE                                834
835 835          C                                                835
836 836          IJTRIG = IJTRIG + 1                                  836
837 837          IICOLR( IJTRIG ) = NEC                              837
838 838          C                                                838
839 839          ELSE IF( IKKE . EQ . 2 ) THEN                       839
840 840          C                                                840
841 841          C          BEGINING THE DELETION PROCESS IF KSD HAS A VERTEX ON THE BOUNDARY 841
842 842          C          THE FIRST LOOP IS AROUND VERTEX KKV1 . 842
843 843          C                                                843
844 844          IVV = KKV1                                           844
845 845          IE = IEIN                                            845
846 846          IVIN = JE( 2 , IE )                                  846
847 847          XXYIIB = XE( 1 , IE )                                847
848 848          IV1 = JE( 1 , IE )                                    848
849 849          IF( IV1 . EQ . IVV ) THEN                            849
850 850          ISI = JE( 3 , IE )                                  850
851 851          ELSE                                                 851
852 852          ISI = JE( 4 , IE )                                  852
853 853          END IF                                             853
854 854          IS = ISI                                             854
855 855          C                                                855
856 856          C          290 CONTINUE                               856
857 857          C                                                857
858 858          ITRIG = ITRIG + 1                                     858
859 859          ISCRSS( ITRIG ) = IS                                 859
860 860          C                                                860
861 861          IETRIG = IETRIG + 1                                  861
862 862          IECRSS( IETRIG ) = IE                               862
863 863          C                                                863
864 864          IF(                                                864
865 865          .          HYDFLX( IS , 4 ) . GT . FLUXPP . OR .    865
866 866          .          HYDFLX( IS , 2 ) . GT . FLUXUU . OR .    866
867 867          .          HYDFLX( IS , 1 ) . GT . FLUXRR . OR .    867
868 868          .          KSDEL( IS ) . GT . NIDUMP . OR .         868
869 869          .          XS( 3 , IS ) . GT . AREVGG ) THEN      869
870 870          INDCTR = 3                                          870
871 871          RETURN                                              871
872 872          END IF                                             872
873 873          C                                                873
874 874          DO 300 IR = 1 , 3                                     874
875 875          JR = MOD( IR , 3 ) + 1                               875
876 876          IEA = IABS( JS( JR + 3 , IS ) )                     876
877 877          IF( IEA . EQ . IE ) THEN                            877
878 878          IIR = MOD( JR , 3 ) + 4                             878
879 879          IEI = JS( IIR , IS )                                879
880 880          IEIB = IABS( IEI )                                   880
881 881          XEIEB = XE( 1 , IEIB )                               881
882 882          XYLNGT = XYLNGT + XEIEB                             882
883 883          IF( XYLONG . LT . XEIEB ) XYLONG = XEIEB          883
884 884          IF( XYSHRT . GT . XEIEB ) XYSHRT = XEIEB          884
885 885          IJTRIG = IJTRIG + 1                                  885
886 886          IICOLR( IJTRIG ) = IEI                             886
887 887          JJR = MOD( JR + 1 , 3 ) + 4                         887

```



```

888 888 IER = IABS( JS( JJR , IS ) ) 888
889 889 C 889
890 890 IV1 = JE( 1 , IER ) 890
891 891 IF( IV1 . EQ . IVV ) THEN 891
892 892 ISR = JE( 3 , IER ) 892
893 893 ELSE 893
894 894 ISR = JE( 4 , IER ) 894
895 895 END IF 895
896 896 END IF 896
897 897 C 897
898 898 300 CONTINUE 898
899 899 C 899
900 900 IF( IER . NE . IKKE3 ) THEN 900
901 901 IS = ISR 901
902 902 IE = IER 902
903 903 GO TO 290 903
904 904 END IF 904
905 905 IJTRIG = IJTRIG - 2 905
906 906 C 906
907 907 C FIRST LOOP SURROUNDING KKV1 IS DONE, SECOND LOOP OVER KKV2 START . 907
908 908 C 908
909 909 IVV = KKV2 909
910 910 IE = IABS( IICOLR( IJTRIG + 1 ) ) 910
911 911 IV1 = JE( 1 , IE ) 911
912 912 IF( IV1 . EQ . IVV ) THEN 912
913 913 ISI = JE( 3 , IE ) 913
914 914 ELSE 914
915 915 ISI = JE( 4 , IE ) 915
916 916 END IF 916
917 917 IS = ISI 917
918 918 C 918
919 919 ILOOP = 0 919
920 920 310 CONTINUE 920
921 921 IDOUBL = IABS( IICOLR( IJTRIG ) ) 921
922 922 C 922
923 923 ITRIG = ITRIG + 1 923
924 924 ISCRSS( ITRIG ) = IS 924
925 925 C 925
926 926 IETRIG = IETRIG + 1 926
927 927 IECRSS( IETRIG ) = IE 927
928 928 C 928
929 929 IF( 929
930 930 . HYDFLX( IS , 4 ) . GT . FLUXPP . OR . 930
931 931 . HYDFLX( IS , 2 ) . GT . FLUXUU . OR . 931
932 932 . HYDFLX( IS , 1 ) . GT . FLUXRR . OR . 932
933 933 . KSDEL( IS ) . GT . MIDUMP . OR . 933
934 934 . XS( 3 , IS ) . GT . AREVGG ) THEN 934
935 935 INOCTR = 3 935
936 936 RETURN 936
937 937 END IF 937
938 938 C 938
939 939 DO 320 IR = 1 , 3 939
940 940 JR = MOD( IR , 3 ) + 1 940
941 941 IEA = IABS( JS( JR + 3 , IS ) ) 941
942 942 IF( IEA . EQ . IE ) THEN 942
943 943 IIR = MOD( JR , 3 ) + 4 943
944 944 IEI = JS( IIR , IS ) 944
945 945 IEIB = IABS( IEI ) 945
946 946 XEIEB = XE( 1 , IEIB ) 946
947 947 XYLNGT = XYLNGT + XEIEB 947
948 948 IF( XYLONG . LT . XEIEB ) XYLONG = XEIEB 948
949 949 IF( XYSHRT . GT . XEIEB ) XYSHRT = XEIEB 949
950 950 ILOOP = ILOOP + 1 950
951 951 IF( ILOOP . EQ . 1 . AND . IDOUBL . EQ . IEIB ) THEN 951
952 952 JLOOP = 1 952
953 953 IETRIG = IETRIG + 1 953
954 954 IECRSS( IETRIG ) = IDOUBL 954
955 955 IJTRIG = IJTRIG - 1 955
956 956 IF( IEI . GT . 0 ) THEN 956
957 957 JKVV = JE( 1 , IEIB ) 957
958 958 ELSE 958
959 959 JKVV = JE( 2 , IEIB ) 959
960 960 END IF 960
961 961 JVDDELT = JVDDELT + 1 961

```

```

962 962 IVDELT( JVDDEL ) = JKVV 962
963 963 ILOOP = 0 963
964 964 ELSE 964
965 965 IJTRIG = IJTRIG + 1 965
966 966 IICOLR( IJTRIG ) = IEI 966
967 967 END IF 967
968 968 JJR = MOD( JR + 1 , 3 ) + 4 968
969 969 IER = IABS( JS( JJR , IS ) ) 969
970 970 C 970
971 971 IV1 = JE( 1 , IER ) 971
972 972 IF( IV1 . EQ . IVV ) THEN 972
973 973 ISR = JE( 3 , IER ) 973
974 974 ELSE 974
975 975 ISR = JE( 4 , IER ) 975
976 976 END IF 976
977 977 END IF 977
978 978 C 978
979 979 320 CONTINUE 979
980 980 C 980
981 981 IF( IER . NE . IKKE2 ) THEN 981
982 982 IS = ISR 982
983 983 IE = IER 983
984 984 GO TO 310 984
985 985 END IF 985
986 986 C 986
987 987 IJTRIG = IJTRIG - 1 987
988 988 C 988
989 989 C SECOND LOOP SURROUNDING KKV2 IS DONE, THIRD LOOP OVER KKV3 START . 989
990 990 C 990
991 991 IVV = KKV3 991
992 992 IE = IABS( IICOLR( IJTRIG + 1 ) ) 992
993 993 IV1 = JE( 1 , IE ) 993
994 994 IF( IV1 . EQ . IVV ) THEN 994
995 995 ISI = JE( 3 , IE ) 995
996 996 ELSE 996
997 997 ISI = JE( 4 , IE ) 997
998 998 END IF 998
999 999 IS = ISI 999
1000 1000 C 1000
1001 1001 ILOOP = 0 1001
1002 1002 330 CONTINUE 1002
1003 1003 KDOUBL = IABS( IICOLR( IJTRIG ) ) 1003
1004 1004 C 1004
1005 1005 ITRIG = ITRIG + 1 1005
1006 1006 ISCRSS( ITRIG ) = IS 1006
1007 1007 IETRIG = IETRIG + 1 1007
1008 1008 IECRSS( IETRIG ) = IE 1008
1009 1009 C 1009
1010 1010 IF( 1010
1011 1011 . HYDFLX( IS , 4 ) . GT . FLUXPP . OR . 1011
1012 1012 . HYDFLX( IS , 2 ) . GT . FLUXUU . OR . 1012
1013 1013 . HYDFLX( IS , 1 ) . GT . FLUXRR . OR . 1013
1014 1014 . KSDEL( IS ) . GT . NIDUMP . OR . 1014
1015 1015 . XS( 3 , IS ) . GT . AREVGG ) THEN 1015
1016 1016 INDCTR = 3 1016
1017 1017 RETURN 1017
1018 1018 END IF 1018
1019 1019 C 1019
1020 1020 DO 340 IR = 1 , 3 1020
1021 1021 JR = MOD( IR , 3 ) + 1 1021
1022 1022 IEA = IABS( JS( JR + 3 , IS ) ) 1022
1023 1023 IF( IEA . EQ . IE ) THEN 1023
1024 1024 IIR = MOD( JR , 3 ) + 4 1024
1025 1025 IEI = JS( IIR , IS ) 1025
1026 1026 IEIB = IABS( IEI ) 1026
1027 1027 XEIEB = XE( 1 , IEIB ) 1027
1028 1028 XYLNGT = XYLNGT + XEIEB 1028
1029 1029 IF( XYLONG . LT . XEIEB ) XYLONG = XEIEB 1029
1030 1030 IF( XYSHRT . GT . XEIEB ) XYSHRT = XEIEB 1030
1031 1031 ILOOP = ILOOP + 1 1031
1032 1032 IF( ILOOP . EQ . 1 . AND . KDOUBL . EQ . IEIB ) THEN 1032
1033 1033 JLOOP = 2 1033
1034 1034 IETRIG = IETRIG + 1 1034
1035 1035 IECRSS( IETRIG ) = KDOUBL 1035

```

1036	1036		IJTRIG = IJTRIG - 1	1036
1037	1037		IF(IEI . GT . 0) THEN	1037
1038	1038		JKVV = JE(1 , IEIB)	1038
1039	1039		ELSE	1039
1040	1040		JKVV = JE(2 , IEIB)	1040
1041	1041		END IF	1041
1042	1042		JVDELT = JVDELT + 1	1042
1043	1043		IVDELT(JVDELT) = JKVV	1043
1044	1044		ILOOP = 0	1044
1045	1045		ELSE	1045
1046	1046		IJTRIG = IJTRIG + 1	1046
1047	1047		IICOLR(IJTRIG) = IEI	1047
1048	1048		END IF	1048
1049	1049		JJR = MOD(JR + 1 , 3) + 4	1049
1050	1050		IER = IABS(JS(JJR , IS))	1050
1051	1051	C		1051
1052	1052		IV1 = JE(1 , IER)	1052
1053	1053		IF(IV1 . EQ . IVV) THEN	1053
1054	1054		ISR = JE(3 , IER)	1054
1055	1055		ELSE	1055
1056	1056		ISR = JE(4 , IER)	1056
1057	1057		END IF	1057
1058	1058		END IF	1058
1059	1059	C		1059
1060	1060	340	CONTINUE	1060
1061	1061	C		1061
1062	1062		IF(IER . NE . IKKE3) THEN	1062
1063	1063		IS = ISR	1063
1064	1064		IE = IER	1064
1065	1065		GO TO 330	1065
1066	1066		END IF	1066
1067	1067	C		1067
1068	1068		IETRIG = IETRIG + 1	1068
1069	1069		IECRSS(IETRIG) = IKKE3	1069
1070	1070		IETRIG = IETRIG + 1	1070
1071	1071		IECRSS(IETRIG) = IKKE2	1071
1072	1072	C		1072
1073	1073		IJTRIG = IJTRIG - 1	1073
1074	1074	C		1074
1075	1075	C	THIRD LOOP SURROUNDING KKV3 IS DONE, FOURTH LOOP OVER KKV1 START .	1075
1076	1076	C		1076
1077	1077		IVV = KKV1	1077
1078	1078		IE = IABS(IICOLR(IJTRIG + 1))	1078
1079	1079		IF(JE(5 , IE) . NE . 0) THEN	1079
1080	1080		IER = IE	1080
1081	1081		GO TO 350	1081
1082	1082		END IF	1082
1083	1083		IV1 = JE(1 , IE)	1083
1084	1084		IF(IV1 . EQ . IVV) THEN	1084
1085	1085		ISI = JE(3 , IE)	1085
1086	1086		ELSE	1086
1087	1087		ISI = JE(4 , IE)	1087
1088	1088		END IF	1088
1089	1089		IS = ISI	1089
1090	1090		ISI = 0	1090
1091	1091	C		1091
1092	1092		ILOOP = 0	1092
1093	1093	360	CONTINUE	1093
1094	1094		JDOUBL = IABS(IICOLR(IJTRIG))	1094
1095	1095	C		1095
1096	1096		ITRIG = ITRIG + 1	1096
1097	1097		ISCRSS(ITRIG) = IS	1097
1098	1098		IETRIG = IETRIG + 1	1098
1099	1099		IECRSS(IETRIG) = IE	1099
1100	1100	C		1100
1101	1101		IF(1101
1102	1102	.	HYDFLX(IS , 4) . GT . FLUXPP . OR .	1102
1103	1103	.	HYDFLX(IS , 2) . GT . FLUXUU . OR .	1103
1104	1104	.	HYDFLX(IS , 1) . GT . FLUXRR . OR .	1104
1105	1105	.	KSDELTA(IS) . GT . NIDUMP . OR .	1105
1106	1106	.	XS(3 , IS) . GT . AREVGG) THEN	1106
1107	1107		INDCTR = 3	1107
1108	1108		RETURN	1108
1109	1109		END IF	1109

```

1110 1110 C
1111 1111 DO 370 IR = 1, 3
1112 1112 JR = MOD( IR, 3 ) + 1
1113 1113 IEA = IABS( JS( JR + 3, IS ) )
1114 1114 IF( IEA .EQ. IE ) THEN
1115 1115 IIR = MOD( JR, 3 ) + 4
1116 1116 IEI = JS( IIR, IS )
1117 1117 IEIB = IABS( IEI )
1118 1118 XEIEB = XE( 1, IEIB )
1119 1119 XYLNGT = XYLNGT + XEIEB
1120 1120 IF( XYLNG .LT. XEIEB ) XYLNG = XEIEB
1121 1121 IF( XYSHRT .GT. XEIEB ) XYSHRT = XEIEB
1122 1122 ILOOP = ILOOP + 1
1123 1123 IF( ILOOP .EQ. 1 .AND. JDOUBL .EQ. IEIB ) THEN
1124 1124 JLOOP = 3
1125 1125 IETRIG = IETRIG + 1
1126 1126 IECRSS( IETRIG ) = JDOUBL
1127 1127 IJTRIG = IJTRIG - 1
1128 1128 IF( IEI .GT. 0 ) THEN
1129 1129 JKVV = JE( 1, IEIB )
1130 1130 ELSE
1131 1131 JKVV = JE( 2, IEIB )
1132 1132 END IF
1133 1133 JVDDELT = JVDDELT + 1
1134 1134 IVDDELT( JVDDELT ) = JKVV
1135 1135 ILOOP = 0
1136 1136 ELSE
1137 1137 IJTRIG = IJTRIG + 1
1138 1138 IICOLR( IJTRIG ) = IEI
1139 1139 END IF
1140 1140 JJR = MOD( JR + 1, 3 ) + 4
1141 1141 IER = IABS( JS( JJR, IS ) )
1142 1142 C
1143 1143 IV1 = JE( 1, IER )
1144 1144 IF( IV1 .EQ. IVV ) THEN
1145 1145 ISR = JE( 3, IER )
1146 1146 ELSE
1147 1147 ISR = JE( 4, IER )
1148 1148 END IF
1149 1149 END IF
1150 1150 C
1151 1151 370 CONTINUE
1152 1152 C
1153 1153 IF( ISR .NE. ISI ) THEN
1154 1154 IS = ISR
1155 1155 IE = IER
1156 1156 GO TO 360
1157 1157 END IF
1158 1158 C
1159 1159 350 CONTINUE
1160 1160 C
1161 1161 IETRIG = IETRIG + 1
1162 1162 IECRSS( IETRIG ) = IER
1163 1163 C
1164 1164 ITYPE = JE( 5, IER )
1165 1165 C
1166 1166 XEIEB = XE( 1, IER )
1167 1167 XEIEB = XXYIIB + XEIEB
1168 1168 XYLNGT = XYLNGT + XEIEB
1169 1169 IF( XYLNG .LT. XEIEB ) XYLNG = XEIEB
1170 1170 IF( XYSHRT .GT. XEIEB ) XYSHRT = XEIEB
1171 1171 C
1172 1172 INDCTR = 2
1173 1173 C IF( XYLNG / XYSHRT .GT. 10. .AND. JLOOP .EQ. 0 ) RETURN
1174 1174 C
1175 1175 IV1 = IVIN
1176 1176 IE1 = IICOLR( IJTRIG )
1177 1177 IF( IE1 .GT. 0 ) THEN
1178 1178 IV2 = JE( 2, IE1 )
1179 1179 ELSE
1180 1180 IV2 = JE( 1, - IE1 )
1181 1181 END IF
1182 1182 C
1183 1183 NEC = IECRSS( IETRIG )

```

```

1184 1184      IETRIG = IETRIG - 1
1185 1185 C
1186 1186      JV( 2 , IV2 ) = - NEC
1187 1187      JE( 1 , NEC ) = IV2
1188 1188      JE( 2 , NEC ) = IV1
1189 1189      JE( 4 , NEC ) = 0
1190 1190      JE( 5 , NEC ) = ITYPE
1191 1191 C
1192 1192      IJTRIG = IJTRIG + 1
1193 1193      IICOLR( IJTRIG ) = NEC
1194 1194 C
1195 1195      ELSE IF( IKKE . EQ . 3 ) THEN
1196 1196 C
1197 1197 C      BEGINING THE DELETION PROCESS IF KSD HAS TWO VERTECIS ON THE BOUNDARY
1198 1198 C      BUT THE EDGE THAT CONNECT THEM IS IN THE COMPUTATIONAL DOMAIN.
1199 1199 C      THE FIRST LOOP IS AROUND VERTEX KKV1 .
1200 1200 C
1201 1201      IVV = KKV1
1202 1202      IE = IEINI
1203 1203      XXYIIB = XE( 1 , IE )
1204 1204      IV1 = JE( 1 , IE )
1205 1205      IVINI = JE( 2 , IE )
1206 1206      IF( IV1 . EQ . IVV ) THEN
1207 1207      ISI = JE( 3 , IE )
1208 1208      ELSE
1209 1209      ISI = JE( 4 , IE )
1210 1210      END IF
1211 1211      IS = ISI
1212 1212 C
1213 1213 380 CONTINUE
1214 1214 C
1215 1215      ITRIG = ITRIG + 1
1216 1216      ISCRSS( ITRIG ) = IS
1217 1217      IETRIG = IETRIG + 1
1218 1218      IECRSS( IETRIG ) = IE
1219 1219 C
1220 1220      IF(
1221 1221 .      HYDFLX( IS , 4 ) . GT . FLUXPP . OR .
1222 1222 .      HYDFLX( IS , 2 ) . GT . FLUXUU . OR .
1223 1223 .      HYDFLX( IS , 1 ) . GT . FLUXRR . OR .
1224 1224 .      KSDDEL( IS ) . GT . NIDUMP . OR .
1225 1225 .      XS( 3 , IS ) . GT . AREVGG ) THEN
1226 1226      INDCTR = 3
1227 1227      RETURN
1228 1228      END IF
1229 1229 C
1230 1230      DO 390 IR = 1 , 3
1231 1231      JR = MOD( IR , 3 ) + 1
1232 1232      IEA = IABS( JS( JR + 3 , IS ) )
1233 1233      IF( IEA . EQ . IE ) THEN
1234 1234      IIR = MOD( JR , 3 ) + 4
1235 1235      IEI = JS( IIR , IS )
1236 1236      IEIB = IABS( IEI )
1237 1237      XEIEB = XE( 1 , IEIB )
1238 1238      XYLNLT = XYLNLT + XEIEB
1239 1239      IF( XYLONG . LT . XEIEB ) XYLONG = XEIEB
1240 1240      IF( XYSHRT . GT . XEIEB ) XYSHRT = XEIEB
1241 1241      IJTRIG = IJTRIG + 1
1242 1242      IICOLR( IJTRIG ) = IEI
1243 1243      JJR = MOD( JR + 1 , 3 ) + 4
1244 1244      IER = IABS( JS( JJR , IS ) )
1245 1245 C
1246 1246      IV1 = JE( 1 , IER )
1247 1247      IF( IV1 . EQ . IVV ) THEN
1248 1248      ISR = JE( 3 , IER )
1249 1249      ELSE
1250 1250      ISR = JE( 4 , IER )
1251 1251      END IF
1252 1252      END IF
1253 1253 C
1254 1254 390 CONTINUE
1255 1255 C
1256 1256      IF( IER . NE . IKKE3 ) THEN
1257 1257      IS = ISR

```

```

1258 1258          IE = IER                      1258
1259 1259          GO TO 380                      1259
1260 1260          END IF                          1260
1261 1261          C                                1261
1262 1262          C FIRST LOOP SURROUNDING KKV1 IS DONE, SECOND LOOP OVER KKV2 START . 1262
1263 1263          C                                1263
1264 1264          IJTRIG = IJTRIG - 1              1264
1265 1265          400 CONTINUE                      1265
1266 1266          C                                1266
1267 1267          IEJK = IICOLR( IJTRIG )          1267
1268 1268          IF( IEJK . GT . 0 ) THEN          1268
1269 1269          IVIEJK = JE( 1 , IEJK )          1269
1270 1270          IJEJK5 = JE( 5 , IEJK )         1270
1271 1271          ELSE                               1271
1272 1272          IVIEJK = JE( 2 , -IEJK )           1272
1273 1273          IJEJK5 = JE( 5 , -IEJK )         1273
1274 1274          END IF                          1274
1275 1275          C                                1275
1276 1276          IF( IJEJK5 . EQ . 0 ) THEN       1276
1277 1277          JLOOP = 1                        1277
1278 1278          C                                1278
1279 1279          C INTERMEDIATE LOOP START .     1279
1280 1280          C                                1280
1281 1281          IEJK1 = IABS( IICOLR( IJTRIG - 1 ) ) 1281
1282 1282          IEJK2 = IABS( IEJK )              1282
1283 1283          IETRIG = IETRIG + 1               1283
1284 1284          IECRSS( IETRIG ) = IEJK2         1284
1285 1285          IJTRIG = IJTRIG - 2              1285
1286 1286          IVV = IVIEJK                     1286
1287 1287          JVDDELT = JVDDELT + 1           1287
1288 1288          IVDDELT( JVDDELT ) = IVV         1288
1289 1289          IE = IEJK1                       1289
1290 1290          IV1 = JE( 1 , IE )               1290
1291 1291          IF( IV1 . EQ . IVV ) THEN         1291
1292 1292          ISI = JE( 3 , IE )               1292
1293 1293          ELSE                               1293
1294 1294          ISI = JE( 4 , IE )               1294
1295 1295          END IF                          1295
1296 1296          IS = ISI                        1296
1297 1297          IET = IEJK2                     1297
1298 1298          C                                1298
1299 1299          410 CONTINUE                      1299
1300 1300          C                                1300
1301 1301          ITRIG = ITRIG + 1               1301
1302 1302          ISCRSS( ITRIG ) = IS             1302
1303 1303          C                                1303
1304 1304          IETRIG = IETRIG + 1             1304
1305 1305          IECRSS( IETRIG ) = IE           1305
1306 1306          C                                1306
1307 1307          IF(                              1307
1308 1308          . HYDFLX( IS , 4 ) . GT . FLUXPP . OR . 1308
1309 1309          . HYDFLX( IS , 2 ) . GT . FLUXUU . OR . 1309
1310 1310          . HYDFLX( IS , 1 ) . GT . FLUXRR . OR . 1310
1311 1311          . KSDELTA( IS ) . GT . NIDUMP . OR . 1311
1312 1312          . XS( 3 , IS ) . GT . AREVGG ) THEN 1312
1313 1313          INDCTR = 3                       1313
1314 1314          RETURN                          1314
1315 1315          END IF                          1315
1316 1316          C                                1316
1317 1317          DO 420 IR = 1 , 3                  1317
1318 1318          JR = MOD( IR , 3 ) + 1            1318
1319 1319          IEA = IABS( JS( JR + 3 , IS ) )    1319
1320 1320          IF( IEA . EQ . IE ) THEN          1320
1321 1321          IIR = MOD( JR , 3 ) + 4          1321
1322 1322          IEI = JS( IIR , IS )              1322
1323 1323          IEIB = IABS( IEI )                1323
1324 1324          XEIEB = XE( 1 , IEIB )           1324
1325 1325          XYLNGT = XYLNGT + XEIEB         1325
1326 1326          IF( XYLONG . LT . XEIEB ) XYLONG = XEIEB 1326
1327 1327          IF( XYSHRT . GT . XEIEB ) XYSHRT = XEIEB 1327
1328 1328          IIKK = IABS( IICOLR( IJTRIG ) )  1328
1329 1329          IF( IIKK . EQ . IEIB ) THEN      1329
1330 1330          JLOOP = 2                       1330
1331 1331          IETRIG = IETRIG + 1              1331

```

```

1332 1332      IECRSS( IETRIG ) = IEIB
1333 1333      IJTRIG = IJTRIG - 1
1334 1334      IF( IEI . GT . 0 ) THEN
1335 1335      JKVV = JE( 1 , IEIB )
1336 1336      ELSE
1337 1337      JKVV = JE( 2 , IEIB )
1338 1338      END IF
1339 1339      JVDDELT = JVDDELT + 1
1340 1340      IVDEL( JVDDELT ) = JKVV
1341 1341      ELSE
1342 1342      IJTRIG = IJTRIG + 1
1343 1343      IICOLR( IJTRIG ) = IEI
1344 1344      END IF
1345 1345      JJR = MOD( JR + 1 , 3 ) + 4
1346 1346      IER = IABS( JS( JJR , IS ) )
1347 1347      C
1348 1348      IV1 = JE( 1 , IER )
1349 1349      IF( IV1 . EQ . IVV ) THEN
1350 1350      ISR = JE( 3 , IER )
1351 1351      ELSE
1352 1352      ISR = JE( 4 , IER )
1353 1353      END IF
1354 1354      C
1355 1355      C
1356 1356      420 CONTINUE
1357 1357      C
1358 1358      IF( IER . NE . IET ) THEN
1359 1359      IS = ISR
1360 1360      IE = IER
1361 1361      GO TO 410
1362 1362      END IF
1363 1363      C
1364 1364      GO TO 400
1365 1365      END IF
1366 1366      C
1367 1367      C INTERMEDIATE LOOP IS DONE, SECOND LOOP OVER KKV2 START .
1368 1368      C
1369 1369      IVV = KKV2
1370 1370      IE = IEIN2
1371 1371      IVIN2 = JE( 2 , IE )
1372 1372      IEJJK = IICOLR( IJTRIG )
1373 1373      IV1 = JE( 1 , IE )
1374 1374      IF( IV1 . EQ . IVV ) THEN
1375 1375      ISI = JE( 3 , IE )
1376 1376      ELSE
1377 1377      ISI = JE( 4 , IE )
1378 1378      END IF
1379 1379      IS = ISI
1380 1380      C
1381 1381      430 CONTINUE
1382 1382      C
1383 1383      ITRIG = ITRIG + 1
1384 1384      ISCRSS( ITRIG ) = IS
1385 1385      C
1386 1386      IETRIG = IETRIG + 1
1387 1387      IECRSS( IETRIG ) = IE
1388 1388      C
1389 1389      IF(
1390 1390      . HYDFLX( IS , 4 ) . GT . FLUXPP . OR .
1391 1391      . HYDFLX( IS , 2 ) . GT . FLUXUU . OR .
1392 1392      . HYDFLX( IS , 1 ) . GT . FLUXRR . OR .
1393 1393      . KSDEL( IS ) . GT . NIDUMP . OR .
1394 1394      . XS( 3 , IS ) . GT . AREVGG ) THEN
1395 1395      INDOCTR = 3
1396 1396      RETURN
1397 1397      END IF
1398 1398      C
1399 1399      DO 440 IR = 1 , 3
1400 1400      JR = MOD( IR , 3 ) + 1
1401 1401      IEA = IABS( JS( JR + 3 , IS ) )
1402 1402      IF( IEA . EQ . IE ) THEN
1403 1403      IIR = MOD( JR , 3 ) + 4
1404 1404      IEI = JS( IIR , IS )
1405 1405      IEIB = IABS( IEI )

```

```

1406 1406      XEIEB = XE( 1 , IEIB )
1407 1407      XYLNGT = XYLNGT + XEIEB
1408 1408      IF( XYLONG . LT . XEIEB ) XYLONG = XEIEB
1409 1409      IF( XYSHRT . GT . XEIEB ) XYSHRT = XEIEB
1410 1410      IJTRIG = IJTRIG + 1
1411 1411      IICOLR( IJTRIG ) = IEI
1412 1412      JJR = MOD( JR + 1 , 3 ) + 4
1413 1413      IER = IABS( JS( JJR , IS ) )
1414 1414      C
1415 1415      IV1 = JE( 1 , IER )
1416 1416      IF( IV1 . EQ . IVV ) THEN
1417 1417      ISR = JE( 3 , IER )
1418 1418      ELSE
1419 1419      ISR = JE( 4 , IER )
1420 1420      END IF
1421 1421      END IF
1422 1422      C
1423 1423      440      CONTINUE
1424 1424      C
1425 1425      IF( IER . NE . IKKE2 ) THEN
1426 1426      IS = ISR
1427 1427      IE = IER
1428 1428      GO TO 430
1429 1429      END IF
1430 1430      C
1431 1431      IJTRIG = IJTRIG - 1
1432 1432      C
1433 1433      C      SECOND LOOP SURROUNDING KKV2 IS DONE, THIRD LOOP OVER KKV3 START .
1434 1434      C
1435 1435      IVV = KKV3
1436 1436      IE = IABS( IICOLR( IJTRIG + 1 ) )
1437 1437      IV1 = JE( 1 , IE )
1438 1438      IF( IV1 . EQ . IVV ) THEN
1439 1439      ISI = JE( 3 , IE )
1440 1440      ELSE
1441 1441      ISI = JE( 4 , IE )
1442 1442      END IF
1443 1443      IS = ISI
1444 1444      C
1445 1445      ILOOP = 0
1446 1446      450      CONTINUE
1447 1447      IDOUBL = IABS( IICOLR( IJTRIG ) )
1448 1448      C
1449 1449      ITRIG = ITRIG + 1
1450 1450      ISCRSS( ITRIG ) = IS
1451 1451      IETRIG = IETRIG + 1
1452 1452      IECRSS( IETRIG ) = IE
1453 1453      C
1454 1454      IF(
1455 1455      .      HYDFLX( IS , 4 ) . GT . FLUXPP . OR .
1456 1456      .      HYDFLX( IS , 2 ) . GT . FLUXUU . OR .
1457 1457      .      HYDFLX( IS , 1 ) . GT . FLUXRR . OR .
1458 1458      .      KSDELTA( IS ) . GT . NIDUMP . OR .
1459 1459      .      XS( 3 , IS ) . GT . AREVGG ) THEN
1460 1460      INDOCTR = 3
1461 1461      RETURN
1462 1462      END IF
1463 1463      C
1464 1464      DO 460 IR = 1 , 3
1465 1465      JR = MOD( IR , 3 ) + 1
1466 1466      IEA = IABS( JS( JR + 3 , IS ) )
1467 1467      IF( IEA . EQ . IE ) THEN
1468 1468      IIR = MOD( JR , 3 ) + 4
1469 1469      IEI = JS( IIR , IS )
1470 1470      IEIB = IABS( IEI )
1471 1471      XEIEB = XE( 1 , IEIB )
1472 1472      XYLNGT = XYLNGT + XEIEB
1473 1473      IF( XYLONG . LT . XEIEB ) XYLONG = XEIEB
1474 1474      IF( XYSHRT . GT . XEIEB ) XYSHRT = XEIEB
1475 1475      ILOOP = ILOOP + 1
1476 1476      IF( ILOOP . EQ . 1 . AND . IDOUBL . EQ . IEIB ) THEN
1477 1477      JLOOP = 3
1478 1478      IETRIG = IETRIG + 1
1479 1479      IECRSS( IETRIG ) = IDOUBL

```



```

1480 1480 IJTRIG = IJTRIG - 1 1480
1481 1481 IF( IE1 . GT . 0 ) THEN 1481
1482 1482 JKVV = JE( 1 , IEIB ) 1482
1483 1483 ELSE 1483
1484 1484 JKVV = JE( 2 , IEIB ) 1484
1485 1485 END IF 1485
1486 1486 JVDDEL = JVDDEL + 1 1486
1487 1487 IVDEL( JVDDEL ) = JKVV 1487
1488 1488 ILOOP = 0 1488
1489 1489 ELSE 1489
1490 1490 IJTRIG = IJTRIG + 1 1490
1491 1491 IICOLR( IJTRIG ) = IEI 1491
1492 1492 END IF 1492
1493 1493 JJR = MOD( JR + 1 , 3 ) + 4 1493
1494 1494 IER = IABS( JS( JJR , IS ) ) 1494
1495 1495 C 1495
1496 1496 IVI = JE( 1 , IER ) 1496
1497 1497 IF( IVI . EQ . IVV ) THEN 1497
1498 1498 ISR = JE( 3 , IER ) 1498
1499 1499 ELSE 1499
1500 1500 ISR = JE( 4 , IER ) 1500
1501 1501 END IF 1501
1502 1502 END IF 1502
1503 1503 C 1503
1504 1504 460 CONTINUE 1504
1505 1505 C 1505
1506 1506 IF( IER . NE . IKKE3 ) THEN 1506
1507 1507 IS = ISR 1507
1508 1508 IE = IER 1508
1509 1509 GO TO 450 1509
1510 1510 END IF 1510
1511 1511 C 1511
1512 1512 IETRIG = IETRIG + 1 1512
1513 1513 IECRSS( IETRIG ) = IKKE3 1513
1514 1514 IETRIG = IETRIG + 1 1514
1515 1515 IECRSS( IETRIG ) = IKKE2 1515
1516 1516 C 1516
1517 1517 IJTRIG = IJTRIG - 1 1517
1518 1518 C 1518
1519 1519 C THIRD LOOP SURROUNDING KKV3 IS DONE, FOURTH LOOP OVER KKV1 START . 1519
1520 1520 C 1520
1521 1521 IVV = KKV1 1521
1522 1522 IE = IABS( IICOLR( IJTRIG + 1 ) ) 1522
1523 1523 IF( JE( 5 , IE ) . NE . 0 ) THEN 1523
1524 1524 IER = IE 1524
1525 1525 GO TO 470 1525
1526 1526 END IF 1526
1527 1527 IVI = JE( 1 , IE ) 1527
1528 1528 IF( IVI . EQ . IVV ) THEN 1528
1529 1529 ISI = JE( 3 , IE ) 1529
1530 1530 ELSE 1530
1531 1531 ISI = JE( 4 , IE ) 1531
1532 1532 END IF 1532
1533 1533 IS = ISI 1533
1534 1534 ISI = 0 1534
1535 1535 C 1535
1536 1536 ILOOP = 0 1536
1537 1537 480 CONTINUE 1537
1538 1538 JDDBL = IABS( IICOLR( IJTRIG ) ) 1538
1539 1539 C 1539
1540 1540 ITRIG = ITRIG + 1 1540
1541 1541 ISCRSS( ITRIG ) = IS 1541
1542 1542 IETRIG = IETRIG + 1 1542
1543 1543 IECRSS( IETRIG ) = IE 1543
1544 1544 C 1544
1545 1545 IF( 1545
1546 1546 . HYDFLX( IS , 4 ) . GT . FLUXPP . OR . 1546
1547 1547 . HYDFLX( IS , 2 ) . GT . FLUXUU . OR . 1547
1548 1548 . HYDFLX( IS , 1 ) . GT . FLUXRR . OR . 1548
1549 1549 . KSDEL( IS ) . GT . NIDUMP . OR . 1549
1550 1550 . XS( 3 , IS ) . GT . AREVGG ) THEN 1550
1551 1551 INOCTR = 3 1551
1552 1552 RETURN 1552
1553 1553 END IF 1553

```

```

1554 1554 C
1555 1555 DO 490 IR = 1 , 3
1556 1556 JR = MOD( IR , 3 ) + 1
1557 1557 IEA = IABS( JS( JR + 3 , IS ) )
1558 1558 IF( IEA . EQ . IE ) THEN
1559 1559 IIR = MOD( JR , 3 ) + 4
1560 1560 IEI = JS( IIR , IS )
1561 1561 IEIB = IABS( IEI )
1562 1562 XEIEB = XE( 1 , IEIB )
1563 1563 XYLNGT = XYLNGT + XEIEB
1564 1564 IF( XYLONG . LT . XEIEB ) XYLONG = XEIEB
1565 1565 IF( XYSHRT . GT . XEIEB ) XYSHRT = XEIEB
1566 1566 ILOOP = ILOOP + 1
1567 1567 IF( ILOOP . EQ . 1 . AND . JDOUBL . EQ . IEIB ) THEN
1568 1568 JLOOP = 4
1569 1569 IETRIG = IETRIG + 1
1570 1570 IECRSS( IETRIG ) = JDOUBL
1571 1571 IJTRIG = IJTRIG - 1
1572 1572 IF( IEI . GT . 0 ) THEN
1573 1573 JKVV = JE( 1 , IEIB )
1574 1574 ELSE
1575 1575 JKVV = JE( 2 , IEIB )
1576 1576 END IF
1577 1577 JVDDELT = JVDDELT + 1
1578 1578 IVDEL( JVDDELT ) = JKVV
1579 1579 ILOOP = 0
1580 1580 ELSE
1581 1581 IJTRIG = IJTRIG + 1
1582 1582 IICOLR( IJTRIG ) = IEI
1583 1583 END IF
1584 1584 JJR = MOD( JR + 1 , 3 ) + 4
1585 1585 IER = IABS( JS( JJR , IS ) )
1586 1586 C
1587 1587 IV1 = JE( 1 , IER )
1588 1588 IF( IV1 . EQ . IVV ) THEN
1589 1589 ISR = JE( 3 , IER )
1590 1590 ELSE
1591 1591 ISR = JE( 4 , IER )
1592 1592 END IF
1593 1593 END IF
1594 1594 C
1595 1595 490 CONTINUE
1596 1596 C
1597 1597 IF( ISR . NE . ISI ) THEN
1598 1598 IS = ISR
1599 1599 IE = IER
1600 1600 GO TO 480
1601 1601 END IF
1602 1602 C
1603 1603 470 CONTINUE
1604 1604 C
1605 1605 IETRIG = IETRIG + 1
1606 1606 IECRSS( IETRIG ) = IER
1607 1607 C
1608 1608 ITYPE = JE( 5 , IER )
1609 1609 C
1610 1610 XEIEB = XE( 1 , IER )
1611 1611 XEIEB = XYYIB + XEIEB
1612 1612 XYLNGT = XYLNGT + XEIEB
1613 1613 IF( XYLONG . LT . XEIEB ) XYLONG = XEIEB
1614 1614 IF( XYSHRT . GT . XEIEB ) XYSHRT = XEIEB
1615 1615 C
1616 1616 INDCTR = 2
1617 1617 C IF( XYLONG / XYSHRT . GT . 10 . AND . JLOOP . EQ . 0 ) RETURN
1618 1618 C
1619 1619 JE( 2 , IEJJK ) = IVIN2
1620 1620 C
1621 1621 IV1 = IVIN1
1622 1622 IE1 = IICOLR( IJTRIG )
1623 1623 IF( IE1 . GT . 0 ) THEN
1624 1624 IV2 = JE( 2 , IE1 )
1625 1625 ELSE
1626 1626 IV2 = JE( 1 , - IE1 )
1627 1627 END IF

```

```

1628 1628 C
1629 1629 NEC = IECRSS( IETRIG )
1630 1630 IETRIG = IETRIG - 1
1631 1631 C
1632 1632 JV( 2 , IV2 ) = - NEC
1633 1633 JE( 1 , NEC ) = IV2
1634 1634 JE( 2 , NEC ) = IV1
1635 1635 JE( 4 , NEC ) = 0
1636 1636 JE( 5 , NEC ) = ITYPE
1637 1637 C
1638 1638 IJTRIG = IJTRIG + 1
1639 1639 IICOLR( IJTRIG ) = NEC
1640 1640 C
1641 1641 ELSE IF( IKKE . EQ . 4 ) THEN
1642 1642 print*, 'ikke=4', ksd, ikke
1643 1643 C
1644 1644 C BEGINING THE DELETION PROCESS IF KSD HAS AN EDGE ON THE BOUNDARY
1645 1645 C AND THE THIRD VERTEX IS OLSO ON THE BOUNDARY.
1646 1646 C THE FIRST LOOP IS AROUND VERTEX KKV2 .
1647 1647 C
1648 1648 IVV = KKV2
1649 1649 IE = IEIN1
1650 1650 XXYYIB = XE( 1 , IE ) + XE( 1 , IKKE1 )
1651 1651 IV1 = JE( 1 , IE )
1652 1652 IVIN1 = JE( 2 , IE )
1653 1653 IF( IV1 . EQ . IVV ) THEN
1654 1654 ISI = JE( 3 , IE )
1655 1655 ELSE
1656 1656 ISI = JE( 4 , IE )
1657 1657 END IF
1658 1658 IS = ISI
1659 1659 C
1660 1660 500 CONTINUE
1661 1661 C
1662 1662 ITRIG = ITRIG + 1
1663 1663 ISCRSS( ITRIG ) = IS
1664 1664 IETRIG = IETRIG + 1
1665 1665 IECRSS( IETRIG ) = IE
1666 1666 C
1667 1667 IF(
1668 1668 . HYDFLX( IS , 4 ) . GT . FLUXPP . OR .
1669 1669 . HYDFLX( IS , 2 ) . GT . FLUXUU . OR .
1670 1670 . HYDFLX( IS , 1 ) . GT . FLUXRR . OR .
1671 1671 . KSDEL( IS ) . GT . MIDUMP . OR .
1672 1672 . XS( 3 , IS ) . GT . AREVGG ) THEN
1673 1673 INDOCTR = 3
1674 1674 RETURN
1675 1675 END IF
1676 1676 C
1677 1677 DO 510 IR = 1 , 3
1678 1678 JR = MOD( IR , 3 ) + 1
1679 1679 IEA = IABS( JS( JR + 3 , IS ) )
1680 1680 IF( IEA . EQ . IE ) THEN
1681 1681 IIR = MOD( JR , 3 ) + 4
1682 1682 IEI = JS( IIR , IS )
1683 1683 IEIB = IABS( IEI )
1684 1684 XEIEB = XE( 1 , IEIB )
1685 1685 XYLNGT = XYLNGT + XEIEB
1686 1686 IF( XYLONG . LT . XEIEB ) XYLONG = XEIEB
1687 1687 IF( XYSHRT . GT . XEIEB ) XYSHRT = XEIEB
1688 1688 IJTRIG = IJTRIG + 1
1689 1689 IICOLR( IJTRIG ) = IEI
1690 1690 JJR = MOD( JR + 1 , 3 ) + 4
1691 1691 IER = IABS( JS( JJR , IS ) )
1692 1692 C
1693 1693 IV1 = JE( 1 , IER )
1694 1694 IF( IV1 . EQ . IVV ) THEN
1695 1695 ISR = JE( 3 , IER )
1696 1696 ELSE
1697 1697 ISR = JE( 4 , IER )
1698 1698 END IF
1699 1699 END IF
1700 1700 C
1701 1701 510 CONTINUE

```

```

1702 1702 C
1703 1703 IF( IER . NE . IKKE1 ) THEN
1704 1704 IS = ISR
1705 1705 IE = IER
1706 1706 GO TO 500
1707 1707 END IF
1708 1708 C
1709 1709 C FIRST LOOP SURROUNDING KKV2 IS DONE, SECOND LOOP OVER KKV3 START .
1710 1710 C
1711 1711 IJTRIG = IJTRIG - 1
1712 1712 520 CONTINUE
1713 1713 C
1714 1714 IEJK = IICOLR( IJTRIG )
1715 1715 IF( IEJK . GT . 0 ) THEN
1716 1716 IVIEJK = JE( 1 , IEJK )
1717 1717 IJEJK5 = JE( 5 , IEJK )
1718 1718 ELSE
1719 1719 IVIEJK = JE( 2 , -IEJK )
1720 1720 IJEJK5 = JE( 5 , -IEJK )
1721 1721 END IF
1722 1722 C
1723 1723 IF( IJEJK5 . EQ . 0 ) THEN
1724 1724 C
1725 1725 C INTERMEDIATE LOOP START .
1726 1726 C
1727 1727 JLOOP = 1
1728 1728 IEJKI = IABS( IICOLR( IJTRIG - 1 ) )
1729 1729 IEJK2 = IABS( IEJK )
1730 1730 IETRIG = IETRIG + 1
1731 1731 IECRSS( IETRIG ) = IEJK2
1732 1732 IJTRIG = IJTRIG - 2
1733 1733 IVV = IVIEJK
1734 1734 JVDDELT = JVDDELT + 1
1735 1735 IVDEL( JVDDELT ) = IVV
1736 1736 IE = IEJKI
1737 1737 IV1 = JE( 1 , IE )
1738 1738 IF( IV1 . EQ . IVV ) THEN
1739 1739 ISI = JE( 3 , IE )
1740 1740 ELSE
1741 1741 ISI = JE( 4 , IE )
1742 1742 END IF
1743 1743 IS = ISI
1744 1744 IET = IEJK2
1745 1745 C
1746 1746 530 CONTINUE
1747 1747 C
1748 1748 ITRIG = ITRIG + 1
1749 1749 ISCRSS( ITRIG ) = IS
1750 1750 C
1751 1751 IETRIG = IETRIG + 1
1752 1752 IFCRSS( IETRIG ) = IE
1753 1753 C
1754 1754 IF(
1755 1755 . HYDFLX( IS , 4 ) . GT . FLUXPP . OR .
1756 1756 . HYDFLX( IS , 2 ) . GT . FLUXUU . OR .
1757 1757 . HYDFLX( IS , 1 ) . GT . FLUXRR . OR .
1758 1758 . KSDFLT( IS ) . GT . NIDUMP . OR .
1759 1759 . XS( 3 , IS ) . GT . AREVGG ) THEN
1760 1760 INDCR = 3
1761 1761 RETURN
1762 1762 END IF
1763 1763 C
1764 1764 DO 540 IR = 1 , 3
1765 1765 JR = MOD( IR , 3 ) + 1
1766 1766 IEA = IABS( JS( JR + 3 , IS ) )
1767 1767 IF( IEA . EQ . IE ) THEN
1768 1768 IIR = MOD( JR , 3 ) + 4
1769 1769 IEI = JS( IIR , IS )
1770 1770 IEIB = IABS( IEI )
1771 1771 XEIEB = XE( 1 , IEIB )
1772 1772 XYLNGT = XYLNGT + XEIEB
1773 1773 IF( XYLONG . LT . XEIEB ) XYLONG = XEIEB
1774 1774 IF( XYSHRT . GT . XEIEB ) XYSHRT = XEIEB
1775 1775 IIKK = IABS( IICOLR( IJTRIG ) )

```

```

1776 1776      IF( IIKK . EQ . IEIB ) THEN
1777 1777      JLOOP = 2
1778 1778      IETRIG = IETRIG + 1
1779 1779      IECRSS( IETRIG ) = IEIB
1780 1780      IJTRIG = IJTRIG - 1
1781 1781      IF( IEI . GT . 0 ) THEN
1782 1782      JKVV = JE( 1 , IEIB )
1783 1783      ELSE
1784 1784      JKVV = JE( 2 , IEIB )
1785 1785      END IF
1786 1786      JVDELT = JVDELT + 1
1787 1787      IVDELT( JVDELT ) = JKVV
1788 1788      ELSE
1789 1789      IJTRIG = IJTRIG + 1
1790 1790      IICOLR( IJTRIG ) = IEI
1791 1791      END IF
1792 1792      JJR = MOD( JR + 1 , 3 ) + 4
1793 1793      IER = IABS( JS( JJR , IS ) )
1794 1794      C
1795 1795      IV1 = JE( 1 , IER )
1796 1796      IF( IV1 . EQ . IVV ) THEN
1797 1797      ISR = JE( 3 , IER )
1798 1798      ELSE
1799 1799      ISR = JE( 4 , IER )
1800 1800      END IF
1801 1801      END IF
1802 1802      C
1803 1803      540 CONTINUE
1804 1804      C
1805 1805      IF( IER . NE . IET ) THEN
1806 1806      IS = ISR
1807 1807      IE = IER
1808 1808      GO TO 530
1809 1809      END IF
1810 1810      C
1811 1811      GO TO 520
1812 1812      END IF
1813 1813      C
1814 1814      C INTERMEDIATE LOOP IS DONE, SECOND LOOP OVER KKV3 START .
1815 1815      C
1816 1816      IVV = KKV3
1817 1817      IE = IEIN2
1818 1818      IVIN2 = JE( 2 , IE )
1819 1819      IEJJK = IICOLR( IJTRIG )
1820 1820      IV1 = JE( 1 , IE )
1821 1821      IF( IV1 . EQ . IVV ) THEN
1822 1822      ISI = JE( 3 , IE )
1823 1823      ELSE
1824 1824      ISI = JE( 4 , IE )
1825 1825      END IF
1826 1826      IS = ISI
1827 1827      C
1828 1828      550 CONTINUE
1829 1829      C
1830 1830      ITRIG = ITRIG + 1
1831 1831      ISCRSS( ITRIG ) = IS
1832 1832      C
1833 1833      IETRIG = IETRIG + 1
1834 1834      IECRSS( IETRIG ) = IE
1835 1835      C
1836 1836      IF(
1837 1837      . HYDFLX( IS , 4 ) . GT . FLUXPP . OR .
1838 1838      . HYDFLX( IS , 2 ) . GT . FLUXUU . OR .
1839 1839      . HYDFLX( IS , 1 ) . GT . FLUXRR . OR .
1840 1840      . KSDELTA( IS ) . GT . NIDUMP . OR .
1841 1841      . XS( 3 , IS ) . GT . AREVGG ) THEN
1842 1842      INDCTR = 3
1843 1843      RETURN
1844 1844      END IF
1845 1845      C
1846 1846      DO 560 IR = 1 , 3
1847 1847      JR = MOD( IR , 3 ) + 1
1848 1848      IEA = IABS( JS( JR + 3 , IS ) )
1849 1849      IF( IEA . EQ . IE ) THEN

```

```

1850 1850      IIR = MOD( JR , 3 ) + 4      1850
1851 1851      IEI = JS( IIR , IS )      1851
1852 1852      IEIB = IABS( IEI )      1852
1853 1853      XEIEB = XE( 1 , IEIB )      1853
1854 1854      XYLNGT = XYLNGT + XEIEB      1854
1855 1855      IF( XYLONG . LT . XEIEB ) XYLONG = XEIEB      1855
1856 1856      IF( XYSHRT . GT . XEIEB ) XYSHRT = XEIEB      1856
1857 1857      IJTRIG = IJTRIG + 1      1857
1858 1858      IICOLR( IJTRIG ) = IEI      1858
1859 1859      JJR = MOD( JR + 1 , 3 ) + 4      1859
1860 1860      IER = IABS( JS( JJR , IS ) )      1860
1861 1861      C      1861
1862 1862      IV1 = JE( 1 , IER )      1862
1863 1863      IF( IV1 . EQ . IVV ) THEN      1863
1864 1864      ISR = JE( 3 , IER )      1864
1865 1865      ELSE      1865
1866 1866      ISR = JE( 4 , IER )      1866
1867 1867      END IF      1867
1868 1868      END IF      1868
1869 1869      C      1869
1870 1870      560      CONTINUE      1870
1871 1871      C      1871
1872 1872      IF( IER . NE . IKKE3 ) THEN      1872
1873 1873      IS = ISR      1873
1874 1874      IE = IER      1874
1875 1875      GO TO 550      1875
1876 1876      END IF      1876
1877 1877      C      1877
1878 1878      IJTRIG = IJTRIG - 1      1878
1879 1879      IETRIG = IETRIG + 1      1879
1880 1880      IECRSS( IETRIG ) = IKKE3      1880
1881 1881      IETRIG = IETRIG + 1      1881
1882 1882      IECRSS( IETRIG ) = IKKE1      1882
1883 1883      C      1883
1884 1884      C      SECOND LOOP SURROUNDING KKV3 IS DONE, THIRD LOOP OVER KKV1 START .      1884
1885 1885      C      1885
1886 1886      IVV = KKV1      1886
1887 1887      IE = IABS( IICOLR( IJTRIG + 1 ) )      1887
1888 1888      IF( JE( 5 , IE ) . NE . 0 ) THEN      1888
1889 1889      IER = IE      1889
1890 1890      GO TO 570      1890
1891 1891      END IF      1891
1892 1892      IV1 = JE( 1 , IE )      1892
1893 1893      IF( IV1 . EQ . IVV ) THEN      1893
1894 1894      ISI = JE( 3 , IE )      1894
1895 1895      ELSE      1895
1896 1896      ISI = JE( 4 , IE )      1896
1897 1897      END IF      1897
1898 1898      IS = ISI      1898
1899 1899      ISI = 0      1899
1900 1900      C      1900
1901 1901      580      CONTINUE      1901
1902 1902      C      1902
1903 1903      ITRIG = ITRIG + 1      1903
1904 1904      ISCRSS( ITRIG ) = IS      1904
1905 1905      IETRIG = IETRIG + 1      1905
1906 1906      IECRSS( IETRIG ) = IE      1906
1907 1907      C      1907
1908 1908      IF(      1908
1909 1909      .      HYDFLX( IS , 4 ) . GT . FLUXPP . OR .      1909
1910 1910      .      HYDFLX( IS , 2 ) . GT . FLUXUU . OR .      1910
1911 1911      .      HYDFLX( IS , 1 ) . GT . FLUXRR . OR .      1911
1912 1912      .      KSDEL( IS ) . GT . NIDUMP . OR .      1912
1913 1913      .      XS( 3 , IS ) . GT . AREVGG ) THEN      1913
1914 1914      INDCR = 3      1914
1915 1915      RETURN      1915
1916 1916      END IF      1916
1917 1917      C      1917
1918 1918      DO 590 IR = 1 , 3      1918
1919 1919      JR = MOD( IR , 3 ) + 1      1919
1920 1920      IEA = IABS( JS( JR + 3 , IS ) )      1920
1921 1921      IF( IEA . EQ . IE ) THEN      1921
1922 1922      IIR = MOD( JR , 3 ) + 4      1922
1923 1923      IEI = JS( IIR , IS )      1923

```

1924	1924	IEIB = IABS(IEI)	1924
1925	1925	XEIEB = XE(1 , IEIB)	1925
1926	1926	XYLNGT = XYLNGT + XEIEB	1926
1927	1927	IF(XYLNG . LT . XEIEB) XYLNG = XEIEB	1927
1928	1928	IF(XYSHRT . GT . XEIEB) XYSHRT = XEIEB	1928
1929	1929	IJTRIG = IJTRIG + 1	1929
1930	1930	IICOLR(IJTRIG) = IEI	1930
1931	1931	JJR = MOD(JR + 1 , 3) + 4	1931
1932	1932	IER = IABS(JS(JJR , IS))	1932
1933	1933	C	1933
1934	1934	IV1 = JE(1 , IER)	1934
1935	1935	IF(IV1 . EQ . IVV) THEN	1935
1936	1936	ISR = JE(3 , IER)	1936
1937	1937	ELSE	1937
1938	1938	ISR = JE(4 , IER)	1938
1939	1939	END IF	1939
1940	1940	END IF	1940
1941	1941	C	1941
1942	1942	590 CONTINUE	1942
1943	1943	C	1943
1944	1944	IF(ISR . NE . ISI) THEN	1944
1945	1945	IS = ISR	1945
1946	1946	IE = IER	1946
1947	1947	GO TO 580	1947
1948	1948	END IF	1948
1949	1949	C	1949
1950	1950	570 CONTINUE	1950
1951	1951	C	1951
1952	1952	IETRIG = IETRIG + 1	1952
1953	1953	IECRSS(IETRIG) = IER	1953
1954	1954	C	1954
1955	1955	ITYPE = JE(5 , IER)	1955
1956	1956	C	1956
1957	1957	XEIEB = XE(1 , IER)	1957
1958	1958	XEIEB = XYYI8 + XEIEB	1958
1959	1959	XYLNGT = XYLNGT + XEIEB	1959
1960	1960	IF(XYLNG . LT . XEIEB) XYLNG = XEIEB	1960
1961	1961	IF(XYSHRT . GT . XEIEB) XYSHRT = XEIEB	1961
1962	1962	C	1962
1963	1963	INDCTR = 2	1963
1964	1964	C IF(XYLNG / XYSHRT . GT . 10 . . AND . JLOOP . EQ . 0) RETURN	1964
1965	1965	C	1965
1966	1966	JE(2 , IEJKK) = IVIN2	1966
1967	1967	C	1967
1968	1968	IV1 = IVIN1	1968
1969	1969	IE1 = IICOLR(IJTRIG)	1969
1970	1970	IF(IE1 . GT . 0) THEN	1970
1971	1971	IV2 = JE(2 , IE1)	1971
1972	1972	ELSE	1972
1973	1973	IV2 = JE(1 , - IE1)	1973
1974	1974	END IF	1974
1975	1975	C	1975
1976	1976	NEC = IECRSS(IETRIG)	1976
1977	1977	IETRIG = IETRIG - 1	1977
1978	1978	C	1978
1979	1979	JV(2 , IV2) = - NEC	1979
1980	1980	JE(1 , NEC) = IV2	1980
1981	1981	JE(2 , NEC) = IV1	1981
1982	1982	JE(4 , NEC) = 0	1982
1983	1983	JE(5 , NEC) = ITYPE	1983
1984	1984	C	1984
1985	1985	IJTRIG = IJTRIG + 1	1985
1986	1986	IICOLR(IJTRIG) = NEC	1986
1987	1987	C	1987
1988	1988	ELSE IF(IKKE . EQ . 5) THEN	1988
1989	1989	print*, 'ikke=5', ksd, ikke	1989
1990	1990	C	1990
1991	1991	C BEGINNING THE DELETION PROCESS IF KSD HAS AN EDGE ON THE BOUNDARY	1991
1992	1992	C AND THE THIRD VERTEX IS ALSO ON THE BOUNDARY.	1992
1993	1993	C THE FIRST LOOP IS AROUND VERTEX KKV3 .	1993
1994	1994	C	1994
1995	1995	IVV = KKV3	1995
1996	1996	IE = IEIN2	1996
1997	1997	XYYI8 = XE(1 , IE)	1997

```

1998 1998          IV1 = JE( 1 , IE )          1998
1999 1999          IVIN1 = JE( 2 , IE )        1999
2000 2000          IF( IV1 . EQ . IVV ) THEN    2000
2001 2001          ISI = JE( 3 , IE )          2001
2002 2002          ELSE                        2002
2003 2003          ISI = JE( 4 , IE )          2003
2004 2004          END IF                      2004
2005 2005          IS = ISI                    2005
2006 2006          C                          2006
2007 2007          600      CONTINUE            2007
2008 2008          C                          2008
2009 2009          ITRIG = ITRIG + 1           2009
2010 2010          ISCRSS( ITRIG ) = IS        2010
2011 2011          IETRIG = IETRIG + 1         2011
2012 2012          IECRSS( IETRIG ) = IE       2012
2013 2013          C                          2013
2014 2014          IF(                          2014
2015 2015          .   HYDFLX( IS , 4 ) . GT . FLUXPP . OR . 2015
2016 2016          .   HYDFLX( IS , 2 ) . GT . FLUXUU . OR . 2016
2017 2017          .   HYDFLX( IS , 1 ) . GT . FLUXRR . OR . 2017
2018 2018          .   KSDFLT( IS ) . GT . NIDUMP . OR . 2018
2019 2019          .   XS( 3 , IS ) . GT . AREVGG ) THEN 2019
2020 2020          INDCTR = 3                   2020
2021 2021          RETURN                       2021
2022 2022          END IF                      2022
2023 2023          C                          2023
2024 2024          DO 610 IR = 1 , 3           2024
2025 2025          JR = MOD( IR , 3 ) + 1       2025
2026 2026          IEA = IABS( JS( JR + 3 , IS ) ) 2026
2027 2027          IF( IEA . EQ . IE ) THEN     2027
2028 2028          IIR = MOD( JR , 3 ) + 4     2028
2029 2029          IEI = JS( IIR , IS )        2029
2030 2030          IEIB = IABS( IEI )           2030
2031 2031          XEIEB = XE( 1 , IEIB )       2031
2032 2032          XYLNGT = XYLNGT + XEIEB     2032
2033 2033          IF( XYLONG . LT . XEIEB ) XYLONG = XEIEB 2033
2034 2034          IF( XYSHRT . GT . XEIEB ) XYSHRT = XEIEB 2034
2035 2035          IJTRIG = IJTRIG + 1         2035
2036 2036          IICOLR( IJTRIG ) = IEI      2036
2037 2037          JJR = MOD( JR + 1 , 3 ) + 4 2037
2038 2038          IER = IABS( JS( JJR , IS ) ) 2038
2039 2039          C                          2039
2040 2040          IV1 = JE( 1 , IER )           2040
2041 2041          IF( IV1 . EQ . IVV ) THEN    2041
2042 2042          ISR = JE( 3 , IER )          2042
2043 2043          ELSE                        2043
2044 2044          ISR = JE( 4 , IER )          2044
2045 2045          END IF                      2045
2046 2046          END IF                      2046
2047 2047          C                          2047
2048 2048          610      CONTINUE            2048
2049 2049          C                          2049
2050 2050          IF( IER . NE . IKKE2 ) THEN  2050
2051 2051          IS = ISR                      2051
2052 2052          IE = IER                     2052
2053 2053          GO TO 600                    2053
2054 2054          END IF                      2054
2055 2055          C                          2055
2056 2056          C      FIRST LOOP SURROUNDING KKV3 IS DONE, SECOND LOOP OVER KKV2 START . 2056
2057 2057          C                          2057
2058 2058          IJTRIG = IJTRIG - 1         2058
2059 2059          620      CONTINUE            2059
2060 2060          C                          2060
2061 2061          IEJK = IICOLR( IJTRIG )       2061
2062 2062          IF( IEJK . GT . 0 ) THEN       2062
2063 2063          IVIEJK = JE( 1 , IEJK )       2063
2064 2064          IJEJK5 = JE( 5 , IEJK )      2064
2065 2065          ELSE                        2065
2066 2066          IVIEJK = JE( 2 , -IEJK )      2066
2067 2067          IJEJK5 = JE( 5 , -IEJK )     2067
2068 2068          END IF                      2068
2069 2069          C                          2069
2070 2070          IF( IJEJK5 . EQ . 0 ) THEN    2070
2071 2071          C                          2071

```



```

2072 2072 C   INTERMEDIATE LOOP START .
2073 2073 C
2074 2074       JLOOP = 1
2075 2075       IEJKI = IABS( IICOLR( IJTRIG - 1 ) )
2076 2076       IEJK2 = IABS( IEJK )
2077 2077       IETRIG = IETRIG + 1
2078 2078       IECRSS( IETRIG ) = IEJK2
2079 2079       IJTRIG = IJTRIG - 2
2080 2080       IVV = IVIEJK
2081 2081       JVDELTA = JVDELTA + 1
2082 2082       IVDELTA( JVDELTA ) = IVV
2083 2083       IE = IEJKI
2084 2084       IV1 = JE( 1 , IE )
2085 2085       IF( IV1 .EQ. IVV ) THEN
2086 2086       ISI = JE( 3 , IE )
2087 2087       ELSE
2088 2088       ISI = JE( 4 , IE )
2089 2089       END IF
2090 2090       IS = ISI
2091 2091       IET = IEJK2
2092 2092 C
2093 2093 630 CONTINUE
2094 2094 C
2095 2095       ITRIG = ITRIG + 1
2096 2096       ISCRSS( ITRIG ) = IS
2097 2097 C
2098 2098       IETRIG = IETRIG + 1
2099 2099       IECRSS( IETRIG ) = IE
2100 2100 C
2101 2101       IF(
2102 2102       .   HYDFLX( IS , 4 ) .GT. FLUXPP .OR.
2103 2103       .   HYDFLX( IS , 2 ) .GT. FLUXUU .OR.
2104 2104       .   HYDFLX( IS , 1 ) .GT. FLUXRR .OR.
2105 2105       .   KSDELTA( IS ) .GT. NIDUMP .OR.
2106 2106       .   XS( 3 , IS ) .GT. AREVGG ) THEN
2107 2107       INDCR = 3
2108 2108       RETURN
2109 2109       END IF
2110 2110 C
2111 2111       DO 640 IR = 1 , 3
2112 2112       JR = MOD( IR , 3 ) + 1
2113 2113       IEA = IABS( JS( JR + 3 , IS ) )
2114 2114       IF( IEA .EQ. IE ) THEN
2115 2115       IIR = MOD( JR , 3 ) + 4
2116 2116       IEI = JS( IIR , IS )
2117 2117       IEIB = IABS( IEI )
2118 2118       XEIEB = XE( 1 , IEIB )
2119 2119       XYLNGT = XYLNGT + XEIEB
2120 2120       IF( XYLONG .LT. XEIEB ) XYLONG = XEIEB
2121 2121       IF( XYSHRT .GT. XEIEB ) XYSHRT = XEIEB
2122 2122       IIKK = IABS( IICOLR( IJTRIG ) )
2123 2123       IF( IIKK .EQ. IEIB ) THEN
2124 2124       JLOOP = 2
2125 2125       IETRIG = IETRIG + 1
2126 2126       IECRSS( IETRIG ) = IEIB
2127 2127       IJTRIG = IJTRIG - 1
2128 2128       IF( IEI .GT. 0 ) THEN
2129 2129       JKVV = JE( 1 , IEIB )
2130 2130       ELSE
2131 2131       JKVV = JE( 2 , IEIB )
2132 2132       END IF
2133 2133       JVDELTA = JVDELTA + 1
2134 2134       IVDELTA( JVDELTA ) = JKVV
2135 2135       ELSE
2136 2136       IJTRIG = IJTRIG + 1
2137 2137       IICOLR( IJTRIG ) = IEI
2138 2138       END IF
2139 2139       JJR = MOD( JR + 1 , 3 ) + 4
2140 2140       IER = IABS( JS( JJR , IS ) )
2141 2141 C
2142 2142       IV1 = JE( 1 , IER )
2143 2143       IF( IV1 .EQ. IVV ) THEN
2144 2144       ISR = JE( 3 , IER )
2145 2145       ELSE

```

```

2146 2146          ISR = JE( 4 , IER )                2146
2147 2147          END IF                            2147
2148 2148          END IF                            2148
2149 2149          C                                2149
2150 2150          640          CONTINUE                2150
2151 2151          C                                2151
2152 2152          IF( IER . NE . IET ) THEN          2152
2153 2153          IS = ISR                            2153
2154 2154          IE = IER                            2154
2155 2155          GO TO 630                            2155
2156 2156          END IF                            2156
2157 2157          C                                2157
2158 2158          GO TO 620                            2158
2159 2159          END IF                            2159
2160 2160          C                                2160
2161 2161          C          INTERMEDIATE LOOP IS DONE, SECOND LOOP OVER KKV2 START . 2161
2162 2162          C                                2162
2163 2163          IVV = KKV2                            2163
2164 2164          IE = IEIN1                            2164
2165 2165          XYYIC = XE( 1 , IE ) + XE( 1 , IKKE1 ) + XE( 1 , IEIB ) 2165
2166 2166          XYLNGT = XYLNGT + XYYIC - XE( 1 , IEIB ) 2166
2167 2167          IF( XYLONG . LT . XYYIC ) XYLONG = XYYIC 2167
2168 2168          IF( XYSHRT . GT . XYYIC ) XYSHRT = XYYIC 2168
2169 2169          IVIN2 = JE( 2 , IE )                  2169
2170 2170          IEJJK = IICOLR( IJTRIG )              2170
2171 2171          IV1 = JE( 1 , IE )                    2171
2172 2172          IF( IV1 . EQ . IVV ) THEN            2172
2173 2173          ISI = JE( 3 , IE )                  2173
2174 2174          ELSE                                  2174
2175 2175          ISI = JE( 4 , IE )                  2175
2176 2176          END IF                            2176
2177 2177          IS = ISI                            2177
2178 2178          C                                2178
2179 2179          650          CONTINUE                2179
2180 2180          C                                2180
2181 2181          ITRIG = ITRIG + 1                    2181
2182 2182          ISCRSS( ITRIG ) = IS                 2182
2183 2183          C                                2183
2184 2184          IETRIG = IETRIG + 1                  2184
2185 2185          IECRSS( IETRIG ) = IE              2185
2186 2186          C                                2186
2187 2187          IF(                                2187
2188 2188          .   HYDFLX( IS , 4 ) . GT . FLUXPP . OR . 2188
2189 2189          .   HYDFLX( IS , 2 ) . GT . FLUXUU . OR . 2189
2190 2190          .   HYDFLX( IS , 1 ) . GT . FLUXRR . OR . 2190
2191 2191          .   KSDELTA( IS ) . GT . NIDUMP . OR . 2191
2192 2192          .   XS( 3 , IS ) . GT . AREVGG ) THEN 2192
2193 2193          INDOCTR = 3                          2193
2194 2194          RETURN                              2194
2195 2195          END IF                            2195
2196 2196          C                                2196
2197 2197          DO 660 IR = 1 , 3                    2197
2198 2198          JR = MOD( IR , 3 ) + 1                2198
2199 2199          IEA = IABS( JS( JR + 3 , IS ) )      2199
2200 2200          IF( IEA . EQ . IE ) THEN            2200
2201 2201          IIR = MOD( JR , 3 ) + 4            2201
2202 2202          IEI = JS( IIR , IS )                2202
2203 2203          IEIB = IABS( IEI )                  2203
2204 2204          XEIEB = XE( 1 , IEIB )              2204
2205 2205          XYLNGT = XYLNGT + XEIEB            2205
2206 2206          IF( XYLONG . LT . XEIEB ) XYLONG = XEIEB 2206
2207 2207          IF( XYSHRT . GT . XEIEB ) XYSHRT = XEIEB 2207
2208 2208          IJTRIG = IJTRIG + 1                2208
2209 2209          IICOLR( IJTRIG ) = IEI              2209
2210 2210          JJR = MOD( JR + 1 , 3 ) + 4        2210
2211 2211          IER = IABS( JS( JJR , IS ) )        2211
2212 2212          C                                2212
2213 2213          IV1 = JE( 1 , IER )                  2213
2214 2214          IF( IV1 . EQ . IVV ) THEN            2214
2215 2215          ISR = JE( 3 , IER )                  2215
2216 2216          ELSE                                  2216
2217 2217          ISR = JE( 4 , IER )                  2217
2218 2218          END IF                            2218
2219 2219          END IF                            2219

```

```

2220 2220 C
2221 2221 660 CONTINUE 2220
2222 2222 C 2221
2223 2223 IF( IER . NE . IKKE2 ) THEN 2222
2224 2224 IS = ISR 2223
2225 2225 IE = IER 2224
2226 2226 GO TO 650 2225
2227 2227 END IF 2226
2228 2228 C 2227
2229 2229 IJTRIG = IJTRIG - 1 2228
2230 2230 IETRIG = IETRIG + 1 2229
2231 2231 IECRSS( IETRIG ) = IKKE2 2230
2232 2232 IETRIG = IETRIG + 1 2231
2233 2233 IECRSS( IETRIG ) = IKKE1 2232
2234 2234 C 2233
2235 2235 SECOND LOOP SURROUNDING KKV2 IS DONE, THIRD LOOP OVER KKV3 START . 2234
2236 2236 C 2235
2237 2237 IVV = KKV3 2236
2238 2238 IE = IABS( IICOLR( IJTRIG + 1 ) ) 2237
2239 2239 IF( JE( 5 , IE ) . NE . 0 ) THEN 2238
2240 2240 IER = IE 2239
2241 2241 GO TO 670 2240
2242 2242 END IF 2241
2243 2243 IV1 = JE( 1 , IE ) 2242
2244 2244 IF( IV1 . EQ . IVV ) THEN 2243
2245 2245 ISI = JE( 3 , IE ) 2244
2246 2246 ELSE 2245
2247 2247 ISI = JE( 4 , IE ) 2246
2248 2248 END IF 2247
2249 2249 IS = ISI 2248
2250 2250 ISI = 0 2249
2251 2251 C 2250
2252 2252 680 CONTINUE 2251
2253 2253 C 2252
2254 2254 ITRIG = ITRIG + 1 2253
2255 2255 ISCRSS( ITRIG ) = IS 2254
2256 2256 IETRIG = IETRIG + 1 2255
2257 2257 IECRSS( IETRIG ) = IE 2256
2258 2258 C 2257
2259 2259 IF( 2258
2260 2260 . HYDFLX( IS , 4 ) . GT . FLUXPP . OR . 2259
2261 2261 . HYDFLX( IS , 2 ) . GT . FLUXJU . OR . 2260
2262 2262 . HYDFLX( IS , 1 ) . GT . FLUXRR . OR . 2261
2263 2263 . KSDEL( IS ) . GT . NIDUMP . OR . 2262
2264 2264 . XS( 3 , IS ) . GT . AREVGG ) THEN 2263
2265 2265 INDCTR = 3 2264
2266 2266 RETURN 2265
2267 2267 END IF 2266
2268 2268 C 2267
2269 2269 DO 690 IR = 1 , 3 2268
2270 2270 JR = MOD( IR , 3 ) + 1 2269
2271 2271 IEA = IABS( JS( JR + 3 , IS ) ) 2270
2272 2272 IF( IEA . EQ . IE ) THEN 2271
2273 2273 IIR = MOD( JR , 3 ) + 4 2272
2274 2274 IEI = JS( IIR , IS ) 2273
2275 2275 IEIB = IABS( IEI ) 2274
2276 2276 XEIEB = XE( 1 , IEIB ) 2275
2277 2277 XYLNGT = XYLNGT + XEIEB 2276
2278 2278 IF( XYLONG . LT . XEIEB ) XYLONG = XEIEB 2277
2279 2279 IF( XYSHRT . GT . XEIEB ) XYSHRT = XEIEB 2278
2280 2280 IJTRIG = IJTRIG + 1 2279
2281 2281 IICOLR( IJTRIG ) = IEI 2280
2282 2282 JJR = MOD( JR + 1 , 3 ) + 4 2281
2283 2283 IER = IABS( JS( JJR , IS ) ) 2282
2284 2284 C 2283
2285 2285 IV1 = JE( 1 , IER ) 2284
2286 2286 IF( IV1 . EQ . IVV ) THEN 2285
2287 2287 ISR = JE( 3 , IER ) 2286
2288 2288 ELSE 2287
2289 2289 ISR = JE( 4 , IER ) 2288
2290 2290 END IF 2289
2291 2291 END IF 2290
2292 2292 C 2291
2293 2293 690 CONTINUE 2292

```

```

2294 2294 C
2295 2295 IF( ISR . NE . ISI ) THEN
2296 2296 IS = ISR
2297 2297 IE = IER
2298 2298 GO TO 680
2299 2299 END IF
2300 2300 C
2301 2301 670 CONTINUE
2302 2302 C
2303 2303 IETRIG = IETRIG + 1
2304 2304 IECRSS( IETRIG ) = IER
2305 2305 C
2306 2306 ITYPE = JE( 5 , IER )
2307 2307 C
2308 2308 XEIEB = XE( 1 , IER )
2309 2309 XEIEB = XYYIB + XEIEB
2310 2310 XYLNGT = XYLNGT + XEIEB
2311 2311 IF( XYLONG . LT . XEIEB ) XYLONG = XEIEB
2312 2312 IF( XYSHRT . GT . XEIEB ) XYSHRT = XEIEB
2313 2313 C
2314 2314 INOCTR = 2
2315 2315 C IF( XYLONG / XYSHRT . GT . 10. . AND . JLOOP . EQ . 0 ) RETURN
2316 2316 C
2317 2317 JE( 2 , IEJKK ) = IVIN2
2318 2318 C
2319 2319 IV1 = IVIN1
2320 2320 IE1 = IICOLR( IJTRIG )
2321 2321 IF( IE1 . GT . 0 ) THEN
2322 2322 IV2 = JE( 2 , IE1 )
2323 2323 ELSE
2324 2324 IV2 = JE( 1 , - IE1 )
2325 2325 END IF
2326 2326 C
2327 2327 NEC = IECRSS( IETRIG )
2328 2328 IETRIG = IETRIG - 1
2329 2329 C
2330 2330 JV( 2 , IV2 ) = - NEC
2331 2331 JE( 1 , NEC ) = IV2
2332 2332 JE( 2 , NEC ) = IV1
2333 2333 JE( 4 , NEC ) = 0
2334 2334 JE( 5 , NEC ) = ITYPE
2335 2335 C
2336 2336 IJTRIG = IJTRIG + 1
2337 2337 IICOLR( IJTRIG ) = NEC
2338 2338 C
2339 2339 END IF
2340 2340 C
2341 2341 C LOOP OVER TRIANGLE KSD IS DONE
2342 2342 C
2343 2343 C ELIMINATING THE DELETED TRIANGLES FROM JSDELTA ARRAY
2344 2344 C
2345 2345 LSDELTA = 0
2346 2346 DO 1520 IS = 1 , JSDELTA
2347 2347 JSP = JSDELTA( IS )
2348 2348 ILOOP = 0
2349 2349 IF( JSP . EQ . 0 ) THEN
2350 2350 ILOOP = 1
2351 2351 ELSE
2352 2352 DO 1525 IKS = 1 , ITRIG
2353 2353 ISP = ISCRSS( IKS )
2354 2354 IF( JSP . EQ . ISP ) ILOOP = 1
2355 2355 1525 CONTINUE
2356 2356 END IF
2357 2357 IF( ILOOP . EQ . 0 ) THEN
2358 2358 LSDELTA = LSDELTA + 1
2359 2359 JSDELTA( LSDELTA ) = JSP
2360 2360 END IF
2361 2361 1520 CONTINUE
2362 2362 ISDELTA = LSDELTA
2363 2363 C
2364 2364 JVDELTA = JVDELTA + 1
2365 2365 IVDELTA( JVDELTA ) = KV1
2366 2366 JVDELTA = JVDELTA + 1
2367 2367 IVDELTA( JVDELTA ) = KV2

```

```

2368 2368          JVDEL = JVDEL + 1
2369 2369          IVDEL( JVDEL ) = KV3
2370 2370          C
2371 2371          DO 700 IE = 1 , IJTRIG
2372 2372             IEM = IABS( IICOLR( IE ) )
2373 2373             JUE( IE ) = IEM
2374 2374             IV1 = JE( 1 , IEM )
2375 2375             IV2 = JE( 2 , IEM )
2376 2376             IEE1 = JV( 2 , IV1 )
2377 2377             IEE2 = JV( 2 , IV2 )
2378 2378             IF( IEE1 .GT. 0 ) JV( 2 , IV1 ) = IEM
2379 2379             IF( IEE2 .GT. 0 ) JV( 2 , IV2 ) = IEM
2380 2380          700 CONTINUE
2381 2381          C
2382 2382          JTRIG = IJTRIG
2383 2383          ISTOP = 0
2384 2384          NSINTL = 0
2385 2385          C
2386 2386          JJTRIG = IJTRIG
2387 2387          DO 710 IE = 1 , JTRIG
2388 2388             IEM = IICOLR( IE )
2389 2389             IF( IEM .GT. 0 ) THEN
2390 2390                 JUV( IE ) = JE( 1 , IEM )
2391 2391             ELSE
2392 2392                 JUV( IE ) = JE( 2 , - IEM )
2393 2393             END IF
2394 2394             IITRIG( IE ) = JUV( IE )
2395 2395          710 CONTINUE
2396 2396          C
2397 2397          720 CONTINUE
2398 2398          C
2399 2399          JTRIGP = JTRIG + 1
2400 2400          DO 730 IE = 1 , JTRIG
2401 2401             IEM = IICOLR( IE )
2402 2402             IF( IEM .GT. 0 ) THEN
2403 2403                 JUV( IE ) = JE( 1 , IEM )
2404 2404             ELSE
2405 2405                 JUV( IE ) = JE( 2 , - IEM )
2406 2406             END IF
2407 2407          730 CONTINUE
2408 2408          C
2409 2409          AREMIN = 1000000.
2410 2410          IEMIN = 1
2411 2411          DO 740 IE = 1 , JTRIG
2412 2412          C
2413 2413             IEM = MOD( IE - 1 , JTRIG ) + 1
2414 2414             IEP = MOD( IE , JTRIG ) + 1
2415 2415             IEI = MOD( IE + 1 , JTRIG ) + 1
2416 2416          C
2417 2417             IV1 = JUV( IEM )
2418 2418             IV2 = JUV( IEP )
2419 2419             IV3 = JUV( IEI )
2420 2420          C
2421 2421             X1 = XV( 1 , IV1 ) - XV( 1 , IV2 )
2422 2422             Y1 = XV( 2 , IV1 ) - XV( 2 , IV2 )
2423 2423             X2 = XV( 1 , IV3 ) - XV( 1 , IV2 )
2424 2424             Y2 = XV( 2 , IV3 ) - XV( 2 , IV2 )
2425 2425             XSIN = ( X2 * Y1 - X1 * Y2 )
2426 2426             XCOS = ( X1 * X2 + Y1 * Y2 )
2427 2427             XCOT = XCOS / ( XSIN + 1.E-8 )
2428 2428             IF( XSIN .LT. 0. .AND. XCOT .LT. AREMIN ) THEN
2429 2429                 AREMIN = XCOT
2430 2430             IEMIN = IE
2431 2431             END IF
2432 2432             ANGLE( IE ) = XSIN / ( ABS( XCOS ) + 1.E-7 )
2433 2433          C
2434 2434          740 CONTINUE
2435 2435          C
2436 2436          DO 750 IE = 1 , JTRIG
2437 2437             IEP = MOD( IE - IEMIN + JTRIG , JTRIG ) + 1
2438 2438             JEN( IEP ) = IICOLR( IE )
2439 2439             ANGLER( IEP ) = ANGLE( IE )
2440 2440          750 CONTINUE
2441 2441          C

```

```

2442 2442 DO 760 IE = 1 , JTRIG 2442
2443 2443 IICOLR( IE ) = JEN( IE ) 2443
2444 2444 ANGLE( IE ) = ANGLER( IE ) 2444
2445 2445 760 CONTINUE 2445
2446 2446 C 2446
2447 2447 IFINAL = 0 2447
2448 2448 IEI = 1 2448
2449 2449 DO 770 IE = 1 , JTRIG 2449
2450 2450 SANGLE = ANGLE( IE ) 2450
2451 2451 IANGLE( IE ) = - 1 2451
2452 2452 IF( SANGLE . GT . 1.E-2 ) IANGLE( IE ) = 1 2452
2453 2453 770 CONTINUE 2453
2454 2454 C 2454
2455 2455 DO 780 IE = 1 , JTRIG 2455
2456 2456 IEM = MOD( IE - 1 , JTRIG ) + 1 2456
2457 2457 IEP = MOD( IE , JTRIG ) + 1 2457
2458 2458 IKM = MOD( IE + 1 , JTRIG ) + 1 2458
2459 2459 KEM = IANGLE( IEM ) 2459
2460 2460 KEP = IANGLE( IEP ) 2460
2461 2461 KKM = IANGLE( IKM ) 2461
2462 2462 IF( KEM . EQ . - 1 . AND . 2462
2463 2463 KEP . EQ . 1 . AND . 2463
2464 2464 KKM . EQ . - 1 . AND . IFINAL . EQ . 0 ) THEN 2464
2465 2465 IEI = IKM 2465
2466 2466 IFINAL = 1 2466
2467 2467 END IF 2467
2468 2468 780 CONTINUE 2468
2469 2469 C 2469
2470 2470 IF( IFINAL . EQ . 0 ) THEN 2470
2471 2471 DO 790 IE = 1 , JTRIG 2471
2472 2472 IEM = MOD( IE - 1 , JTRIG ) + 1 2472
2473 2473 IEP = MOD( IE , JTRIG ) + 1 2473
2474 2474 KEM = IANGLE( IEM ) 2474
2475 2475 KEP = IANGLE( IEP ) 2475
2476 2476 IF( KEM . EQ . - 1 . AND . KEP . EQ . 1 . AND . 2476
2477 2477 IFINAL . EQ . 0 ) THEN 2477
2478 2478 IEI = MOD( IE + 1 , JTRIG ) + 1 2478
2479 2479 IFINAL = 1 2479
2480 2480 END IF 2480
2481 2481 790 CONTINUE 2481
2482 2482 END IF 2482
2483 2483 C 2483
2484 2484 IF( IFINAL . EQ . 0 ) THEN 2484
2485 2485 ANGMIN = 1000000. 2485
2486 2486 DO 800 IE = 1 , JTRIG 2486
2487 2487 XANGLE = ANGLE( IE ) 2487
2488 2488 SANGLE = ABS( XANGLE - 1. ) 2488
2489 2489 IF( XANGLE . GT . 0. . AND . SANGLE . LT . ANGMIN ) THEN 2489
2490 2490 IEI = MOD( IE , JTRIG ) + 1 2490
2491 2491 ANGMIN = SANGLE 2491
2492 2492 END IF 2492
2493 2493 800 CONTINUE 2493
2494 2494 END IF 2494
2495 2495 C 2495
2496 2496 DO 810 IE = 1 , JTRIG 2496
2497 2497 IEP = MOD( IE - IEI + JTRIGP , JTRIG ) + 1 2497
2498 2498 JEN( IEP ) = IICOLR( IE ) 2498
2499 2499 ANGLER( IEP ) = ANGLE( IE ) 2499
2500 2500 810 CONTINUE 2500
2501 2501 C 2501
2502 2502 DO 820 IE = 1 , JTRIG 2502
2503 2503 ANGLE( IE ) = ANGLER( IE ) 2503
2504 2504 IICOLR( IE ) = JEN( IE ) 2504
2505 2505 820 CONTINUE 2505
2506 2506 C 2506
2507 2507 DO 830 IE = 1 , JTRIG 2507
2508 2508 IEM = JEN( IE ) 2508
2509 2509 IF( IEM . GT . 0 ) THEN 2509
2510 2510 JUV( IE ) = JE( 1 , IEM ) 2510
2511 2511 ELSE 2511
2512 2512 JUV( IE ) = JE( 2 , - IEM ) 2512
2513 2513 END IF 2513
2514 2514 830 CONTINUE 2514
2515 2515 C 2515

```

```

2516 2516 IF( JTRIG . EQ . 3 ) THEN
2517 2517 C
2518 2518 NSC = ISCRSS( ITRIG )
2519 2519 ITRIG = ITRIG - 1
2520 2520 NSINTL = NSINTL + 1
2521 2521 INVTRG( NSINTL ) = NSC
2522 2522 JS( 1 , NSC ) = JUV( 1 )
2523 2523 JS( 2 , NSC ) = JUV( 2 )
2524 2524 JS( 3 , NSC ) = JUV( 3 )
2525 2525 JS( 4 , NSC ) = JEN( 1 )
2526 2526 JS( 5 , NSC ) = JEN( 2 )
2527 2527 JS( 6 , NSC ) = JEN( 3 )
2528 2528 C
2529 2529 IV1 = JS( 1 , NSC )
2530 2530 IV2 = JS( 2 , NSC )
2531 2531 IV3 = JS( 3 , NSC )
2532 2532 AX = XV( 1 , IV2 ) - XV( 1 , IV1 )
2533 2533 AY = XV( 2 , IV2 ) - XV( 2 , IV1 )
2534 2534 BX = XV( 1 , IV3 ) - XV( 1 , IV1 )
2535 2535 BY = XV( 2 , IV3 ) - XV( 2 , IV1 )
2536 2536 XS( 3 , NSC ) = 0.5 * ( AX * BY - AY * BX )
2537 2537 C
2538 2538 SAREA( NSC ) = 1. / XS( 3 , NSC )
2539 2539 XXC = ( XV( 1 , IV1 ) + XV( 1 , IV2 ) + XV( 1 , IV3 ) ) *
2540 2540 THIRD
2541 2541 YYC = ( XV( 2 , IV1 ) + XV( 2 , IV2 ) + XV( 2 , IV3 ) ) *
2542 2542 THIRD
2543 2543 XS( 1 , NSC ) = XXC
2544 2544 XS( 2 , NSC ) = YYC
2545 2545 HYDFLX( NSC , 4 ) = 0.
2546 2546 HYDFLX( NSC , 1 ) = 0.
2547 2547 HYDFLX( NSC , 2 ) = 0.
2548 2548 KSDEL( NSC ) = 1
2549 2549 C
2550 2550 DO 840 IR = 1 , MHQ
2551 2551 HYDV( NSC , IR ) = ( HYDVVV( IV1 , IR ) +
2552 2552 HYDVVV( IV2 , IR ) +
2553 2553 HYDVVV( IV3 , IR ) ) * THIRD
2554 2554 840 CONTINUE
2555 2555 C
2556 2556 HDUM = 1. / ( HYDV( NSC , 1 ) + 1.E-12 )
2557 2557 HYDV( NSC , 2 ) = HYDV( NSC , 2 ) * HDUM
2558 2558 HYDV( NSC , 3 ) = HYDV( NSC , 3 ) * HDUM
2559 2559 HYDV( NSC , 4 ) = ( HYDV( NSC , 4 ) -
2560 2560 .5 * HYDV( NSC , 1 ) *
2561 2561 ( HYDV( NSC , 2 ) * HYDV( NSC , 2 ) +
2562 2562 HYDV( NSC , 3 ) * HYDV( NSC , 3 ) ) ) *
2563 2563 ( HYDV( NSC , 5 ) - 1. )
2564 2564 C
2565 2565 ISTOP = 1
2566 2566 C
2567 2567 ELSE IF( JTRIG . EQ . 4 ) THEN
2568 2568 C
2569 2569 NSC = ISCRSS( ITRIG )
2570 2570 ITRIG = ITRIG - 1
2571 2571 NSINTL = NSINTL + 1
2572 2572 INVTRG( NSINTL ) = NSC
2573 2573 NEC = IECRSS( IETRIG )
2574 2574 IETRIG = IETRIG - 1
2575 2575 C
2576 2576 IJTRIG = IJTRIG + 1
2577 2577 JUE( IJTRIG ) = NEC
2578 2578 C
2579 2579 JE( 1 , NEC ) = JUV( 1 )
2580 2580 JE( 2 , NEC ) = JUV( 3 )
2581 2581 JE( 5 , NEC ) = 0
2582 2582 C
2583 2583 JS( 1 , NSC ) = JUV( 1 )
2584 2584 JS( 2 , NSC ) = JUV( 2 )
2585 2585 JS( 3 , NSC ) = JUV( 3 )
2586 2586 JS( 4 , NSC ) = JEN( 1 )
2587 2587 JS( 5 , NSC ) = JEN( 2 )
2588 2588 JS( 6 , NSC ) = - NEC
2589 2589 C

```

```

2590 2590 NSC = ISCRSS( ITRIG ) 2590
2591 2591 ITRIG = ITRIG - 1 2591
2592 2592 NSINTL = NSINTL + 1 2592
2593 2593 INVTRG( NSINTL ) = NSC 2593
2594 2594 C 2594
2595 2595 JS( 1 , NSC ) = JUV( 1 ) 2595
2596 2596 JS( 2 , NSC ) = JUV( 3 ) 2596
2597 2597 JS( 3 , NSC ) = JUV( 4 ) 2597
2598 2598 JS( 4 , NSC ) = NEC 2598
2599 2599 JS( 5 , NSC ) = JEN( 3 ) 2599
2600 2600 JS( 6 , NSC ) = JEN( 4 ) 2600
2601 2601 C 2601
2602 2602 DO 850 IKR = 1 , 2 2602
2603 2603 NSS = INVTRG( NSINTL + 1 - IKR ) 2603
2604 2604 IV1 = JS( 1 , NSS ) 2604
2605 2605 IV2 = JS( 2 , NSS ) 2605
2606 2606 IV3 = JS( 3 , NSS ) 2606
2607 2607 AX = XV( 1 , IV2 ) - XV( 1 , IV1 ) 2607
2608 2608 AY = XV( 2 , IV2 ) - XV( 2 , IV1 ) 2608
2609 2609 BX = XV( 1 , IV3 ) - XV( 1 , IV1 ) 2609
2610 2610 BY = XV( 2 , IV3 ) - XV( 2 , IV1 ) 2610
2611 2611 XS( 3 , NSS ) = 0.5 * ( AX * BY - AY * BX ) 2611
2612 2612 C 2612
2613 2613 SAREA( NSS ) = 1. / XS( 3 , NSS ) 2613
2614 2614 XXC = ( XV( 1 , IV1 ) + XV( 1 , IV2 ) + XV( 1 , IV3 ) ) * 2614
2615 2615 THIRD 2615
2616 2616 YYC = ( XV( 2 , IV1 ) + XV( 2 , IV2 ) + XV( 2 , IV3 ) ) * 2616
2617 2617 THIRD 2617
2618 2618 XS( 1 , NSS ) = XXC 2618
2619 2619 XS( 2 , NSS ) = YYC 2619
2620 2620 HYDFLX( NSS , 4 ) = 0. 2620
2621 2621 HYDFLX( NSS , 1 ) = 0. 2621
2622 2622 HYDFLX( NSS , 2 ) = 0. 2622
2623 2623 KSDELTA( NSS ) = 1 2623
2624 2624 C 2624
2625 2625 DO 860 IR = 1 , MHQ 2625
2626 2626 HYDV( NSS , IR ) = ( HYDVVV( IV1 , IR ) + 2626
2627 2627 HYDVVV( IV2 , IR ) + 2627
2628 2628 HYDVVV( IV3 , IR ) ) * THIRD 2628
2629 2629 860 CONTINUE 2629
2630 2630 C 2630
2631 2631 HDUM = 1. / ( HYDV( NSS , 1 ) + 1.E-12 ) 2631
2632 2632 HYDV( NSS , 2 ) = HYDV( NSS , 2 ) * HDUM 2632
2633 2633 HYDV( NSS , 3 ) = HYDV( NSS , 3 ) * HDUM 2633
2634 2634 HYDV( NSS , 4 ) = ( HYDV( NSS , 4 ) - 2634
2635 2635 .5 * HYDV( NSS , 1 ) ) * 2635
2636 2636 ( HYDV( NSS , 2 ) * HYDV( NSS , 2 ) + 2636
2637 2637 HYDV( NSS , 3 ) * HYDV( NSS , 3 ) ) * 2637
2638 2638 ( HYDV( NSS , 5 ) - 1. ) 2638
2639 2639 C 2639
2640 2640 850 CONTINUE 2640
2641 2641 ISTOP = 1 2641
2642 2642 C 2642
2643 2643 ELSE 2643
2644 2644 C 2644
2645 2645 NSC = ISCRSS( ITRIG ) 2645
2646 2646 ITRIG = ITRIG - 1 2646
2647 2647 NSINTL = NSINTL + 1 2647
2648 2648 INVTRG( NSINTL ) = NSC 2648
2649 2649 NEC = IECRSS( IETRIG ) 2649
2650 2650 IETRIG = IETRIG - 1 2650
2651 2651 C 2651
2652 2652 IJTRIG = IJTRIG + 1 2652
2653 2653 JUE( IJTRIG ) = NEC 2653
2654 2654 C 2654
2655 2655 JE( 1 , NEC ) = JUV( 1 ) 2655
2656 2656 JE( 2 , NEC ) = JUV( 3 ) 2656
2657 2657 JE( 5 , NEC ) = 0 2657
2658 2658 C 2658
2659 2659 JS( 1 , NSC ) = JUV( 1 ) 2659
2660 2660 JS( 2 , NSC ) = JUV( 2 ) 2660
2661 2661 JS( 3 , NSC ) = JUV( 3 ) 2661
2662 2662 JS( 4 , NSC ) = JEN( 1 ) 2662
2663 2663 JS( 5 , NSC ) = JEN( 2 ) 2663

```



```

2664 2664 JS( 6 , NSC ) = - NEC 2664
2665 2665 C 2665
2666 2666 IICOLR( 1 ) = NEC 2666
2667 2667 JTRIG = JTRIG - 1 2667
2668 2668 DO 870 IEE = 2 , JTRIG 2668
2669 2669 IICOLR( IEE ) = JEN( IEE + 1 ) 2669
2670 2670 870 CONTINUE 2670
2671 2671 C 2671
2672 2672 IV1 = JS( 1 , NSC ) 2672
2673 2673 IV2 = JS( 2 , NSC ) 2673
2674 2674 IV3 = JS( 3 , NSC ) 2674
2675 2675 AX = XV( 1 , IV2 ) - XV( 1 , IV1 ) 2675
2676 2676 AY = XV( 2 , IV2 ) - XV( 2 , IV1 ) 2676
2677 2677 BX = XV( 1 , IV3 ) - XV( 1 , IV1 ) 2677
2678 2678 BY = XV( 2 , IV3 ) - XV( 2 , IV1 ) 2678
2679 2679 XS( 3 , NSC ) = 0.5 * ( AX * BY - AY * BX ) 2679
2680 2680 C 2680
2681 2681 SAREA( NSC ) = 1. / XS( 3 , NSC ) 2681
2682 2682 . XXC = ( XV( 1 , IV1 ) + XV( 1 , IV2 ) + XV( 1 , IV3 ) ) * 2682
2683 2683 . THIRD 2683
2684 2684 . YYC = ( XV( 2 , IV1 ) + XV( 2 , IV2 ) + XV( 2 , IV3 ) ) * 2684
2685 2685 . THIRD 2685
2686 2686 XS( 1 , NSC ) = XXC 2686
2687 2687 XS( 2 , NSC ) = YYC 2687
2688 2688 HYDFLX( NSC , 4 ) = 0. 2688
2689 2689 HYDFLX( NSC , 1 ) = 0. 2689
2690 2690 HYDFLX( NSC , 2 ) = 0. 2690
2691 2691 KSDEL( NSC ) = 1 2691
2692 2692 C 2692
2693 2693 DO 880 IR = 1 , MHQ 2693
2694 2694 HYDV( NSC , IR ) = ( HYDVVV( IV1 , IR ) + 2694
2695 2695 . HYDVVV( IV2 , IR ) + 2695
2696 2696 . HYDVVV( IV3 , IR ) ) * THIRD 2696
2697 2697 880 CONTINUE 2697
2698 2698 C 2698
2699 2699 HDUM = 1. / ( HYDV( NSC , 1 ) + 1.E-12 ) 2699
2700 2700 HYDV( NSC , 2 ) = HYDV( NSC , 2 ) * HDUM 2700
2701 2701 HYDV( NSC , 3 ) = HYDV( NSC , 3 ) * HDUM 2701
2702 2702 HYDV( NSC , 4 ) = ( HYDV( NSC , 4 ) - 2702
2703 2703 . .5 * HYDV( NSC , 1 ) ) * 2703
2704 2704 . ( HYDV( NSC , 2 ) * HYDV( NSC , 2 ) + 2704
2705 2705 . HYDV( NSC , 3 ) * HYDV( NSC , 3 ) ) * 2705
2706 2706 . ( HYDV( NSC , 5 ) - 1. ) 2706
2707 2707 C 2707
2708 2708 END IF 2708
2709 2709 IF( ISTOP .EQ. 0 ) GO TO 720 2709
2710 2710 C 2710
2711 2711 DO 890 ISS = 1 , NSINTL 2711
2712 2712 IS = INVTRG( ISS ) 2712
2713 2713 DO 890 IR = 4 , 6 2713
2714 2714 IE = JS( IR , IS ) 2714
2715 2715 IF( IE .GT. 0 ) THEN 2715
2716 2716 JE( 3 , IE ) = IS 2716
2717 2717 ELSE 2717
2718 2718 JE( 4 , - IE ) = IS 2718
2719 2719 END IF 2719
2720 2720 890 CONTINUE 2720
2721 2721 C 2721
2722 2722 DO 900 IENN = 1 , IJTRIG 2722
2723 2723 IEN = JUE( IENN ) 2723
2724 2724 JV1 = JE( 1 , IEN ) 2724
2725 2725 JV2 = JE( 2 , IEN ) 2725
2726 2726 AX = XV( 1 , JV2 ) - XV( 1 , JV1 ) 2726
2727 2727 AY = XV( 2 , JV2 ) - XV( 2 , JV1 ) 2727
2728 2728 XE( 1 , IEN ) = SQRT( AX * AX + AY * AY ) 2728
2729 2729 XEREV = 1. / XE( 1 , IEN ) 2729
2730 2730 XN( IEN ) = AY * XEREV 2730
2731 2731 YN( IEN ) = - AX * XEREV 2731
2732 2732 ISSR = JE( 4 , IEN ) 2732
2733 2733 ISSL = JE( 3 , IEN ) 2733
2734 2734 C 2734
2735 2735 IF( JE( 5 , IEN ) .NE. 0 ) THEN 2735
2736 2736 C 2736
2737 2737 AA = XV( 1 , JV2 ) - XV( 1 , JV1 ) 2737

```

```

2738 2738      BB = XV( 2 , JV2 ) - XV( 2 , JV1 )      2738
2739 2739      XEL = XS( 1 , ISSL )                    2739
2740 2740      YEL = XS( 2 , ISSL )                    2740
2741 2741      CC = XEL - XV( 1 , JV1 )                2741
2742 2742      DD = YEL - XV( 2 , JV1 )                2742
2743 2743      EE = ( AA * CC + BB * DD ) * XEREV * XEREV 2743
2744 2744      XER = XV( 1 , JV1 ) + AA * EE           2744
2745 2745      YER = XV( 2 , JV1 ) + BB * EE           2745
2746 2746      AX = XER - XEL                          2746
2747 2747      AY = YER - YEL                          2747
2748 2748      XE( 2 , IEN ) = SQRT( AX * AX + AY * AY ) 2748
2749 2749      XEREV = 1. / XE( 2 , IEN )                2749
2750 2750      XXN( IEN ) = AX * XEREV                   2750
2751 2751      YYN( IEN ) = AY * XEREV                   2751
2752 2752      XE( 2 , IEN ) = 2. * XE( 2 , IEN )        2752
2753 2753      XYMIDL( IEN ) = .5                      2753
2754 2754      XMIDL( IEN ) = XER                       2754
2755 2755      YMIDL( IEN ) = YER                       2755
2756 2756      C                                         2756
2757 2757      ELSE                                     2757
2758 2758      C                                         2758
2759 2759      XER = XS( 1 , ISSR )                      2759
2760 2760      YER = XS( 2 , ISSR )                      2760
2761 2761      XEL = XS( 1 , ISSL )                      2761
2762 2762      YEL = XS( 2 , ISSL )                      2762
2763 2763      C                                         2763
2764 2764      AA = XV( 1 , JV2 ) - XV( 1 , JV1 )        2764
2765 2765      BB = XV( 2 , JV2 ) - XV( 2 , JV1 )        2765
2766 2766      CC = XEL - XER                          2766
2767 2767      DD = YEL - YER                          2767
2768 2768      ACA = XER - XV( 1 , JV1 )                2768
2769 2769      DBD = YER - XV( 2 , JV1 )                2769
2770 2770      EE = ( ACA * DD - DBD * CC ) / ( AA * DD - BB * CC ) 2770
2771 2771      XMIDL( IEN ) = XV( 1 , JV1 ) + AA * EE    2771
2772 2772      YMIDL( IEN ) = XV( 2 , JV1 ) + BB * EE    2772
2773 2773      C                                         2773
2774 2774      XEMID = XMIDL( IEN ) - XEL                 2774
2775 2775      YEMID = YMIDL( IEN ) - YEL                 2775
2776 2776      C                                         2776
2777 2777      AX = XER - XEL                          2777
2778 2778      AY = YER - YEL                          2778
2779 2779      XE( 2 , IEN ) = SQRT( AX * AX + AY * AY ) 2779
2780 2780      XEREV = 1. / XE( 2 , IEN )                2780
2781 2781      XXN( IEN ) = AX * XEREV                   2781
2782 2782      YYN( IEN ) = AY * XEREV                   2782
2783 2783      C                                         2783
2784 2784      XYMIDL( IEN ) = SQRT( XEMID * XEMID + YEMID * YEMID ) * XEREV 2784
2785 2785      C                                         2785
2786 2786      END IF                                   2786
2787 2787      C                                         2787
2788 2788      900      CONTINUE                          2788
2789 2789      C                                         2789
2790 2790      C      ORDER THE DELETED VERTECIS IN A DECEDED ORDER IN AN ARRAY 2790
2791 2791      C      NVDEL                                2791
2792 2792      C                                         2792
2793 2793      KFLIP = JVDEL                                2793
2794 2794      DO 910 KK = 1 , JVDEL                          2794
2795 2795      IFLIP = 1                                     2795
2796 2796      NVDEL( KK ) = IVDEL( 1 )                    2796
2797 2797      DO 920 KI = 1 , KFLIP                        2797
2798 2798      IF( IVDEL( KI ) . GT . NVDEL( KK ) ) THEN    2798
2799 2799      NVDEL( KK ) = IVDEL( KI )                  2799
2800 2800      IFLIP = KI                                  2800
2801 2801      END IF                                     2801
2802 2802      920      CONTINUE                          2802
2803 2803      ISS = 0                                       2803
2804 2804      DO 930 KI = 1 , KFLIP                        2804
2805 2805      IF( KI . NE . IFLIP ) THEN                2805
2806 2806      ISS = ISS + 1                               2806
2807 2807      IVDEL( ISS ) = IVDEL( KI )                2807
2808 2808      END IF                                     2808
2809 2809      930      CONTINUE                          2809
2810 2810      KFLIP = KFLIP - 1                          2810
2811 2811      910      CONTINUE                          2811

```

2812	2812	C		2812
2813	2813	C	ORDER THE DELETED EDGES IN A DECENDED ORDER IN AN ARRAY	2813
2814	2814	C	NECRSS	2814
2815	2815	C		2815
2816	2816		KFLIP = IETRIG	2816
2817	2817		DO 940 KK = 1 , IETRIG	2817
2818	2818		IFLIP = 1	2818
2819	2819		NECRSS(KK) = IECRSS(1)	2819
2820	2820		DO 950 KI = 1 , KFLIP	2820
2821	2821		IF(IECRSS(KI) . GT . NECRSS(KK)) THEN	2821
2822	2822		NECRSS(KK) = IECRSS(KI)	2822
2823	2823		IFLIP = KI	2823
2824	2824		END IF	2824
2825	2825	950	CONTINUE	2825
2826	2826		ISS = 0	2826
2827	2827		DO 960 KI = 1 , KFLIP	2827
2828	2828		IF(KI . NE . IFLIP) THEN	2828
2829	2829		ISS = ISS + 1	2829
2830	2830		IECRSS(ISS) = IECRSS(KI)	2830
2831	2831		END IF	2831
2832	2832	960	CONTINUE	2832
2833	2833		KFLIP = KFLIP - 1	2833
2834	2834	940	CONTINUE	2834
2835	2835	C		2835
2836	2836	C	ORDER THE DELETED CELLS IN A DECENDED ORDER IN AN ARRAY	2836
2837	2837	C	NSCRSS	2837
2838	2838	C		2838
2839	2839		KFLIP = ITRIG	2839
2840	2840		DO 970 KK = 1 , ITRIG	2840
2841	2841		IFLIP = 1	2841
2842	2842		NSCRSS(KK) = ISCRSS(1)	2842
2843	2843		DO 980 KI = 1 , KFLIP	2843
2844	2844		IF(ISCRSS(KI) . GT . NSCRSS(KK)) THEN	2844
2845	2845		NSCRSS(KK) = ISCRSS(KI)	2845
2846	2846		IFLIP = KI	2846
2847	2847		END IF	2847
2848	2848	980	CONTINUE	2848
2849	2849		ISS = 0	2849
2850	2850		DO 990 KI = 1 , KFLIP	2850
2851	2851		IF(KI . NE . IFLIP) THEN	2851
2852	2852		ISS = ISS + 1	2852
2853	2853		ISCRSS(ISS) = ISCRSS(KI)	2853
2854	2854		END IF	2854
2855	2855	990	CONTINUE	2855
2856	2856		KFLIP = KFLIP - 1	2856
2857	2857	970	CONTINUE	2857
2858	2858	C		2858
2859	2859		DO 1000 KI = 1 , JVDEL	2859
2860	2860		IVDEL(KI) = NV + 1 - KI	2860
2861	2861	1000	CONTINUE	2861
2862	2862	C		2862
2863	2863		DO 1010 KI = 1 , IETRIG	2863
2864	2864		IECRSS(KI) = NE + 1 - KI	2864
2865	2865	1010	CONTINUE	2865
2866	2866	C		2866
2867	2867		DO 1020 KI = 1 , ITRIG	2867
2868	2868		ISCRSS(KI) = NS + 1 - KI	2868
2869	2869	1020	CONTINUE	2869
2870	2870	C		2870
2871	2871	C	IT MAKE SURE THAT VERTICES THAT ARE TO BE DELETED ARE NOT	2871
2872	2872	C	REPLACED BY VERTICES THAS ARE TO BE DELETED ALSO	2872
2873	2873	C		2873
2874	2874		DO 1030 KI = 1 , JVDEL	2874
2875	2875		IVM = NVDEL(KI)	2875
2876	2876		DO 1030 KK = 1 , JVDEL	2876
2877	2877		JVM = IVDEL(KK)	2877
2878	2878		IF(IVM . EQ . JVM . AND . KK . NE . KI) THEN	2878
2879	2879		IVDUM = IVDEL(KI)	2879
2880	2880		IVDEL(KI) = IVM	2880
2881	2881		IVDEL(KK) = IVDUM	2881
2882	2882		END IF	2882
2883	2883	1030	CONTINUE	2883
2884	2884	C		2884
2885	2885	C	IT MAKE SURE THAT EDGES THAT ARE TO BE DELETED ARE NOT	2885

```

2886 2886 C REPLACED BY EDGES THAS ARE TO BE DELETED ALSO 2886
2887 2887 C 2887
2888 2888 DO 1040 KI = 1 , IETRIG 2888
2889 2889 IEM = NECRSS( KI ) 2889
2890 2890 DO 1040 KK = 1 , IETRIG 2890
2891 2891 JEM = IECRSS( KK ) 2891
2892 2892 IF( IEM . EQ . JEM . AND . KK . NE . KI ) THEN 2892
2893 2893 IEDUM = IECRSS( KI ) 2893
2894 2894 IECRSS( KI ) = IEM 2894
2895 2895 IECRSS( KK ) = IEDUM 2895
2896 2896 END IF 2896
2897 2897 1040 CONTINUE 2897
2898 2898 C 2898
2899 2899 C IT MAKE SURE THAT CELLS THAT ARE TO BE DELETED ARE NOT 2899
2900 2900 C REPLACED BY CELLS THAS ARE TO BE DELETED ALSO 2900
2901 2901 C 2901
2902 2902 DO 1050 KI = 1 , ITRIG 2902
2903 2903 ISM = NSCRSS( KI ) 2903
2904 2904 DO 1050 KK = 1 , ITRIG 2904
2905 2905 JSM = ISCRSS( KK ) 2905
2906 2906 IF( ISM . EQ . JSM . AND . KK . NE . KI ) THEN 2906
2907 2907 ISDUM = ISCRSS( KI ) 2907
2908 2908 ISCRSS( KI ) = ISM 2908
2909 2909 ISCRSS( KK ) = ISDUM 2909
2910 2910 END IF 2910
2911 2911 1050 CONTINUE 2911
2912 2912 C 2912
2913 2913 C IVDDEL(*) SEQUENCE OF VERTICES TO BE DELETED END OF LIST 2913
2914 2914 C NVDEL(*) SEQUENCE OF VERTICES TO BE REPLACED CURRENT IN LIST 2914
2915 2915 C ISCRSS(*) SEQUENCE OF TRIANGLES TO BE DELETED END OF LIST 2915
2916 2916 C NSCRSS(*) SEQUENCE OF TRIANGLES TO BE REPLACED CURRENT IN LIST 2916
2917 2917 C IECRSS(*) SEQUENCE OF EDGES TO BE DELETED END OF LIST 2917
2918 2918 C NECRSS(*) SEQUENCE OF EDGES TO BE REPLACED CURRENT IN LIST 2918
2919 2919 C 2919
2920 2920 DO 1060 KI = 1 , JVDEL 2920
2921 2921 IVM = NVDEL( KI ) 2921
2922 2922 JVM = IVDDEL( KI ) 2922
2923 2923 C 2923
2924 2924 XV( 1 , IVM ) = XV( 1 , JVM ) 2924
2925 2925 XV( 2 , IVM ) = XV( 2 , JVM ) 2925
2926 2926 JV( 1 , IVM ) = JV( 1 , JVM ) 2926
2927 2927 C 2927
2928 2928 DO 1060 IR = 1 , MHQ 2928
2929 2929 HYDVVV( IVM , IR ) = HYDVVV( JVM , IR ) 2929
2930 2930 1060 CONTINUE 2930
2931 2931 C 2931
2932 2932 NVN = NV - JVDEL 2932
2933 2933 NEM = NE - IETRIG 2933
2934 2934 NSM = NS - ITRIG 2934
2935 2935 C 2935
2936 2936 C UPDATE THE EDGES AND CELLS THAT ARE CONNECTED TO THE DELETED 2936
2937 2937 C VERTICES 2937
2938 2938 C 2938
2939 2939 JNVEDG = 0 2939
2940 2940 JNVTRG = 0 2940
2941 2941 DO 1070 JVOL = 1 , JVDEL 2941
2942 2942 IVDL = NVDEL( JVOL ) 2942
2943 2943 NVDL = IVDDEL( JVOL ) 2943
2944 2944 IF( IVDL . NE . NVDL ) THEN 2944
2945 2945 IE = JV( 2 , NVDL ) 2945
2946 2946 IF( IE . GT . 0 ) THEN 2946
2947 2947 C 2947
2948 2948 IV1 = JE( 1 , IE ) 2948
2949 2949 IF( IV1 . EQ . NVDL ) THEN 2949
2950 2950 ISI = JE( 3 , IE ) 2950
2951 2951 ELSE 2951
2952 2952 ISI = JE( 4 , IE ) 2952
2953 2953 END IF 2953
2954 2954 IS = ISI 2954
2955 2955 C 2955
2956 2956 JNVEDG = JNVEDG + 1 2956
2957 2957 INVDEL( JNVEDG ) = IE 2957
2958 2958 JNVTRG = JNVTRG + 1 2958
2959 2959 INVTRG( JNVTRG ) = IS 2959

```

2960	2960	C		2960
2961	2961	1090	CONTINUE	2961
2962	2962	C		2962
2963	2963		DO 1080 IR = 1 , 3	2963
2964	2964		JR = MOD(IR , 3) + 1	2964
2965	2965		IEA = IABS(JS(JR + 3 , IS))	2965
2966	2966		IF(IEA . EQ . IE) THEN	2966
2967	2967		JJR = MOD(JR + 1 , 3) + 4	2967
2968	2968		IER = IABS(JS(JJR , IS))	2968
2969	2969	C		2969
2970	2970		IV1 = JE(1 , IER)	2970
2971	2971		IF(IV1 . EQ . NVOL) THEN	2971
2972	2972		ISR = JE(3 , IER)	2972
2973	2973		ELSE	2973
2974	2974		ISR = JE(4 , IER)	2974
2975	2975		END IF	2975
2976	2976		END IF	2976
2977	2977	C		2977
2978	2978	1080	CONTINUE	2978
2979	2979	C		2979
2980	2980		IF(ISR . NE . ISI) THEN	2980
2981	2981		IS = ISR	2981
2982	2982		IE = IER	2982
2983	2983	C		2983
2984	2984		JNVEDG = JNVEDG + 1	2984
2985	2985		INVEDG(JNVEDG) = IE	2985
2986	2986		JNVTRG = JNVTRG + 1	2986
2987	2987		INVTRG(JNVTRG) = IS	2987
2988	2988	C		2988
2989	2989		GO TO 1090	2989
2990	2990		END IF	2990
2991	2991	C		2991
2992	2992		ELSE	2992
2993	2993	C		2993
2994	2994		IE = - IE	2994
2995	2995		IV1 = JE(1 , IE)	2995
2996	2996		IF(IV1 . EQ . NVOL) THEN	2996
2997	2997		ISI = JE(3 , IE)	2997
2998	2998		ELSE	2998
2999	2999		ISI = JE(4 , IE)	2999
3000	3000		END IF	3000
3001	3001		IS = ISI	3001
3002	3002		ISI = 0	3002
3003	3003	C		3003
3004	3004		JNVEDG = JNVEDG + 1	3004
3005	3005		INVEDG(JNVEDG) = IE	3005
3006	3006		JNVTRG = JNVTRG + 1	3006
3007	3007		INVTRG(JNVTRG) = IS	3007
3008	3008	C		3008
3009	3009	1100	CONTINUE	3009
3010	3010	C		3010
3011	3011		DO 1110 IR = 1 , 3	3011
3012	3012		JR = MOD(IR , 3) + 1	3012
3013	3013		IEA = IABS(JS(JR + 3 , IS))	3013
3014	3014		IF(IEA . EQ . IE) THEN	3014
3015	3015		JJR = MOD(JR + 1 , 3) + 4	3015
3016	3016		IER = IABS(JS(JJR , IS))	3016
3017	3017	C		3017
3018	3018		IV1 = JE(1 , IER)	3018
3019	3019		IF(IV1 . EQ . NVOL) THEN	3019
3020	3020		ISR = JE(3 , IER)	3020
3021	3021		ELSE	3021
3022	3022		ISR = JE(4 , IER)	3022
3023	3023		END IF	3023
3024	3024		END IF	3024
3025	3025	C		3025
3026	3026	1110	CONTINUE	3026
3027	3027	C		3027
3028	3028		IF(ISR . NE . ISI) THEN	3028
3029	3029		IS = ISR	3029
3030	3030		IE = IER	3030
3031	3031	C		3031
3032	3032		JNVEDG = JNVEDG + 1	3032
3033	3033		INVEDG(JNVEDG) = IE	3033

```

3034 3034          JNVTRG = JNVTRG + 1          3034
3035 3035          INVTRG( JNVTRG ) = IS        3035
3036 3036          C                            3036
3037 3037          GO TO 1100                    3037
3038 3038          END IF                        3038
3039 3039          C                            3039
3040 3040          JNVEDG = JNVEDG + 1          3040
3041 3041          INVEDG( JNVEDG ) = IER      3041
3042 3042          C                            3042
3043 3043          END IF                        3043
3044 3044          END IF                        3044
3045 3045          1070 CONTINUE                3045
3046 3046          C                            3046
3047 3047          NSMPT = INVTRG( 1 )          3047
3048 3048          C                            3048
3049 3049          DO 1120 IE = 1 , JNVEDG        3049
3050 3050          IEE = INVEDG( IE )           3050
3051 3051          DO 1120 IIDG = IE + 1 , JNVEDG 3051
3052 3052          IF( INVEDG( IIDG ) . EQ . IEE ) THEN 3052
3053 3053          INVEDG( IIDG ) = 0           3053
3054 3054          END IF                        3054
3055 3055          1120 CONTINUE                3055
3056 3056          C                            3056
3057 3057          IEDUM = 0                      3057
3058 3058          DO 1130 IIDG = 1 , JNVEDG      3058
3059 3059          IF( INVEDG( IIDG ) . NE . 0 ) THEN 3059
3060 3060          IEDUM = IEDUM + 1             3060
3061 3061          INVEDG( IEDUM ) = INVEDG( IIDG ) 3061
3062 3062          END IF                        3062
3063 3063          1130 CONTINUE                3063
3064 3064          JNVEDG = IEDUM                3064
3065 3065          C                            3065
3066 3066          DO 1140 IS = 1 , JNVTRG        3066
3067 3067          ISS = INVTRG( IS )            3067
3068 3068          DO 1140 IITG = IS + 1 , JNVTRG 3068
3069 3069          IF( INVTRG( IITG ) . EQ . ISS ) THEN 3069
3070 3070          INVTRG( IITG ) = 0           3070
3071 3071          END IF                        3071
3072 3072          1140 CONTINUE                3072
3073 3073          C                            3073
3074 3074          ISDUM = 0                      3074
3075 3075          DO 1150 IITG = 1 , JNVTRG      3075
3076 3076          IF( INVTRG( IITG ) . NE . 0 ) THEN 3076
3077 3077          ISDUM = ISDUM + 1             3077
3078 3078          INVTRG( ISDUM ) = INVTRG( IITG ) 3078
3079 3079          END IF                        3079
3080 3080          1150 CONTINUE                3080
3081 3081          JNVTRG = ISDUM                3081
3082 3082          C                            3082
3083 3083          C UPDATE THE VERTECIS AND CELLS THAT ARE CONNECTED TO THE DELETED 3083
3084 3084          C EDGES                        3084
3085 3085          C                            3085
3086 3086          DO 1160 IE = 1 , IETRIG        3086
3087 3087          IES = IECRSS( IE )            3087
3088 3088          C                            3088
3089 3089          IV = JE( 1 , IES )             3089
3090 3090          IER = JV( 2 , IV )             3090
3091 3091          IIN = ISIGN( 1 , IER )         3091
3092 3092          IEE = IABS( IER )             3092
3093 3093          IEM = IEE                       3093
3094 3094          DO 1170 KK = 1 , IETRIG        3094
3095 3095          JEM = IECRSS( KK )             3095
3096 3096          IF( IEE . EQ . JEM ) IEM = NECRSS( KK ) 3096
3097 3097          1170 CONTINUE                3097
3098 3098          JV( 2 , IV ) = IIN * IEM       3098
3099 3099          C                            3099
3100 3100          IV = JE( 2 , IES )             3100
3101 3101          IER = JV( 2 , IV )             3101
3102 3102          IIN = ISIGN( 1 , IER )         3102
3103 3103          IEE = IABS( IER )             3103
3104 3104          IEM = IEE                       3104
3105 3105          DO 1180 KK = 1 , IETRIG        3105
3106 3106          JEM = IECRSS( KK )             3106
3107 3107          IF( IEE . EQ . JEM ) IEM = NECRSS( KK ) 3107

```

3108	3108	1180	CONTINUE	3108
3109	3109		JV(2 , IV) = IIN * IEM	3109
3110	3110	C		3110
3111	3111	1160	CONTINUE	3111
3112	3112	C		3112
3113	3113		DO 1190 KK = 1 , JJTRIG	3113
3114	3114		IVV = IITRIG(KK)	3114
3115	3115		DO 1190 JVDL = 1 , JVDL	3115
3116	3116		IVDL = NVDEL(JVDL)	3116
3117	3117		NVDL = IVDEL(JVDL)	3117
3118	3118		IF(IVV . EQ . NVDL) IITRIG(KK) = IVDL	3118
3119	3119	1190	CONTINUE	3119
3120	3120	C		3120
3121	3121		DO 1200 JVDL = 1 , JVDL	3121
3122	3122		IVDL = NVDEL(JVDL)	3122
3123	3123		NVDL = IVDEL(JVDL)	3123
3124	3124		JV(2 , IVDL) = JV(2 , NVDL)	3124
3125	3125	1200	CONTINUE	3125
3126	3126	C		3126
3127	3127		DO 1210 IS = 1 , JNVTRG	3127
3128	3128		ISS = INVTRG(IS)	3128
3129	3129	C		3129
3130	3130		IV = JS(1 , ISS)	3130
3131	3131		IVM = IV	3131
3132	3132		DO 1220 KI = 1 , JVDL	3132
3133	3133		JVM = IVDEL(KI)	3133
3134	3134		IF(IV . EQ . JVM) IVM = NVDEL(KI)	3134
3135	3135	1220	CONTINUE	3135
3136	3136		JS(1 , ISS) = IVM	3136
3137	3137	C		3137
3138	3138		IV = JS(2 , ISS)	3138
3139	3139		IVM = IV	3139
3140	3140		DO 1230 KI = 1 , JVDL	3140
3141	3141		JVM = IVDEL(KI)	3141
3142	3142		IF(IV . EQ . JVM) IVM = NVDEL(KI)	3142
3143	3143	1230	CONTINUE	3143
3144	3144		JS(2 , ISS) = IVM	3144
3145	3145	C		3145
3146	3146		IV = JS(3 , ISS)	3146
3147	3147		IVM = IV	3147
3148	3148		DO 1240 KI = 1 , JVDL	3148
3149	3149		JVM = IVDEL(KI)	3149
3150	3150		IF(IV . EQ . JVM) IVM = NVDEL(KI)	3150
3151	3151	1240	CONTINUE	3151
3152	3152		JS(3 , ISS) = IVM	3152
3153	3153	C		3153
3154	3154	1210	CONTINUE	3154
3155	3155	C		3155
3156	3156		DO 1250 IE = 1 , JNVEDG	3156
3157	3157		IEE = INVEDG(IE)	3157
3158	3158	C		3158
3159	3159		IV = JE(1 , IEE)	3159
3160	3160		IVM = IV	3160
3161	3161		DO 1260 KI = 1 , JVDL	3161
3162	3162		JVM = IVDEL(KI)	3162
3163	3163		IF(IV . EQ . JVM) IVM = NVDEL(KI)	3163
3164	3164	1260	CONTINUE	3164
3165	3165		JE(1 , IEE) = IVM	3165
3166	3166	C		3166
3167	3167		IV = JE(2 , IEE)	3167
3168	3168		IVM = IV	3168
3169	3169		DO 1270 KI = 1 , JVDL	3169
3170	3170		JVM = IVDEL(KI)	3170
3171	3171		IF(IV . EQ . JVM) IVM = NVDEL(KI)	3171
3172	3172	1270	CONTINUE	3172
3173	3173		JE(2 , IEE) = IVM	3173
3174	3174	C		3174
3175	3175	1250	CONTINUE	3175
3176	3176	C		3176
3177	3177	C	UPDATE THE VERTECIS AND EDGES THAT ARE CONNECTED TO THE DELETED	3177
3178	3178	C	CELSS	3178
3179	3179	C		3179
3180	3180		DO 1280 IS = 1 , ITRIG	3180
3181	3181	C		3181

3182	3182		ISE = ISCRSS(IS)	3182
3183	3183	C		3183
3184	3184		IE = IABS(JS(4 , ISE))	3184
3185	3185		ISS = JE(3 , IE)	3185
3186	3186		ISM = ISS	3186
3187	3187		DO 1290 KI = 1 , ITRIG	3187
3188	3188		JSM = ISCRSS(KI)	3188
3189	3189		IF(ISS . EQ . JSM) ISM = NSCRSS(KI)	3189
3190	3190	1290	CONTINUE	3190
3191	3191		JE(3 , IE) = ISM	3191
3192	3192	C		3192
3193	3193		ISS = JE(4 , IE)	3193
3194	3194		ISM = ISS	3194
3195	3195		DO 1300 KI = 1 , ITRIG	3195
3196	3196		JSM = ISCRSS(KI)	3196
3197	3197		IF(ISS . EQ . JSM) ISM = NSCRSS(KI)	3197
3198	3198	1300	CONTINUE	3198
3199	3199		JE(4 , IE) = ISM	3199
3200	3200	C		3200
3201	3201		IE = IABS(JS(5 , ISE))	3201
3202	3202		ISS = JE(3 , IE)	3202
3203	3203		ISM = ISS	3203
3204	3204		DO 1310 KI = 1 , ITRIG	3204
3205	3205		JSM = ISCRSS(KI)	3205
3206	3206		IF(ISS . EQ . JSM) ISM = NSCRSS(KI)	3206
3207	3207	1310	CONTINUE	3207
3208	3208		JE(3 , IE) = ISM	3208
3209	3209	C		3209
3210	3210		ISS = JE(4 , IE)	3210
3211	3211		ISM = ISS	3211
3212	3212		DO 1320 KI = 1 , ITRIG	3212
3213	3213		JSM = ISCRSS(KI)	3213
3214	3214		IF(ISS . EQ . JSM) ISM = NSCRSS(KI)	3214
3215	3215	1320	CONTINUE	3215
3216	3216		JE(4 , IE) = ISM	3216
3217	3217	C		3217
3218	3218		IE = IABS(JS(6 , ISE))	3218
3219	3219		ISS = JE(3 , IE)	3219
3220	3220		ISM = ISS	3220
3221	3221		DO 1330 KI = 1 , ITRIG	3221
3222	3222		JSM = ISCRSS(KI)	3222
3223	3223		IF(ISS . EQ . JSM) ISM = NSCRSS(KI)	3223
3224	3224	1330	CONTINUE	3224
3225	3225		JE(3 , IE) = ISM	3225
3226	3226	C		3226
3227	3227		ISS = JE(4 , IE)	3227
3228	3228		ISM = ISS	3228
3229	3229		DO 1340 KI = 1 , ITRIG	3229
3230	3230		JSM = ISCRSS(KI)	3230
3231	3231		IF(ISS . EQ . JSM) ISM = NSCRSS(KI)	3231
3232	3232	1340	CONTINUE	3232
3233	3233		JE(4 , IE) = ISM	3233
3234	3234	C		3234
3235	3235	1280	CONTINUE	3235
3236	3236	C		3236
3237	3237		DO 1350 IE = 1 , IETRIG	3237
3238	3238		IES = IECRSS(IE)	3238
3239	3239	C		3239
3240	3240		IS = JE(3 , IES)	3240
3241	3241		ISS = IS	3241
3242	3242		DO 1360 KI = 1 , ITRIG	3242
3243	3243		ISM = NSCRSS(KI)	3243
3244	3244		IF(IS . EQ . ISM) ISS = ISCRSS(KI)	3244
3245	3245	1360	CONTINUE	3245
3246	3246	C		3246
3247	3247		IF(ISS . NE . 0) THEN	3247
3248	3248	C		3248
3249	3249		IER = JS(4 , ISS)	3249
3250	3250		IEE = IABS(IER)	3250
3251	3251		IEM = IEE	3251
3252	3252		DO 1370 KI = 1 , IETRIG	3252
3253	3253		JEM = IECRSS(KI)	3253
3254	3254		IF(IEE . EQ . JEM) IEM = NECRSS(KI)	3254
3255	3255	1370	CONTINUE	3255

3256	3256		JS(4 , ISS) = ISIGN(1 , IER) * IEM	3256
3257	3257	C		3257
3258	3258		IER = JS(5 , ISS)	3258
3259	3259		IEE = IABS(IER)	3259
3260	3260		IEM = IEE	3260
3261	3261		DO 1380 KI = 1 , IETRIG	3261
3262	3262		JEM = IECRSS(KI)	3262
3263	3263		IF(IEE . EQ . JEM) IEM = NECRSS(KI)	3263
3264	3264	1380	CONTINUE	3264
3265	3265		JS(5 , ISS) = ISIGN(1 , IER) * IEM	3265
3266	3266	C		3266
3267	3267		IER = JS(6 , ISS)	3267
3268	3268		IEE = IABS(IER)	3268
3269	3269		IEM = IEE	3269
3270	3270		DO 1390 KI = 1 , IETRIG	3270
3271	3271		JEM = IECRSS(KI)	3271
3272	3272		IF(IEE . EQ . JEM) IEM = NECRSS(KI)	3272
3273	3273	1390	CONTINUE	3273
3274	3274		JS(6 , ISS) = ISIGN(1 , IER) * IEM	3274
3275	3275	C		3275
3276	3276		END IF	3276
3277	3277	C		3277
3278	3278		IS = JE(4 , IES)	3278
3279	3279		ISS = IS	3279
3280	3280		DO 1400 KI = 1 , ITRIG	3280
3281	3281		ISM = NSCRSS(KI)	3281
3282	3282		IF(IS . EQ . ISM) ISS = ISCRSS(KI)	3282
3283	3283	1400	CONTINUE	3283
3284	3284	C		3284
3285	3285		IF(ISS . NE . 0) THEN	3285
3286	3286	C		3286
3287	3287		IER = JS(4 , ISS)	3287
3288	3288		IEE = IABS(IER)	3288
3289	3289		IEM = IEE	3289
3290	3290		DO 1410 KI = 1 , IETRIG	3290
3291	3291		JEM = IECRSS(KI)	3291
3292	3292		IF(IEE . EQ . JEM) IEM = NECRSS(KI)	3292
3293	3293	1410	CONTINUE	3293
3294	3294		JS(4 , ISS) = ISIGN(1 , IER) * IEM	3294
3295	3295	C		3295
3296	3296		IER = JS(5 , ISS)	3296
3297	3297		IEE = IABS(IER)	3297
3298	3298		IEM = IEE	3298
3299	3299		DO 1420 KI = 1 , IETRIG	3299
3300	3300		JEM = IECRSS(KI)	3300
3301	3301		IF(IEE . EQ . JEM) IEM = NECRSS(KI)	3301
3302	3302	1420	CONTINUE	3302
3303	3303		JS(5 , ISS) = ISIGN(1 , IER) * IEM	3303
3304	3304	C		3304
3305	3305		IER = JS(6 , ISS)	3305
3306	3306		IEE = IABS(IER)	3306
3307	3307		IEM = IEE	3307
3308	3308		DO 1430 KI = 1 , IETRIG	3308
3309	3309		JEM = IECRSS(KI)	3309
3310	3310		IF(IEE . EQ . JEM) IEM = NECRSS(KI)	3310
3311	3311	1430	CONTINUE	3311
3312	3312		JS(6 , ISS) = ISIGN(1 , IER) * IEM	3312
3313	3313	C		3313
3314	3314		END IF	3314
3315	3315	C		3315
3316	3316	1350	CONTINUE	3316
3317	3317	C		3317
3318	3318		DO 1440 IE = 1 , IETRIG	3318
3319	3319		IEM = NECRSS(IE)	3319
3320	3320		JEM = IECRSS(IE)	3320
3321	3321	C		3321
3322	3322		DO 1450 IK = 1 , 5	3322
3323	3323		JE(IK , IEM) = JE(IK , JEM)	3323
3324	3324	1450	CONTINUE	3324
3325	3325	C		3325
3326	3326		XE(1 , IEM) = XE(1 , JEM)	3326
3327	3327		XE(2 , IEM) = XE(2 , JEM)	3327
3328	3328	C		3328
3329	3329		XN(IEM) = XN(JEM)	3329

```

3330 3330 YN( IEM ) = YN( JEM ) 3330
3331 3331 XXN( IEM ) = XXN( JEM ) 3331
3332 3332 YYN( IEM ) = YYN( JEM ) 3332
3333 3333 XMIDL( IEM ) = XMIDL( JEM ) 3333
3334 3334 YMIDL( IEM ) = YMIDL( JEM ) 3334
3335 3335 XYMIDL( IEM ) = XYMIDL( JEM ) 3335
3336 3336 1440 CONTINUE 3336
3337 3337 C 3337
3338 3338 DO 1460 IS = 1 , ITRIG 3338
3339 3339 ISM = NSCRSS( IS ) 3339
3340 3340 JSM = ISCRSS( IS ) 3340
3341 3341 C 3341
3342 3342 DO 1470 IK = 1 , 6 3342
3343 3343 JS( IK , ISM ) = JS( IK , JSM ) 3343
3344 3344 1470 CONTINUE 3344
3345 3345 C 3345
3346 3346 XS( 1 , ISM ) = XS( 1 , JSM ) 3346
3347 3347 XS( 2 , ISM ) = XS( 2 , JSM ) 3347
3348 3348 XS( 3 , ISM ) = XS( 3 , JSM ) 3348
3349 3349 C 3349
3350 3350 SAREA( ISM ) = SAREA( JSM ) 3350
3351 3351 KSDEL( ISM ) = KSDEL( JSM ) 3351
3352 3352 C 3352
3353 3353 DO 1480 IK = 1 , MHQ 3353
3354 3354 HYDV( ISM , IK ) = HYDV( JSM , IK ) 3354
3355 3355 1480 CONTINUE 3355
3356 3356 C 3356
3357 3357 HYDFLX( ISM , 4 ) = HYDFLX( JSM , 4 ) 3357
3358 3358 HYDFLX( ISM , 1 ) = HYDFLX( JSM , 1 ) 3358
3359 3359 HYDFLX( ISM , 2 ) = HYDFLX( JSM , 2 ) 3359
3360 3360 C 3360
3361 3361 1460 CONTINUE 3361
3362 3362 C 3362
3363 3363 NV = NVM 3363
3364 3364 NE = NEM 3364
3365 3365 NS = NSM 3365
3366 3366 C 3366
3367 3367 DO 1490 IENN = 1 , IJTRIG 3367
3368 3368 IE = JUE( IENN ) 3368
3369 3369 DO 1490 KI = 1 , IETRIG 3369
3370 3370 JEM = IECRSS( KI ) 3370
3371 3371 IF( IE . EQ . JEM ) JUE( IENN ) = NECRSS( KI ) 3371
3372 3372 1490 CONTINUE 3372
3373 3373 C 3373
3374 3374 DO 1540 IENN = 1 , JJTRIG 3374
3375 3375 IVV = IITRIG( IENN ) 3375
3376 3376 IF( JV( 1 , IVV ) . NE . 3 ) CALL RELAXY( IVV ) 3376
3377 3377 1540 CONTINUE 3377
3378 3378 C 3378
3379 3379 DO 1500 IENN = 1 , IJTRIG 3379
3380 3380 IE = JUE( IENN ) 3380
3381 3381 CALL RECNC( IE , IDONE , ITL , ITR , JAA , JAB , JAC , JAD ) 3381
3382 3382 CALL RECNC( JA , JADONE , ITL , ITR , JAA , JAB , JAC , JAD ) 3382
3383 3383 CALL RECNC( JB , JBDONE , ITL , ITR , JBA , JBB , JBC , JBD ) 3383
3384 3384 CALL RECNC( JC , JCDONE , ITL , ITR , JCA , JCB , JCC , JCD ) 3384
3385 3385 CALL RECNC( JD , JDDONE , ITL , ITR , JDA , JDB , JDC , JDD ) 3385
3386 3386 1500 CONTINUE 3386
3387 3387 C 3387
3388 3388 DO 1510 IPRTCL = 1 , NPT 3388
3389 3389 ISM = IJKPRT( IPRTCL ) 3389
3390 3390 DO 1510 KI = 1 , ITRIG 3390
3391 3391 JSM = NSCRSS( KI ) 3391
3392 3392 IF( ISM . EQ . JSM ) IJKPRT( IPRTCL ) = NSMNPT 3392
3393 3393 1510 CONTINUE 3393
3394 3394 C 3394
3395 3395 C UPDATE THE JSDEL( IS ) 3395
3396 3396 C 3396
3397 3397 DO 1530 IS = 1 , ISDEL( IS ) 3397
3398 3398 JSP = JSDEL( IS ) 3398
3399 3399 DO 1530 KI = 1 , ITRIG 3399
3400 3400 JSM = ISCRSS( KI ) 3400
3401 3401 IF( JSP . EQ . JSM ) JSDEL( IS ) = NSCRSS( KI ) 3401
3402 3402 1530 CONTINUE 3402
3403 3403 C 3403

```

3404	3404		INDCTR = 1		3404
3405	3405	C			3405
3406	3406	C	--- EXIT POINT FROM SUBROUTINE -----		3406
3407	3407	C			3407
3408	3408	C	-----		3408
3409	3409		RETURN		3409
3410	3410	C	-----		3410
3411	3411	C			3411
3412	3412	C	---		3412
3413	3413		END		3413

APPENDIX C
COPIES OF PUBLICATIONS



AIAA 89-2446

**A REVIEW OF PROPULSION APPLICATIONS
OF THE PULSED DETONATION ENGINE
CONCEPT**

**S. EIDELMAN, W. GROSSMANN AND I. LOTTATI
SCIENCE APPLICATIONS INTERNATIONAL CORP.
APPLIED PHYSICS OPERATION
MCLEAN, VA**

**AIAA/ASME/SAE/ASEE
25th Joint Propulsion Conference
Monterey, CA / July 10-12, 1989**

A REVIEW OF PROPULSION APPLICATIONS OF THE PULSED DETONATION ENGINE CONCEPT

S. Eidelman, W. Grossmann and I. Lottati

Applied Physics Operation
Science Applications International Corporation
1710 Goodridge Dr., McLean, VA 22102

Introduction

The early development leading to practical propulsion engines was almost completely associated with steady state engine concepts. Unsteady concepts, which initially appeared promising, never evolved from the conceptual state and have remained for the most part unexplored. The early work in unsteady propulsion suffered from a lack of appropriate analytical and design tools, a condition which seriously impeded the advancement of the unsteady concepts to a practical stage.

In this paper we review the historical development of unsteady propulsion by concentrating on one particular concept, the intermittent detonation engine, and discuss current research activities in this area. A review of the literature¹⁻²⁴ reveals that a significant body of experimental and theoretical research exists in the area of unsteady propulsion. However, this research has not been extended to the point where a conclusive quantitative comparison can be made between impulsive engine concepts and steady state concepts. For example, the analysis given in References 8-11 of the performance of a detonation engine concept does not include frequency dependence, nor any analysis of losses due to multi-cycle operation. A new generation of analytical and computational tools exists today and allows us to revisit and analyse such issues with a high degree of confidence. Numerical simulation has developed to the state where it can now provide time dependent two and three dimensional modeling of complex internal flow processes^{20,24,25} and will eventually result in tools for systematically analysing and optimising engineering design. In addition to a review of applications of the Pulsed Detonation Engine Concept here we will report results of a numerical study of the gasdynamics of a model of an air-breathing detonation engine with detailed analysis of the nonsteady flow pattern. This study was performed using new unsteady CFD tools which we will also describe.

Our paper is structured as follows: 1) historical review of the pulsed detonation development efforts; 2) description of the basic phenomenology of the air-breathing Pulsed Detonation Engine concept; 3) description of the

mathematical formulation and new numerical scheme used to simulate the problem; 4) discussion of the simulation results; and 5) conclusions.

Historical Review

Constant Volume Combustion

From the very early development of jet-propulsion engines it was known that an engine based on a constant volume combustion process achieves higher efficiency than a constant pressure engine. This follows from a thermodynamic analysis of the engine cycle.¹

Constant volume combustion was used in gas turbine engines at the beginning of this century, and the first gas turbine engines in commercial use were based on the constant volume cycle. Jet-propulsion engines were one of the applications of the constant volume cycle (or explosion cycle) which was explored in the late 1940s.² Although the explosion cycle operates at a larger pressure variation in the combustion chamber than in a pulse-jet^{3,4}, the cycle actually realized in these engines was not a fully constant volume one since the combustion chamber was open ended². In Reference 2 the maximum pressure ratio measured in an explosion cycle engine was 3:1, whereas the pressure ratio for the same mixture under the assumption of a constant volume cycle would be 8:1. Also, this engine was limited by the available frequency of cycles, which in turn is limited by the reaction rate. A simple calculation² showed that if the combustion time could be reduced in this engine from 0.006 sec to 0.003 sec, the thrust per pound of mixture would increase 100%. Thus the explosion-cycle engine has two main disadvantages:

- Constrained volume combustion (as distinguished from constant volume combustion) does not take full advantage of the pressure rise characteristic of the constant volume combustion process.
- The frequency of the explosion cycle is limited by the reaction rate, which is only slightly higher than the deflagrative combustion rate.

The main advantage of the constant pressure cycle is that it leads to engine configurations with steady state

processes of injection of the fuel and oxidiser, combustion of the mixture, and expansion of the combustion products. These stages can be easily identified and the engine designer can optimise them on the basis of relatively simple steady state considerations:

At the same time an engine based on constant volume combustion will have an intermittent mode of operation, which may complicate its design and optimisation. We are interested in the question of whether this complication is worth the potential gains in engine efficiency.

Pulsed Detonation Engine as an Ultimate Constant Volume Combustion Concept

The detonation process, due to the very high rate of reaction, permits construction of a propulsion engine in which the constant volume process can be fully realised. In detonative combustion, the strong shock wave, which is part of the detonation wave, acts like a valve between the detonation products and the fresh charge. The speed of the detonation wave is about two orders of magnitude higher than the speed of a typical deflagration. This allows the design of propulsion engines with a very high power density. Usually, each detonation is initiated separately by a fully controlled ignition device, and the cycle frequency can be changed over a wide range of values. This also means that a device based on a detonative combustion cycle can be scaled and its operating parameters can be modified for a range of required output conditions. There have been numerous attempts to take advantage of detonative combustion for engine applications. In the following we give a description of the most relevant past experimental and analytical studies of the detonation engine concept.

Hoffmann's Report.

The first reported work on intermittent detonation is attributed to Hoffmann⁵ in 1940. He operated an intermittent detonation test stand with acetylene-oxygen and bensine-oxygen mixtures. The addition of water vapor was used to prevent the highly sensitive acetylene-oxygen mixture from premature detonation. Hoffmann⁵ indicated the importance of the spark plug location in reference to tube length and diffuser length. It was found that a continuous injection of the combustible mixture leads to only a narrow range of ignition frequencies which will produce an intermittent detonation cycle. These frequencies are governed by the time required for the mixture to reach the igniter, time of transition from deflagration to detonation, and time of expansion of the detonation products. Hoffmann attempted to find the optimum cycle frequency experimentally. It was discovered that detonation-tube firing occurred at lower frequencies than the spark-plug energising frequencies indicating that the

injection flow rate and ignition were out of phase. Events prevented further work by Hoffmann and co-workers.

Nicholls Experiments.

A substantial effort in intermittent detonation engine research was done by a group headed by J. A. Nicholls⁶⁻¹⁰ of The University of Michigan beginning in the early 50's. The most relevant work concerns a set of experiments carried out in a six foot long detonation tube⁶. The detonation tube was constructed from a one inch internal diameter stainless steel tube. The fuel and oxidiser were injected under pressure from the left end of the tube and ignited at the some distance down stream. The tube was mounted on a pendulum platform, suspended by support wires. Thrust for single detonations was measured by detecting tube (platform) movement relative to a stationary pointer. For multi-cycle detonations thrust measurement was achieved by mounting the thrust end of the tube to the free end of the cantilever beam. In addition to direct thrust measurements the temperature on the inner wall of the detonation tube was measured.

Fuel mixtures of hydrogen/oxygen, hydrogen/air, acetylene-oxygen and acetylene-air mixtures were used. The gaseous oxidiser and fuel were continuously injected at the closed end wall of the detonation tube and three fixed flow rates were used. Under these conditions the only parameters which could be varied were the fuel/oxidiser ratio and frequency of ignition. A maximum gross thrust of $\approx 3.2lb$ was measured in hydrogen/air mixture at the frequency of ≈ 30 detonations per second. The most promising results were demonstrated for the H_2 /Air mixture, where a fuel specific impulse of $I_{sp} = 2100$ sec was reached. The maximum frequency of detonations obtained in all experiments was 35 Hz. The temperature measurements on the inner wall showed that for the highest frequency of detonations the temperature did not exceed 800° F.

In their later work,^{8,9,10} the University of Michigan group concentrated on development of the Rotating Detonation Wave Rocket Motor. No further work on the pulsed detonation cycle was pursued.

Krsvcki Experiments

In a setup somewhat similar to Nicholl's, L. J. Krsvcki¹¹ performed an experimental investigation of intermittent detonations with frequencies up to 60 cps. An attempt was also made to analyze the basic phenomena using unsteady gas dynamic theory. Krsvcki's attempt to analyze the basic phenomena relied on wave diagrams to trace characteristics, assumptions of isentropic flow for detonation and expansion, and incompressible flow for mixture injection processes. The most convincing

data from the experiments is the measurement of thrust for a range of initiation frequencies and mixture flow rates. Unfortunately no direct pressure measurement in the device are reported so that only indirect evidence exists of the nature of the process observed.

The basic test stand used by Krzycki is very similar to that used by Nicholls et al.⁶ The length of the detonation tube and internal diameter were exactly the same as those in Nicholl's experiments. A Propane/Air mixture was continuously injected through a reversed-flow diffuser for better mixing, and ignited at the some distance from the injection point by an automobile spark plug. The spark frequency was varied from 1 to 60 cps. The spark plug power output was varied inversely with the initiation frequency and at the frequency of 60 cps was only 0.65 Joule. This fact alone eliminated the possibility of direct initiation of the detonation wave by the spark and consequently all of the experiments must have been based on transition from deflagration to detonation. According to experimental data and theory,¹² for direct initiation of a mixture of propane-air at the detonability limits, an energy release on the order of 10^6 Joules is required. Thus, the required deflagration-detonation transition region length would have been prohibitively large for the propane-air mixture. It follows that in all of the experiments a substantial part of the process was deflagrative. This resulted in low efficiency, and negligible thrust. Krzycki repeated the experiments of Nicholls using exactly the same size detonation tube and basically the same rates of injection of the detonable mixture. Krzycki's experimental results are very well documented, allowing a clear picture of the physical processes occurring in the tube to be deduced. A conclusion, arrived at by the author, was that thrust was possible from such a device but practical applications did not appear promising. It is unfortunate that, possibly based on Krzycki's extensive but misleading results, all experimental work related to the pulsed detonation engine concept stopped at this time.

Work Reported in Russian sources on Pulse Detonation Devices

A review of the Russian literature has not uncovered work concerning applications of pulsed detonation devices to propulsion. However there are numerous reports of applications of such devices for producing nitrogen oxide¹³ (an old Zeldovich idea to bind nitrogen directly from air to produce fertilizers) and as rock crushing devices¹⁴.

Korovin et al.¹³ provide a most interesting account of the operation of a commercial detonation reactor. The main objective of this study was to examine the efficiency of thermal oxidation of nitrogen in an intermittent detonative process as well as an assessment of such techno-

logical issues as the fatigue of the reactor parts exposed to the intermittent detonation waves over a prolonged time. The reactor consisted of a tube with an inner diameter of 16 mm and length 1.3 m joined by a conical diffuser to a second tube with an inner diameter of 70 mm and length 3 m. The entire detonation reactor was submerged in running water. The detonation mixture was introduced at the end wall of the small tube. CH_4 , O_2 and N_2 comprised the mixture composition and the mixture ratios were varied during the continuous operation of the reactor. The detonation wave velocity was measured directly by piezoelectric sensors placed in the small and large tubes. The detonation initiation frequency in the reactor was 2-16 Hz. It is reported that the apparatus operated without significant changes for 2000 hours.

Smirnov and Boichenko¹⁴ studied intermittent detonations of gasoline-air mixtures in a 3 m long and 22 mm inner diameter tube operating in the 6-8 Hz ignition frequency range. The main motivation of this work was to improve the efficiency of a commercial rock crushing apparatus based on intermittent detonations of the gasoline-air mixtures.¹⁵ The authors investigated the dependence of the length of the transitional region from deflagration to detonation on the initial temperature of the mixture.

As a result of the information contained in the Soviet reports, it can be concluded that reliable commercial devices based on intermittent detonations can be constructed and operated.

Development of the Blast Propulsion System at JPL

Work at the Jet Propulsion Laboratory (JPL) by Back, Varzi and others¹⁶⁻¹⁹ concerned an experimental and theoretical study of the feasibility of a rocket thruster using intermittent detonations of solid explosive useful for propulsion in dense or high-pressure atmospheres of certain solar system planets. The JPL work was directed at very specific applications; however, the studies¹⁷⁻¹⁹ addressed some key issues of devices using unsteady process such as propulsion efficiency. The JPL studies have important implications to pulsed detonation propulsion systems.

Reference 19 gives the basic description of the test stand used. In this work a Deta sheet type C explosive was detonated inside a small detonation chamber attached to nozzles of various length and geometry. The nozzles, complete with firing plug, were mounted in a containment vessel which could be pressurised with the mixture of various inert gases from vacuum to 70 atm. The apparatus measured directly the thrust generated by single detonations of a small amount of solid explosive charge expanding into conical or straight nozzles.

Thrust and specific impulse was measured by a pendulum balance system.

Results obtained from an extensive experimental study of the explosively driven rocket have led to the following conclusions. First, rockets with long nozzles show increasing specific impulse with increasing ambient pressure in CO_2 and N_2 . Short nozzles, on the other hand, show that specific impulse is independent of ambient pressure. Most importantly, most of the experiments obtained a relatively high specific impulse of 250 seconds and larger. This result is all the more striking since the detonation of a solid explosive yields a relatively low energy release of approximately 1000 cal/gm compared with 3000 cal/gm obtained in hydrogen oxygen combustion. Thus, it can be concluded that the total losses in a thruster based on unsteady expansion are not prohibitive and, in principle, very efficient propulsion systems operating on intermittent detonations are possible.

Detonation Engine Studies at Naval Postgraduate School

A modest exploratory study of a propulsion device utilizing detonation phenomena was conducted at the Naval Postgraduate School.²⁰⁻²³ During this study, several fundamentally new elements were introduced to the concept distinguishing the new device from previous ones.

First, it is important to note that the experimental apparatus constructed by Helman et al.²¹ was the first successful self aspirating air breathing detonation device. Intermittent detonation frequencies of 25 Hz were obtained. This frequency was in phase with the fuel mixture injection through timed fuel valve opening and spark discharge. The feasibility of intermittent injection was established. Pressure measurements showed conclusively that a detonation process occurred at the frequency chosen for fuel injection. Further, self aspiration was shown to be effective. Finally, the effectiveness of a primary detonation as a driver for the main detonation was clearly demonstrated. Although the NPS studies were abbreviated, many of the technical issues considered to be essential for efficient intermittent detonation propulsion were addressed with positive results.

Simulations of Pulsed Detonation Engine Cycle at NASA-Ames Center

Recently Cambier and Adelman²⁴ carried out numerical simulations of a pulsed detonation engine cycle taking into account finite rate chemistry. Unfortunately, the simulations were restricted to a quasi-one dimensional model. The configuration considered had a 6 cm inner diameter 50 cm long main chamber which was attached to a 43 cm diverging nozzle. It was assumed that

a stoichiometric mixture of hydrogen/air at 3.0 atmospheres is injected from an inlet on the closed end wall of the detonation chamber. At such conditions Cambier and Adelman estimated a large range of possible detonation frequencies of engine operation up to 667 Hz. The origin of this estimate is not clear from their work, since according to their simulations, the detonation, expansion and fresh charge fill requires 2.5 msec. This value leads to a maximum frequency of 400 Hz. The simulated engine performance yielded a large average thrust of 893 lb and an unusually high specific impulse of 6507 sec. These simulations were the first to demonstrate the use of modern CFD methods to address the technical issues associated with unsteady pulsed detonation concepts.

In the remaining sections we discuss a particular propulsion concept based on the results of the experiments of Helman et al.²² and describe a computational study of its performance characteristics. The unsteady numerical scheme used for the study made use of unique simulation techniques; the key ingredients of these techniques are also described.

A Generic Pulsed Detonation Engine

The generic device we consider here is a small engine 15 cm long and 15 cm in diameter. The combustible gas mixture is injected at the closed end of the detonation chamber and a detonation wave propagates through the mixture. The size of the engine suggests a small payload, but the concept can be extended to larger payloads simply by scaling up the size of the detonation chamber and possibly combining a number of engines into one large propulsion engine. A key issue in the pulsed detonation engine concept is the design of the main detonation chamber. The detonation chamber geometry determines the propulsion efficiency and the duration of the cycle (frequency of detonations). Since the fresh charge for the generic engine is supplied from the external flow field, the efficiency of the engine depends on the interaction of the surrounding flow with the internal flow dynamics. The range of the physical processes requiring simulation in order to model the complex flow phenomena associated with the detonation engine performance is very broad.

A partial list is:

1. Initiation and propagation of the detonation wave inside the chamber,
2. Expansion of the detonation products from the chamber into the air stream around the chamber at flight Mach numbers.
3. Reverse flow from the surrounding air into the chamber resulting from over expansion of the detonation products,
4. Pressure buildup in the chamber due to reverse flow. The flow pattern inside the chamber during post-exhaust pressure buildup determines the strategy

for mixing the next detonation charge,

5. Strong mutual interaction between the flow processes inside the chamber and flow around the engine.

All of these processes are interdependent and their timing is crucial to the engine efficiency. Thus, unlike simulations of steady state engines, the phenomena described above can not be evaluated independently.

The need to resolve the flow regime inside the chamber accounting for nozzles, air inlets etc., and at the same time resolve the flow around the engine, where the flow regime varies from high subsonic, locally transonic and supersonic, makes it a challenging computational problem.

The main issue is to determine the timing of the air intake for the fresh gas charge. It is sufficient to assume inviscid flow for the purpose of simulating the expansion of the detonation products and fresh gas intake. In the following we present the first results of an inviscid simulation of the detonation cycle in a cylindrical chamber. First, we describe our computational method for solving the time dependent Euler equations used in the study.

The Unsteady Euler Solver

A new second order algorithm for solving the Euler equations on an unstructured grid was used in our study of the detonation concept. The approach is based on first and second order Godunov methods. The method leads to an extremely efficient and fast Flow Solver which is fully vectorised and easily lends itself to parallelisation. The low memory requirements and speed of the method are due to the use of a unique data structure.

Until recently most CFD simulations were carried out with logically structured grids. Vectorisation and/or parallelisation did not present a problem. The increased need for simulation of flow phenomena in the vicinity of complex geometrical bodies and surfaces has led to the development of CFD codes for logically unstructured grids. The most successful of these unstructured grid codes are based on finite elements or finite volume methods. For an unstructured grid in two-dimensions, the computational domain is usually covered by triangles and the indices of the arrays containing the values of the hydrodynamic flow quantities are not related directly to the actual geometric location of a node. The calculations performed on unstructured grids evolve around the elemental grid shape (e.g. the triangle for two-dimensional problems) and there is no obvious pattern to the order in which the local integrations should be performed. Explicit integration of hydrodynamic problems on an unstructured grid requires that a logical substructure should be created which identifies the locations in the global arrays of all the local quantities necessary for the integration of one element. This usually results in a large

price in computational efficiency, in memory requirements, and in code complexity. As a consequence, vectorisation for the conventional unstructured grid methods has concentrated on rearrangement of the data structure in a manner such that these locally centered data structures appear as global arrays. This can be done to some extent using machine dependent Gather-Scatter operations.^{25,26} Additional optimisation can be achieved using localisation and search algorithms. However, these methods are complex and result in marginal improvement. Most optimised unstructured codes to date run considerably slower and require an order of magnitude more memory per grid cell than their structured counterparts. Parallelisation of the conventional unstructured codes is even more difficult, there is very little experience with unstructured codes on massively parallel computers.

The method we have developed overcomes these difficulties and results in code with speed and memory requirements comparable to those found in structured grid codes. Moreover, the ability to construct grids with arbitrary resolution leads to a flexibility in dealing with complex geometries not attainable with structured grids. The essence of the method is based on independent flux calculation across the edges of a dual baricentric grid, followed by node integration. This approach is order independent. Below we give the essential details of our algorithm; a complete description follows later.

Basic Integration Algorithm.

We begin by describing the first order Godunov method for the system of two-dimensional (axi-symmetric) Euler equations written in conservation law form as

$$\frac{\partial \bar{Q}}{\partial t} + \frac{\partial \bar{F}}{\partial z} + \frac{\partial \bar{G}}{\partial r} = -\frac{1}{r} \bar{C}, \quad (1)$$

where,

$$\bar{Q} = \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ e \end{pmatrix}, \quad \bar{F} = \begin{pmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ (e + p)u \end{pmatrix}, \quad \bar{G} = \begin{pmatrix} \rho v \\ \rho vu \\ \rho v^2 + p \\ (e + p)v \end{pmatrix},$$

$$\bar{C} = \begin{pmatrix} \rho v \\ \rho vu \\ \rho v^2 \\ (e + p)v \end{pmatrix}.$$

Here u and v are the z and r velocity vector components, p is the pressure, ρ is the density and e is the total energy of the fluid per unit volume. It is assumed that a mixed (initial conditions, boundary conditions) problem is properly posed for the set of equations (1)

and that an initial distribution of the fluid parameters is given at $t = 0$ and some boundary conditions defining a unique solution are specified on the boundary of the computational domain.

We look for a solution of the system of equations represented by Eq. 1 in the computational domain covered by an unstructured grid. As an example, Fig. 1a shows the unstructured triangular grid used in the pulsed detonation engine simulation. Here most of the computational effort is committed to the resolution of the flow inside the engine detonation chamber and in the immediate vicinity of the nozzle. In Figure 1b an enlargement of the nozzle region is shown, illustrating the ability to represent geometry of arbitrary complexity and with localized resolution.

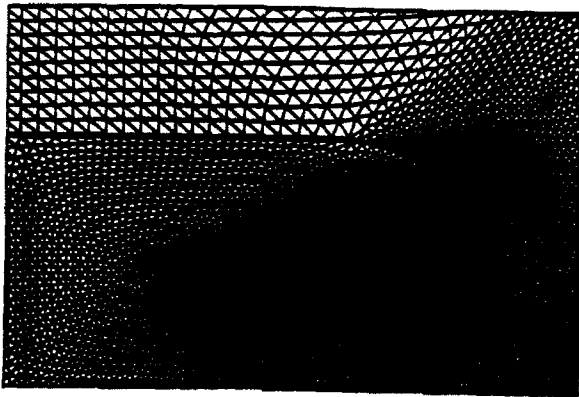


Figure 1a Computational domain and grid used in simulation of PDE operation.

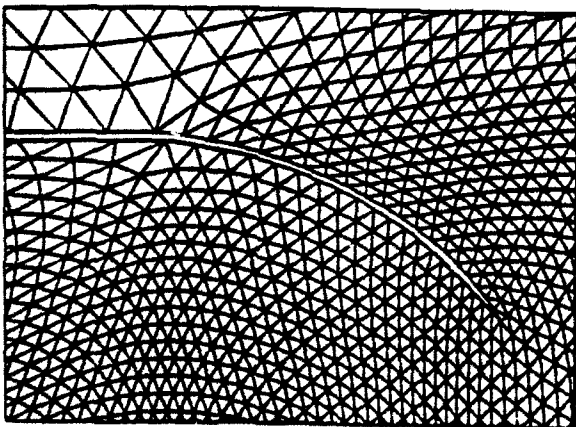


Figure 1b Enlargement of computational grid in the vicinity of the PDE nozzle.

Fig. 2 displays a fragment of the computational domain with the corresponding dual grid. The secondary or dual grid is formed by connecting the baricenters of the primary mesh, thus forming polygons around the primary vertices.

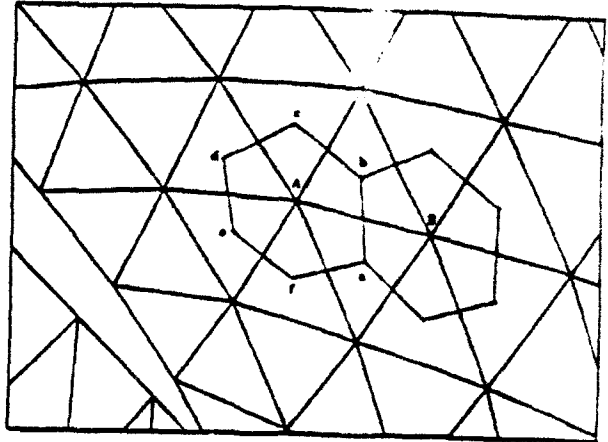


Figure 2 The primary (triangles) and secondary (polygons) unstructured grids.

We have found, as have others,²⁷ that the best practical representation of the integration volumes is obtained when the dual grid is formed by connecting baricenters of the triangles. Integration by the Godunov method²⁸ can be divided into two basic steps: 1. Calculation of the fluxes at the edges of the secondary grid using solutions of a set of one dimensional Riemann problems; 2. Integration of the system of partial differential equations which amounts to addition of all the fluxes for every polygon at a particular time step.

To define the fluxes for the grid shown in Fig. 2 at every edge of the main grid it is necessary to solve the corresponding Riemann problem. For example, to define the flux at the edge ab , we solve the Riemann problem between points A and B . The solution of this problem is in coordinates local to the edge of the dual grid ab so that the tangential component of velocity will be directed along this edge (ab). Implementation of our approach requires maintaining strict consistency when defining the "left" and "right" states for the Riemann problems at the edges ab , bc , cd , de , ef , and fa . For this reason we define not only the location of the vertices and lengths of the edges but also the direction of the edges with respect to the primary grid. For the clockwise integration pattern in the same Polygon, point A will be the "right" state for all the Riemann problems related to this point and the neighbor will represent the "left" side of the diaphragm.

It is easy to see that the flux calculation is based on information at only two nodes and requires single geometrical parameters defining the edge of the secondary

grid that dissects the line connecting the two points. Thus, we can calculate all the values needed for flux calculation in one loop over all edges of the primary grid without any details related to the geometrical structures which these edges form. This in turn assures parallelisation or vectorisation of the algorithm for the bulk of the calculations involving the Riemann solver that provides the first order flux. The only procedure not readily parallelisable is the integration of the fluxes for the flow variables at the vertices of the grid. Here we use the "edge coloring" technique which allows us to split the flux addition loop into 7 or 8 loops for edges of different color. Each of these loops is usually large enough not to impair vectorisation. At this stage all the fluxes are added with their correct sign corresponding to the chosen direction of integration within the cell. The amount of calculation required here is minimal since the fluxes are known and need only to be multiplied at each time step by a simple factor and added to the vertex quantity.

Second Order Integration Algorithm

The second order solver is constructed along lines similar to that from the first-order method. At each cell edge the Riemann problem is solved for some specified pair of left and right conditions. The solution to this Riemann problem is then used in the calculation of fluxes which are added later to advance to the next integration step. The extension to second order is achieved by using extrapolation in space and time to obtain time-centered left and right limiting values as inputs for the Riemann problem. The basic implementation of the method of calculation of second order accurate fluxes is fundamentally the same as for one dimensional cases. The only difference is in the method of obtaining linear extrapolation of the flow variables as a first guess of their value at the edges of the dual grid. To obtain the first guess we need to know the gradient of some gasdynamical parameter U at the vertices of the primary mesh. The value of ∇U can be evaluated by using a linear path integral along the edges which delineates the finite volume associated with the vertex. For vertex A in Figure 2 :

$$\int_A \nabla U dA = \oint_l U n dl \quad (2)$$

where integration along the path l in this case is equivalent to integration along the edges ab, bc, cd, de, ef, fa . Knowing the gradient of the gasdynamic parameter in the volume related to vertex A will allow us to extrapolate the values of this parameter at any location within the volume. This permits us to evaluate the first guess for U at the edges of the dual grid. The final four steps of the implementation of the second order algorithm has been described previously.²⁸

A schematic flow chart of the basic steps of the second order algorithm implementation is shown in Figure 3.

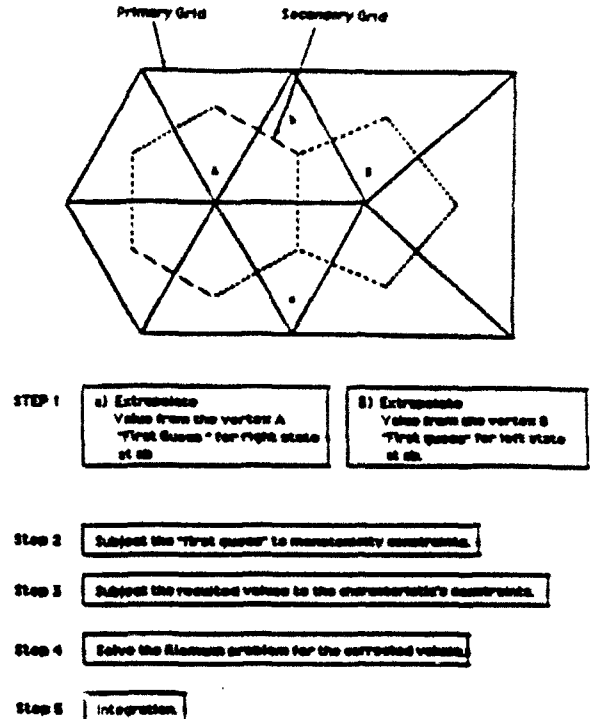


Figure 3 Grid schematic and outline of steps for second order Godunov method.

Simulations of the Generic Pulsed Detonation Engine

In this section we present sample results of simulations of the generic PDE device using the numerical code described in the preceding section. In Figure 1a the computational domain containing the PDE main detonation chamber is shown covered with the unstructured grid. In our sample simulation we have chosen a small ≈ 15 cm long and ≈ 15 cm internal diameter cylindrical chamber with a small converging nozzle. This geometry is one of a number of the geometries we have analyzed in a parametric study whose goal is to evaluate and optimize a typical PDE device. The device shown in Figure 1a does not represent the optimum and is given here to illustrate our methodology. We consider a situation when the PDE serves as a main thruster for a vehicle traveling in air with the velocity of $M = 0.9$ and located at the aft end of the vehicle. The main objectives of the simulations presented here are:

1. To find the maximum cycle frequency. This is determined by the time required from detonation, exhaust of combustion products and intake of fresh charge for the next detonation.

2. To calculate the thrust produced during each cycle and the integrated thrust as a function of time.

The simulation begins at $t = 0$ when a strong detonation wave is initiated inside the detonation chamber. Initially the detonation wave travels from the open aft end of the chamber towards the interior with a maximum velocity of $1800 \frac{m}{sec}$ and maximum pressure of $20 \cdot 10^5 Pa$. The distribution of pressure, velocity, and density of the detonation wave is defined through the selfsimilar solution for a planar detonation wave. The wave was directed towards the interior of the chamber to capture the kinetic energy of the wave and to prolong exposure of the inner chamber walls to the high pressure. In Figure 4a simulation results are shown at time $t = 0.19 \text{ msec}$ in the form of pressure contours and particle paths from different locations inside and outside the detonation chamber. From the pressure contour plots we observe that the shock reflection from the inner wall has taken place and detonation products are expanding into the ambient airstream. The flow inside the chamber is choked due to the converging nozzle and the maximum pressure behind the shock is $\approx 8 \text{ atm}$. The pressure inside the chamber is less than 3 atm. The strong expansion of the detonation products into the ambient airstream produces a shock wave with a spherical like front rapidly decaying in strength. As a result of the interaction of the expanding detonation products with the external flow a large toroidal vortex is created. The vortex is carried away quickly from the chamber by the external flow and by its own flow momentum.

In Figure 4a we also show particle paths for the particles introduced inside the chamber and outside just above the nozzle. Examination of these trajectories allows us to follow the dynamics of the chamber evacuation and refill. In order to track the detonation products we initially place marker particles inside the chamber at three cross sections in clusters of four distributed equally normal to the detonation chamber axis. Each particle has a different color; however, particles in the same cluster have the same shade of color. At the three chosen cross sections we have designated shades of red, yellow, and blue for the particles located correspondingly at the left end, center and beginning of the nozzle cross sections of the chamber. The movement of these particles is shown by connecting them with a continuous line beginning with particle location at $t = 0$ to the present time. In Figure 4a we observe that at time $t = 0.19 \cdot 10^{-3} \text{ sec}$ all particles originally in the nozzle cross section and three of the particles originally in the mid section have left the detonation chamber. However, particles originally introduced on the inner wall of the chamber have only advanced to the nozzle region.

We use a different technique for observing the motion of the ambient gas outside the chamber. Here a

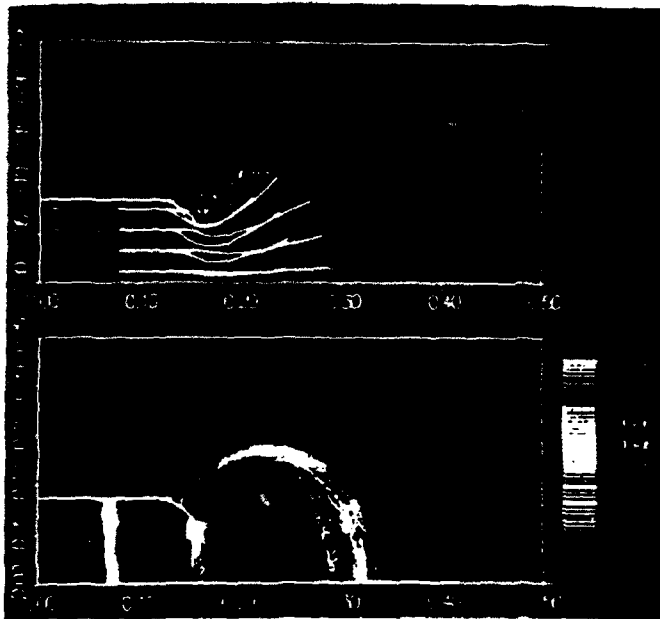
cluster of seven particles is introduced every $0.5 \cdot 10^{-4}$ seconds in the external flow above the nozzle. All such particles are traced as they move with the flow until they leave the computational domain. At any given time only the current location of the particle is displayed, and since the particles are introduced periodically with time there is a large number of particles to trace. We assign a color to every cluster of external particles to keep track of the time when they were introduced in the calculation. The colors vary from magenta for those particles introduced early in calculation, to blue for those introduced shortly at the time before the end of a detonation cycle. In Figure 4a corresponding to very early times, only one cluster of external particles is visible. This cluster was introduced at $t = 0$ and is tracking the expanding flow of the detonation products.

In Figure 4b the simulation results are shown for $t = 1.7 \cdot 10^{-3} \text{ sec}$. The pressure contours show that a shock wave develops at the external edge of the nozzle as a result of a strong expansion of the Mach 0.9 external flow. A result of overexpansion of the detonation products is that the pressure inside the detonation chamber is lower than the ambient pressure, causing the shock to be located lower on the external surface of the nozzle. The external flow about the chamber has a stagnation point on the axis of symmetry downstream at $\approx 25 \text{ cm}$. At this time as it is evident from the particle trajectories that most of the detonation products have left the chamber. Figure 4b shows one continuous trace of the particles originating at the back wall of the detonation chamber having advanced well ahead of the stagnation point in the external flow.

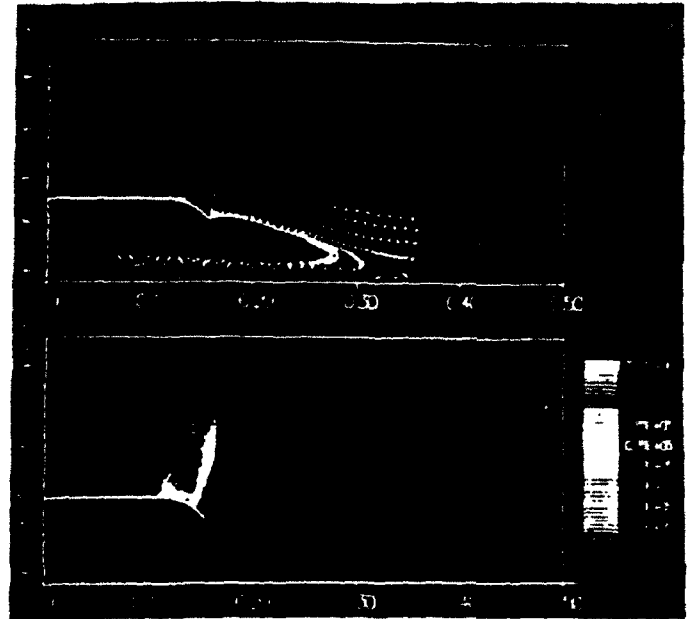
The marker particles released outside and just above the nozzles exit show two distinct flow paths. One path takes the flow past the stagnation point to the right of the detonation chamber; this flow path is marked by the four upper particle traces. Another flow path, marked by three lower particles released close to the nozzle surface is deflected towards the detonation chamber exit. Figure 4b shows this deflected stream approaching the detonation chamber nozzle. The magenta color of these particles indicates they were released at $\approx 0.5 \cdot 10^{-3} \text{ sec}$.

Figure 4c corresponds to the simulation time $t = 0.47 \cdot 10^{-3} \text{ sec}$. The pressure inside the chamber has risen $\approx 1 \text{ atm}$. Higher pressure at the chamber exit has caused the shock standing on the external surface of the nozzle to move upwards. The particles marking the movement of fresh air into the chamber show these to be well inside with some reflecting from the end wall giving a second stagnation point for the reversed fresh airflow.

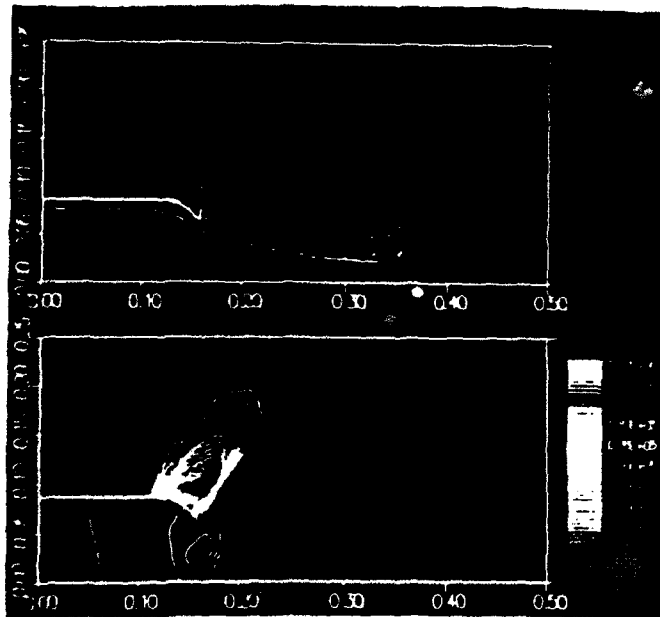
Figure 4d corresponds to the end of the first cycle when the detonation chamber should be filled with fresh charge and ready for the next detonation. In this figure the particle paths indicate that the chamber refills in a



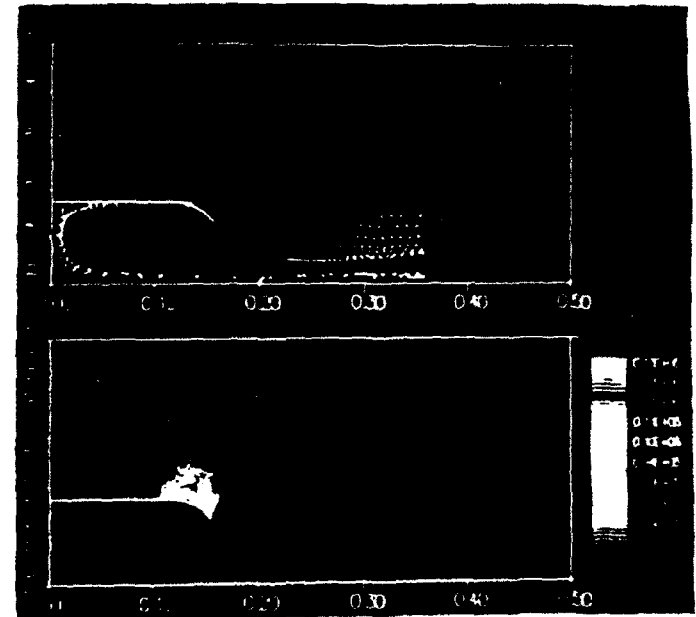
a) $t = 0.19$ msec,



c) $t = 4.7$ msec,



b) $t = 1.7$ msec,



d) $t = 7.4$ msec, end of first detonation cycle.

Figure 4 Pressure contours and particle paths for various times during the PDE simulation;
 a) $t = 0.19$ msec, b) $t = 1.7$ msec,
 c) $t = 4.7$ msec, d) $t = 7.4$ msec, end of first detonation cycle.

pattern suitable for fast mixing of the fuel-air mixture. We conjecture that fuel injection along the chamber axis will promote fast fuel-air mixing. We can see in Figure 4d that the farther injection of the external air flow inside the chamber stopped, and from that point on the mixture composition in the chamber will be fixed.

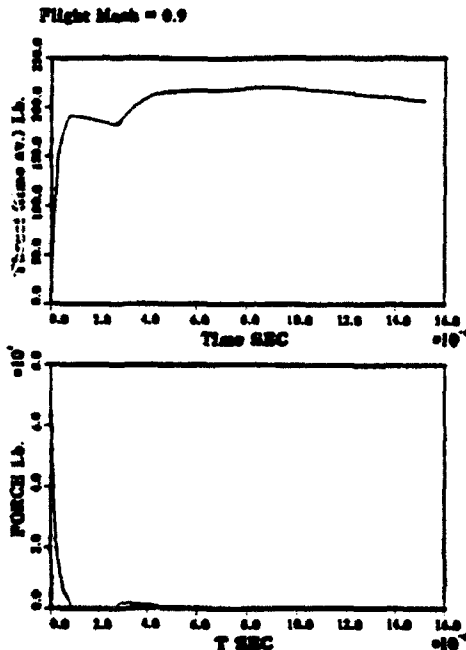


Figure 5 Thrust and force generated by PDE as function of time.

In Figure 5 total force and time averaged thrust generated by the device in the simulations discussed previously are shown as a function of time. The time averaged thrust is based on the total time for one cycle. As seen in Figure 5, initially a very large force of $\approx 7 \times 10^4$ lb is felt on the end wall of the detonation chamber. This is a result of the inwardly moving detonation wave used in our simulation. Very early during the sequence, this wave reflects from the left wall of the detonation chamber generating briefly a large force. This force rapidly decays and at $t \approx 1.0 \times 10^{-4}$ sec changes sign due to interaction of the strong shock wave with the converging nozzle. This effect is noticeable in the thrust data; the average thrust decreases somewhat after reaching levels of ≈ 200 lbs. The shock partially reflects from the converging nozzle walls and generates a wave moving to the left wall. The reflected wave thereafter generates positive thrust from $t \approx 3.0 \times 10^{-4}$ sec. Finally thrust levels reach the maximum of 225 lbs. and then decays slowly as a result of the cross sectional drag force. The simulations predict that to sustain this level of thrust will require a detonation frequency of about 150 Hz.

Conclusions

The main intent of the present study was to carry out a review of the relevant literature in the area of detonation propulsion, to assess the state-of-the-art, and to recommend future research based on our findings. We have reviewed the literature and presented our summary in first section of this paper. Our initial conclusion from the review is that there is a substantial body of evidence leading toward the possibility of producing propulsion engines with significant thrust levels based on an intermittent detonation.

Most of the historical attempts at producing thrust based on the intermittent detonation cycle were carried out with the same basic experimental setup; namely, a long straight detonation tube employing forced fuel injection at the closed tube end. We have discussed the many reasons why such a device cannot take proper advantage of the physical processes associated with detonation.

The experiments performed at the Naval Postgraduate School using a self-aspirating mode of operation for pulsed detonation thruster produced very useful results which, upon further examination, provide us with a route towards practical propulsion engines of variable thrust levels which are both controllable and scalable.

We have explored some of the implications of the possible applications of the self aspirating detonation engine concept and have developed a suitable numerical simulation code to be used as a design, analysis and evaluation tool. In fact, the preliminary analysis of a candidate detonation chamber flow properties was shown to be dominated completely by unsteady gasdynamics. An attempt to understand the flow properties based on any steady state model or one-dimensional unsteady analytical model will miss such important aspects as fuel-air mixing and, shock reflection from internal geometrical obstacle such as the converging nozzle. The unsteady simulation code developed during the course of our study is a necessary tool that we plan to use in a study leading to a feasible prototype engine design realizing the full potential of the intermittent detonation process.

Acknowledgements

The authors would express appreciation to Drs. Adam Drobot and Aharon Friedman for helpful suggestions and advice. The work reported on here was supported by DARPA under contract N66001-88-D-0088.

REFERENCES

1. Stodola, A., *Steam and Gas Turbines*, McGraw-Hill Inc., 1927.
2. Zipkin, M. A., and Lewis G. W., "Analytical and Experimental Performance of an Explosion-Cycle Combustion Chamber of a Jet Propulsion Engine," NACA TN-1702, Sept. 1948.
3. Shults-Grunow, F., "Gas-Dynamic Investigation of the Pulse-Jet Tube," NACA TM-1131, Feb. 1947.
4. Zinn, B. T., Miller, N. Carvelho, J. A., and Daniel B. R., "Pulsating Combustion of Coal in Rijke Type Combustor," 19th International Symposium on Combustion, 1197-1203, 1982.
5. Hoffmann, N., "Reaction Propulsion by Intermittent Detonative Combustion," Ministry of Supply, Volkenrode Translation, 1940.
6. Nicholls, J. A., Wilkinson, H. R. and Morrison, R. B. "Intermittent Detonation as a Thrust-Producing Mechanism," *Jet Propulsion*, 27, 534-541, 1957.
7. Dunlap, R., Brehm, R. L. and Nicholls, J. A., "A Preliminary Study of the Application of Steady State Detonative Combustion of a Reaction Engine", *ARS J.*, 28, 451-456, 1958.
8. Nicholls, J. A., Gullen, R. E. and Ragland K. W., "Feasibility Studies of a Rotating Detonation Wave Rocket Motor," *Journal of Spacecrafts and Rockets*, 3, 893-898, 1966.
9. Adamson, T. C. and Olsson, G. R., "Performance Analysis of a Rotating Detonation Wave Rocket Engine," *Astronautica Acta*, 13, 405-415, 1967.
10. Shen, P. I., and Adamson, T. C., "Theoretical Analysis of a Rotating Two-Phase Detonation in Liquid Rocket Motors," *Astronautica Acta*, 17, 715-728, 1972.
11. Krzycki, L. J., Performance Characteristics of an Intermittent Detonation Device, Navwepe Report 7655, U. S. Naval Ordnance Test Station, China Lake, California 1962.
12. Matsui, H., and Lee, J. H., "On the Measure of the Relative Detonation Hazards of Gaseous Fuel-Oxygen and Air Mixtures," Seventeenth Symposium (International) on Combustion, 1269-1280, 1978.
13. Korovin, L. N., Losev A., S. G. Ruban and Smekhov, G. D. "Combustion of Natural Gas in a Commercial Detonation Reactor," *Fiz. Gor. Vstryva*, Vol. 17, No.3, p.86, 1981.
14. Smirnov, N. N., Boichenko, A. P., "Transition from Deflagration to Detonation in Gasoline-Air Mixtures," *Fiz. Gor. Vstryva*, 22, No.2, 65-67, 1986.
15. Lobanov, D. P., Fonbershtein, E. G., Ekomasov, S. P., "Detonation of Gasoline-Air Mixtures in Small Diameter Tubes," *Fiz. Gor. Vstryva*, 12, No.3, 446, 1976.
16. Back, L. H., "Application of Blast Wave Theory to Explosive Propulsion," *Acta Astronautica*, 2, No 5/6, 391-407, 1975.
17. Varsi, G., Back, L. H., and Kim, K., "Blast Wave in a Nossle for Propulsion Applications," *Acta Astronautica*, 3, 141-156, 1976.
18. Kim, K., Varsi, G., Back and L. H., "Blast Wave Analysis for Detonation Propulsion," *AIAA Journal*, Vol. 10, Oct. 1977.
19. Back L. H., Dowler, W. L. and Varsi, G., "Detonation Propulsion Experiments and Theory," *AIAA Journal* Vol. 21 Oct. 1983.
20. Eidelman, S., Shreeve, R. P., "Numerical Modeling of the Nonsteady Thrust Produced by Intermittent Pressure Rise in a Diverging Channel," *ASME FED-Vol. 18, Multi-Dimensional Fluid Transient*, p.77, 1984.
21. Eidelman, S., "Rotary Detonation Engine," U.S. Patent 4 741 154, 1988.
22. Helman, D., Shreeve, R. P., and Eidelman, S., "Detonation Pulse Engine," *AIAA-86-1683*, 24th Joint Propulsion Conference, Huntsville, 1986.
23. Monks, S. A., "Preliminary Assessment of a Rotary Detonation Engine Concept," MSc Thesis, Naval Postgraduate School, Monterey, California, Sept. 1983.
24. Camblier, T. L. and Adelman, N. G., "Preliminary Numerical Simulations of a Pulsed Detonation Wave Engine," *AIAA-88-2960*, *AIAA 29th Joint Propulsion Conference*, Boston 1988.
25. R. Lohner, K. Morgan, and D.C. Zienkiewicz, "Finite Element Methods for High Speed Flows," *AIAA 7th Computational Fluid Dynamics Conference*, Cincinnati, Ohio, *AIAA Paper 85-1531* (1985).
26. R. Lohner and K. Morgan, "Improved Adaptive Refinement Strategies for Finite Element Aerodynamic Computations," *AIAA 29th Aerospace Sciences Meeting*, Reno, Nevada, *AIAA Paper 86-0499* (1986).
27. T.J. Barth and D.C. Jespersen, "The Design and Application of Upwind Schemes on Unstructured Meshes," *27th Aerospace Sciences Meeting*, *AIAA-89-0366*, Reno, Nevada.
28. S. Eidelman, P. Collela, and R.P. Shreeve, "Application of the Godunov Method and It's Second Order Extension to Cascade Flow Modeling," *AIAA Journal*, v. 22, 10, 1984.



AIAA-90-0460

**COMPUTATIONAL ANALYSIS OF
PULSED DETONATION ENGINES
AND APPLICATIONS**

S. EIDELMAN, W. GROSSMANN
AND I. LOTTATI
SCIENCE APPLICATIONS
INTERNATIONAL CORPORATION
McLEAN, VA

28th Aerospace Sciences Meeting

January 8-11, 1990/Reno, Nevada

COMPUTATIONAL ANALYSIS OF PULSED DETONATION ENGINES AND APPLICATIONS

S. Eidelman, W. Grossmann and I. Lottati
Applied Physics Operation
Science Applications International Corporation
1710 Goodridge Drive
McLean, Virginia 22102

1. Introduction

This paper presents the results of a computational fluid dynamic simulation/ parameter study of the SAIC Pulsed Detonation Engine (PDE) concept. Results from computer simulations of generic PDE geometries over a wide range of subsonic and supersonic flight Mach numbers indicate that potentially practical detonation engines can now be conceptualized and optimized for specific flight requirements and missions. Specifically, the study shows that primary propulsion for aerodynamic vehicles of the PENAID variety may be possible at Mach numbers $0.5 < M < 0.8$, thrust levels on the order of 100 pounds and a specific fuel consumption of the order of 1 lb./lb.hr.). The predicted performance places the PDE propulsion concept in a strongly competitive position compared with present day small turbojets. The PDE concept has the added attractiveness of rapid variable thrust control, no moving parts and the potential for low cost manufacturing. Finally, the PDE concept is scalable over a wide range of engine sizes and thrust levels. For example, it is theoretically possible to produce PDE engines on the order of one to several inches in diameter and thrusts on the order of pounds, as well as devices which provide thousands of pounds thrust.

A literature search of past research on related concepts and devices uncovered important information which proved useful in pursuing our present study. A review of the literature¹⁻²⁴ reveals that a significant body of experimental and theoretical research exists in the area of unsteady propulsion. However, this research was not sufficiently extensive to provide a conclusive quantitative comparison between impulsive engine concepts and steady state concepts. In addition, the computational and analytical techniques were not sufficiently developed in the past to treat the inherently unsteady flows of pulsed engines. A new generation of analytical and computational tools exists today, allowing us to revisit and analyze these devices with a high degree of confidence.

Our paper is organized into the following sections: 2) description of the basic phenomenology of the air-breathing Pulsed Detonation Engine concept; 3) discus-

sion of the numerical simulation results; and 4) conclusions. Details of the mathematical formulation of the simulation and a discussion of the numerical code used in the present study are given elsewhere.^{25,26}

2. The Generic Pulsed Detonation Engine

A detonation process, due to the very high rate of reaction, leads to a propulsion concept in which the constant volume process can be fully realized. In detonative combustion, the strong shock wave, which is part of the detonation wave, acts like a valve between the detonation products and fresh charge. The speed of the detonation wave is about two orders of magnitude higher than the speed of a typical deflagration. This allows the design of propulsion engines with a very high power density. Each detonation has to be initiated separately by a fully controlled ignition device, with a wide range of variable cycle frequencies. There is no theoretical restriction on the range of operating frequencies; they are uncoupled from acoustical chamber resonancies. This is very important feature of the constant volume detonation process that differentiates it from the process occurring in a pulse-jet;³⁻⁴ the pulse jet cycle is tuned to the acoustical resonances of the combustion chamber. This leads to a lack of scalability for the pulse jet concept.

A physical restriction dictating the range of detonation frequency arises from the rate at which the fuel/air mixture can be introduced into the detonation chamber. This also means that a device based on a detonative combustion cycle can be scaled and its operating parameters can be modified for a range of required output conditions. There have been numerous attempts to take advantage of detonative combustion for engine applications. The most recent and successful of these attempts was carried out at the Naval Postgraduate School (NPS) by Helman et al.²² During this study, several fundamentally new elements were introduced to the concept distinguishing the NPS research device from previous studies. First, it is important to note that the NPS experimental apparatus was the first successful self aspirating air breathing detonation device. Intermittent detonation frequencies of 25 Hz were obtained. This frequency

was in phase with the fuel mixture injection through timed fuel valve opening and spark ignition. The feasibility of intermittent injection was established. Pressure measurements showed conclusively that a detonation process occurred at the frequency chosen for fuel injection. Further, self aspiration was shown to be effective. Finally, the effectiveness of a primary detonation as a driver for the main detonation was clearly demonstrated. Although the NPS studies were abbreviated, many of the technical issues considered to be essential for efficient intermittent detonation propulsion were addressed with positive results.

As a result of the survey of past research on intermittent detonation devices, we have focussed our attention on the NPS experiments of Helman et al.²² The remainder of this paper is concerned with a computer simulation of performance characteristics of such a device. We have chosen a generic geometry, applicable to certain present day vehicle and mission requirements, and have parametrically varied key features which affect performance and assessed the effects of these variations.

The generic device we consider here is a small engine 15 cm long and 15 cm in diameter. Figure 1 shows a schematic of the basic detonation chamber attached to the aft end of a generic aerodynamic vehicle. The combustible gas mixture is injected at the closed end of the detonation chamber and a detonation wave propagates through the mixture. The size of the engine suggests a small payload or aerodynamic vehicle, but the concept can be extended to larger payloads simply by scaling up the size of the detonation chamber and possibly combining a number of chambers into one larger engine. As an example, a PENAIID vehicle has been conceptualized requiring an engine with a diameter of roughly 15 cm and a useful continuous thrust at Mach 0.8 approximately 60-90 pounds. Such an engine should have a specific fuel consumption in the range of 1.7 to 1.9 lb. fuel/hr per pound and an endurance on the order of 10-30 minutes. These specifications are met by present day small turbojets. Hence, in order to be competitive, a PDE must at least meet these requirements. Should this prove to be the case, a PDE with no moving parts would be a very attractive engine from the point of view of performance, ease of manufacturing and cost.

A key issue in the pulsed detonation engine concept is the design of the main detonation chamber. The detonation chamber geometry determines the propulsion efficiency and the duration of the cycle (frequency of detonations). Since the fresh charge for the generic engine is supplied from the external flow field, the efficiency of the engine depends on the interaction of the surrounding flow with the internal flow dynamics. The range of the physical process requiring simulation in order to model the complex flow phenomena associated with the deto-

nation engine performance is very broad. A partial list is:

1. Initiation and propagation of the detonation wave inside the chamber,
2. Expansion of the detonation products from the chamber into the air stream around the chamber at flight Mach numbers,
3. Fresh air intake from the surrounding air into the chamber.
4. The flow pattern inside the chamber during post-exhaust pressure buildup which determines the strategy for mixing the next detonation charge,
5. Strong mutual interaction between the flow inside the chamber and surrounding the engine.

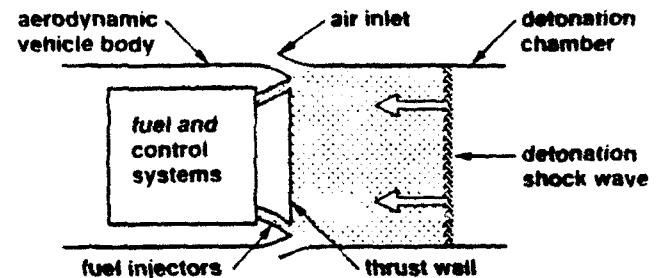


Figure 1. Schematic of the generic PDE showing detonation chamber, inlet, detonation wave, fuel injectors and position relative to an aerodynamic vehicle.

All of these processes are interdependent, and interaction and timing are crucial to engine efficiency. Thus, unlike simulations of steady state engines, the phenomena described above can not be evaluated independently.

The need to resolve the flow regime inside the chamber accounting for nozzles, air inlets etc., and at the same time resolve the flow outside and surrounding the engine, where the flow regime varies from high subsonic, locally transonic and supersonic, makes it a challenging computational problem.

The single most important issue is to determine the timing of the air intake for the fresh charge leading to repetitive detonations. It is sufficient to assume inviscid flow for the purpose of simulating the expansion of the detonation products and fresh air intake. The assumption of inviscid flow makes the task of numerically simulating the PDE flow phenomena somewhat easier than if a fully viscous flow model were employed. For the size of the generic device studied in this work the effects of viscous boundary layers are negligible with the exception of possible boundary layer effects on the valve and inlet geometries discussed subsequently. Boundary layer effects on the present results are discussed later.

3. Results of Simulational Parameter Study

As mentioned, and as shown in Figure 1, we have chosen a small ≈ 15 cm long and ≈ 15 cm internal diameter cylindrical chamber as the basic device. In the figure, the detonation chamber including detonation wave and inlet valves are shown schematically. The PDE, as envisioned in the present study, would most naturally be applicable for a class of aerodynamic vehicles such as target drones and PENAID missiles among others. The exact details of this basic chamber geometry were modified during the course of the study in order to obtain the required aerodynamic or propulsion effects; however, these modifications did not significantly change the total internal volume of the chamber. Thus, the performance results for the different cases can be compared holding the chamber volume constant. The schematic shown in Figure 1 does not represent an optimum configuration and is given here mainly to illustrate our methodology. We consider a situation where the PDE serves as the main thruster for an aerodynamic vehicle traveling in air with Mach numbers between $M = 0.2$ and $M = 5.0$, and is located at the aft end of the vehicle. The main objectives of our study are:

1. To calculate the thrust produced during each cycle and the integrated thrust as a function of time,
2. To find the maximum cycle frequency. This is determined by the time required from detonation to the final exhaust of combustion products and intake of fresh charge for the next detonation,
3. To evaluate the parametric dependence of the thrust and detonation frequency on the flight Mach number and detonation chamber geometry details.

In addition to the technical objectives outlined above, we have set another goal for our study. We require that the best, but by no means optimum, configuration produce a minimum of 60 pounds thrust at an operating frequency of 140 cycles per second. The definition of best is that configuration which satisfies the technical objectives outlined above and meets the operational goal of 60 pounds thrust and 140 Hz frequency over the flight regimes from $M=0.2$ to $M=0.9$. (The Mach number range corresponds to that of a PENAID missile mission profile.)

To achieve these objectives we have conducted a comprehensive parametric simulational study of the PDE performance. We have studied PDE engine performance for a range of Mach numbers with two separate initial detonation locations in the chamber and for various geometry modifications. In addition to the range of subsonic Mach numbers we have examined PDE performance in the supersonic regime, $2 < M < 5$. The geometry modifications included converging exhaust nozzles, inlets and dynamic valves. A computer simulation code was developed and optimized for a Stellar graphics work-

station to carry out the analysis. In addition, a particle-tracing package was developed and implemented in the code. This allowed us to analyze the flow pattern inside and outside the detonation chamber, the main sources creating this pattern as a function of time, and the composition of the resulting gas mixture (air/detonation products).

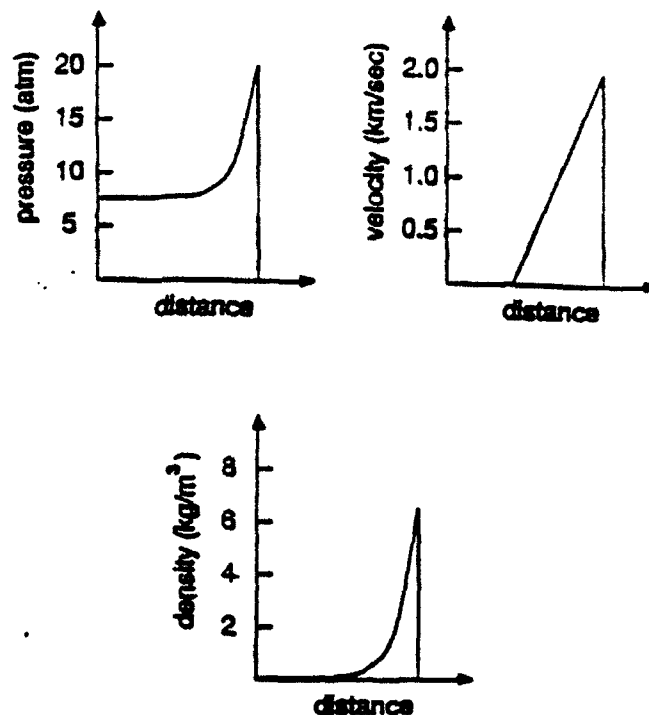


Figure 2. Distribution of gasdynamic parameters behind the detonation wave according to a 1-D self-similar solution.

First we will describe in detail the results for a typical simulation, Case 1, and illustrate the main features of our analysis.

Case 1. The simulation begins at $t = 0$ when a strong detonation wave is initiated inside the detonation chamber. The detonation chamber for this case includes a simple annular inlet which remains open during operation. The external freestream Mach number is 0.8. The specific fuel chosen for the present simulations is ethylene. The chemical reaction occurring in the ethylene/air detonation process is given by:



The detonability limits of ethylene in air range from 4 to 12% by volume and depend somewhat on temperature and pressure. We assume for the sake of simplicity that

the fuel/air ratio is 6% by volume. Because the detonation initiation and propagation (detonative combustion) takes place several orders of magnitude faster than any of the other flow processes in or surrounding the device, finite rate chemistry is not included in the simulations. Instead the equation of state for the flow in the chamber immediately after detonative combustion was adjusted to represent the correct physical state of the combustion products. Initially the detonation wave travels from the closed end of the chamber towards the open aft end with a maximum velocity of 1800 m/sec and maximum pressure of $20 \times 10^5 Pa$. The initiation of the main detonation wave is assumed to take place via a device proposed and successfully implemented by Helman, *et al.*;²² namely, a primary detonation is established in a small tube containing an oxygen rich mixture. This mixture requires a low initiation energy but will sustain a detonation which, in turn, is used to trigger the main detonation wave. We do not model the detonation tube; but, we assume that such a device is present to trigger the main detonation at $t = 0$. The distribution of pressure, velocity, and density of the detonation wave is defined through the selfsimilar solution for a planar detonation wave. A schematic of the detonation wave distribution in space in Figure 2 shows pressure, temperature and velocity as a function of the spatial extent of the detonation wave.

In Figure 3a simulation results are shown at time $t = 0.64 \text{ m/sec}$ in the form of pressure contours and particle paths from different locations inside and outside the detonation chamber. The free stream Mach Number is 0.8. From the pressure contour plots we observe that the detonation shock wave has left the chamber and is freely expanding outwardly in the external flow. The strong expansion of the detonation products into the ambient airstream produces a shock wave with a spherical-like front that rapidly decays in strength away from the source. A large toroidal vortex is created as a result of the interaction of the expanding detonation products with the external flow. The vortex is carried away quickly from the chamber by the external flow and by its own momentum. At the time shown in Figure 3a, the detonation products are almost fully expanded into the ambient air and the maximum pressure at the front of the shock wave is 1.2 atm. As a result of this expansion the detonation products inside the detonation chamber are overexpanded and their pressure is 0.45 atm.

In the upper frame of Figure 3a we show particle paths for the marker particles introduced inside the chamber and outside just above the nozzle exit. Examination of these trajectories allows us to follow the dynamics of the chamber evacuation and subsequent refill. In order to track the detonation products we initially place marker particles inside the chamber at three separate cross sections in clusters of four. Each particle

has a different color; however, particles in the same cluster have the same shade of color. At the three chosen cross sections we have designated shades of red, yellow, and blue for the particles located correspondingly at the left chamber end, center and aft end of the nozzle cross section. The movement of these particles is shown by connecting them with a continuous line beginning with particle location at $t = 0$ to the present time. In Figure 3a we observe that at time $t = 0.64 \times 10^{-3} \text{ sec}$ all particles originally in the nozzle cross section (the cross section at the aft end) and three of the particles originally in the mid section have left the detonation chamber. However, particles originally introduced at the inner end wall of the chamber (red traces) have only advanced to the nozzle region.

We use a different particle technique for observing the motion of the ambient gas outside the chamber. Here a cluster of seven particles is introduced every 0.5×10^{-4} seconds in the external flow above the nozzle. All such particles are traced as they move with the flow until they leave the computational domain. At any given time only the current location of the particle is displayed, and since the particles are introduced periodically with time, there are many particles to trace. We assign a color to every cluster of external particles to keep track of the time when they were introduced in the calculation. The colors vary from magenta for those particles introduced early in the calculation, to blue for those introduced near the end of a detonation cycle. In Figure 3a, which corresponds to early times, only 12 clusters of external particles are visible. These clusters were introduced from $t = 0$ to 0.6×10^{-3} second, vary from magenta to red in color, and are tracking the expanding flow of the detonation products.

In Figure 3b the simulation results for the same case are shown for $t = 1.4 \times 10^{-3} \text{ sec}$. The pressure contours show that a strong stagnation point develops on the axis of symmetry downstream at $\approx 25 \text{ cm}$ as a result of a strong expansion of the Mach 0.8 external flow around the engine. At this time it is evident from the particle trajectories that most of the detonation products have left the chamber. Figure 3b also shows that only traces of the particles originating at the back wall of the detonation chamber are left in the computational domain. These particles advanced to the aft end of the chamber and then following the contraction of the over expanded detonation products, reversed their flow direction. The pressure contour plots in Figure 3b show the formation of an additional stagnation point at the closed end wall of the detonation chamber resulting from the inverse flow of the detonation products. The average pressure in the chamber is below ambient and is $\approx 0.55 \text{ atm}$.

The marker particles released outside and just above the nozzle exit show two distinct flow paths. One path

takes the flow past the stagnation point to the far right of the detonation chamber; this flow path is marked by the four upper particle traces. Another flow path, marked by three lower particles released close to the external wall of the chamber, are deflected from the stagnation region towards the detonation chamber exit. The magenta color of these particles indicates they were released at $t \approx 0.6 \cdot 10^{-3}$ sec.

Figure 3c corresponds to the simulation time $t = 2.2 \cdot 10^{-3}$ sec. The pressure inside the chamber has risen to ≈ 0.8 atm. The stagnation region at the closed end of the detonation chamber continues to develop and has produced a compression wave moving toward the open end of the chamber. The particles marking the movement of fresh air into the chamber show these to be well inside the chamber, with some reflecting from the end wall and contributing to the pressure at the second stagnation point. The circular motion of a few of the detonation products particles (red solid lines), indicates that the detonation products which did not expand with the first shock wave are now "trapped" inside the detonation chamber.

Figures 3d and 3e correspond to the end of the first cycle when the detonation chamber should be filled with fresh charge and ready for the next detonation. However, these figures indicate that the fresh air refill is not totally satisfactory for this chamber configuration. The marker particle paths indicate that the chamber refill is incomplete and at a time of $t = 4.7 \cdot 10^{-3}$ sec the refill process has essentially stopped. As a result, only about a third of the detonation chamber volume has enough fresh air for the next detonation cycle.

In Figure 4 the total force and time averaged thrust generated by the device in the simulations just discussed, are shown as a function of time. The time averaged thrust is based on the total time for one cycle defined as $7.0 \cdot 10^{-3}$ sec. This time is equivalent to a detonation frequency of 140 Hz. As seen in the figure, initially a very large force of $\approx 3.2 \cdot 10^3$ lb is felt on the end wall of the detonation chamber. This force is a result of the high pressure behind the detonation wave. It rapidly decays and at $t \approx 0.5 \cdot 10^{-3}$ sec changes sign due to over expansion and dynamic pressure of the external flow. This effect is noticeable in the thrust data; the average thrust increases rapidly but decreases after reaching levels of ≈ 55 lbs. At the end of the simulation the thrust is actually negative ≈ -20 lbs. The average thrust for one cycle in this case will be ≈ 10 lbs.

The simulation just described has served to illustrate the information generated with the numerical simulations. For the remaining simulations, emphasis was placed on determining the effects of propagation direction of the main detonation wave, effects of inlet and valve geometry, detonation chamber geometry and Mach

number. Many of the simulations produced unsatisfactory results from the point of view of ineffective fresh air refill and hence either not enough fresh charge for repetitive detonations or too slow a refill resulting in low detonation frequency. We give below examples of successful simulations at Mach 0.8, Case 2 and Mach 2, Case 3.

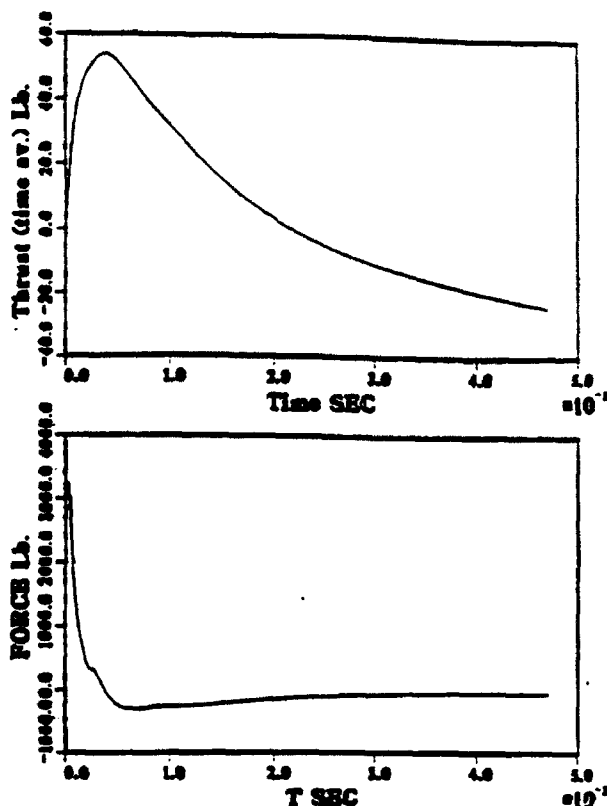


Figure 4. Time averaged thrust and force data from simulation of Case 1.

Case 2. The results from all simulations show that, irrespective of the inlet geometry, but with a straight nozzle and initial detonation position at the nozzle exit plane, sufficient thrust levels can be produced. A remaining problem in view of the objectives is to demonstrate that enough fresh air can be injected into the chamber to produce the required conditions for intermittent detonation at a frequency of 140 Hz. To accomplish this, we have considered a contoured inlet in the periphery of the end wall of the detonation chamber. The details of this inlet geometry and the computational grid are shown in Figure 5.

For the initial tests with this inlet no attempt was made to optimize the inlet geometry for a given flow regime. Figures 6a-f present results for the simulation of the chamber geometry shown in Figure 5.

The flight Mach number in this case is 0.8. The initial detonation wave is launched inwards and its energy parameters are the same as in all previous cases. In Figure 6a we see two distinct shock waves expanding into ambient air: one generated by expansion from the aft of the chamber and another produced by the expansion through the inlet. We also notice some particles tracing the motion of the detonation products flowing out through the inlet. In Figure 6b, at time $t = 0.7 \times 10^{-3}$ sec., fresh air is noted entering the chamber through the inlet. At this time the dominant pressure in the chamber is 0.77 atm. Figure 6c shows that at the time $t = 1.4 \times 10^{-3}$ sec, 3/4 of the detonation chamber is filled with fresh air. The strong air jet entering the chamber impinges the axis of symmetry, creating two large vortices which rotate in opposite directions. Such vortical motion would promote effective fuel-air mixing in the chamber. In Figure 6d, $t = 2.4 \times 10^{-3}$ sec., the fresh air stream begins to exit the chamber. At this point the mixture inside the chamber has achieved the required conditions for the next detonation. This result translates to a sustained detonation frequency of ≈ 400 Hz. In Figures 6e-f we follow the later evolution of the flow pattern inside and outside the chamber. We observe strong air flow through the inlet with a strong recirculation pattern, which will assure fuel air mixing even if the fuel is injected into the chamber with a delay to sustain intermittent detonation at a lower frequency. In Figure 7 thrust and force simulated for the last case are shown as a function of time. First we notice that the maximum thrust for this case is ≈ 70 lbs., somewhat lower than for the cases with a very simple annular inlet and completely flat end walls.

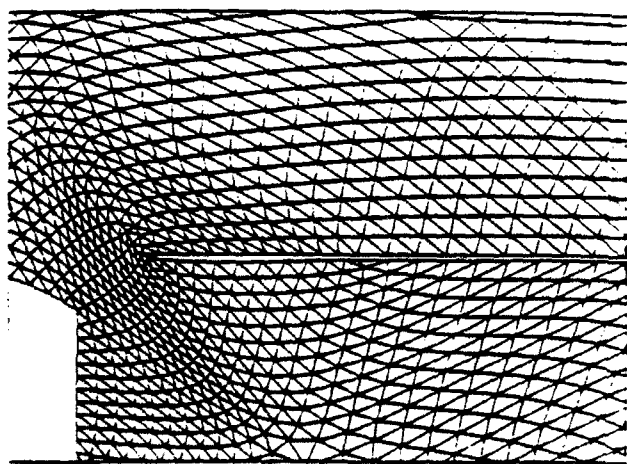


Figure 5. Computational grid for the inlet geometry used in simulation of Case 2.

This results from a reduction of the area normal to the propagation direction of the detonation wave due to the inlet geometry. It is surprising that the case with the inlet results in a reduction of the average thrust as a function of time that is almost the same as for a case without inlet at the same Mach number (≈ 90 lb reduction without the inlet and 100 lb reduction with the inlet). This strongly indicates that the generic inlet we have just considered will not contribute significantly to the drag produced by the chamber dynamics and interaction with the ambient flow. The cycle average thrust generated by the PDE based on 150 Hz operation frequency in this case is ≈ 100 lbs. This value is somewhat larger than the thrust targeted for this study.

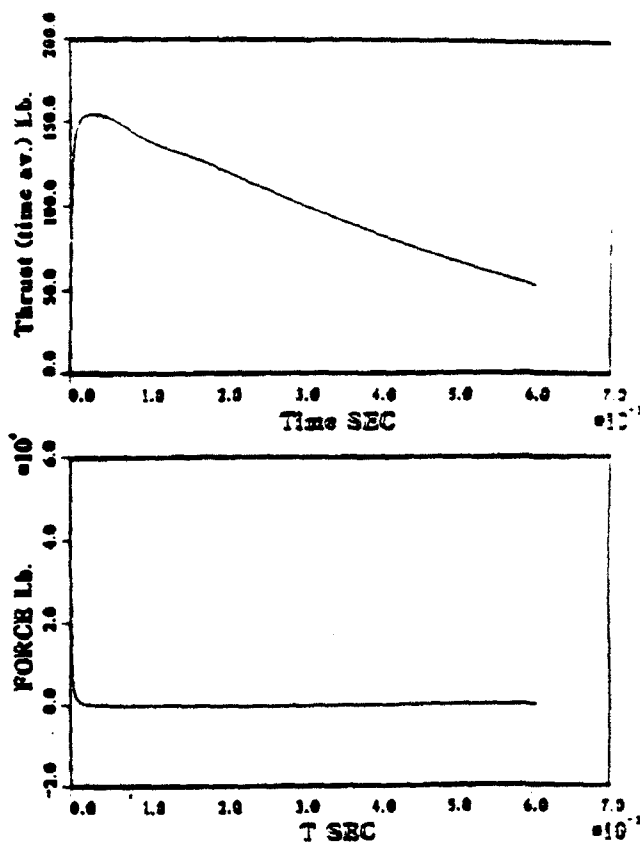


Figure 7. Time averaged thrust and force data from simulation of Case 2.

Case 3. Results for a Mach 2.0 simulation of the same geometry as in the previous simulation; but with a more geometrically complex inlet are shown in Figures 8a-b. The inlet geometry for this case was determined from near choked flow conditions in the throat region of the inlet. In addition to the pressure contours and particle paths, in this case we also show velocity vectors.

We observe in these figures that the detonation chamber is quickly filled with fresh air at the time $t = 1.3 \times 10^{-3}$ sec., which corresponds to a detonation frequency of 700 Hz. In practice this high frequency will be difficult to realize, because of the mixing and initiation problems. In Figure 9 we show thrust and force results for this simulation. We observe that after 3.0×10^{-3} sec. the net average thrust is still 50 lbs.

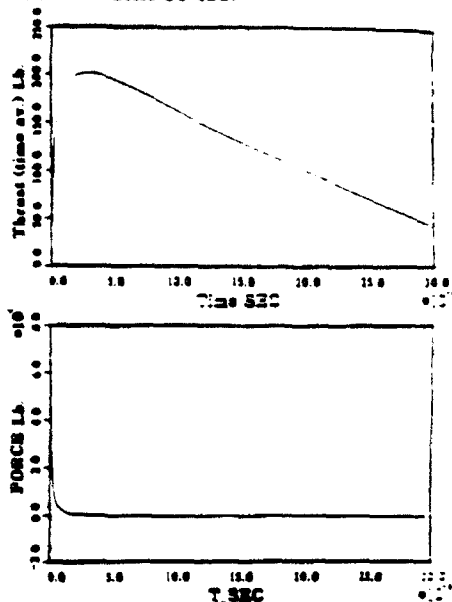


Figure 9. Time averaged thrust and force data from simulation of Case 3, 140 Hz detonation frequency.

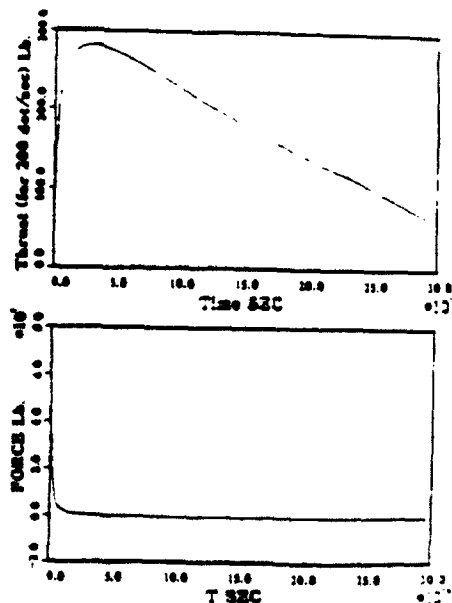


Figure 10. Time averaged thrust and force data from simulation of Case 3, 200 Hz detonation frequency.

The cycle averaged thrust based on 140 Hz detonation frequency, for this simulation is ≈ 70 lbs. However, as

previously mentioned the fresh air refill time allows a much higher frequency of detonations. Figure 10 shows the same results as Figure 9, but calculated for a 200 Hz cycle frequency. In this case the maximum average thrust is ≈ 280 lbs. and the net cycle averaged thrust is ≈ 100 lbs. This result indicates the promising potential of the PDE concept for supersonic propulsion.

4. Conclusions

In this section we present our conclusions reached after carrying out a review of past research on detonative propulsion and a detailed numerical simulation of a generic pulsed detonation engine (PDE) device. The primary conclusion is that the PDE shows promising potential in providing primary propulsion for a range of present day aerodynamic vehicles such as target drones, PENNAID missiles and other smart missiles that require loitering and throttling capability. The operating flight regimes of such a propulsion engine may extend from the low subsonic to supersonic regimes.

Most of the past attempts at producing thrust based on an intermittent detonation cycle were carried out with the same basic experimental set-up; namely, a long straight detonation tube employing forced fuel injection at the closed tube end. We have pointed out the reasons²⁵ why such a device cannot take proper advantage of the physical processes associated with detonative combustion. We have also indicated that, because of the conclusions reached during experiments with such devices, the development of intermittent detonative propulsion was adversely prejudiced and stalled at an early stage.

The experiments performed at the Naval Postgraduate School based on a self-aspirating mode of operation for a pulsed detonation thruster produced very useful results which, upon further examination, provide us with a route towards practical detonation engines of variable thrust levels that are both controllable and scalable. A generic PDE device based on the NPS experiments was conceptualized and served as the basic model for a comprehensive series of numerical simulations. The goal of the simulations was to understand the parametric dependence of the PDE device variables on propulsion performance such as thrust and detonation cycle frequency.

The principle conclusions drawn from the simulation results are as follows. First, the target thrust and cycle frequency of 60-90 pounds and 140 Hz, respectively, have been realized in the simulations. These target values were dictated by knowledge of present day requirements for planned aerodynamic vehicles such as PENNAID devices. Before proceeding, it is appropriate to mention again that the performance of the PDE device is governed entirely by unsteady flow processes. None of the wave averaging effects which had been predicted

by previous studies were found and, it was shown dramatically that the internal (detonation, expansion, refill and mixing) flow processes are directly coupled to the external (shock formation, stagnation point formation, vortex shedding, etc.) flow processes. These two flows must be simultaneously analyzed if a reliable estimate of performance is to be determined. The present study is the first fully unsteady computational analysis of an intermittent detonation scheme with realistic geometry and external flow computed self-consistently.

The simulations further showed that the best thrust performance was realized when the full kinetic energy of the detonation wave was captured on the thrust surface (the closed end wall of the detonation chamber). This indicates that the detonation initiation must be controlled; the ignition must take place in the vicinity of the exit plane of the chamber resulting in initial propagation of the wave towards the chamber wall. The magnitude of the total and time averaged thrust is a strong function of the strength of the wave, the cross-sectional area of the end wall normal to the wave direction, and a weak function of the specific geometrical details of such variables as valve or inlet shape. The simulations also showed that for most situations involving simple inlets (flat cylindrically symmetric openings in the chamber external wall) the thrust data was independent of whether the valve intermittently opens or remains open during the full cycle. This leads to the possibility of a permanently open valve and a no moving parts manifestation of a PDE device. The thrust data indicates a dependence on the external flight conditions, e.g. Mach number. The Mach number plays a role in the wave drag that the geometry of the PDE will incur; the details of the valve and inlet configurations figure prominently in the total wave drag.

On the other hand the simulations showed that the timing of the fresh air refilling required to recharge the chamber for subsequent detonations is a strong function of the details of the valve and inlet geometry, the expansion of the combustion products, the resulting over-expansion of the chamber flow and, the external flow regime and interaction of the external flow with the internal flow. For subsonic flight, Mach 0.2-0.9, the fresh air entering the chamber comes from two separate principal flow processes; one comes from the flow through any valve or inlet and the other comes from the self-aspiration or reverse flow from the aft end of the chamber due to strong over-expansion. All these processes are interdependent, as reported in Section 3, and, in order to search for a given performance in a given device requires variation of many parameters. The simulation results obtained to date provide an understanding of the effects caused by variation of the above mentioned parameters, and with the information available we are able to conclude that a PDE propulsion unit can be optimized

(although no optimization studies were carried out) for a given flight regime. In order to find an optimum configuration satisfying given performance over a wide flight regime a more extensive simulation study will be required. It was mentioned earlier that the simulations presented here were carried out under the assumption of inviscid flow; boundary layer effects were not included. The addition of boundary layers to the PDE engine inlets and valves, the only components where boundary layers will be significant, will lead to increased performance. Roughly the same amount of fresh air will flow into the over expanded detonation chamber but at a somewhat slower rate and in a pattern that will promote enhanced circulation and hence fuel/air mixing. We return to the issue of optimization below.

We give now results from sample performance calculations of the application of the PDE device to proposed aerodynamic vehicles such as a PENAID missile based on the results from our simulations. These predictions are based on point design data for an inlet geometry which has not been optimized. We believe that increased performance can be found through a systematic optimization of the PDE device characteristics. First we consider the Mach 0.8 case and the inlet described in Case 2.

The maximum operation frequency for the device is 400 Hz. The following performance is a consequence of the simulation data:

For a frequency of 100 Hz.:

Thrust79 lb.
Fuel flow rate025 lb./sec.
Fuel weight for 12 min.18 lb.
Oxygen weight.	1.8 lb.
Fuel for detonation tube.06 lb.
Total oxygen and fuel weight.	20.4 lb.
Total engine weight	30.2 lb.
Specific fuel consumption.	1.14 lb./(lb.*hr.)

Assuming the PDE device geometry is kept fixed, a higher detonation frequency will result in a linear increase in thrust and fuel flow rate at the same specific fuel consumption. For example, if the detonation frequency is increased to 200 Hz., the performance data are:

Thrust	1.57 lb.
Fuel flow rate05 lb./sec.
Fuel weight for 12 min36 lb.
Oxygen weight	3.6 lb.
Fuel for detonation tube.	1.2 lb.
Total oxygen and fuel weight	40.8 lb.
Total engine weight.	54.4 lb.
Specific fuel consumption	1.14 lb./(lb.*hr.)

At lower Mach numbers, $M=0.5$, the maximum operating frequencies will be lower since the external dynamic pressure responsible for supplying fresh air to the chamber is also lower. For the device under considera-

tion here the maximum frequency is 250 Hz.

For a frequency of 100 Hz.:

Thrust	100 lb.
Fuel flow rate	0.025 lb./sec.
Fuel weight for 12 min18 lb.
Oxygen weight.	1.8 lb.
Fuel for detonation tube.06 lb.
Total oxygen and fuel weight	20.4 lb.
Total engine weight.	30.2 lb.
Specific fuel consumption.	0.9 lb./(lb.*hr.)

Again, if the frequency is increased the thrust will increase linearly; operation at 200 Hz. yields:

Thrust	200 lb.
Fuel flow rate	0.05 lb./sec.
Fuel weight for 12 min36 lb.
Oxygen weight	3.6 lb.
Fuel for detonation tube	1.2 lb.
Total oxygen and fuel weight.	40.8 lb.
Total engine weight.	54.2 lb.
Specific fuel consumption	0.9 lb./(lb.*hr.)

The examples of performance of PDE devices given above are based on point design conditions arising from the simulations discussed in Section 3 of this report. They cannot be extrapolated with any degree of reliability to other conditions or configurations. We conclude however, that the performance computed for the indicated device is encouraging from the point of view of thrust, thrust control, simplicity of the device (no moving parts) and specific fuel consumption (SFC). The specific fuel consumption computed above is competitive with present day small turbojet engines. The SFC for a PDE could be significantly lower than for small turbojets (SFC's for small turbojets are in the range of 1.8-2.0 lb./(lb.*hr.)). Thus, for a given mission and vehicle, a PDE propulsion unit would be more fuel efficient resulting in increased range. Moreover, if the expected thrust control in PDE's is realizable, it may be possible to produce propulsion units that can slow down, loiter and maneuver and finally accelerate to full thrust again rapidly.

A final conclusion can be made concerning the application of PDE's to supersonic vehicles. As shown in the simulations the ability to refill the detonation chamber with fresh air charge is a very strong function of valve and inlet geometry. Refilling may also be somewhat enhanced by the self-aspiration effect, but; to a much less extent than in the subsonic case. The example of supersonic operation discussed in Section 3 shows that care must be taken in design of the inlet or valve configuration. The flow in the chamber must allow for refill and fuel/air mixing. More than likely choked flow conditions will be required at the inlet entrance to the chamber. This could lead to complications in the design

of a PDE with simple geometry; choked flow conditions are a function of the external Mach number and a fixed inlet will be optimal only for a small range of the operating envelope. On the other hand, if a given vehicle is to fly at supersonic speeds and is launched at supersonic speeds, this problem may not appear. Further, if the given vehicle is launched at subsonic speeds and a booster is used to bring it up to the required supersonic operating speed, the problem may again not appear. We conclude that the PDE has potential for the supersonic flight regime and it is not excluded that a configuration can be found which will operate over the flight regimes $0.2 < \text{Mach number} < 3$ in a fuel efficient manner.

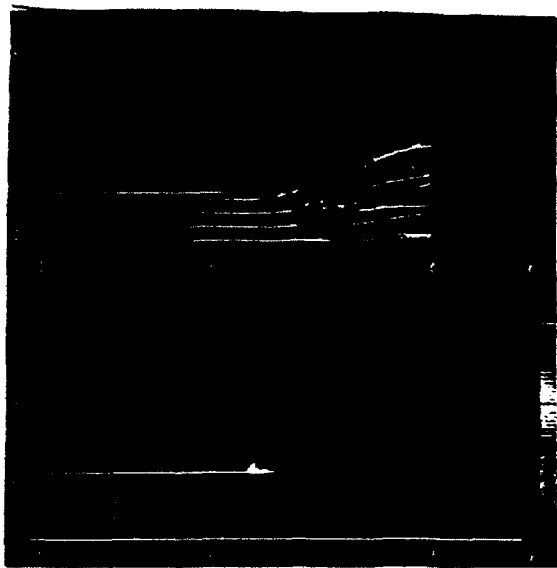
Finally it is appropriate to speculate that the PDE concept is a candidate for a hybrid propulsion device. Consider the following scenario. At low altitudes, up to 30-50 km, and at speeds ranging from low supersonic to hypersonic ($2 < \text{Mach number} < 10$) an air breathing engine can operate. Above these conditions air breathing is not effective and rocket propulsion is required. A PDE can operate in an air breathing mode as long as the external conditions allow it, and when no longer possible, the detonation chamber may be considered a rocket chamber in which detonation occurs with the fuel and oxygen supplied from on-board storage. Similar considerations have been made for NASP propulsion; serious penalties are made in that large quantities of fuel must be carried. However, for vehicles such as the current Pegasus, a PDE propulsion device may be attractive from the point of view of thrust control over a large portion of the flight envelope.

Acknowledgements

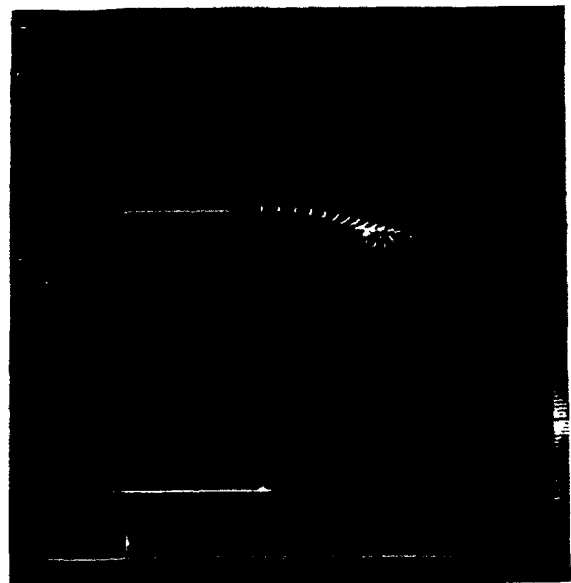
The authors would like to express appreciation to Drs. Adam Drobot and Aharon Friedman of SAIC for helpful suggestions and advice during the course of this work. The work reported on here was supported by DARPA under contract N66001-88-D-0088. The authors would like to thank Col. Doc Daugherty for his encouragement and interest in this project.

References

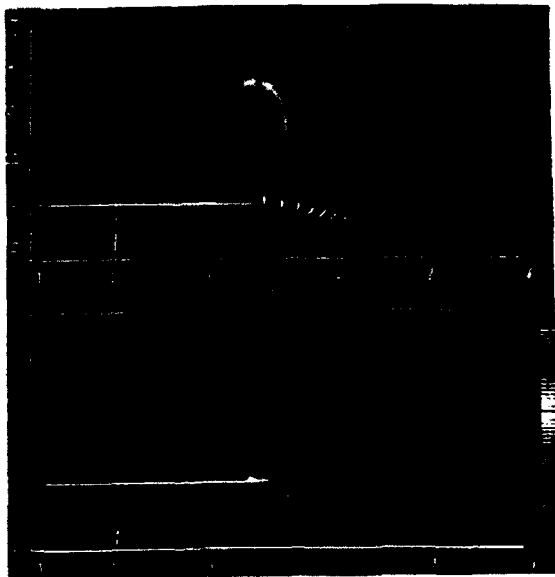
1. Stodola, A., *Steam and Gas Turbines*. McGraw-Hill Inc., 1927.
2. Zipkin, M. A., and Lewis G. W., "Analytical and Experimental Performance of an Explosion-Cycle Combustion Chamber of a Jet Propulsion Engine." NACA TN-1702, Sept. 1948.
3. Shultz-Grunow, F., "Gas-Dynamic Investigation of the Pulse-Jet Tube," NACA TM-1131, Feb. 1947.
4. Zinn, B. T., Miller, N. Carvelho, J. A., and Daniel B. R., "Pulsating Combustion of Coal in Rijke Type Combustor," 19th International Symposium on Combustion, 1197-1203, 1982.
5. Hoffmann, N., "Reaction Propulsion by Intermittent Detonative Combustion," Ministry of Supply, Volkenrode Translation, 1940.
6. Nicholls, J. A., Wilkinson, H. R. and Morrison, R. B. "Intermittent Detonation as a Thrust-Producing Mechanism." *Jet Propulsion*, 27, 534-541, 1957.
7. Dunlap, R., Brehm, R. L. and Nicholls, J. A., "A Preliminary Study of the Application of Steady State Detonative Combustion of a Reaction Engine", *ARS J.*, 28, 451-456, 1958.
8. Nicholls, J. A., Gullen, R. E. and Ragland K. W., "Feasibility Studies of a Rotating Detonation Wave Rocket Motor," *Journal of Spacecrafts and Rockets*, 3, 893-898, 1966.
9. Adamson, T. C. and Olsson, G. R., "Performance Analysis of a Rotating Detonation Wave Rocket Engine," *Astronautica Acta*, 13, 405-415, 1967.
10. Shen, P. I., and Adamson, T. C., "Theoretical Analysis of a Rotating Two-Phase Detonation in Liquid Rocket Motors," *Astronautica Acta*, 17, 715-728, 1972.
11. Krzycki, L. J., Performance Characteristics of an Intermittent Detonation Device, Navweps Report 7655, U. S. Naval Ordnance Test Station, China Lake, California 1962.
12. Matsui, H., and Lee, J. H., "On the Measure of the Relative Detonation Hazards of Gaseous Fuel-Oxygen and Air Mixtures," Seventeenth Symposium (International) on Combustion, 1269-1280, 1978.
13. Korovin, L. N., Losev A., S. G. Ruban and Smekhov, G. D. "Combustion of Natural Gas in a Commercial Detonation Reactor," *Fiz. Gor. Vzryva* 17, 86 (1981).
14. Smirnov, N. N., Boichenko, A. P., "Transition from Deflagration to Detonation in Gasoline-Air Mixtures," *Fiz. Gor. Vzryva* 22, 65 (1986).
15. Lobanov, D. P., Fonbershtein, E. G., Ekomasov, S. P., "Detonation of Gasoline-Air Mixtures in Small Diameter Tubes," *Fiz. Gor. Vzryva* 12, 446 (1976).
16. Back, L. H., "Application of Blast Wave Theory to Explosive Propulsion," *Acta Astronautica* 2, 391 (1975).
17. Varsi, G., Back, L. H., and Kim, K., "Blast Wave in a Nozzle for Propulsion Applications," *Acta Astronautica*, 3, 141 (1976).
18. Kim, K., Varsi, G., Back and L. H., "Blast Wave Analysis for Detonation Propulsion," *AIAA Journal* 10, Oct. 1977.
19. Back L. H., Dowler, W. L. and Varsi, G., "Detonation Propulsion Experiments and Theory," *AIAA Journal* 21, Oct. 1983.
20. Eidelman, S., Shreeve, R. P., "Numerical Modeling of the Nonsteady Thrust Produced by Intermittent Pressure Rise in a Diverging Channel," *ASME FED Multi-Dimensional Fluid Transient* 18, 77 (1984).
21. Eidelman, S., "Rotary Detonation Engine." U.S. Patent 4 741 154, 1988.
22. Helman, D., Shreeve, R. P., and Eidelman, S., "Detonation Pulse Engine," AIAA-86-1683, 24th Joint Propulsion Conference, Huntsville, 1986.
23. Monks, S. A., "Preliminary Assessment of a Rotary Detonation Engine Concept," MSc Thesis, Naval Postgraduate School, Monterey, California, Sept. 1983.
24. Camblier, T. L. and Adelman, N. G., "Preliminary Numerical Simulations of a Pulsed Detonation Wave Engine," AIAA-88-2960, AIAA 28th Joint Propulsion Conference, Boston 1988.
25. Eidelman, S., W. Grossmann, I. Lottati, "A Review of Propulsion Applications of the Pulsed Detonation Engine Concept," AIAA 89-2466, AIAA/ASME/SAE/ASEE 25th Joint Propulsion Conference, Monterey, CA, July 10-12, 1989 (to be published in *AIAA Journal of Propulsion*).
26. Lottati, I., S. Eidelman, A. Drobot, "A Fast Unstructured Grid Second Order Godunov Solver (FUGGS)," Paper AIAA 90-0699, 28th Aerospace Sciences Meeting, Reno, CA, Jan 8-11, 1990.



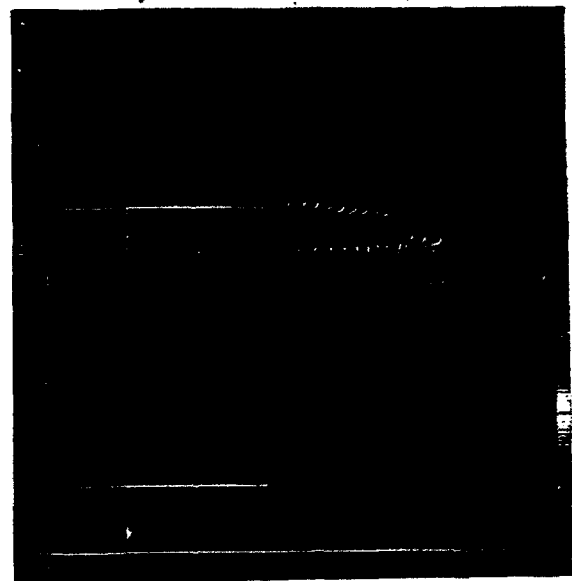
a) $t = 0.64$ msec,



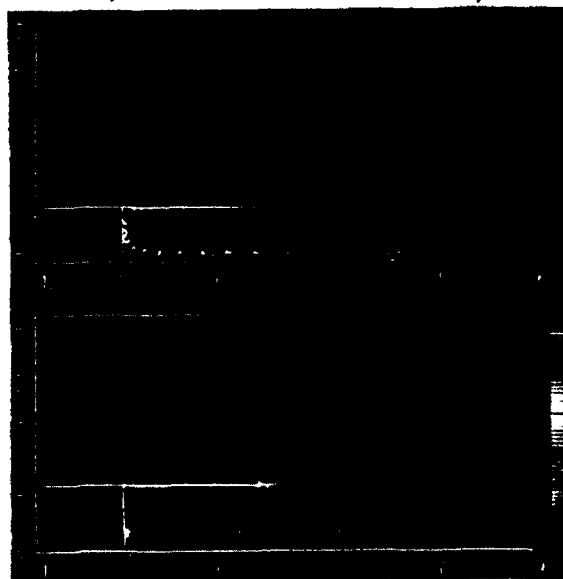
b) $t = 1.4$ msec,



c) $t = 2.2$ msec,



d) $t = 3.4$ msec,

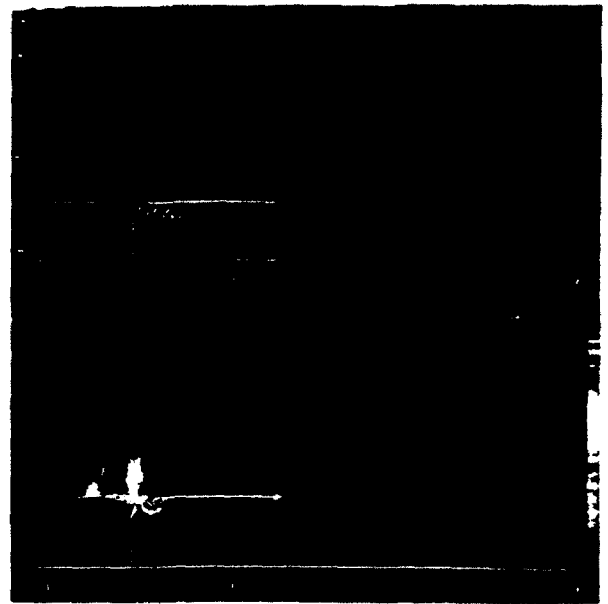


e) $t = 4.7$ msec,

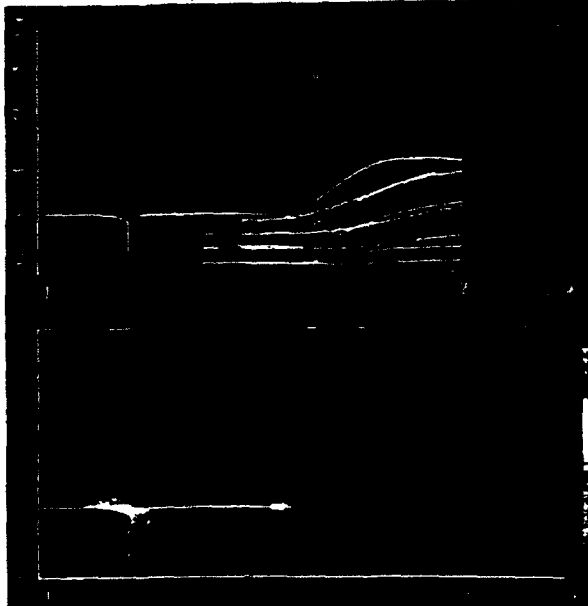
Figure 3 Pressure contours and marker particle paths for Case 1, $M = 0.8$, no inlet, inward initial detonation location.



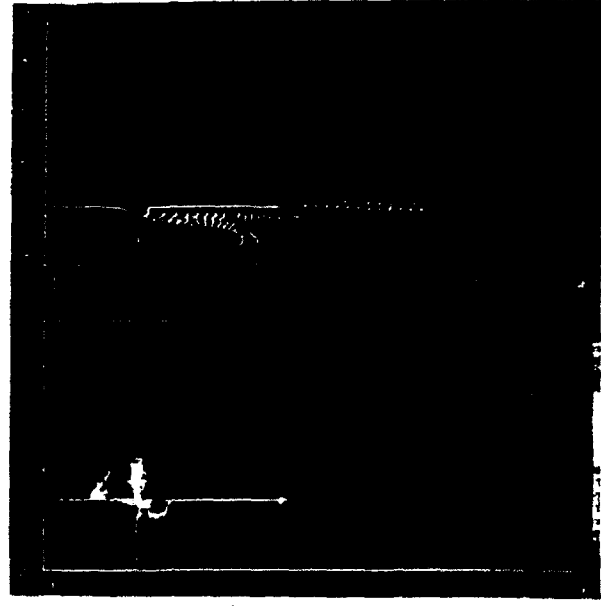
a) $t = 0.33$ msec,



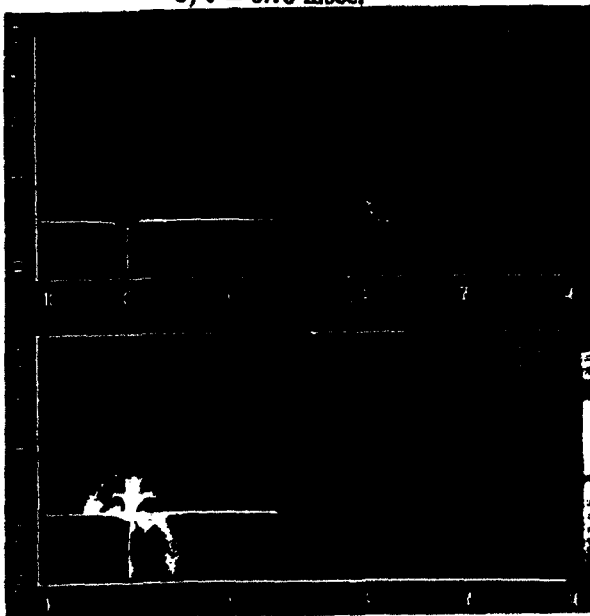
d) $t = 2.4$ msec,



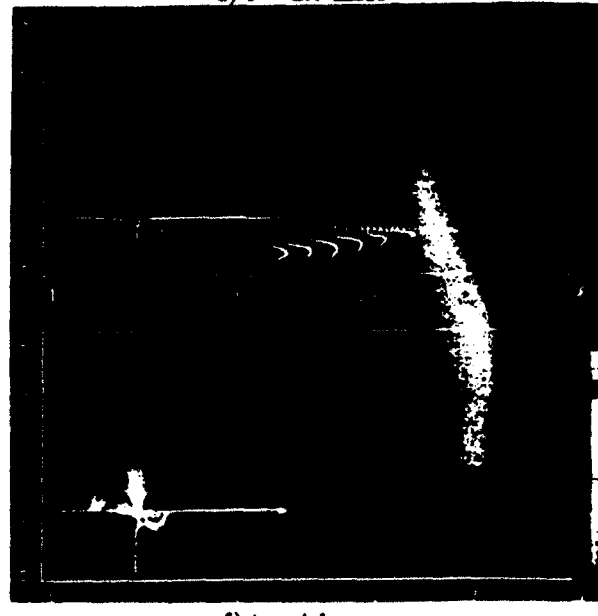
b) $t = 0.70$ msec,



e) $t = 2.7$ msec

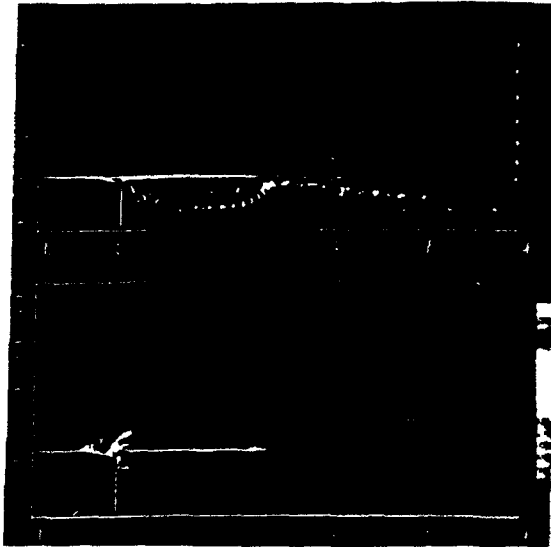


c) $t = 1.4$ msec,

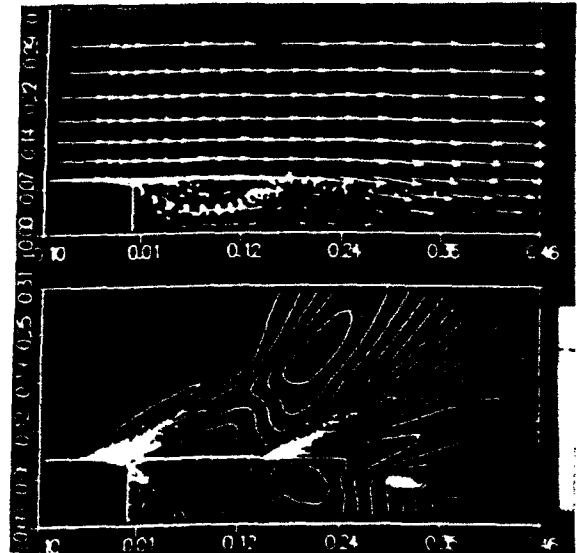


f) $t = 4.1$ msec

Figure 6 Pressure contours and marker particle paths for Case 2, $M = 0.8$, inlet geometry of Figure 5, outward initial detonation location.



a) $t = 1.3$ msec,



b) $t = 2.1$ msec,

Figure 8 Pressure contours and marker particle paths for Case 3, $M = 2.0$, sculptured inlet, outward initial detonation location.



AIAA-90-0699

**A FAST UNSTRUCTURED GRID SECOND ORDER
GODUNOV SOLVER
(FUGGS)**

Itzhak Lottati, Shmuel Eidelman, and Adam Drobot
Science Applications International Corporation
McLean, VA

28th Aerospace Sciences Meeting

January 8-11, 1990/Reno, Nevada

A FAST UNSTRUCTURED GRID SECOND ORDER GODUNOV SOLVER (FUGGS)

Itzhak Lotati, Shmuel Eidelman, and Adam Drobot

Applied Physics Operation
Science Applications International Corporation
McLean, Virginia 22102

ABSTRACT

We describe a new technique for solving Euler's equations on an unstructured triangular grid with arbitrary connectivity. The formulation is based on Godunov methods and is second order accurate. The use of a unique data structure leads to an easily vectorized and parallelized code with speed and memory requirements comparable to those found with logically structured grids. The new algorithm has been tested for a wide range of flow conditions ranging from low speed subsonic flow to hypersonic flow with Mach number 32. The results obtained are comparable to or better than those obtained with leading flow solvers in all of the regimes tested. The code contains no free parameters and can be used in complex flow problems where a variety of flow conditions may be encountered.

INTRODUCTION

This paper introduces a new second order algorithm for solving the Euler equations on an unstructured grid, using an approach based on first and second order Godunov methods. The formulation presented here leads to an extremely efficient and fast Flow Solver which is fully vectorized and easily lends itself to parallelization. The low memory requirements and speed of the method are due to the use of a unique data structure.

Explicit hydrodynamic numerical algorithms are easily adapted to Massively Parallel Computers (MPC) for logically structured grids. This is a consequence of the fact that the calculation of the flow quantities are locally determined. For logically structured quadrilateral grids, the integration algorithm or Flow Solver computes the new flow values at the grid cell nodes (or centers) using the values of the flow parameters from the previous timestep employing four or more of the adjacent nodes. Higher order structured solvers are usually more computationally intensive, but retain the ability of the solver algorithm to be separated into several distinct steps, each of which can easily be vectorized and parallelized.

Until recently, most CFD simulations were carried out with logically structured grids and consequently vectorization and/or parallelization did not present a problem. The increased need for simulation of flow phenomena in the vicinity of complex geometrical bodies and surfaces has led to the emergence of CFD codes based on logically unstructured grids. The most successful of these unstructured grid codes are based on finite elements [1-6] or finite volume [7-12] methods.

Unstructured grid CFD computations in two-dimensions usually decompose the simulation domain into triangular elements. The physical location of the triangular elements and the accompanying list of vertices and edges is random with respect to the element index, making it necessary to maintain an indirect addressing system containing the connectivity information.

Calculations performed on unstructured grids evolve around the elemental grid shape (e.g. the triangle for two-dimensional problems); there is no obvious pattern to the order in which the local integrations should be performed. Explicit integration of hydrodynamic problems on an unstructured grid requires that a logical substructure be created identifying the locations in the global arrays of all the local quantities necessary for the integration of one element. As a result, there is usually a significant cost in computational efficiency, memory requirement, and code complexity. Approaches to vectorization for the conventional unstructured grid methods have concentrated on rearrangement of the data structure in a manner such that these locally centered data structures appear as global arrays. This can be done to some extent using machine dependent Gather-Scatter operations. Additional optimization can be achieved using localization and search algorithms [13]. However, these methods are complex and result in marginal performance. To date, most optimized unstructured codes have run considerably slower and require an order of magnitude more memory per grid cell than their structured counterparts. Parallelization of the conventional unstructured codes is even more difficult, and there is very little experience with unstructured codes on Massively Parallel Computers.

The method we describe in this paper overcomes these difficulties and results in code with speed and memory requirements comparable to those found in structured grid codes. Moreover, the ability to construct grids with arbitrary resolution leads to a flexibility in dealing with complex geometries which is not attainable with structured grids. The essence of the method is based on independent flux calculation across the edges of a dual baricentric grid, followed by node integration. This approach allows the flux and integration calculations to be performed on global arrays, coded as large vector loops, and is independent of element position on the unstructured grid.

In this paper we discuss our choice for data structure, the numerical algorithm (for first and second order solvers), and the results of test calculations. In realistic CFD

problems the physical domain may contain regions that span all flow regimes. It is very important that the numerical code be able to perform well over the full range of flow parameters with no a priori code "refinement." This is especially true of complex problems where flow conditions cannot be easily assessed in all subdomains. A robust code has clear advantages if it is possible to apply it with confidence under such circumstances. We have chosen four test problems to benchmark and validate the FUGGS code. These include: i) a subsonic flow case for steady flow with $M = 0.2$; ii) supersonic steady flow with $M = 2.0$; iii) hypersonic steady flow with $M = 32.0$; and iv) transit hypersonic flow in a shock tube and over a wedge. For all of the test cases the method developed resulted in accurate solutions comparable to or better than reported in the literature by other leading CFD researchers. At the same time, the combination of using unstructured methods and our specific implementation yielded the lowest utilization of computer time and memory needed to achieve a given level of accuracy.

DATA STRUCTURE

On an unstructured grid, the data that describes the connectivity of a grid and the associated geometrical coefficients can represent a considerable overhead on memory usage. We have implemented a rather simple data structure which permits efficient finite difference integration of fluid quantities with only one level of indirection. For two dimensions, the data consists of lists of vertices, edges, and triangles. The physical quantities are stored at vertex locations. The vertex list consists of: the vertex positions (x,y) , the fluid variables, the vertex volume, and workspace. The edge data is composed of: the addresses of the two vertices which form an edge, a vector which indicates the normal to the face that crosses an edge, the face area, and storage for the fluxes. The face is formed by joining the baricenters of the adjoining triangles which lie along the edge. This is the only data required in performing an iteration step. For convenience and ease of diagnostics, we have also maintained a list of triangles, including the positions of the baricenters, and the addresses of both the vertices and edges which form a triangle.

The data structure is compatible with algorithms which decompose the solution of the Euler equations into two steps. The first is determination of the fluxes. This can be realized by a loop over edges where the fluid quantities along the edge can be fetched through the indirect addressing of vertex data. The second step is to integrate the fluxes which contribute to the vertex. There are two options here: one is to maintain a list of flux elements at each vertex and to perform a loop over vertices and then fluxes to each vertex; the other is to again have a loop over edges where each contribution to the vertex is done as a random fetch and store using the appropriate vertex addresses stored by each edge.

BASIC INTEGRATION ALGORITHM

We begin by describing the first order Godunov method for a system of two-dimensional Euler equations written in conservation law form as

$$\frac{\partial \vec{Q}}{\partial t} + \frac{\partial \vec{F}}{\partial x} + \frac{\partial \vec{G}}{\partial y} = 0, \quad (1)$$

where,

$$\vec{Q} = \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ e \end{pmatrix}, \quad \vec{F} = \begin{pmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ (e + p)u \end{pmatrix}, \quad \vec{G} = \begin{pmatrix} \rho v \\ \rho vu \\ \rho v^2 + p \\ (e + p)v \end{pmatrix}.$$

Here u and v are the x and y velocity vector components, p is the pressure, ρ is the density and e is the total energy of the fluid per unit volume. It is assumed that a mixed (initial conditions, boundary conditions) problem is properly posed for the set of equations (1), and that an initial distribution of the fluid parameters is given at $t = 0$, and the boundary conditions defining a unique solution are specified for the computational domain.

We seek a solution of the system of equations represented by Eq. (1) on a computational domain which is decomposed into triangles with arbitrary connectivity. An overwhelming advantage of using this method of domain decomposition is the ability to resolve extremely complicated geometries where the characteristic dimensions of subdomain features can vary over many orders of magnitude.

As an example, Figure 1 shows an unstructured triangular grid used in the simulation of flow for a new generation of the wide body tennis rackets with 21 cross string rows represented as solid circles and a tennis ball. In Figure 1a a blowup of the region near the racket surface is shown. This example illustrates the ability of the unstructured grids to represent geometry of arbitrary complexity and with localized resolution.

There are several options possible for storing physical information on an unstructured triangular grid: i) vertex centered; and ii) triangle centered on either a baricentric or Voronoi node. The selection of a specific grid structure offers two contrasting approaches. The first is to place the effort on creating an optimal grid, as is the case with Voronoi - Delauney meshes, while the second is to rely on the robustness of the integration algorithm. For complex configurations it is more difficult to achieve an optimum Voronoi-Delauney mesh and we have therefore opted for a simple baricentric grid.

This grid can always be constructed for a set of arbitrary triangles. The integration algorithm we have constructed can easily be implemented for both vertex and baricentered control volumes. Figure 2 displays a fragment of such a computational domain with the corresponding dual grid. The secondary or dual grid is formed by connecting the baricenters of the primary mesh, thus forming finite polygons around the primary vertices. Independent of the remarks made in Ref. 17 concerning usefulness of Dirichlet tessellation, we have confirmed that the best practical representation of the integration volume is obtained when the dual grid is formed by connecting baricenters of the triangles.

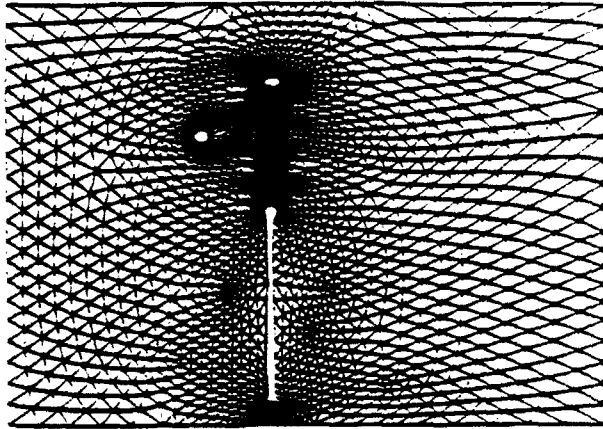


Figure 1 An illustration of the ability of a triangular grid to efficiently resolve the geometric complexity and features of objects with disparate spatial scales.

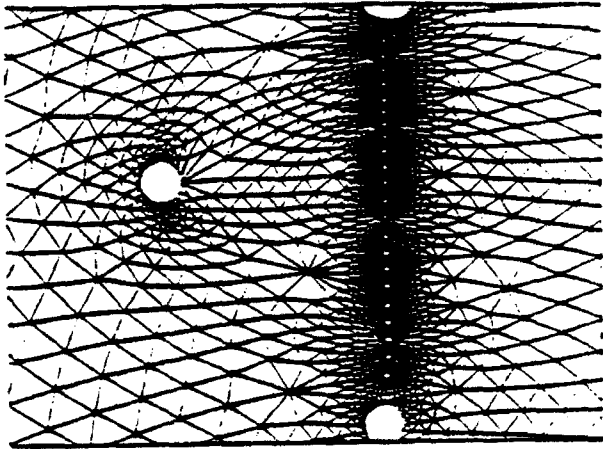


Figure 1a Detail showing the features of a wide body tennis racket simulation including 22 strings and a tennis ball.

In keeping with the philosophy that the overall scheme should be able to perform in all flow regimes with no prior tuning of "free" parameters, we have chosen Godunov methods for performing the numerical integration of the Euler equations in the control volume. These schemes are self consistent and do not contain any adjustable knobs. The superior performance of Godunov type schemes for logically structured grids is well documented and the advantages can be readily realized on triangular grids. Integration by the Godunov method consists of two basic steps: i) determination of the fluxes on the faces of the dual grid, which defines the control volume. This is accomplished by solving a set of one-dimensional Riemann problems along triangle edges; and ii) integration of the system of partial differential equations, which now amounts to a summation of all the fluxes for the vertex-centered control volume at each timestep.

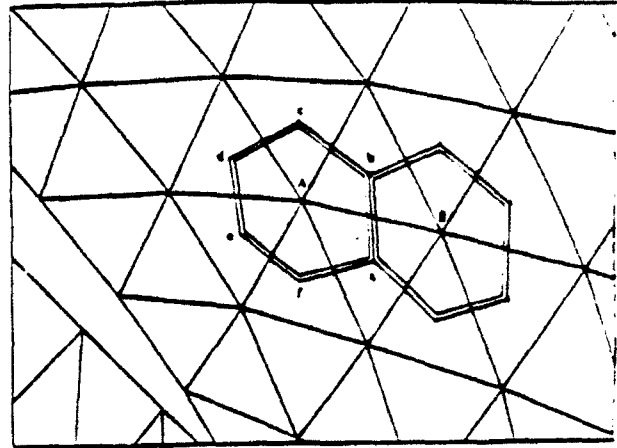


Figure 2 A triangular grid and the baricentered dual grid which defines the control volume. The fluxes are found on the faces of the control volume on each edge joining adjacent vertex points.

To define the fluxes flowing into the control volume shown in Figure 2, it is necessary to solve the Riemann problem along every edge of the primary grid and transverse to the faces of the dual grid. For example, to define the flux through the face ab , we solve the Riemann problem between vertices A and B . The solution of this problem is in coordinates local to the face of the dual grid ab so that the tangential component of velocity will be directed along ab . Implementation of our approach requires maintaining strict consistency when defining the "left" and "right" states for the Riemann problem at the faces ab , bc , cd , de , ef , and fa . For this reason we define not only the location of the vertices and areas of the faces but also the direction of the areas with respect to the primary grid edges. For the clockwise integration pattern in a polygon, vertex A will be the "left" state for all the Riemann problems related to this point and the neighbors will represent the "right" sides of the diaphragm.

It is easy to see that the flux calculation is based on information at only two nodes and requires simple geometrical parameters defining the face of the secondary grid which dissects the line connecting the two points. Thus, we can find all the values needed for the flux calculation in one vector loop over all edges of the primary grid without requiring any details related to the geometrical structures which these edges form. This in turn assures parallelization or vectorization of the algorithm for the bulk of the calculations involving the Riemann solver, which provides the first order fluxes.

The only procedure not obviously parallelizable is the integration of the fluxes for the flow variables at the vertices of the grid.

This operation requires a random fetch and store which can lead to conflicts that impair both parallelization and vectorization. Several common methods have been developed to deal with this difficulty. A practical approach is to split the integration of the fluxes into a small number of independent loops through the use of "edge" coloring. The number of loops necessary is determined by the maximum connectivity of any vertex in the domain and is

usually 7 or 8. Each of these loops is usually large enough not to impair vectorization. At this stage all the fluxes are added with their correct sign corresponding to the chosen direction of integration within the cell. The amount of computation required here is minimal since the fluxes are known and need only to be multiplied at each time step by a simple factor and added to the vertex quantity. This simple procedure results in a first order solver which is fully vectorized.

SECOND ORDER INTEGRATION ALGORITHM

The second order solver is constructed along lines similar to that of the first-order method. At each cell face the Riemann problem is solved for the appropriate pair of left and right conditions. The solution to the Riemann problem is then used in calculation of fluxes, which are to be integrated later to advance to the next integration step. The extension to second order is achieved by using extrapolation in space and time to obtain time-centered left and right limiting values as inputs for the Riemann problem. The basic implementation of the method for finding the second order accurate fluxes is the same as for the one dimensional case and can be found in Refs. 14 and 16. The difference is the method of obtaining linear extrapolations for the flow variables as a first guess of their value on the faces of the dual grid. To obtain the initial guess we need to know the gradient of each gasdynamical parameter U at the vertices of the primary mesh. The value ∇U can be evaluated by using the linear path integral around the finite volume associated with the vertex. For vertex A in Figure 2:

$$\int_A \nabla U dA = \oint_l U \hat{n} dl$$

where integration along the path l in this case is equivalent to integration along the lines ab, bc, cd, de, ef, fa , and where \hat{n} is a unit vector pointing outward from the control volume centered at A and normal to the integration path l . Knowing the gradient of the gasdynamic parameter in the volume related to vertex A allows us to extrapolate the values of this parameter at any location within the volume. This permits us to evaluate the first guess for U at the edges of the dual grid. The rest of the implementation of the second order algorithm is the same as described in Refs. 8 and 9. This includes monotonicity constraints similar to those introduced by VanLeer [15] and characteristic constraints described in Refs. 14 and 16.

A schematic of the basic steps for the second order algorithm is shown in Figure 3. This consists of five steps. They are: i) the calculation of the linearly extrapolated values at each side of the control volume faces using the left and right adjacent vertices and the values for each quantity and its gradient; ii) limiting the quantities obtained based on a monotonicity constraint; iii) a further limiter based on the solution of a one dimensional characteristic equation, which assures that the extrapolation does not violate the characteristics; iv) solution of the Riemann problem for the

final extrapolated values with the limiters applied; and v) integration over the control volume.

The advantages of the method described will be demonstrated in the following section. The inclusion of the characteristic limiter has significantly improved the treatment of contact discontinuities and is the first such implementation on a triangular mesh.

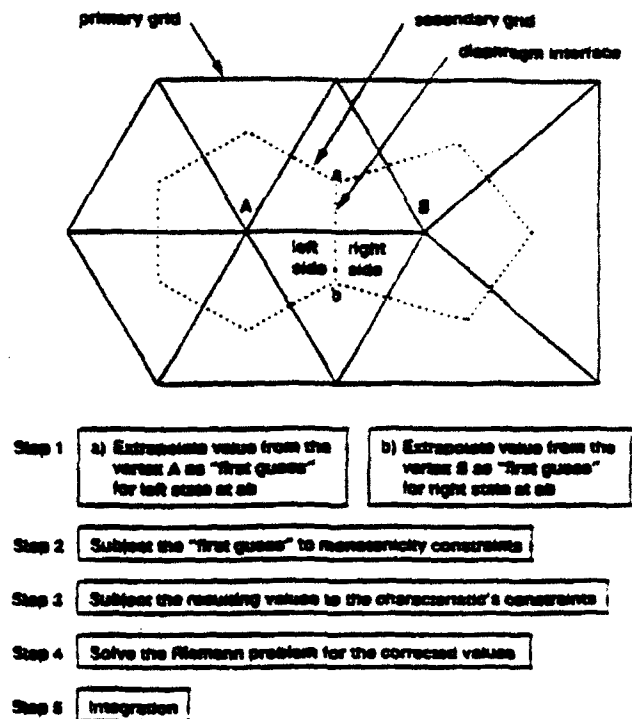


Figure 3 Schematic for stepwise implementation of the second order Godunov method on an unstructured grid.

RESULTS FOR TEST PROBLEMS

We have picked a set of test problems to demonstrate the performance of the FUGGS code for unsteady shock wave problems, and for subsonic, supersonic and hypersonic steady state flows. The cases in the chosen examples have analytical solutions that can be used to quantify the accuracy of the code and to validate the performance. This set of problems is frequently used by other CFD researchers and forms a basis for comparing FUGGS with other techniques.

a. Unsteady Shock Problem

As a first test we have chosen a case of planar shock wave propagation in a channel.

A section of the grid used for this test problem is shown in Figure 4. The total grid contained ~ 2000 vertices with a resolution of 100 points in the direction of propagation. We simulated a simple shock tube problem on

this grid where the gasdynamic parameters to the left and right of the diaphragm have the following values:

$$\begin{aligned} P_l &= 1.0; \rho_l = 1.0; U_l = 0; \\ V_l &= 0; \gamma_l = 1.4; \\ P_r &= 0.1; \rho_r = 0.125; U_r = 0; \\ V_r &= 0; \gamma_r = 1.4. \end{aligned}$$

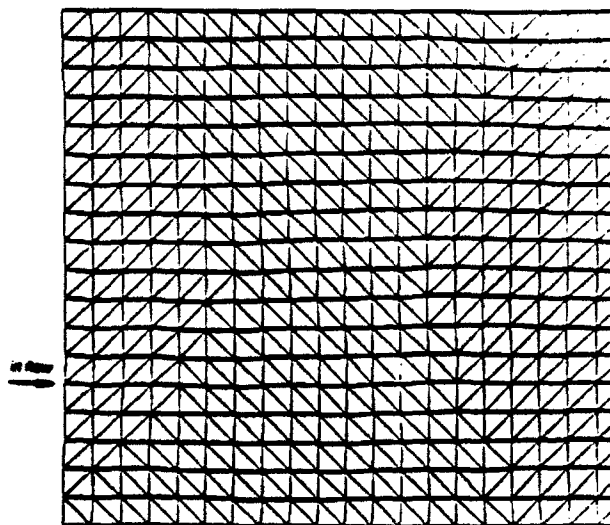


Figure 4 Section of grid from the unsteady shock problem.

This one dimensional problem was simulated on a rather ill formed grid (from the viewpoint of connectivity). Consequently the quality of the solution depended on the flow solver for accuracy. For the triangular shape of the elementary cell, planar shock and rarefaction waves generated by the solution always propagate at conflicting angles with respect to four out of the six edges of the control volume. The triangular grid chosen for this simple test problem therefore indicates the accuracy of FUGGS for shock waves of arbitrary shape and orientation moving through the computational domain. The density distribution found from three different versions of FUGGS is shown in Figure 5 as a function of x along the median cross section of the grid. The three cases are: i) first order Godunov method; ii) second order Godunov; and iii) second order Godunov with the characteristic limiter. The data displayed in the figure represents a loss of resolution due to interpolation of the actual grid values to the projected midsectional line. It is clear from Figure 5 that the final implementation of FUGGS with characteristic constraints yields the best results for contact discontinuities. The code also maintains the one dimensional structure for the shock in all three cases described above. The accurate representation of the density is also typical for all the other gasdynamic parameters.

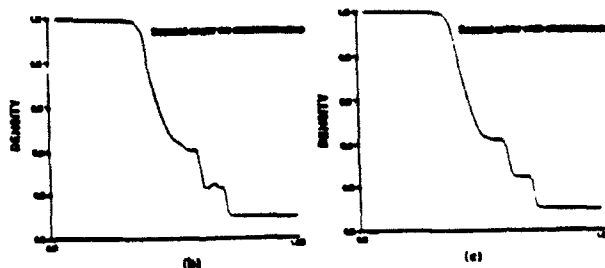
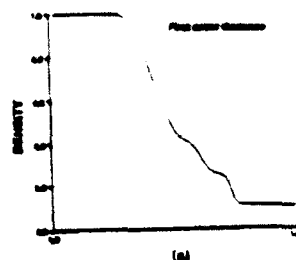


Figure 5 Solution to the density distribution of shock problem with three different versions of FUGGS: a) First Order Godunov, b) Second Order Godunov without characteristics and c) Second Order Godunov with characteristics.

b. Shock on Wedge

Here we demonstrate the performance of the methods for steady supersonic flow simulations. An analytical solution from oblique shock wave theory exists and can serve as an unambiguous comparison with the numerical simulation.

The initial grid for the shock on wedge problem is shown in Figure 6. This gridding results in ~ 500 vertices and ~ 800 triangles. The wedge angle in Figure 6 is 10° . The incoming flow enters the computational domain normal to the left boundary at Mach number $M=2$. Figures 7a and 7b show isomach lines for the steady flow solution from the first and second order Godunov solvers on the original grid. Comparing these two solutions we can see that the second order solution substantially improves the shock resolution. However, it is obvious that the grid density is too small to adequately resolve the oblique shock wave in both cases.

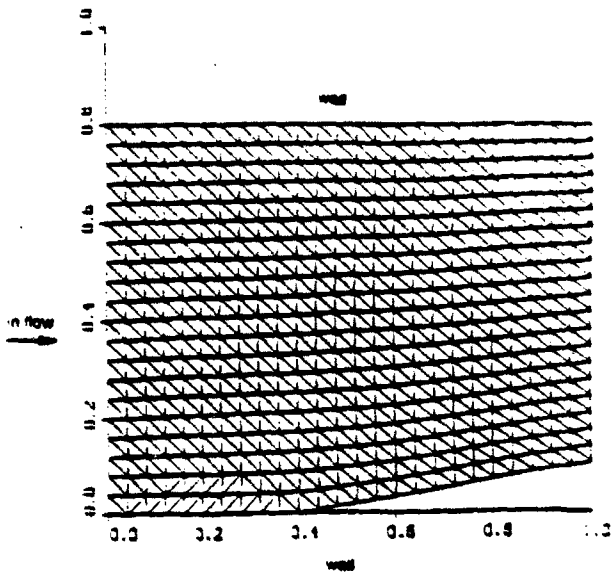


Figure 6 Coarse grid for shock on wedge problem.

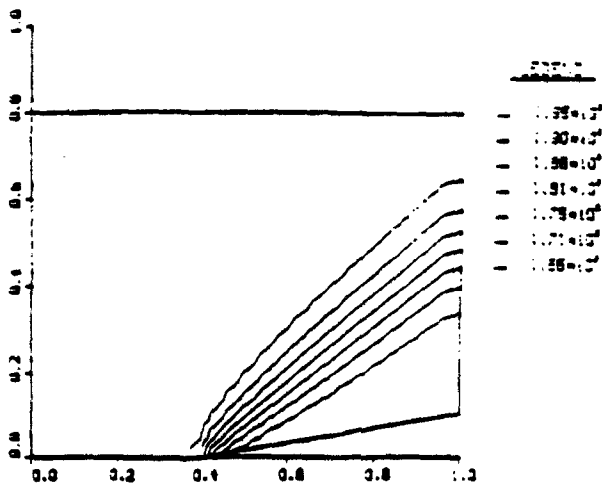


Figure 7a First order Godunov solution for the coarse grid shown in Figure 6.

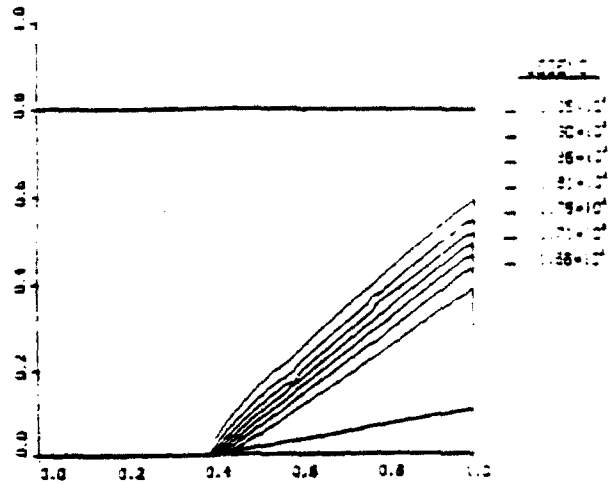


Figure 7b Second order Godunov solution for the grid shown in Figure 6.

To improve the accuracy a higher grid density is required in the region of discontinuity. This is achieved by subdividing the original elements of the grid in regions of large changes in flow parameters.

A variety of criteria can be devised to identify regions which require mesh refinement. An example is given in Ref. 2 where a preset condition is imposed on the resolution from local derivatives of the flow parameters. The implementation of this criteria in FUGGS would have led to a significant loss of computational efficiency because the stencil for the Laplacian is nonlocal and would require more than one level of indirectness. Instead we used a simple parameter variation criteria based on the local variation in pressure or density to select the grid regions needing refinement. Figure 8 shows an enhanced grid derived from the mesh shown in Figure 6 by two levels of subdivision. The number of triangles in this case increase from 800 to 1200. Figure 9 shows isomach lines of the solution using the second order method for the same shock on wedge problem as in Figures 7a and 7b. The improvement in shock resolution is dramatically noticeable. This problem also illustrates the ability of unstructured grid methods to provide local resolution for important flow features, without requiring excessive overhead for other regions of the computational domain.

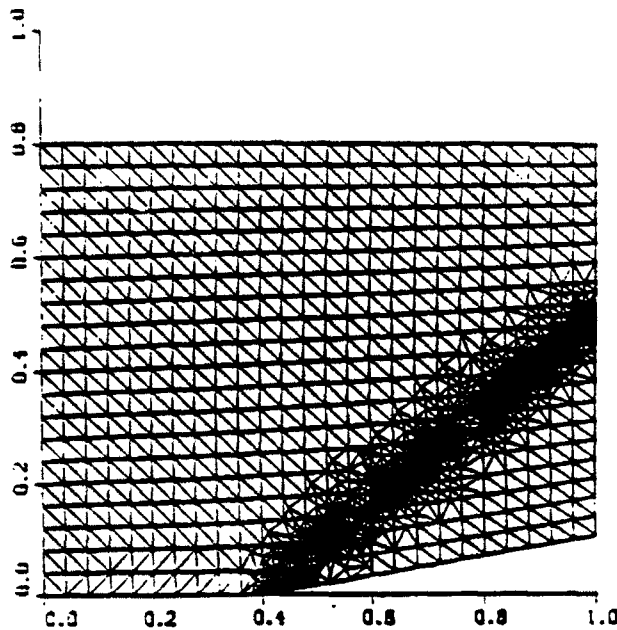


Figure 8 Improved grid for the shock on wedge problem with two levels of refinement based on 5% variation in local value.

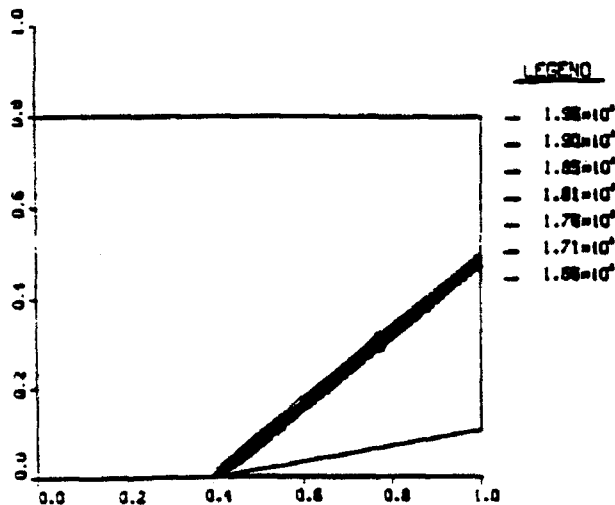


Figure 9 Second order Godunov solution for the shock on wedge problem using a grid with two levels of refinement.

c. Subsonic Flow

A challenging test problem to assess the performance of Euler codes for subsonic flow has been suggested by Pulliam [19]. He has computationally simulated a steady subsonic flow over an ellipse with major to minor axis ratio of 6:1. The numerical solution of Euler equations reported for this case at $M_\infty = 0.2$ with angle of attack $\alpha = 5^\circ$ produced a lift coefficient of $C_L = 1.545$. As is well known from D'Alembert's Paradox, inviscid flow at low Mach numbers should yield $C_L = 0$ and have zero drag for a profile

of an arbitrary shape. For this reason the problem posed by Pulliam is a good indicator of the accuracy and amount of artificial dissipation introduced by a numerical algorithm. Moreover, while a Euler solver is not meant to treat potential flow, a general purpose solver should be capable of simulating such flow conditions if they occur in a portion of a given problem without resorting to a different algorithm. In making a transition to full Navier-Stokes treatment, the use of a Euler solver is an essential step; it is important to have confidence that the artificial viscosity introduced does not dominate the solution.

For the case under consideration, it is very important to understand in detail the potential flow solution over an ellipse. Fortunately, the analytical solution is available and is relatively simple. The complex potential for the flow over a cylindrical ellipse is given by the following [20]:

$$F(z) = -\frac{1}{2} M_\infty (a+b) e^{-i\alpha} \left[\frac{z + \sqrt{z^2 - (a^2 - b^2)}}{a+b} + \frac{z - \sqrt{z^2 - (a^2 - b^2)}}{a-b} \right]$$

where $Z = x+iy$ and M_∞ is the Mach number. By taking the gradient of the potential we can find the velocity flow field explicitly:

$$\frac{U}{U_\infty} = \frac{(1+\lambda) \sin(\theta+\alpha) \sin \theta}{\lambda^2 \cos^2 \theta + \sin^2 \theta}$$

$$\frac{V}{U_\infty} = \frac{(1+\lambda) \lambda \sin(\theta+\alpha) \cos \theta}{\lambda^2 \cos^2 \theta + \sin^2 \theta}$$
(3)

where $\lambda = b/a$ is the ratio of minor to major axis, θ is the angle in polar coordinates from the center of the ellipse, and α is the angle of attack.

In examining this equation, we find that the maximum value of velocity is a strong function of λ . For an ellipse with $\lambda = 1/6$, the maximum value V/U_∞ occurs at $\theta = 0$ or π and where $V_{MAX}/U_\infty = 7 \sin \alpha$. For a flat plate where $\lambda \rightarrow 0$ the maximum velocity is infinite. The angle α defines the distance between the stagnation point where the velocity is zero (at $\theta = -\alpha$ and $\pi - \alpha$) and the point where the velocity is maximum (at $\theta = 0$ and π). For the case selected by Pulliam the distance between the point with minimum and maximum velocity is 0.19% of the length of the major axis.

This means that the gradient of velocity along the major axis of the ellipse in the vicinity of stagnation points is extremely high. With ~ 1000 points uniformly distributed on the surface of the ellipse, only one grid spacing is available to resolve both the stagnation point and the point at which maximum velocity occurs. Even though one would normally construct a nonuniform grid in the vicinity of the stagnation point, we estimate that enormous computational resources would be required to resolve the characteristic scale length for this problem. Traditional methods encounter difficulties in this situation because spatial splitting leads to a poor estimate of the gradient and

the low connectivity of the mesh introduces spurious vorticity.

We performed two simulations for the conditions described by Pulliam. The number of nodes used on the surface of the ellipse is the same as in Ref. [19]. The grid is shown in Figure 10 for these simulations in the region immediately proximate to the ellipse. This grid is of poor quality and highly distorted; contains ~ 6000 vertices and ~ 130 points on the surface of the ellipse. The results are shown in the form of pressure contours in Figures 11 and 12 for the first order and second order solvers respectively. In the case of the first order algorithm, we obtained a lift coefficient of $C_L = 0.29$. The pressure contours for this simulation are not smooth, attributable to the low level of accuracy of the solver. The same situation resulted in $C_L = 0.252$ when computed with the second order solver, and as can be seen in Figure 12 the pressure contours are considerably smoother. The result presented by Pulliam was $C_L \approx 1.55$, almost an order of magnitude higher than achieved with FUGGS. This highlights an important quality of our approach: the low generation of artificial viscosity. In comparison the lift obtained by Pulliam is as high as one would expect from thin profile theory and hence would mask real viscosity effects if they were added to the algorithm.

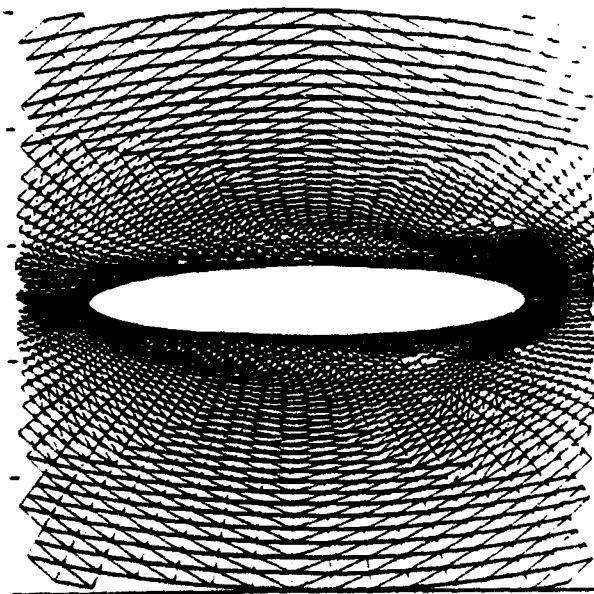


Figure 10 Section of the grid used in simulation of subsonic flow over an ellipse for conditions suggested by Pulliam [19].

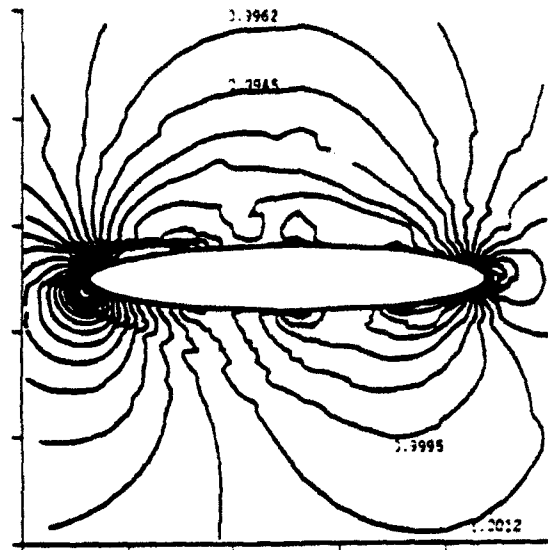


Figure 11 First Order Euler Solution for 6:1 Ellipse. Pressure contours. $\alpha = 5^\circ$; Mach = 0.2; 6065 vertices; $C_L = 0.381$; $C_D = 0.101$.

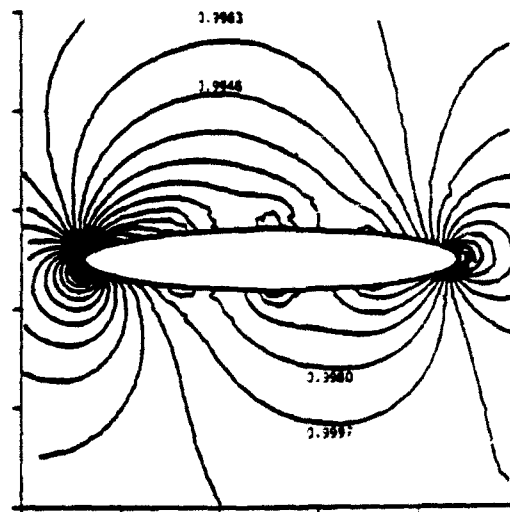


Figure 12 Second Order Euler Solution for 6:1 Ellipse. Pressure Contours. $\alpha = 5^\circ$; Mach = 0.2; 6065 vertices; $C_L = 0.252$; $C_D = 0.004$.

We also simulated flow over a cylinder at $M = 0.2$. The grid for this case is shown in Figure 13. We examined the numerically produced lift with inflow conditions at various angles with respect to the x - axis (0° , 5° , 20° , 45°). The lift coefficient was angle independent and had a value $C_L = 0.76$, almost 20 times smaller than reported by Pulliam, whose results are angle sensitive. With the first order scheme we achieved a lift coefficient of $C_L = 0.47$ with the drag coefficient $C_D \approx 1.49$. For the second order scheme, shown in Figure 14, the drag coefficient was reduced to $C_D = 0.19$ but the lift coefficient increased somewhat to the

value cited above. We also investigated the effects of grid refinement and found that a simple one level of refinement (adding ~ 400 vertices) led to a modest reduction in lift coefficient of about 20%. To reinforce a point made earlier, all of the results were achieved with no "free" parameters to adjust. These parameters are present in many CFD codes in the form of coefficient for artificial viscosity terms present the practitioner with a practical problem of how they should be selected for different flow conditions.

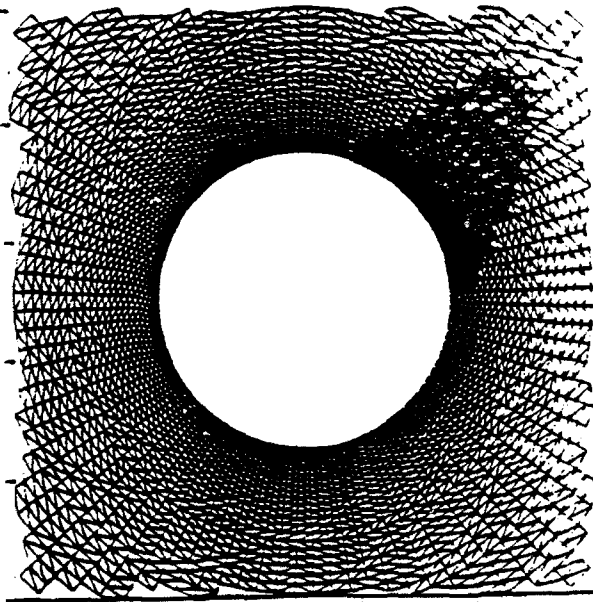


Figure 13 Grid for simulation of flow over a cylinder at varying inflow angles with respect to the mesh.

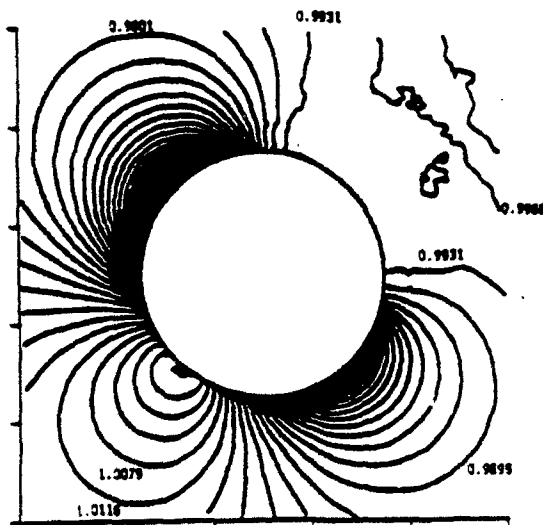


Figure 14 Second Order Euler Solution for a circular cylinder: $\alpha = 45^\circ$; Mach = 0.2; 6311 vertices; $C_L = 0.761$; $C_D = 0.196$.

d. Hypersonic Flow

To demonstrate the versatility of the method for the entire range of flow regimes we have simulated a hypersonic flow test problem. One of the advantages of the Godunov type methods is that for the whole range of calculations performed (from low subsonic flow, supersonic flow, unsteady flow with strong shock, or hypersonic flow at Mach number $M=32$) it is unnecessary to change or adjust the numerical algorithm. In Ref. 21 performance of first and second order Godunov methods has been analyzed for hypersonic flow regimes. There, as a test problem, an analytical solution was used for a hypersonic flow around a flat plate of finite thickness. This solution was obtained based on the analogy between hypersonic flow over a flat plate of finite thickness and a strong planar explosion. Here we will use one expression from Ref. 21 which defines the shape of the shock wave as a function of plate thickness d , γ the adiabatic coefficient, and α a nondimensional scale factor related to the energy released at the stagnation point.

$$Y_{SHOCK} = \left(\frac{1}{2} D_f \frac{dx^2}{2} \right)^{1/3}$$

where D_f is a coefficient of order unity,

$$a = k_1 (\gamma - 1)^{k_2 + k_3 \ln(\gamma - 1)}$$

while $k_1 = 0.36011$, $k_2 = -1.2537$, and $k_3 = -0.1847$.

As a direct comparison we solved the hypersonic flow problem for the same set of conditions as in Ref. 21:

$$U_\infty = 10011 \text{ meters/sec, } p = 98.72 \text{ Pa,}$$

$$\rho = 1.24 \times 10^{-3} \text{ kg/m}^3 \text{ and } \gamma = 1.2.$$

The grid used for this simulation is shown in Figure 15. This grid has ~ 5500 vertices and its spatial resolution at the leading edge of the plate is of the same order as that of a 300 x 60 rectangular grid used in Ref. 12.

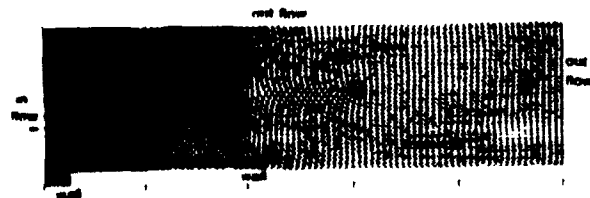


Figure 15 Grid for simulation of hypersonic flow over a flat plate.

In Figure 16 results for this simulation are shown in the form of pressure contours. Figure 16 also shows the location of the analytically calculated shock front by a discrete line (squares). The shock resolution and accuracy of its location are comparable to that obtained in Ref. 21, even

though our triangular grid has less than 1/3 the nodes than the rectangular grid used in Ref. 21. This is due to the fact that in constructing the triangular grid we had the flexibility to place the highest concentration of nodes in the area of the leading edge where the main properties of the flow are established.

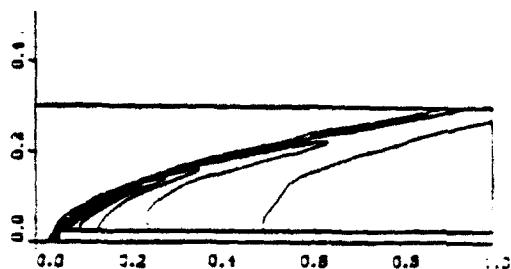


Figure 16 Second order solution for a flat plate. Pressure Contours. Mach = 32; 5509 grid vertices; $P_{max} = 5.0 \times 10^{-4}$; $P_{min} = 98.7 P_a$.

CODES COMPUTATIONAL EFFICIENCY

During the code development effort, great attention was paid to the code data structure, its efficiency and extendability to three dimensional calculations. In fact, the two dimensional version of the code has all the data structures required for the three dimensional simulations. That fact should be factored in comparing our storage overhead figures to those in other codes. Also while developing FUGGS we made a decision not to rely on machine-dependent functions, in order to assure portability. This feature is very important in the current supercomputing environment where a host of powerful parallel supercomputers and super workstations with diverse architecture are available and useful for different aspects of design.

The following performance characteristics have been achieved for the latest version of the FUGGS code:

1. First Order Godunov version:

Memory Requirement	36 places per triangle Includes 5 physical quantities integer indexing arrays all geometric parameters
CPU Performance CRAY XMP-24 STELLAR 1000	15 μ sec/vertex/timestep 79 μ sec/vertex/timestep

2. Second Order Godunov version:

Memory Requirement	39 places per triangle
CPU Performance CRAY XMP-24	45 μ sec/vertex/timestep Monotonicity step 50% Characteristic limiter 15% Riemann Solver 30% Integration 5%
STELLAR 1000	214 μ sec/vertex/timestep

These numbers are provisional since the code is still under development. We feel that further improvements in code performance will be achieved with respect to both timing and storage requirements.

CONCLUSION AND DISCUSSION

We have presented a method for the numerical solution of Euler equations on an unstructured triangular grid. The method was tested for a wide range of flow conditions, from low subsonic flow and unsteady flow with strong shock waves to hypersonic flow with Mach 32. For all these regimes, the method performed extremely well both in terms of solution accuracy and computational efficiency. The method is very robust and does not resort to adjustable computational parameters for the tested range of flow conditions. This is due to the fact that the numerical algorithm in FUGGS is based on Second Order Godunov schemes adapted to triangular grids. The method appears natural for unstructured triangular grids because the greater connectivity intuitively should lead to greater accuracy in eliminating errors introduced by splitting. In a typical hexagonal (or greater) control volume the contribution of fluxes is available from all six adjacent directions as opposed to just two in the case of a rectangular grid. Since the FUGGS method has been implemented on unstructured grids, it is possible to simulate flows over bodies of arbitrary geometry where the grid density can be concentrated in a region of flow discontinuity.

Especially interesting is the code's superior performance for the simulations of subsonic flow. For the test cases calculated here, our method appears to perform better than the leading industry codes like ARC2D and SYMTVD. We think that the two main reasons for the better performance are multidirectional splitting (to distinguish from two directional splitting typical for logically structured quadrilateral grids) and finite volume integration. More should be done to investigate this important aspect of the code's performance.

Historically, Euler solvers were developed to simulate nonisentroic flows for which potential flow assumptions are incorrect. From the original development of numerical methods for the solution of Euler's equations, great effort has been devoted to resolving shocks and contact discontinuities, producing in dramatically improved results for shock wave hydrodynamics. At the same time, attention to the accurate solution of the velocity gradients has been neglected. While these gradients are more difficult to discern than shock waves, they are more prevalent in practical flow

problems and could lead to very significant errors in such important parameters as lift and drag coefficients. In addition, all vorticity and viscosity dominated phenomena depend on accurate solution of the velocity gradients. In view of the performed numerical simulations for subsonic flow over the ellipse and cylinder it is clear that unless these features are resolved, the numerical solution of Euler equations can introduce spurious vorticity, making the results from a full Navier-Stokes implementation impossible.

Acknowledgment

This work was partially supported by Wright-Patterson Research and Development Center under contract No. F33615-88-C-3002 and by DARPA through AFOSR contract No. FY9620-89-C-0087. We would also like to thank Dale Nielsen, Jr. of Lawrence Livermore National Laboratory for the initial gridding software and STELLAR Computer for providing assistance and facilities.

REFERENCES

1. F. Angbrand, V. Boulard, A. Dervieux, J. Periaux, and G. Vuayasundaram, in *Computing Methods in Applied Sciences and Engineering*, edited by R. Glowinski et al. (North-Holland, Amsterdam, 1984).
2. F. Angbrand, V. Boulard, A. Dervieux, J.A. Desideri, J. Periaux, and B. Stouglet, in *Proceedings Ninth International Conference on Numerical Methods in Fluid Dynamics, Saclay, France, 1984*, edited by Soubbaramayer and Boujot, Lecture Notes in Physics, Vol. 218 (Springer-Verlag, 1985).
3. L. Fezoui and B. Stouglet, "A Class of Implicit Upwind Schemes for Euler Simulations with Unstructured Meshes," *J. of Comp. Phys.* **84**, 174-206 (1989).
4. R.A. Shapiro and E.M. Murman, "Adaptive Finite Element Methods for the Euler Equations," AIAA-88-0034, 26th Aerospace Sciences Meeting, Reno, Nevada, 1988.
5. R. Lohner, K. Morgan, and D.C. Zienkiewicz, "Finite Element Methods for High Speed Flows," AIAA 7th Computational Fluid Dynamics Conference, Cincinnati, Ohio, AIAA Paper 85-1531 (1985).
6. R. Lohner and K. Morgan, "Improved Adaptive Refinement Strategies for Finite Element Aerodynamic Computations," AIAA 29th Aerospace Sciences Meeting, Reno, Nevada, AIAA Paper 86-0499 (1986).
7. D. Mavriplis and A. Jameson, "Multigrid Solution of the Two-Dimensional Euler Equations on Unstructured Triangular Meshes, AIAA-87-0353, 1987.
8. T.K. Dukowicz, M.C. Cline, and F.L. Addessio, "A General Topology Godunov Method," *J. of Comp. Phys.* **82**, 29-63 (1989).
9. T.J. Baker and A. Jameson, "A Novel Finite Element Method for the Calculation of Inviscid Flow Over a Complete Aircraft," *Sixth International Symposium on Finite Element Methods in Flow Problems*, Antibes, France (1986).
10. A. Jameson and T.J. Baker, "Improvements to the Aircraft Euler Method," AIAA 25th Aerospace Sciences Meeting, Reno, Nevada, AIAA Paper 87-0452 (1987).
11. T.J. Baker, "Developments and Trends in Three-Dimensional Mesh Generations," *Transonic Symposium*, NASA Langley Research Center, Virginia (1988).
12. A. Jameson, T.J. Baker, and N.P. Weatherill, "Calculation of Inviscid Transonic Flow Over a Complete Aircraft," AIAA 24th Aerospace Sciences Meeting, Reno, Nevada, AIAA Paper 86-0103 (1986).
13. L. Greengard and V. Rokhlin, "A Fast Algorithm for Particle Simulations," *J. Comp. Phys.* **73**, 325-348 (1987).
14. S. Eidelman, P. Collela, and R.P. Shreeve, "Application of the Godunov Method and Its Second Order Extension to Cascade Flow Modeling," *AIAA Journal*, v. **22**, 10 (1984).
15. B. van Leer, "Towards the Ultimate Conservative Difference Scheme, V.A. Second Order Sequel to Godunov's Method," *J. Comp. Phys.* v. **32**, 101-136 (1979).
16. P. Collela and P. Woodward, "The Piecewise Parabolic Method (PPM) for Gasdynamic Simulations," *J. Comp. Phys.* v. **54**, 174-201 (1984).
17. T.J. Barth and D.C. Jespersen, "The Design and Application of Upwind Schemes on Unstructured Meshes," 27th Aerospace Sciences Meeting, AIAA-89-0366, Reno, Nevada (1989).
18. H.M. Glaz, P. Collela, I.I. Glass, and R.L. Deschambault, "A Detailed Numerical, Graphical, and Experimental Study of Oblique Shock Wave Reflections," DNA-TR-86-365, Technical Report, 1986.
19. T.K. Pulliam, "Computational Challenge: Euler Solution for Ellipses," AIAA-89-0469, Reno, Nevada, 1989.
20. Schaum's Outline Series, Fluid Dynamics, by W.F. Hughes and J.A. Brighton, McCraw-Hill Book Co., New York, 1967.
21. S. Eidelman, "Application of the Hypersonic Analogy for Validation of Numerical Simulations," *AIAA Journal* **27**, 11, 1566-1571 (1989).

Reflection of the Triple Point of the Mach
Reflection
in a Planar and Axisymmetric Converging
Channels

Shmuel Eidelman and Itzhak Lottati

Science Applications International Corporation
McLean, Virginia, USA

9th Mach Reflection Symposium
Freiburg
June 1990

Reflection of the Triple Point of the Mach Reflection in Planar and Axisymmetric Converging Channels

Shmuel Eidelman and Itzhak Lottati

Science Applications International Corporation,
McLean, VA. USA

Introduction

Depending on their parameters, the encounter between a planar shock wave and a wedge can produce a classic case of the Mach Reflection. The Mach Reflection has a characteristic triple point, where three shocks and the contact discontinuity coalesce. In a shock tube or in a channel, a developed Mach Reflection can reflect further from the walls opposite the wedge. In this case, the Mach shock of the Mach Reflection will start reflecting when its triple point reaches the wall opposite the wedge. Upon reflection of this shock wave, a secondary Mach Reflection can form. Although the primary Mach Reflection has received considerable attention in scientific literature (Refs. 1,2,3), the phenomenology of the subsequent reflections was gone virtually unnoticed. In our literature review, we found only a short qualitative description of the phenomena by Bazhenova and Gvozdeva (Ref. 4). This omission is unfortunate, since it is a very practical case for propagation of the shock waves in channels of variable cross sections.

The direct simulation of the various cases of Mach Reflection has only become possible in the last decade. This problem is a challenging test for the numerical methods used in Computational Fluid Dynamics (CFD). An impressive demonstration of the capabilities of the direct numerical simulations of Mach Reflection phenomena is given by Glaz et al. (Ref. 5). They demonstrate that all the important phenomenology of the Mach Reflection, including slip line vortex and Mach shock wave bulging, can be simulated directly. This was achieved by using the Second Order Godunov method, numerical technique, developed in 80th, which is extremely robust and allows very accurate simulation of flow discontinuities.

The Second Order Godunov method was implemented on rectangular grids (Ref. 6) and in a few cases on general quadrilateral grids (Ref. 7). This approach has limited application, since the structured quadrilateral grids have great difficulty describing a complicated computational domain with multiple bodies of different geometries and scales. Recently, we have implemented the Second Order Godunov for unstructured triangular grids (Ref. 8). This enables us to combine the robust and accurate numerical algorithm with a gridding technique, allowing us to describe very complex domains with ease and efficiency. In addition, we have developed a novel Dynamic Grid Adaptation methodology which allocates a dense computational grid only to regions where

enhanced resolution is needed to resolve strong gradients in flow parameters. As demonstrated in our paper, this enables an extremely economical allocation of computational resources and accurate simulation of a complicated phenomena like Mach Reflection.

In our study, we numerically simulate the formation of a Double Mach Reflection on a sloped wall of a converging channel, with subsequent reflection of the reflected wave at the straight wall of the channel. Presented here numerical results were obtained with the new numerical technique and we will describe in detail all the important new elements which we have introduced.

The Problem

Figure 1 shows a converging channel with a sloped wall at 27° . The figure illustrates our assumption that a Mach 8.7 shock wave travelling normally to the parallel walls enters the channel at the left hand side. According to analysis presented in Reference 9, this shock will have a Complex Mach Reflection when it encounters the converging wall of the channel. At some stage of the reflection process, the triple point will reach the opposite wall of the channel. Here the Mach stem shock wave will become incident, moving at an angle to the channel wall, as illustrated in Figure 2. The shock and wedge parameters chosen in our problem will cause formation of a secondary Mach reflection. The question is: What form will this secondary reflection take? Bazhenova and Gvozdeva offer a very general description of the anticipated effect, illustrated in Figure 3 (Ref. 4). In this reference, a system of secondary reflections shows the incident and Mach shocks are interchanging their positions with every new reflection, and the strength of the shock waves is increasing. It is not clear from Reference 4 what type of Mach Reflection will form, or how the secondary reflected wave, which expands in already perturbed gas, will be affected by the interactions with the strong slip surfaces located behind the original Mach shock.

We will directly simulate formation of the Mach Reflection at the channel oblique wall, as well as all secondary reflections which will occur according to the conditions outlined above for the channel geometry shown in Figure 1. In addition we will consider cases in which the channel shown in Figure 1 is axisymmetric and will study the same problem for this case. The motivation is further study of the phenomenology of shock wave focusing when a three-dimensional contraction occurs.

In our study we will consider an ideal, inviscid gas which can lead to some distortion of our results compared with experimental data. However, we believe that this simplification will still capture the main phenomenology of wave formation and reflection, and will be of general value to the Mach Reflection Theory.

Numerical Method

In Reference 8 we introduced a new numerical algorithm: FUGGS (Fast Unstructured Second Order Godunov Solver), for solving Euler's equations of gasdynamics on unstructured triangular grids. The algorithm formulated and tested in Reference 8 is vertex-based. Here we will describe a new volume based version of the FUGGS method. The new version of the algorithm as illustrated in our paper, produces considerably more accurate solutions and it is more efficient. This contradicts published results (Ref. 10,11) on implementation of the triangle-based TVD schemes for unstructured triangular grids. The new algorithm has been validated for the range of subsonic, supersonic and hypersonic steady state and transient problems. Here we show only results for Mach Reflection in planar and axisymmetric channels..

The new triangle-based version of the FUGGS algorithm was extended to allow dynamic adaptive grid refinement for transient problems. We will give a description of the dynamic grid adaptation methodology used in FUGGS code.

A three dimensional version of the FUGGS algorithm was developed in an extremely short period of time. This was made possible by the simple structure of the basic algorithm. We will not present simulation results for the three dimensional FUGGS, however, the main elements of the FUGGS algorithm implementation in the three dimensions will be illustrated.

Vertex-Based and Triangle-Based Integration Algorithms

We consider a system of Euler equations written in conservation law form in three dimensions as:

$$\frac{\partial \mathbf{U}}{\partial t} + \frac{\partial \mathbf{f}}{\partial x} + \frac{\partial \mathbf{g}}{\partial y} + \frac{\partial \mathbf{h}}{\partial z} = 0 \quad (1)$$

where

$$\mathbf{U} = \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ \rho E \end{pmatrix}, \quad \mathbf{f} = \begin{pmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ \rho uw \\ \rho uH \end{pmatrix}, \quad \mathbf{g} = \begin{pmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ \rho vw \\ \rho vH \end{pmatrix}, \quad \mathbf{h} = \begin{pmatrix} \rho w \\ \rho uw \\ \rho vw \\ \rho w^2 + p \\ \rho wH \end{pmatrix}$$

Here u , v , and w are the x , y , and z velocity vector components, p is the pressure, ρ is the density and H is the total enthalpy and E is total energy of the fluid. It is assumed that a mixed (initial conditions, boundary conditions) problem is properly posed for the set of equations (1), that an initial distribution of the fluid parameters is given at $t=0$, and the boundary conditions defining a unique solution are specified for the computational domain.

We seek a solution of the system of equations (1) on the computational domain which is decomposed into tetrahedrons (triangles in two dimensions) with arbitrary connectivity. An overwhelming advantage of this method of domain decomposition is the ability to resolve extremely complicated geometries and flow regimes accurately and efficiently. This has been demonstrated in numerous publications on this topic (Ref. 12, 13, 14).

There are several options possible for storing natural physical parameters of the problem on an unstructured tetrahedral or triangular grid. In particular, we have examined: i) vertex centered; and ii) tetrahedron (or triangle) centered. These two approaches, while equivalent from the point of view of the formal numerical representation of the governing equations, lead to different algorithms. As shown below, this will have important consequences not only on data structure and algorithm efficiency, but moreover the different connectivity will affect the overall accuracy of the numerical solution.

In Figure 4, a fragment of the two-dimensional computational domain is shown. Here, together with the original triangular grid (solid lines), the secondary grid (broken line) is shown. This secondary grid is formed by connecting the barycenters of the primary grid. If a vertex based grid is used, the physical parameters of the problem are stored at vertices A, B, C..., and the integration is done for the volumes delineated by the polygons of the secondary grid. For instance, integration volume associated with vertex A is defined by the edges ab, bc, cd, de, ef, fa. For a triangle-based grid the physical parameters will be stored at the nodes of the secondary grid, and integration volume will be the triangle itself. We have shown (Ref. 15) that these two approaches lead to numerical algorithms with different connectivity, accuracy and efficiency. The fundamental algorithm of the second order Godunov method implemented in FUGGS can be illustrated in two dimensions for an edge of the grids control volume shown in Figure 5. The algorithmic steps of the second order Godunov method can be defined as follows:

1. Find the value of the gradient at the vertex point (or at the baricenter of the triangle for the triangle-based version) for the gasdynamic Parameter U;
2. Using the gradient values, find the interpolated values of U at the edges defining the control volume (sides of the triangle for the triangle-based scheme)
3. Limit these interpolated values based on a monotonicity condition (Ref. 16)
4. Subject the resulting values to the characteristic's constraints (Ref. 6)
5. Solve the Riemann problem for the corrected values.

This last step completes the definition of the fluxes at the edges of the control

volume. The flux values can be stored at the edges and the flux calculation loop will be arranged for the list of edges, which is the largest vector in the system. If the algorithm is vertex-based to calculate U^{n+1} values, we will integrate the fluxes at the edges of the secondary grid which define the control volume for the vertex. For the triangle-based algorithm U^{n+1} , value is obtained by integrating the fluxes at the sides of the triangles.

Implementation of the algorithm in three dimensions will have the same basic steps in flux calculation 1-5. To illustrate that point, Figure 6 shows a tetrahedral element of the grid. Here the fluxes are defined on the faces of the tetrahedral at the edge points. At step 1 the gradient is calculated at the barycenter cell point for the tetrahedral. All the rest of the steps are identical to those described above. To find the value of U^{n+1} in the three dimensional case, we will add fluxes defined at the faces of the tetrahedral. Most elements developed for the two dimensional code are applicable to this implementation of the three dimensional algorithm.

Direct Dynamic Refinement Method (DDRM)

Practical numerical simulations of the fluid dynamic problems call for modeling flows over complicated shapes. In addition, important flow features such as shed vortices, shock waves, slip lines and boundary layers usually have widely varied lengths and time scales and need to be resolved. Accurate solution of these problems require computational grids dynamically adapted to the evolving flow feature, and with full control over solution accuracy in the key regions of the computational domain. It is commonly accepted that only unstructured grids can provide full flexibility in obtaining the local grid resolution sufficient to accurately resolve subscale flow features. The five years since the introduction of these grids and methods in CFD research have produced landmark simulations clearly demonstrating their advantages (Ref. 12, 14, 17).

Although a number of research groups have demonstrated application of unstructured grids to simulations of steady state problems (Ref. 14, 17, 18), simulations of time-dependent problems were accomplished by a significantly smaller group (Ref. 19, 20). An adaptive refinement method developed by Lohner (Ref. 20) is based on a hierarchical system of grid refinement/coarsening in which each level of refinement has six possible cases and coarsening three cases of triangular cells formation. Every layer of refinement has a father/son relation with the previous layer, and all these layers of refined mesh move on the basic predefined grid. This technique has the demonstrated capability of carrying out simulations of extremely complex flow regimes. However, its rigid hierarchic approach to generating grid results in some implicit limitations. For example, a dynamically evolving grid will not have an element larger than the cell of the initial grid, or it will be impossible to reduce the cell volume abruptly in some areas without passing through all the necessary level of refinement.

In our paper we will report a new method of dynamic grid adaptation. This method is based on direct refinement and reconnection in the areas of monotonic flow preceding the regions with strong flow gradients. In Figure 7 we have illustrated the basic process of refinement accomplished in the DDRM method. The original grid is shown in Figure 7a. Figure 7b illustrates a one step grid refinement in which a new vertex is introduced into a triangular cell forming three new cells. This is followed by reconnection which modifies the grid in a manner demonstrated in Figure 7c. The process of refinement and reconnection can be continued until the necessary grid resolution is achieved, as illustrated in Figures 7d and 7e. This direct approach to the grid refinement grants extreme flexibility in resolving local flow features. A similar simple method is applied to grid coarsening. In the first step of coarsening the marked vertices, all associated elements of the grid are simply removed, as shown in Figure 8a. During the second step, this void in the grid is filled with new larger triangles (Figure 8b), and then reconnected as shown in Figure 8c.

The Direct Dynamic Refinement Method (DDRM) was implemented for the second order-Godunov method (FUGGS algorithm Ref. 7, 15). Here we demonstrate its performance for a classical Mach Reflection problem.

Results

In Figures 8a, 8b, and 8c, simulation results are shown in the form of density contours for different stages of Mach Reflection in a planar channel. To illustrate the dynamics adaptation of the computational grid to the solution in the same figures, we show the grid as it evolves in time. The numerical solution develops as a classical case of Mach Reflection. Because we have assumed that the gas is ideal with $\gamma = 1.4$, according to Ben-Dor and Glass (Ref. 9) for the shock wave and wedge angle conditions chosen we should have a case of Complex Mach Reflection (CMR). We can observe in Figures 8a, 8b and 8c that the density contours definitely display the pattern of discontinuities attributed to CMR. In these figures we observe a well defined slip line vortex, slip line, triple point and the kink. For real gas, in this case, the Double Mach Reflection should occur. The slip line and slip line vortex will be located close to the Mach shock and will cause the Mach shock bulging (Ref. 5). However the perfect gas assumption will lead to CMR and extensive bulging will not arise, as is accurately predicted in our simulations. It is striking to observe in Figures 8a, 8b, and 8c that the numerical grid closely follows the evolving system of waves, and the high density grid is only observed in the areas of shock waves, slip lines and other flow discontinuities. The result is tremendous savings in both CPU and storage. For example, the grid shown in Figure 8c has only 6000 points (an equivalent of a grid 60x100 in the case of a structured rectangular grid).

Reflection of Mach shock from the wall opposite the wedge will start immediately after the stage shown in Figure 8c. This reflection results in formation of the secondary Mach Reflection which expands towards the channel's oblique wall. In Figure 9a, this secondary Mach Reflection can be clearly identified. In Figure 9b, the blow up of the region of the secondary Mach Reflection is shown. All the distinguishing characteristics of the Mach Reflection can be identified in this figure, including triple point, Mach shock, reflected shock and slip line. In addition to all these features, the secondary Mach Reflection has an additional kink, resulting from interaction of the reflected shock with the slip surface. It is clear that this interaction will affect significantly the dynamics of the secondary reflection.

In Figures 10a, 10b, and 10c, simulation results are shown for the Mach Reflection in an axisymmetric channel which has the same cross section as the planar channel. For direct comparison here the simulation results are presented in the same format as in Figures 8a, 8b, and 8c for the case of a Mach Reflection in a planar channel. The Mach Reflection in Figure 10a is analogous to its planar counterpart in Figures 8a and 8b. In Figure 10b it can be observed that the area of the shock between the triple point and the kink in the reflected shock tilts towards the axis of symmetry of the channel. This is even more pronounced in Figure 10c where the density contours are shown before the secondary reflection starts. It is apparent that the secondary reflection of the Mach shock will occur earlier in the axisymmetric channel than in its planar counterpart. Contraction in the radial direction results in a significant jump in density upon reflection. In Figure 11a we see that at the initial stages of the axisymmetric reflection, maximum density increased three-fold compared with the values observed in the initial reflection. This increase in density affects the increment between the contour levels displayed in Figure 11a and causes the slip line not to show. In Figure 11c a more advanced stage of the secondary reflection is shown. To examine in more detail the features of the secondary reflection in Figures 11b and 11d, we show an enlarged view of the secondary reflection region corresponding to Figures 11a and 11c. In these figures, we can observe the formation of a distinct reflected wave pattern with a characteristic double kink of the reflected wave similar to that seen in the secondary reflection in a planar channel. In the axisymmetric case, the secondary reflection is significantly stronger than in the case of a planar channel. Since this reflected wave propagates along the radius of the channel, it will expand rapidly. This can be observed in Figure 11c where the maximum value of density dropped 30% compared with the maximum in Figure 11a. For the same reason the triple point of the secondary Mach Reflection has advanced much farther towards the oblique wall in Figure 11d than in Figure 9b.

Conclusions

A computer code has been developed for Euler's equations of gas dynamics. This code uses unstructured grids for computational domain decomposition and its integration algorithm is based on the Second Order Godunov method. The code uses the Dynamic Grid Adaptation methodology, allowing economical allocation of computer resources to evolving flow features. In turn, it is then possible to carry out accurate simulations of complicated gas dynamic phenomena with affordable computer resources. Here the code has been demonstrated to produce an accurate simulation of Complex Mach Reflection in planar and axisymmetric channels. We also have simulated the initial stages of the secondary Mach Reflection from the channel wall opposite the oblique wall. In this case we observed new wave structures with a characteristic double kink. The formation of the second kink was a result of the interaction between the secondary reflected wave and the original slip line. It was noted that the dynamics of the secondary reflection is different in the planar and axisymmetric cases. In the axisymmetric case reflection is significantly stronger than in the planar case.

References

1. Ben-Dor, G., and I.I. Glass, "Nonstationary Oblique-Shock Wave Reflections: Actual Isopycnics and Numerical Experiments," *AIAA J.* 16 (1978), pp. 1146-1153.
2. Gvozdeva, L.G., T.V. Bazhenova, O.A. Predvoditeleva, V.P. Fokeev, 1969. Mach Reflection of Shock Waves in Real Gases, *Astron. Acta* 14:503-8.
3. Hornung, H.G., H. Oertel, R.J. Sandeman, 1979. Transmission to Mach Reflexion of Shock Waves in Steady and Pseudosteady Flow with and without Relaxation. *J. Fluid Mech.* 90:541-60.
4. Bazlenova, T.V. and L.G. Gvozdeva, "Unsteady Interactions of Shock Waves," Nauka, Moscow, 1977.
5. Glaz, H.M., P. Colella, I.I. Glass, and R.L. Deschambault, A Detailed Numerical, Graphical, and Experimental Study of Oblique Shock Wave Reflections, DNA-TK-86-365, 1986.
6. Woodward, P.R., and P. Colella, "Numerical Simulation of Two-Dimensional Fluid Flow with Strong Shocks," *J. Comp. Phys.* 54 (1984), pp. 115-173.
7. Eidelman, S., P. Colella, and R.P. Shreeve, "Application of the Godunov Method and Its Second Order Extension to Cascade Flow Modeling," *AIAA Journal*, v. 22, 10 (1984).
8. Lottati, I., S. Eidelman and A. Drobot, "A Fast Unstructured Grid Second Order Godunov Solver (FUGGS)", 28th Aerospace Sciences Meeting, AIAA-90-0699, Reno, NV, 1990.
9. Ben-Dor, G. and I.I. Glass, 1979 Domains and Boundaries of Nonstationary Oblique Shock-Wave Reflexions: 1. Diatomic gas. *J. Fluid Mech.* 92:459-96.
10. Barth, T.J. and D.C Jespersen, "The Design and Application of Upwind Schemes on Unstructured Meshes," 27th Aerospace Sciences Meeting, AIAA-89-0366, Reno, NV, (1989).
11. Mavriplis, D. and A. Jameson. "Multigrid Solution of the Two-Dimensional Euler Equations on Unstructured Triangular Meshes," AIAA-87-0353, 1987.
12. Baum, J.D. and R. Löhner, "Numerical Simulation of Shock- Elevated Box Interaction Using an Adaptive Finite Element Shock Capturing Scheme," AIAA Preprint 89-0653, Presented at the AIAA 27th Aerospace Sciences Meeting, Jan. 8-12, 1989. Reno, NV.

13. Löhner, R., K. Morgan, J. Peraire and M. Vahdati, "Finite Element Flux-Corrected Transport (FEM-FCT) for the Euler and Navier-Stokes Equations" Chapter 6 in Finite Elements in Fluids Vol. VII (R.H. Gallagher, et al. eds.), J. Wiley and Sons (1988).
14. Mevriplis, D.T., "Accurate Multigrid Solutions of the Euler Equations on Unstructured and Adaptive Meshes," AIAA Journal, 2, V. 28, p. 231, 1990.
15. Eidelman, S. and I. Lottati, "Triangle Based FUGGS and its Validation for Two and Three Dimensional Flow Problems," to be presented at 29th Aerospace Sciences Meeting, Reno, NV, 1991.
16. van Leer, B., "Towards the Ultimate Conservative Difference Scheme, V.A. Second Order Sequel to Godunov's Method," J. Comp. Phys. v. 32, 101-136 (1979).
17. Jameson, A., T.J. Baker and N.P. Weatherill, "Calculations of Inviscid Transonic Flow Over a Complete Aircraft." AIAA 24th Aerospace Sciences Meeting, Reno, NV, AIAA Paper 86-0103, January 1986.
18. Peraire, J., M. Vahdati, K. Morgan and O.C. Zienkiewicz - Adaptive Remeshing for Compressible Flow Computations; J. Comp. Phys. 72, 449-466 (1987).
19. Palmerio, B. and A. Dervieux - Application of a FEM Moving Node Adaptive Method to Accurate Shock Capturing; Proc. First Int. Conf. on Numerical Grid Generation in CFD, Landshut, W. Germany, July 14-17, 1986, Pineridge Press.
20. Löhner, R. - Adaptive Remeshing for Transient Problems; Comp. Meth. Appl. Mech. Eng. 75, 195-214 (1989).

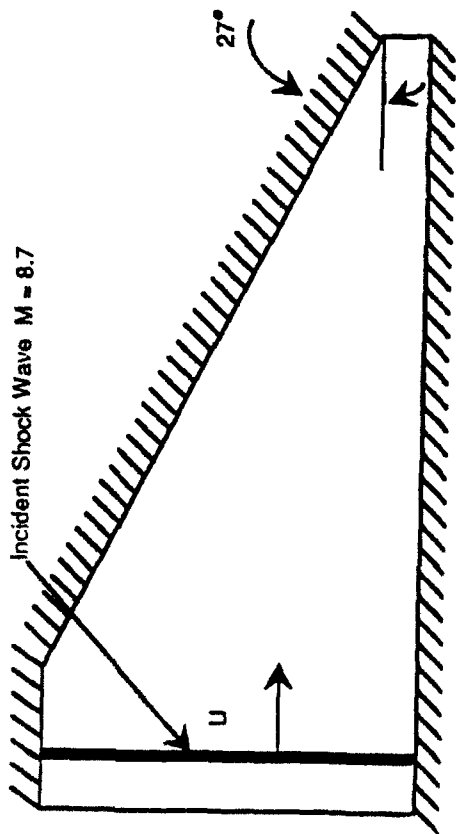


Figure 1. Channel Geometry and Initial Shock Location.

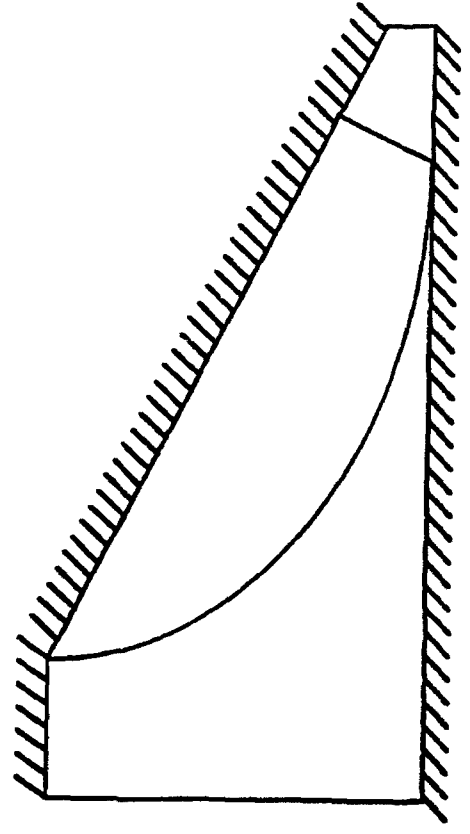


Figure 2. Reflected and Mach Stem Shock Waves at the Start of the Secondary Reflection.

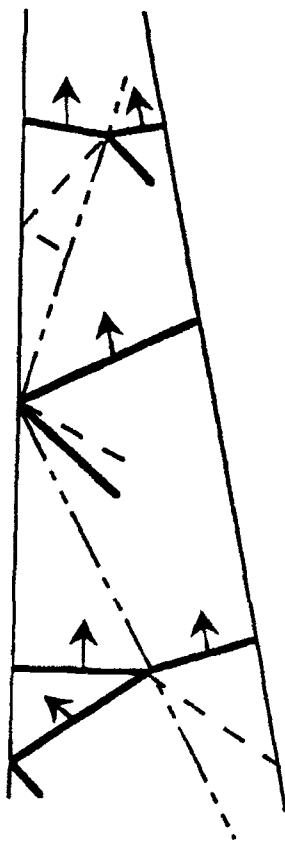


Figure 3. Schematics of Wave Propagation in a converging channel according to Bazhenova and Gvozdeva.

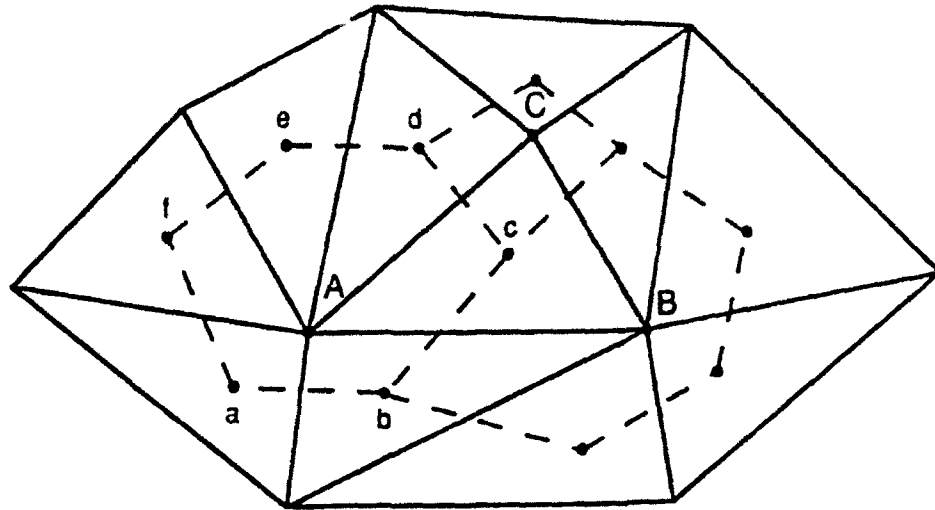
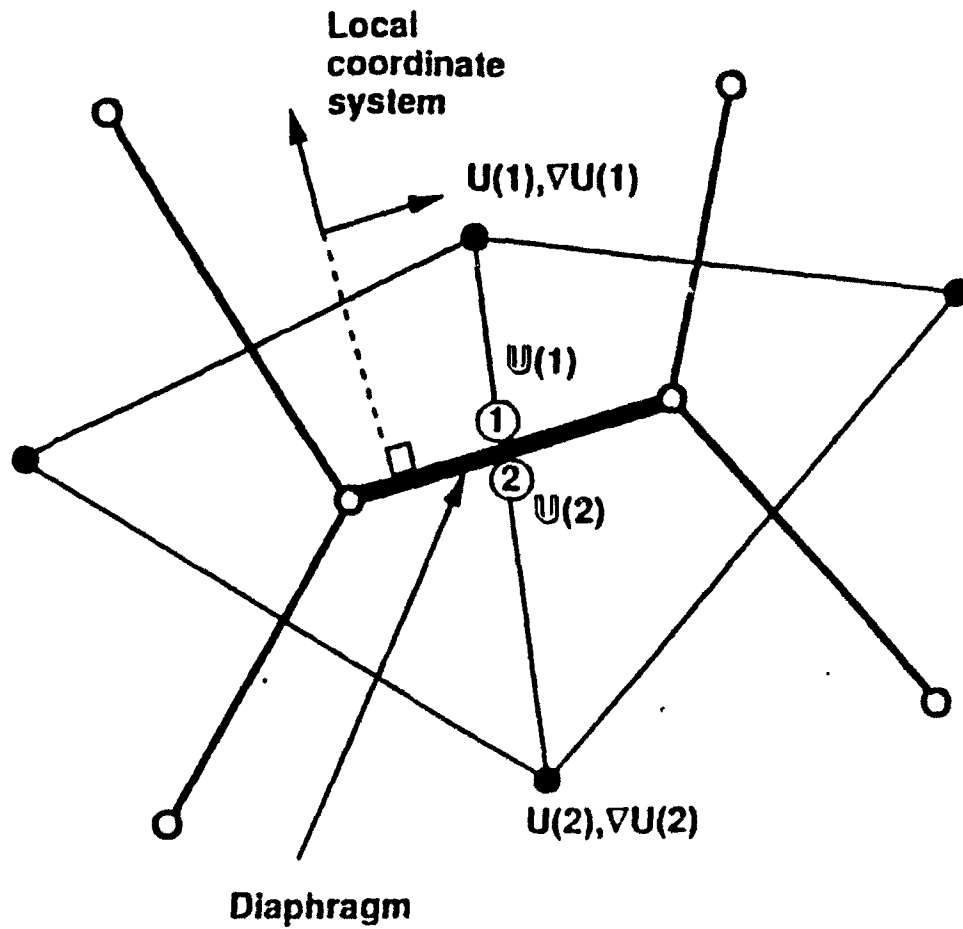
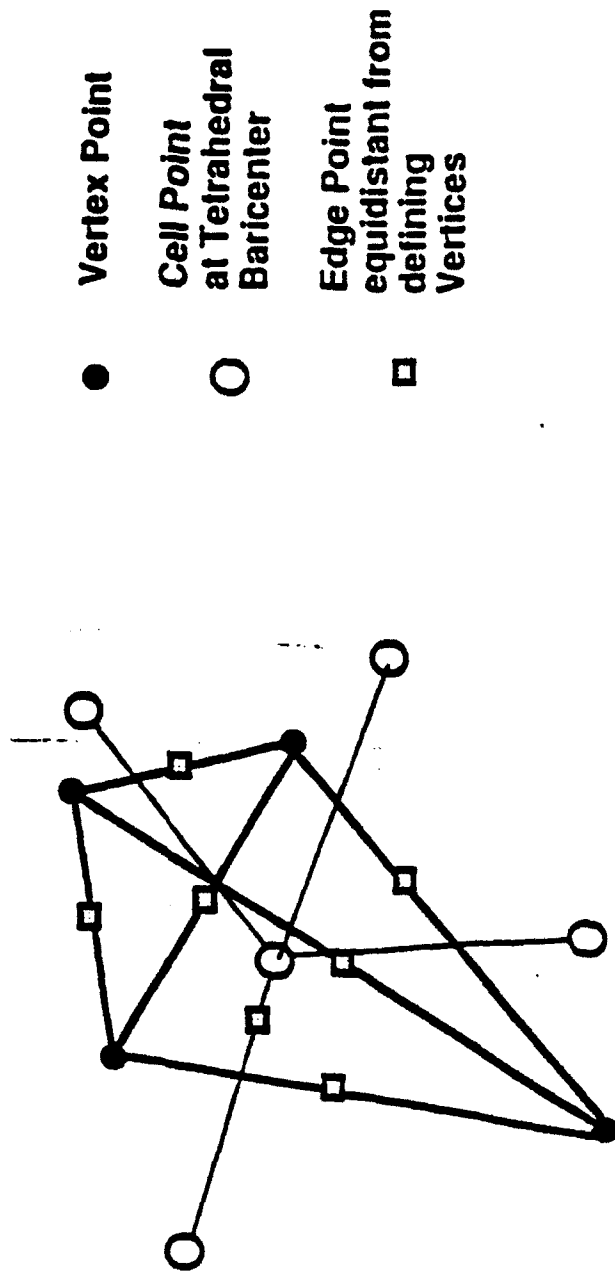


Figure 4.



Second Order Edge Based Flux Calculation

Figure 5.



● Vertex Point

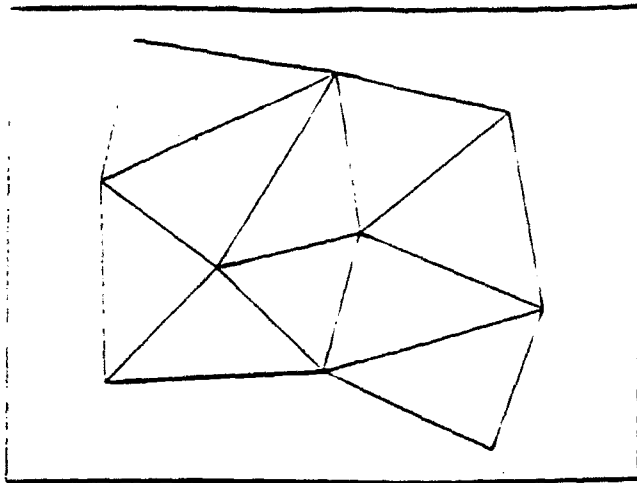
○ Cell Point
at Tetrahedral
Baricenter

□ Edge Point
equidistant from
defining
Vertices

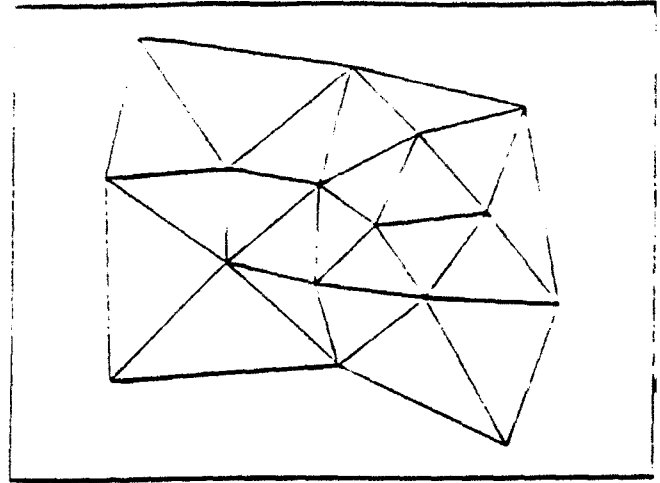
**Tetrahedral Element
Defined by four Vertices
and Baricentric Cell Point**

Scheme for Baricentric Three Dimensional Integration

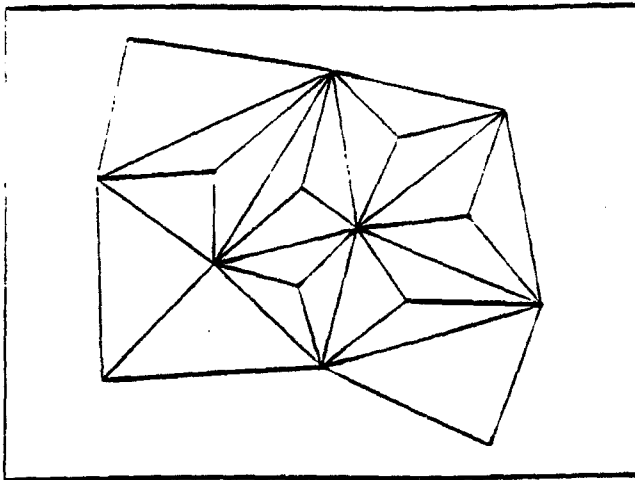
Figure 6.



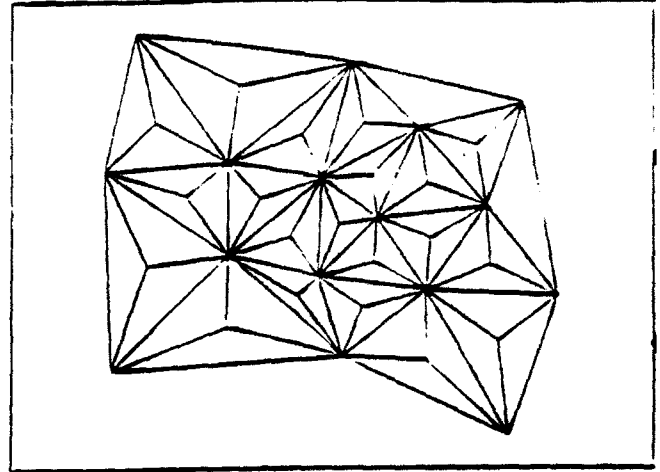
a. Original grid.



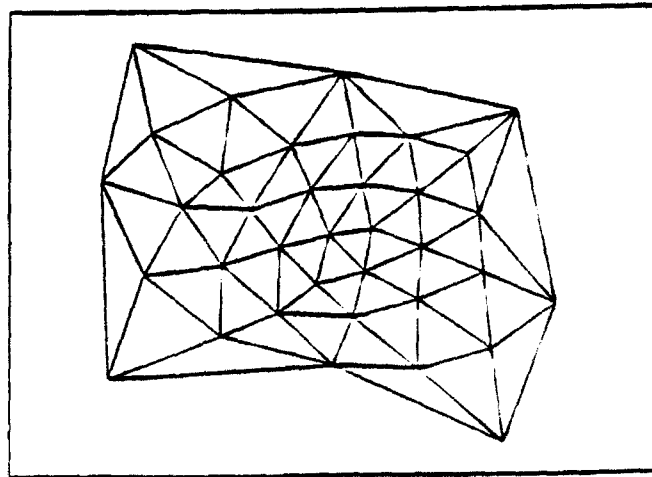
c. Grid after one refinement and one reconnection.



b. Grid after one refinement.

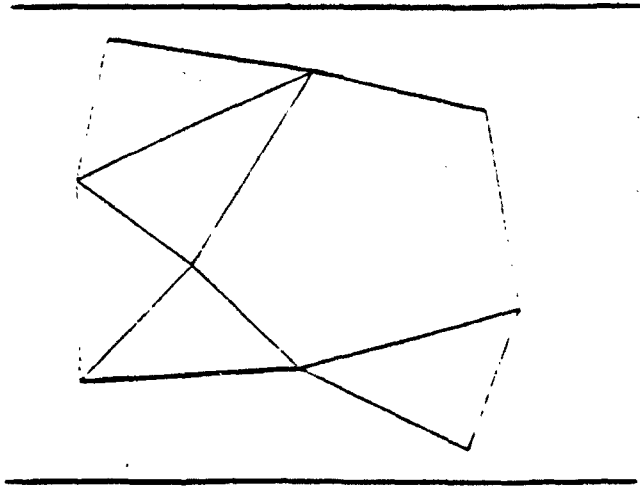


d. Second refinement.

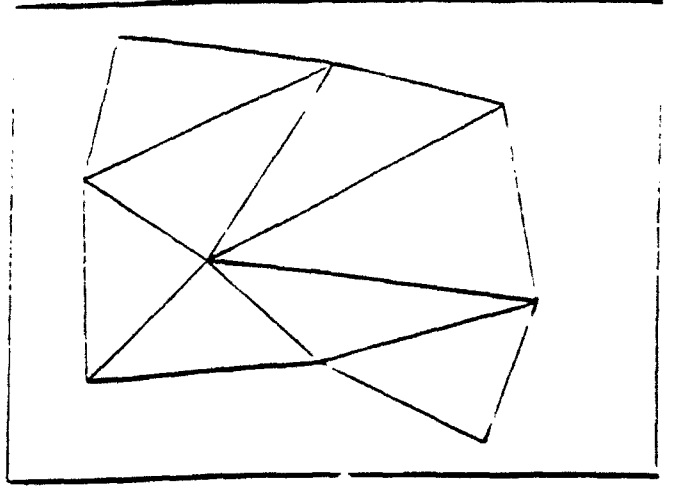


e. Second reconnection.

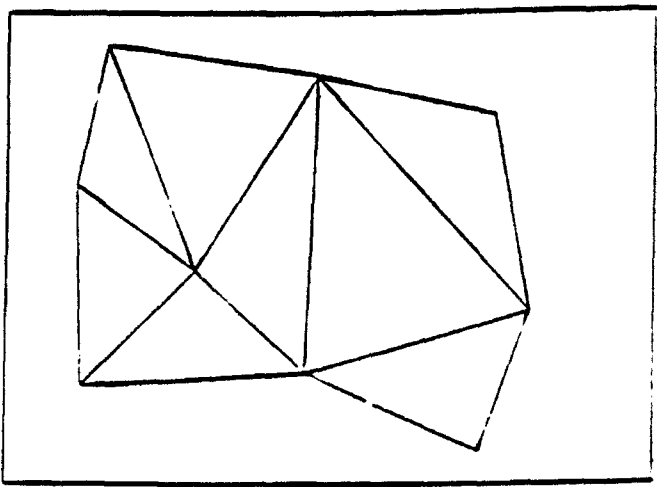
Figure 7. Illustration of the grid refinement process.



a. Point removal.



b. Construction of new cells.



c. Final coarse grid after reconnection.

Figure 8. Illustration of the grid coarsening process.

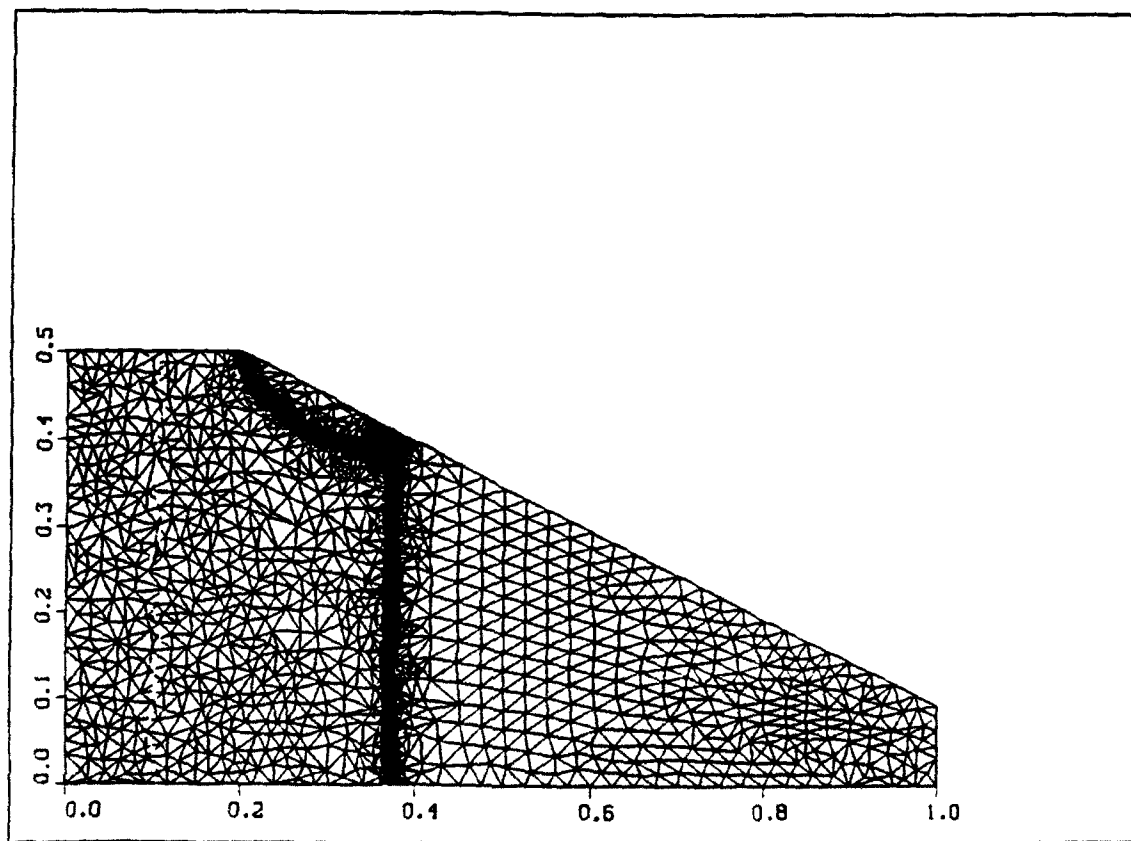
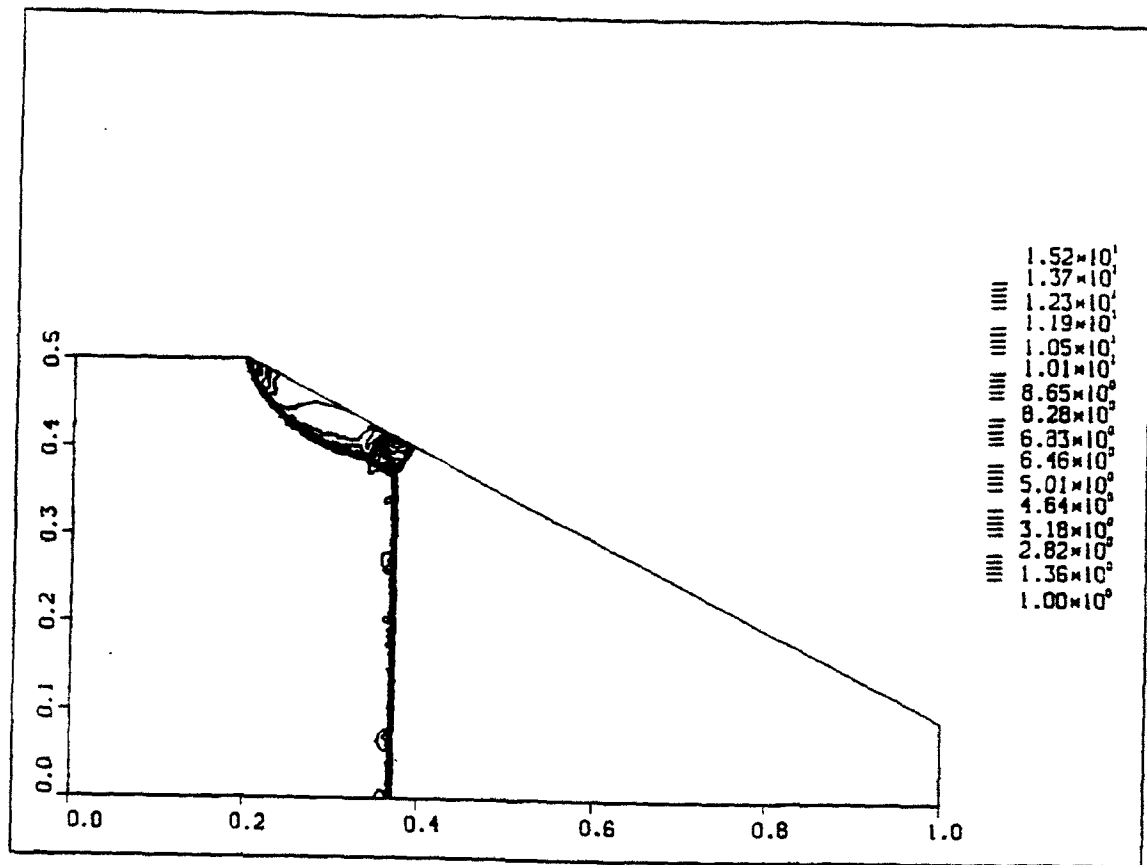
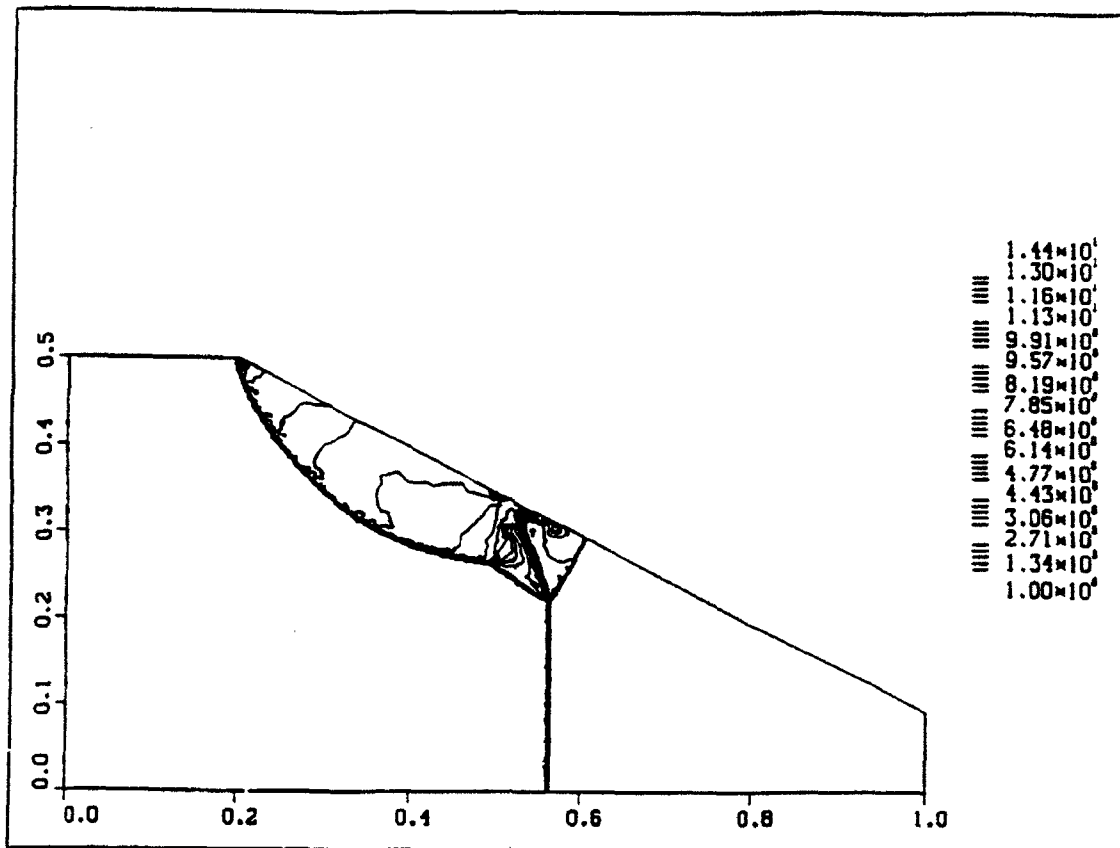
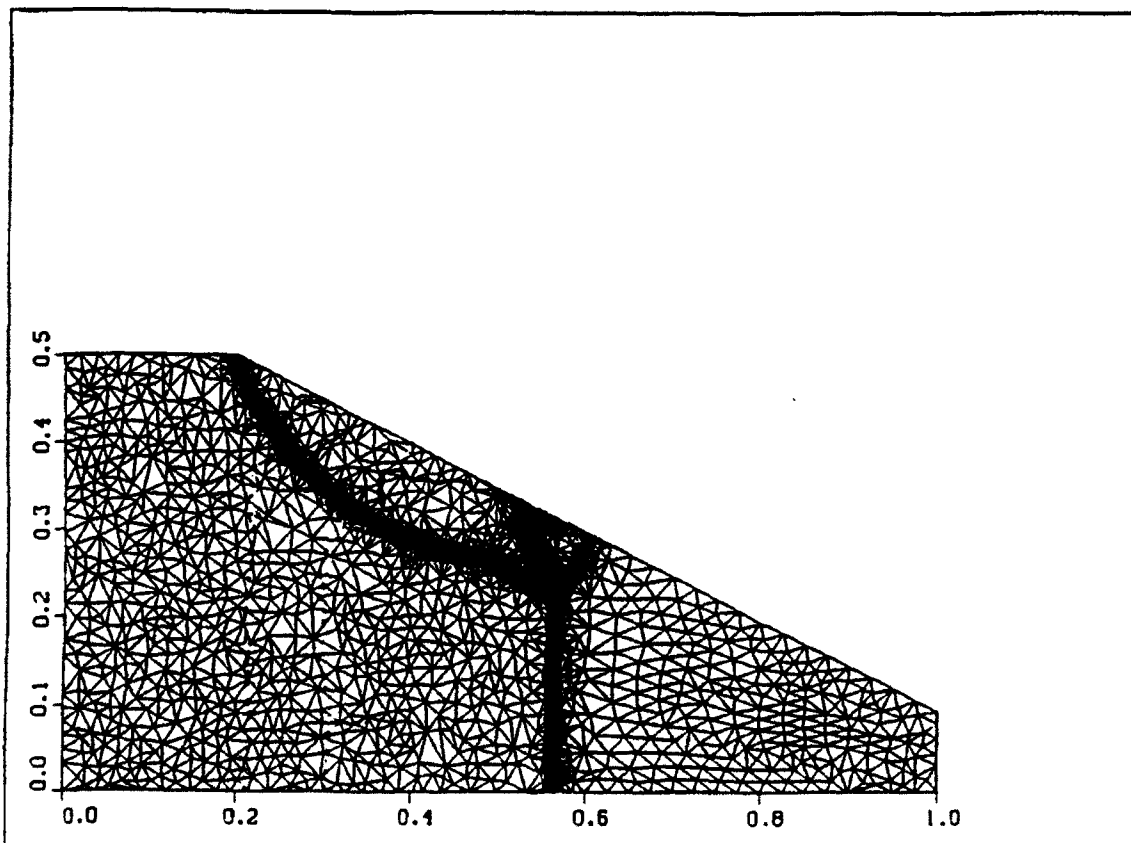


Figure 8a. Mach Reflection in a planar channel. $M_1 = 8.7$; $\alpha = 27^\circ$.

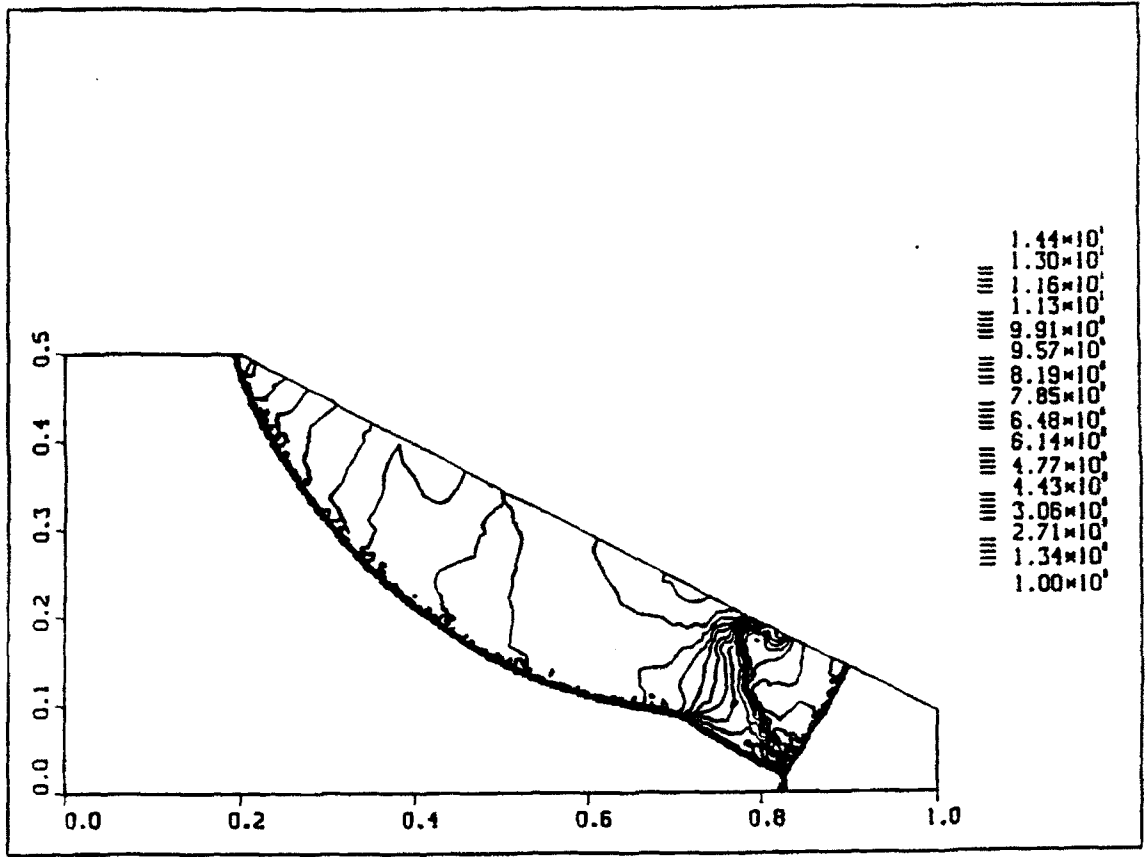


Density contours

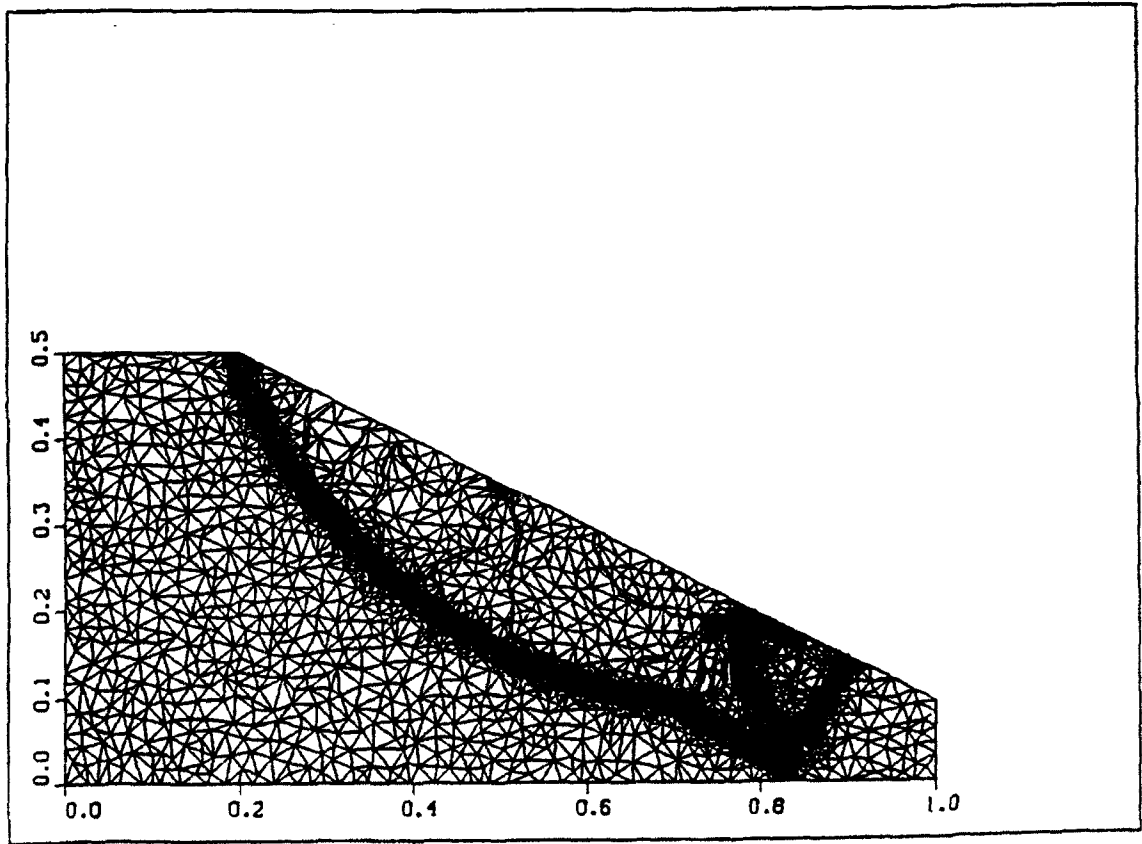


Grid

Figure 8b. Mach Reflection in a planar channel. $M_1 = 8.7$; $\alpha = 27^\circ$.



Density contours



Grid

Figure 8c. Mach Reflection in a planar channel. $M_\infty = 8.7$; $\alpha = 27^\circ$.

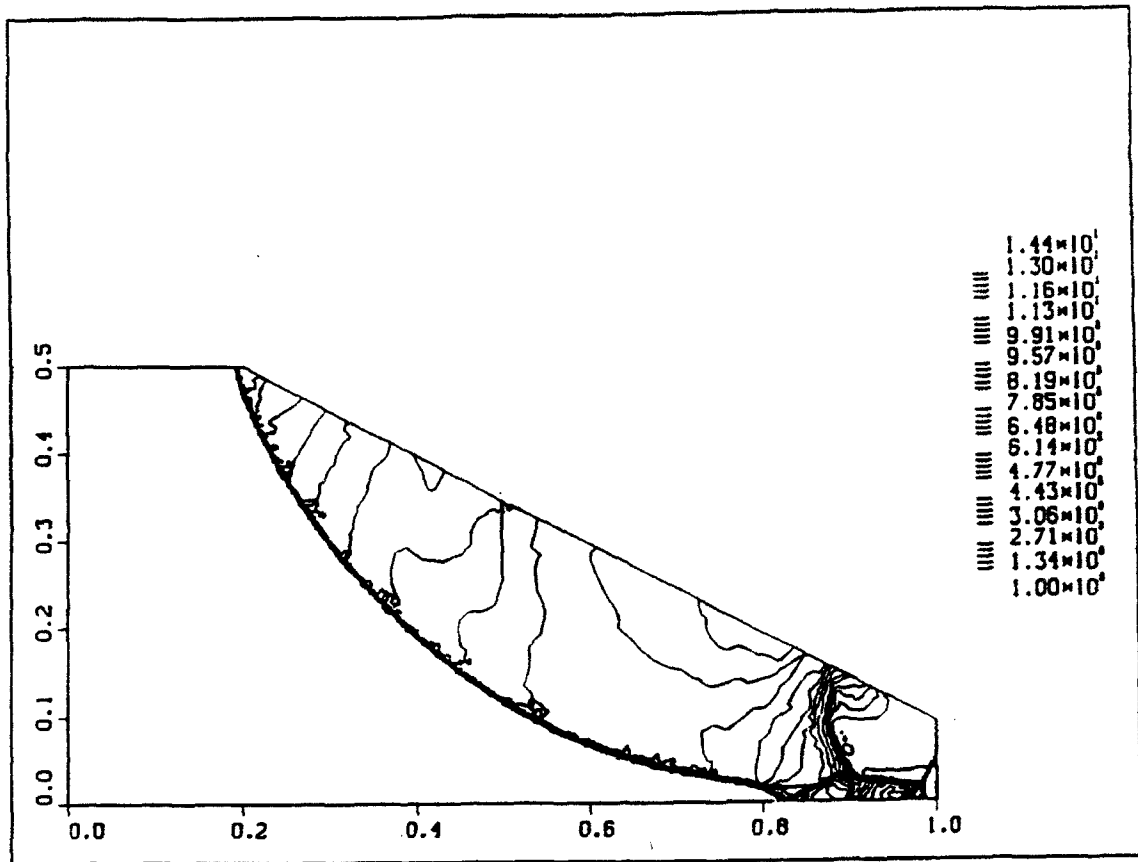
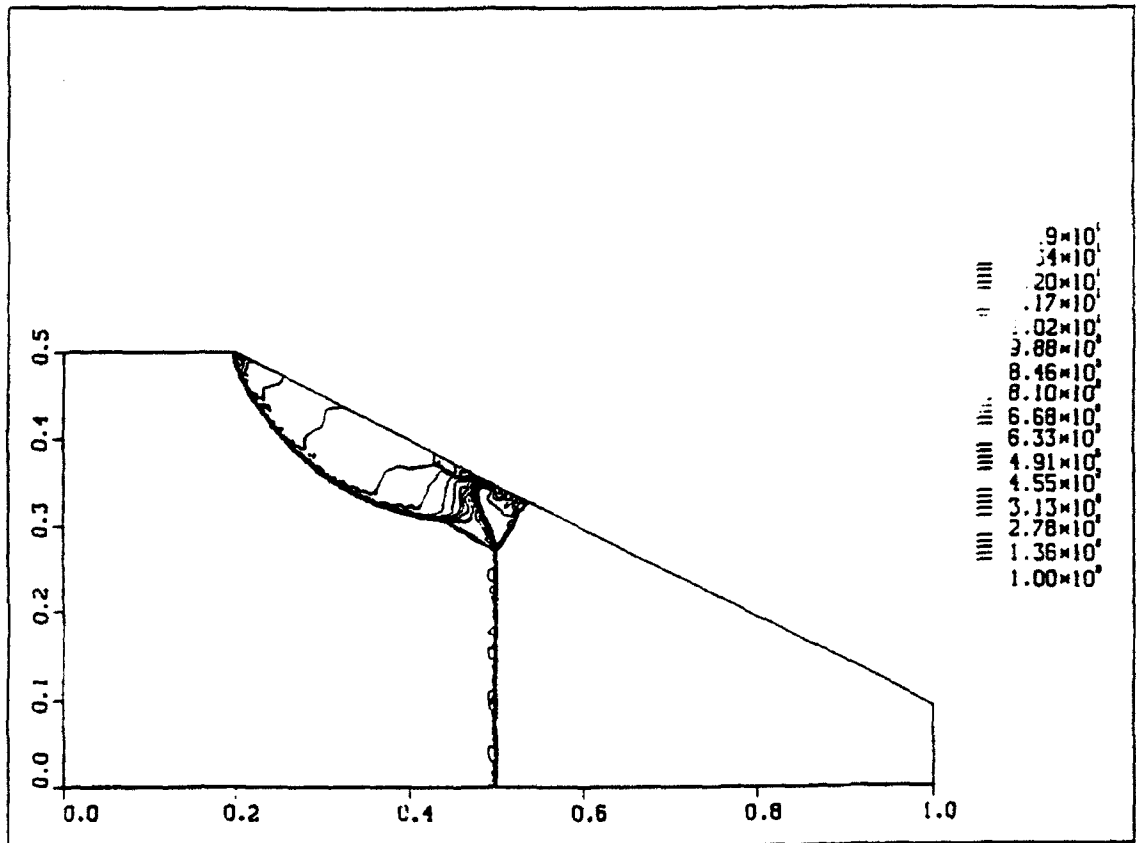


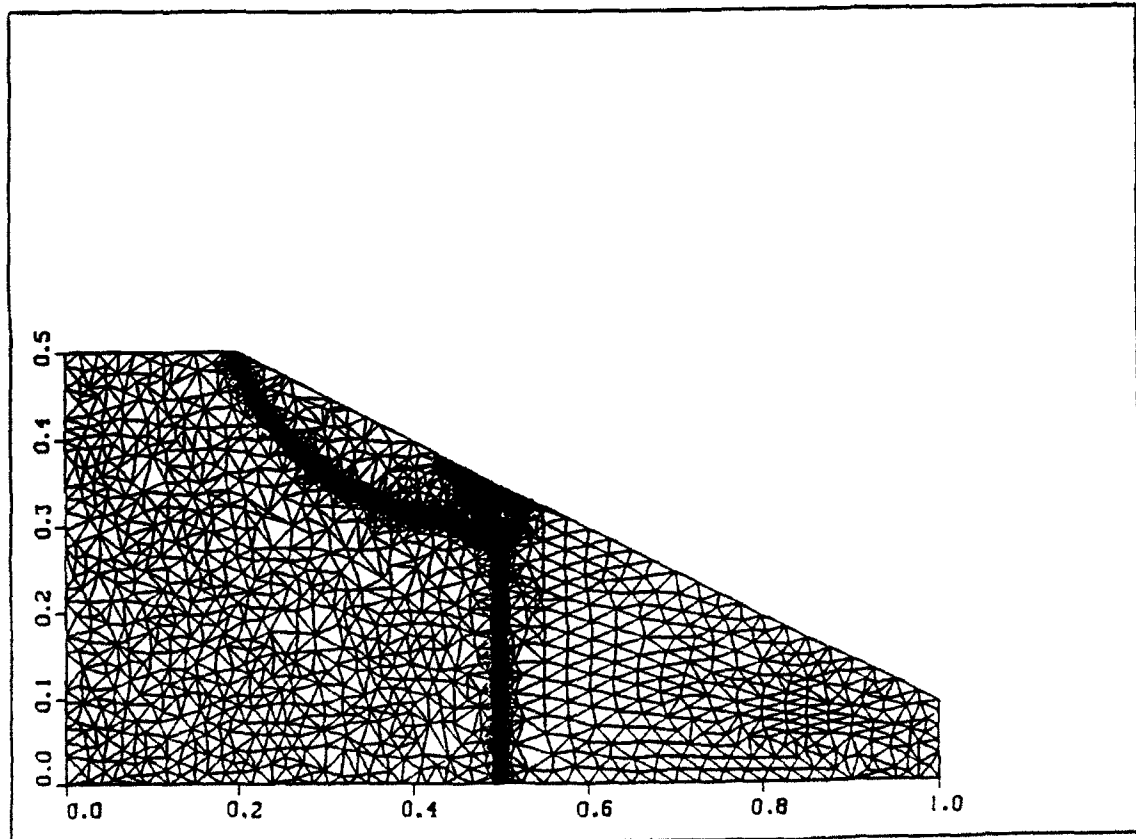
Figure 9a. Secondary Mach Reflection in a planar channel. Density contours.



Figure 9b. A blown up view of the secondary Mach Reflection shown in Figure 9a.

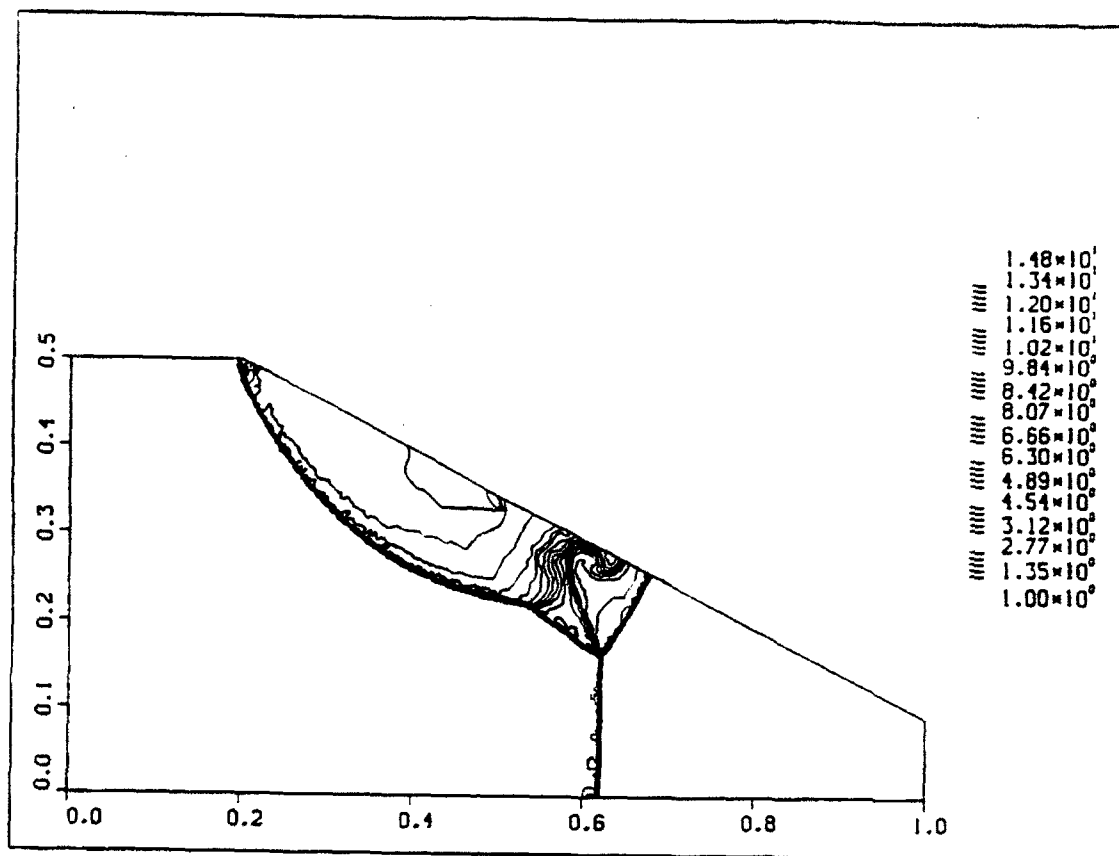


Density contours

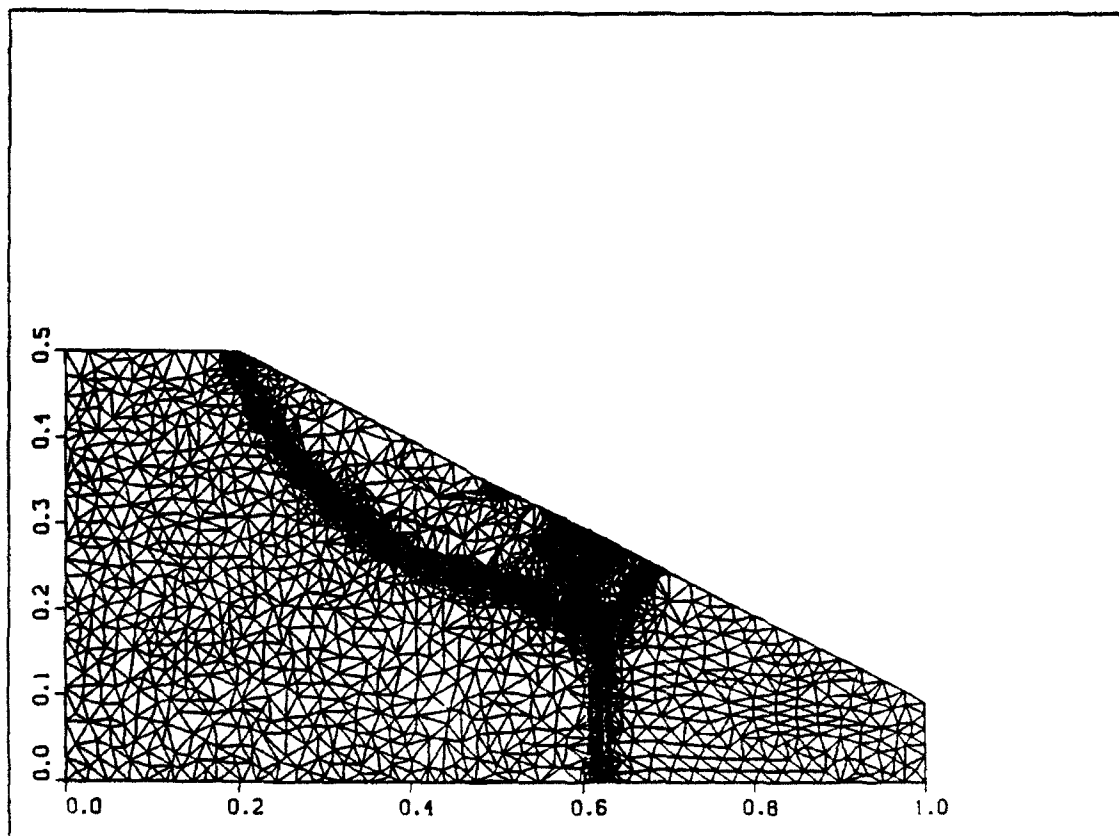


Grid

Figure 10a. Mach Reflection in an axisymmetric channel. $M_1 = 8.7$; $\alpha = 27^\circ$.

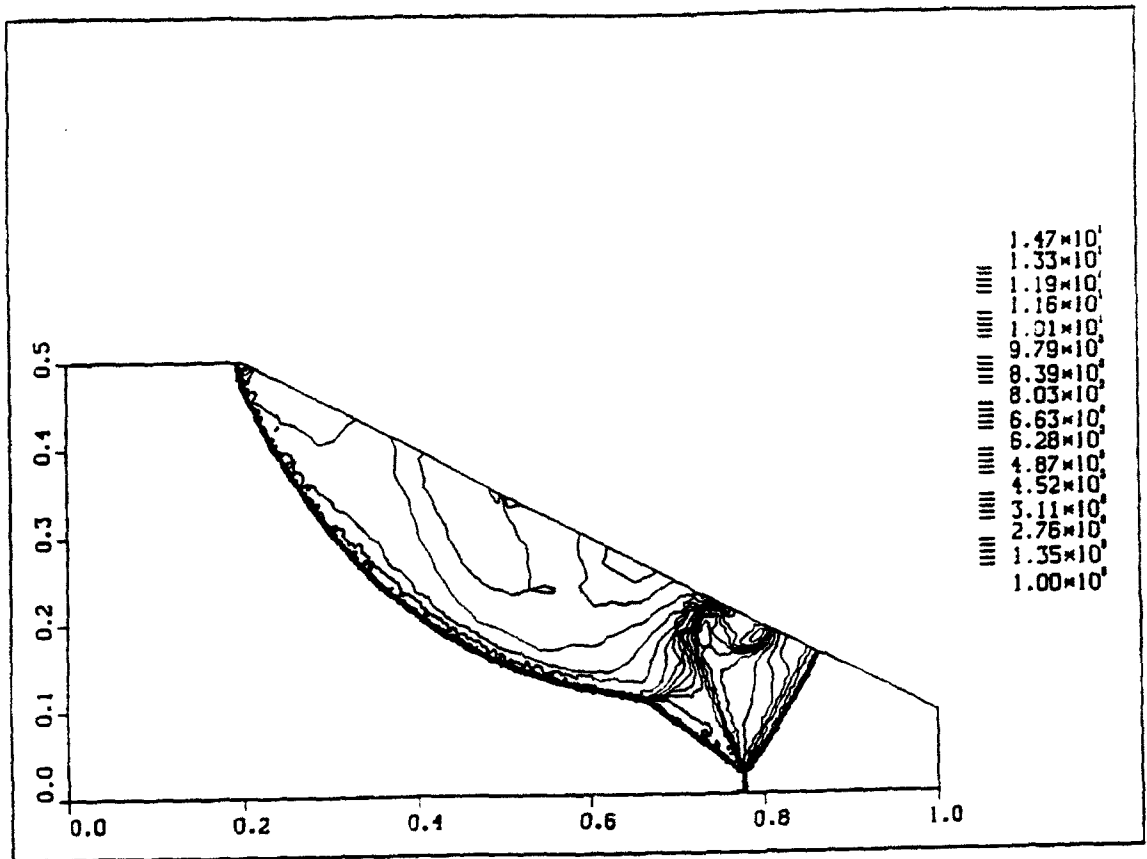


Density contours

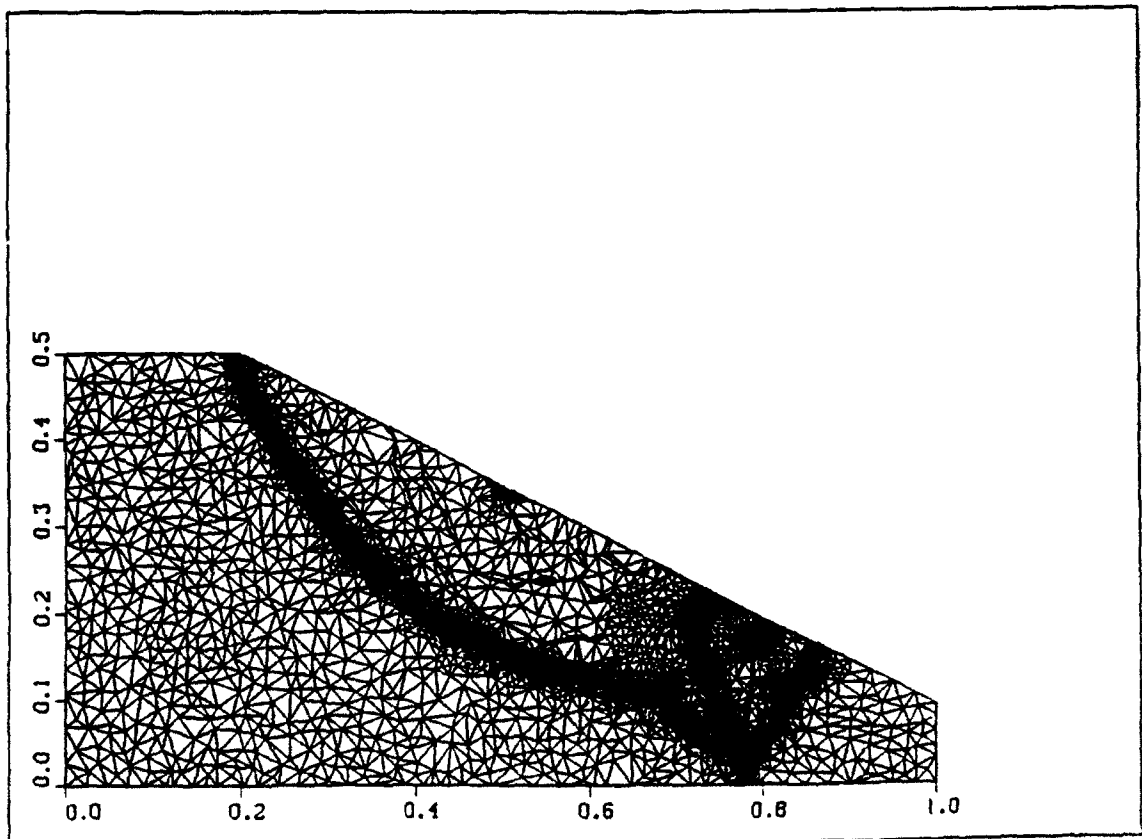


Grid

Figure 10b. Mach Reflection in an axisymmetric channel. $M_1 = 8.7$; $\alpha = 27^\circ$.



Density contours



Grid

Figure 10c. Mach Reflection in an axisymmetric channel. $M_1 = 8.7$; $\alpha = 27^\circ$.

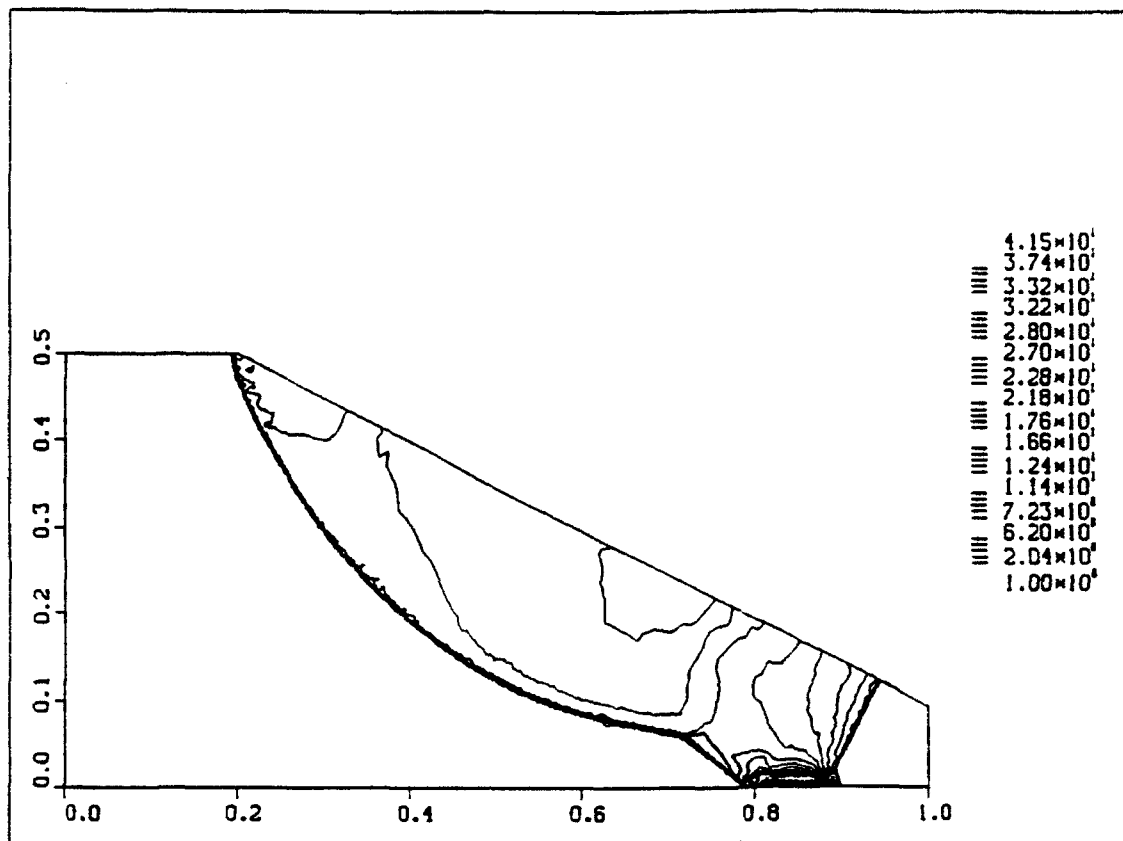


Figure 11a. Start of the secondary Mach Reflection. Axisymmetric channel. Density contours.

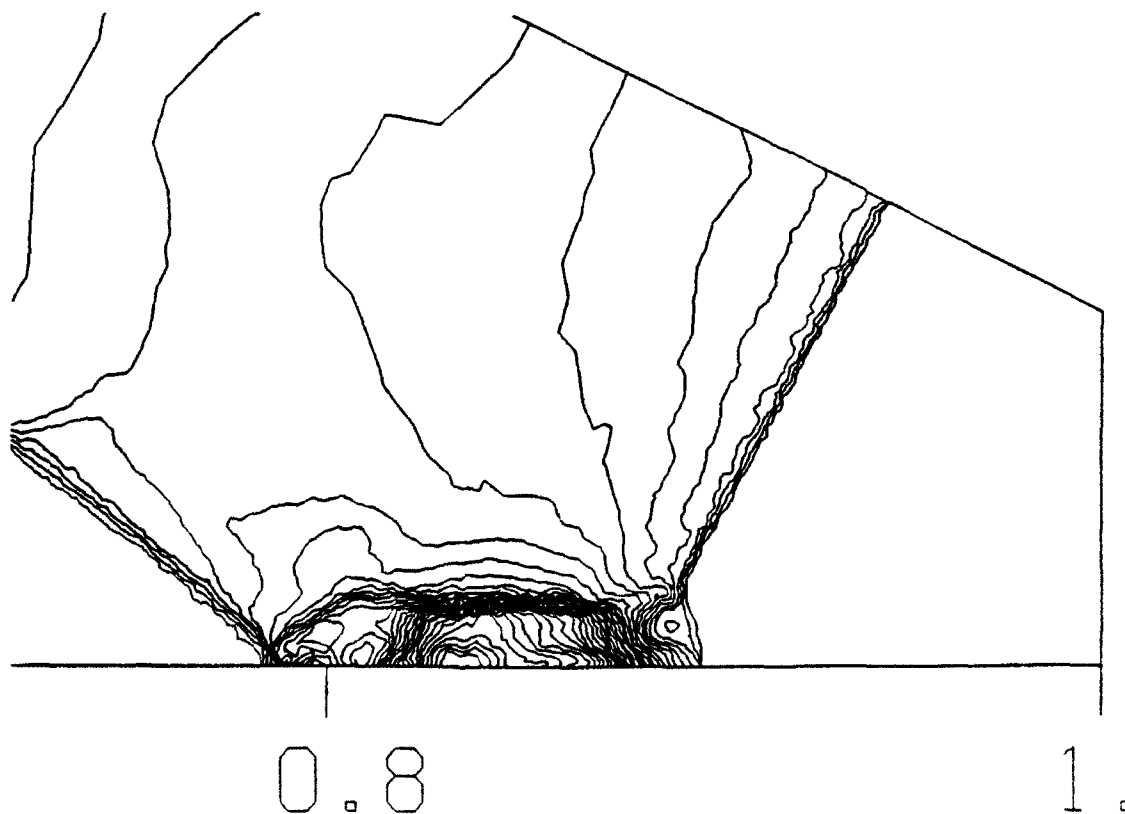


Figure 11b. Blow up of the secondary Mach Reflection area shown in Figure 11a.

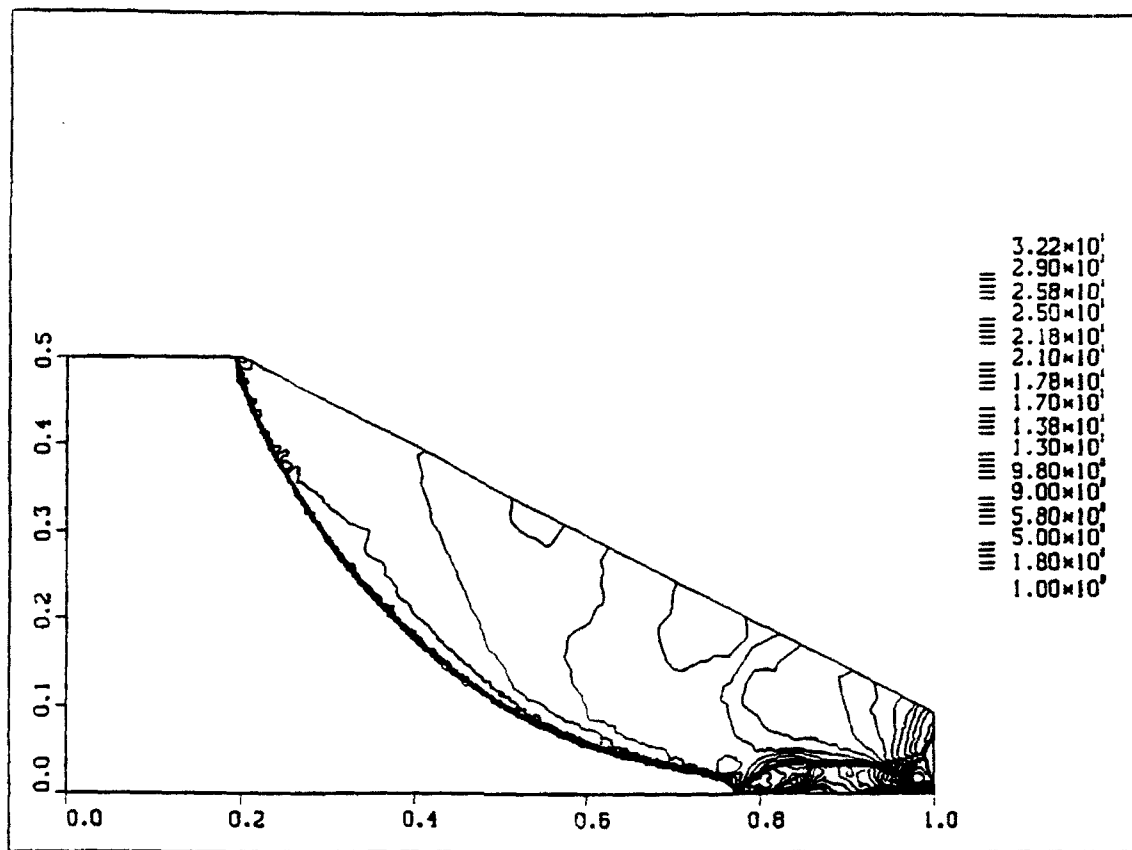


Figure 11c. Secondary Mach Reflection axisymmetric channel. Density contours.

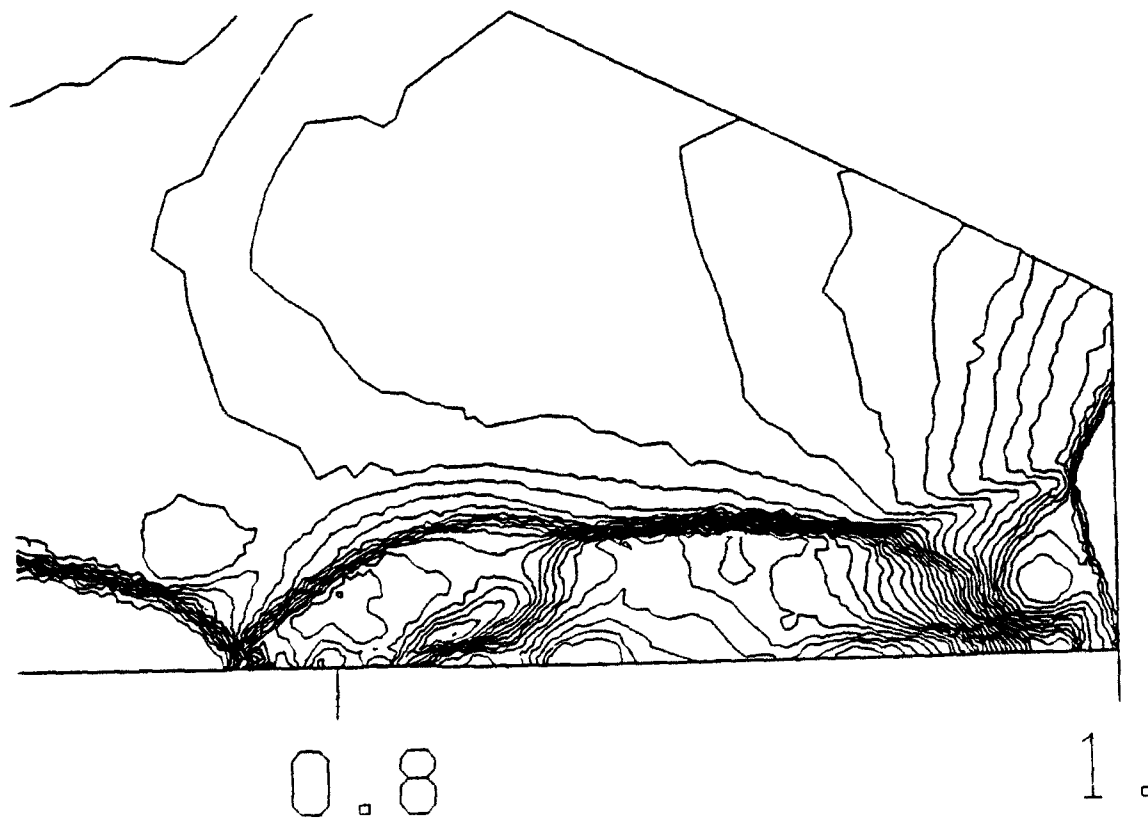


Figure 11d. Blow up of the secondary Mach Reflection area shown in Figure 11c.

Solution of Euler's Equations on Adaptive Grids Using A Fast Unstructured Grid Second Order Godunov Solver (FUGGS)

Itzhak Lottati, Shmuel Eidelman and Adam Drobot
Science Applications International Corporation
1710 Goodridge Drive
McLean, Virginia 22102

Abstract. We describe a new technique for solving Euler's gasdynamic equations on unstructured triangular grids with arbitrary connectivity. The formulation is based on the second order Godunov method. The use of data structure with only one level of indirectness leads to an easily vectorized and parallelized code with a low level of overhead in memory requirement and high computational efficiency. The performance and accuracy of the algorithm has been tested for a very wide range of Mach numbers starting from very low subsonic to high hypersonic flows, without the need to adjust any code parameters. The algorithm was implemented in a vertex based and triangle based scheme. The computational results produced by the triangle based version showed an extremely low level of artificial viscosity.

A new method of direct dynamic refinement of unstructured grids, as described in this paper, allows an automatic adaptation of the grid to regions of pressure or density discontinuity, steep pressure or density gradient, and high vortical activity. Results using the algorithm with dynamic grid refinement are presented.

Flow Solver on an Unstructured Grid

The specific use of triangular meshes provides a very flexible means for simulating flows in extremely complex geometries. The data that identifies a triangular mesh (unstructured grid) provides the flexibility needed to properly discretize the complex geometry of the computational domain, especially on the boundary where the geometry and the implementation of boundary conditions, are extremely crucial for the accuracy of the simulation. The flexibility of unstructured grids enables adaptation to physical features in the flow. The price of resolution results in a local rather than a global penalty. Consequently, it is possible to simulate problems on computers with limited memory and still achieve highly resolved solutions. A typical example, which is illustrated in this paper, is a travelling shock passing over an obstacle. The challenge is to simulate such problems with fine resolution across the shock while limiting the total number of mesh points in the calculation.

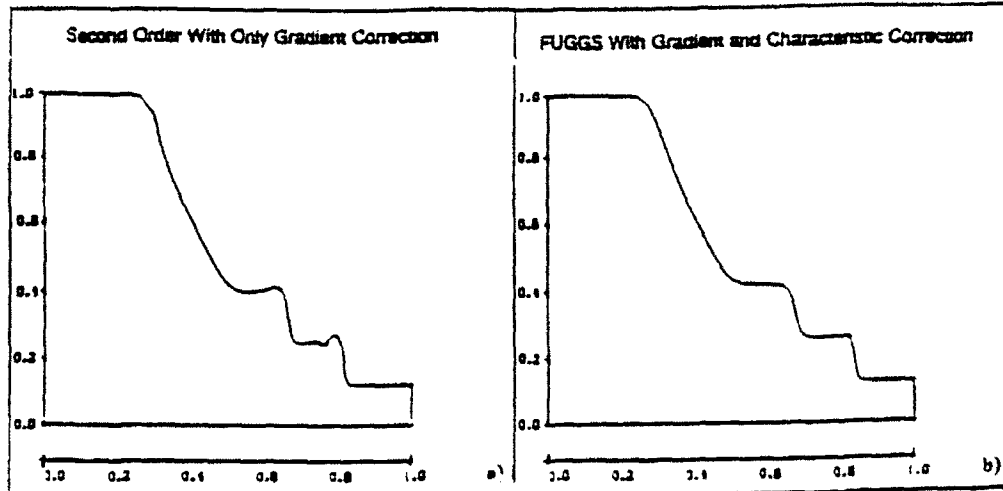


Figure 2: Sod problem, effect of the characteristics correction on the density.

Performance and Validation of FUGGS

FUGGS has proven to be a very robust algorithm capable of high quality solutions while using triangles with large variations of aspect ratios. The code was tested on a variety of unstructured grids and consistently provided results, despite the apparent poor quality of the underlying mesh. We were able to simulate efficiently and accurately a wide spectrum of flow regimes starting from low subsonic to high hypersonic. The code has no free parameters to choose and thus does not require any "tuning" to specific problems. The user has only to specify the boundary conditions (around the grid) and initial flow conditions. The algorithm is fully vectorized and can be easily parallelized in the future. We describe the detailed algorithm below and then present typical results.

Direct Dynamic Refinement Method for Unstructured Triangular Grids

As stated, an unstructured grid is very suitable to implement boundary conditions on complex geometrical shapes and refinement of the grid if necessary. This feature of the unstructured triangular grid is compatible with efficient usage of memory resources. The adaptive grid enables the code to capture moving shocks and high gradient flow features with high resolution. The memory resources available can be very efficiently distributed in the computational domain to accommodate the resolution needed to capture the main features of the physical property of the solution. Dynamic refinement controls the resolution

of the grid according to available memory resources and subject to prescribed priorities. These priorities can be set according to the physical features which the user wishes to emphasize in the simulation. The user has control over the resolution of the physical features resolved in the simulation, without being restricted to the initial grid. The alternative to Direct Dynamic Refinement is the hierarchical dynamic refinement (H refinement) that keeps a history of the initial grid (mother grid) and the subdivision of each level (daughters grid). The H refinement subdivides the initial grid into two or four triangles in each level, and keeps track of the number of subdivision levels each triangle has undertaken. In the H refinement method, one has to keep overhead information on the level of each triangle subdivision, and needs double indirect indexing to keep track of the H refinement process. This slows down the computation by partially disabling the vectorization of the code. As mentioned, the H refinement does rely heavily on the initial grid as it subdivides the mother grid and returns back to it after the passage of the shock.

Direct Dynamic Refinement for capturing the shocks basically requires the refinement to be in the region ahead of the shock. This requirement minimizes the dissipation in the interpolation process when assigning values to the new triangles created in the refined region. Additionally, it requires that the coarsening of the grid should be done after the passage of the shock. In principle, the interpolation and extrapolation in the refinement and coarsening of the grid is done in the region where the flow features are smooth.

The physics of the problem should be involved in the process that identifies the region of refinement and coarsening. One can derive error criteria that will allow grid adaptation to stationary or moving pressure or density discontinuities, region of high vortical activity, etc. For each of the physics features to be resolved, there should be an error indicator that is suited best to capture and identify the region of importance corresponding to this feature.

Criteria for Refinement (Error Indicator)

We have implemented an algorithm with multiple criteria for capturing a variety of features in the physics of the problem to be solved. That means that we were able to derive a number of error indicators that enable identification of moving shock waves or stationary shocks in the computational domain.

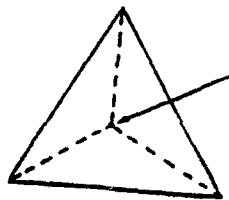
To identify the location of a moving shock, we use the flux of energy or momentum into triangles. The fluxes entering and leaving triangles are the most accurate physical variables computed by the Godunov algorithm for solving the Euler's equations, and are used to update the physical variables for each time step in each triangle. A shock wave means that there is a "step function" change in the cell that is caused by an influx of energy, momentum or density.

Stationary shock can be identified by density gradients computed as required in the second order Godunov algorithm.

The refinement process is done in two ways: i) adding a vertex in the center of a triangle and ii) adding a vertex on an edge of a triangle. Figure 3 illustrates the two alternative ways used to refine the grid. Figure 4 shows an example of the refinement procedure. In the coarsening stage we identify a vertex to be removed. With the point removal, we delete the connecting edges and triangles surrounding the point. The next step is to triangulate the void polygon by creating new triangles using only the vertices of the polygon. Figure 5 shows an example of how the coarsening proceeds.

In the process of refinement and coarsening, we often create triangles with large aspect ratios (the base-to-height maximum ratio for the three edges). We use reconnection to flip the diagonal between two adjacent triangles to obtain triangles with a "better" aspect ratio. This procedure is referred to as the reconnection step in Figs.4 and 5.

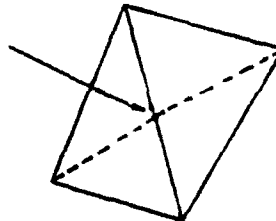
- Adding a vertex in barycenter of triangle.



Advantage: does not effect other triangles.

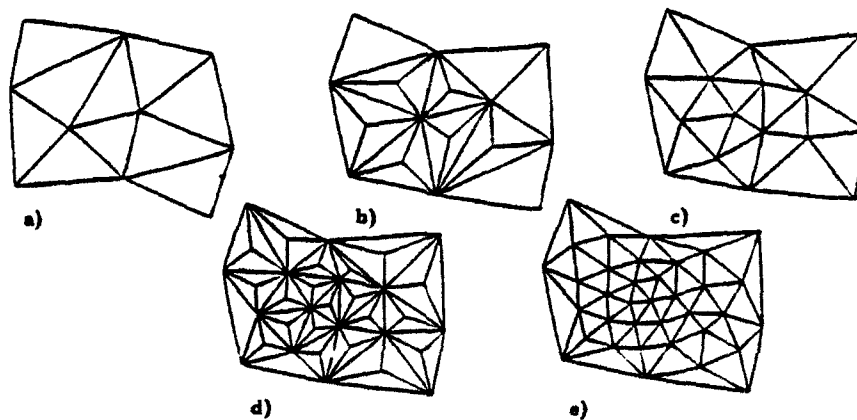
Disadvantage: effects the aspect ratio of the triangles.

- Adding a vertex on the middle of an edge of a triangle.



This method is used on the boundary to improve the triangles with acute angles.

Figure 3: Two Ways of Refinement.



a) Original grid. b) One refinement. c) First reconnection. d) Second refinement. e) Second reconnection.

Figure 4: Illustration of the grid refinement process.

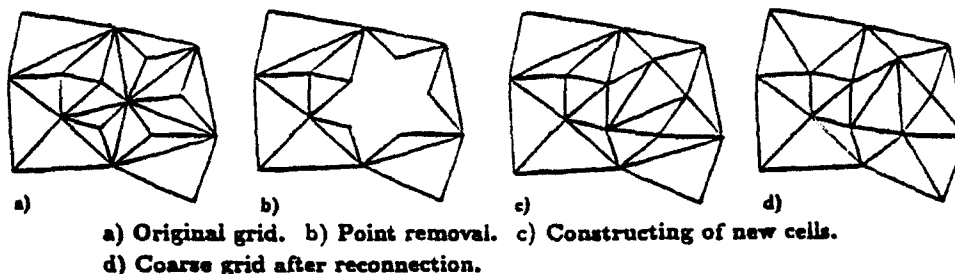


Figure 5: Illustration of the grid coarsening process.

Results

Direct dynamic refinement was used to solve the transient behavior of the flow entering a channel with a double wedge having an inclination of 20° . The flow Mach number entering the channel is 2.5. The flow is from left to right. A sequence of snapshots illustrates the density contours, and the grid for each timestep is given in Figs. 6 (countour plots) and 7 (grid). These figures clearly demonstrate the automatic adaptation of the grid to the moving shocks and the ability to capture the detailed physics of the simulation with very high resolution and minimal memory requirements. The initial grid can clearly be seen to the right of the shock ("ahead") in the early stage of the shock propagation from left to right. The coarsening algorithm is able to produce a reasonable mesh in the region trailing the shock as shown in Fig. 7.

The ability to capture stationary shocks is illustrated in Fig. 8 in which a supersonic free flow ($M = 2.5$) has been run over a diamond shape bump (20° wedge) driven to a steady state. The shock emerges from the first corner (left), the fan of rarefaction waves appears from the apex of the diamond shape bump, and the secondary shock from the second corner (right) is clearly illustrated by the ability of the algorithm to adapt the grid to the physics of the flow. Figure 9 illustrates the sharpness of the reflected shock obtained for an axisymmetric converged channel with an angle of 27° and $M = 8.7$.

The few examples shown here represent a small subset of results obtained with FUGGS. The examples are indicative of the excellent performance that can be achieved for physically complicated situations. We would like to emphasize that these calculations involved no free parameters.

Acknowledgment

We would like to thank Dr. Aharon Friedman and Dr. Marty Fritts for their valuable contributions. This work was partially supported by the Wright Research and Development Center under contract No. F33615-88-C-3002 and by DARPA through contract F49620-89-C-0087 administered by AFOSR.

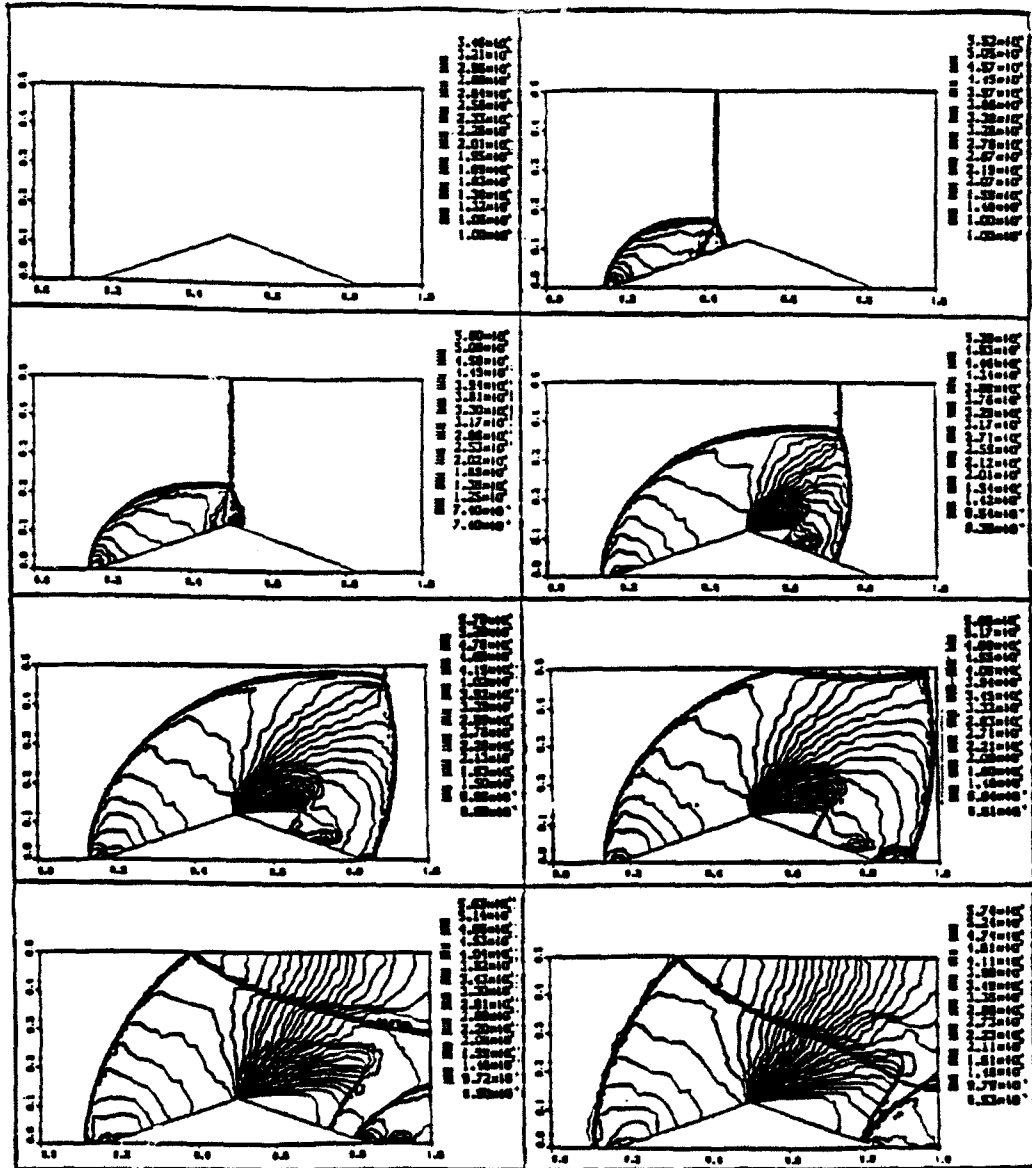


Figure 6. A sequence of density snapshots of contour plots for a propagating shock ($M = 2.5$, wedge angle = 20°).

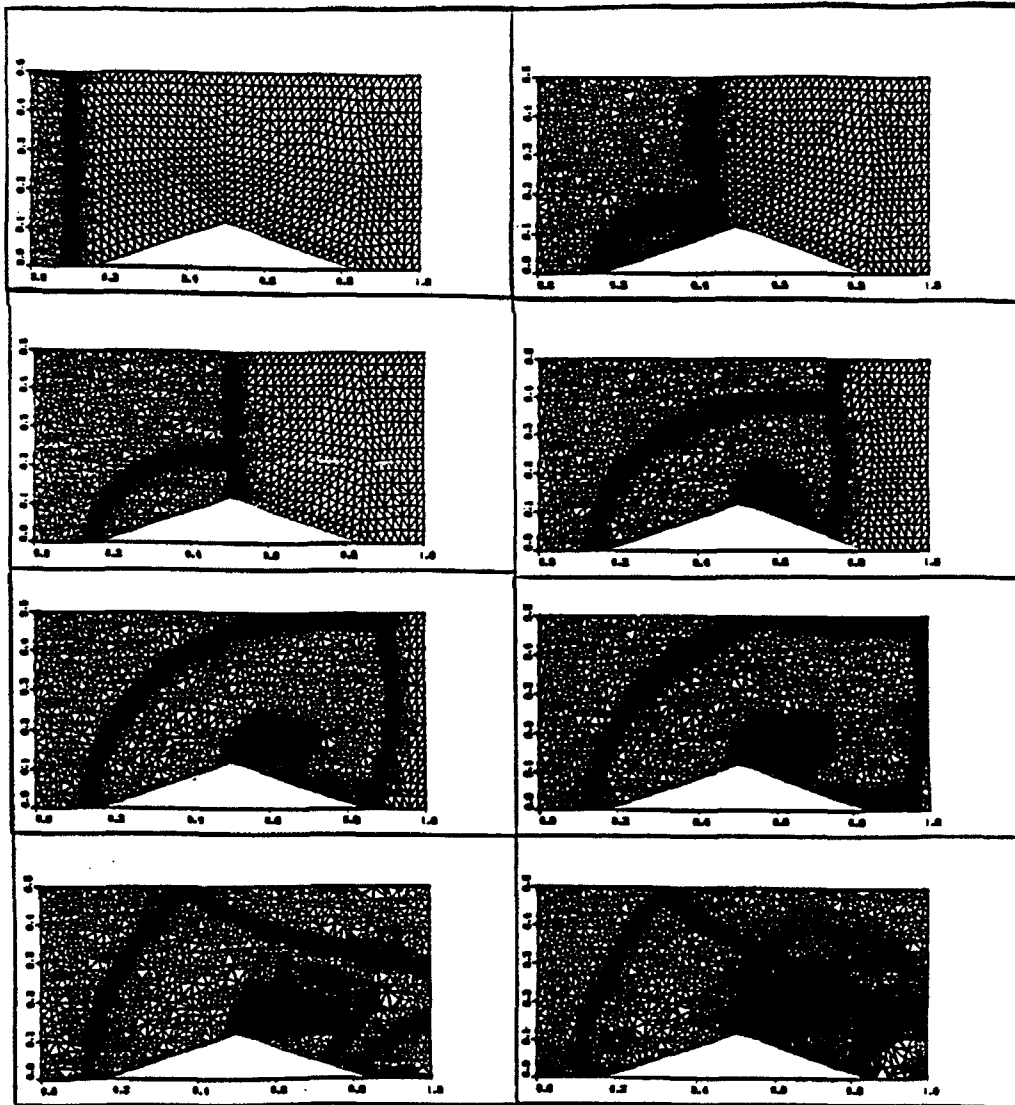


Figure 7. A sequence of grids corresponding to contour plots in figure 6.

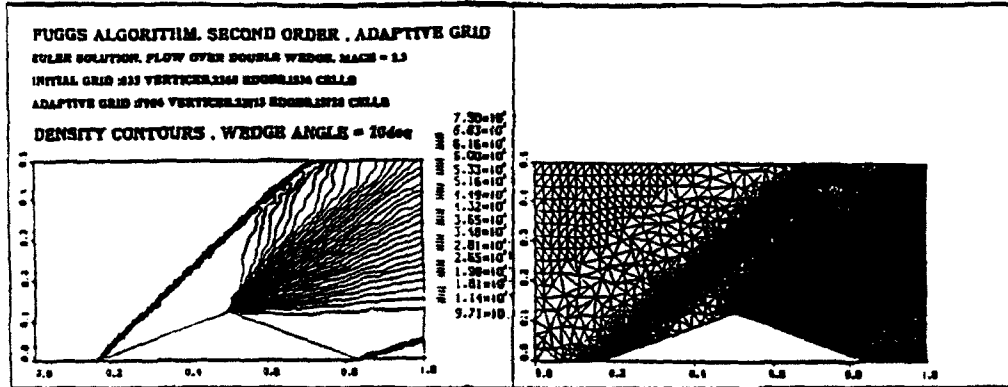


Figure 8. Density contour plot and grid for flow driven to steady state over a double wedge obstacle.

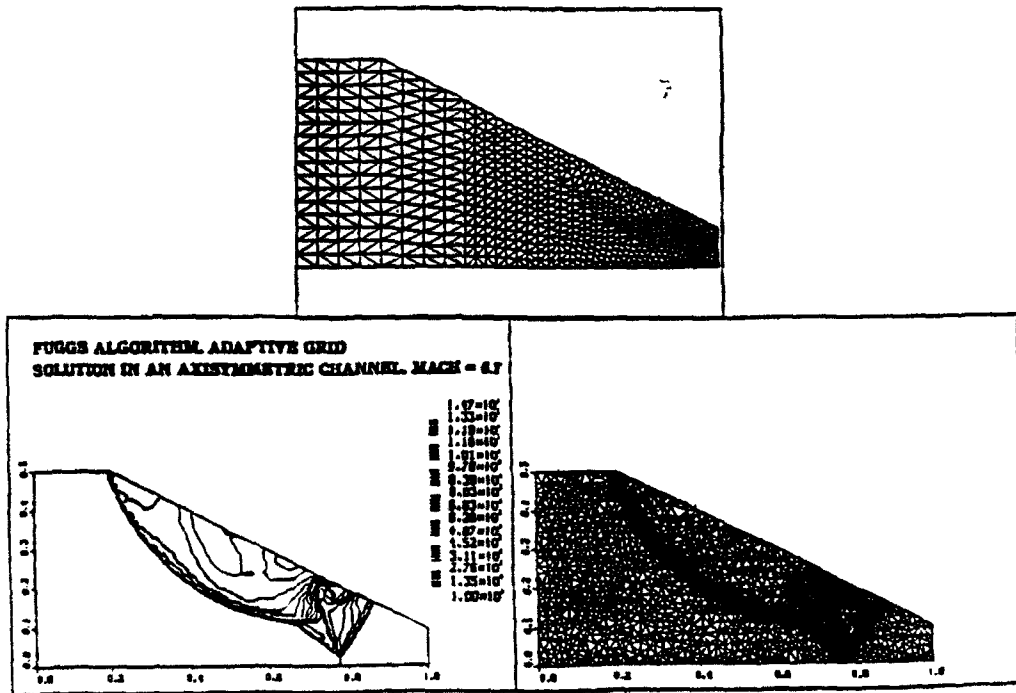


Figure 9. Initial grid, countour plot and the adaptive grid for flow in axisymmetric channel ($M = 6.7$, wedge angle = 27°).



AIAA-90-2420

**Air-Breathing Pulsed Detonation
Engine Concept; A Numerical Study**

S. Eidelman, W. Grossmann and I. Lottati
Science Applications International Corporation,
McLean, Va

**AIAA/SAE/ASME/ASEE
26th Joint Propulsion Conference**

July 16-18, 1990 / Orlando, FL

AIR-BREATHING PULSED DETONATION ENGINE CONCEPT: A NUMERICAL STUDY

S. Eidelman, W. Grossmann and I. Lottati

Applied Physics Operation
Science Applications International Corporation
1710 Goodridge Drive
McLean, Virginia 22102

1. Introduction

The airbreathing Pulsed Detonation Engine (PDE) concept was introduced by us and reported on in the past^{1,2,3}. As described in the previous reports, we have carried out a systematic series of parametric studies of the PDE via Computational Fluid Dynamics (CFD) and have analyzed engine performance over a wide range of flight regimes including subsonic and supersonic flows and physical geometries including various nozzle and air inlets. In addition, we have performed static table top experiments¹ to demonstrate that the principle of pulsed or repetitive detonation can be achieved in a generic PDE configuration. To date, our results indicate that practical engines for certain vehicles and missions can be conceptualized and designed with the information that has already been generated from the studies. Specifically, our studies have shown that the PDE is an excellent candidate for the primary propulsion source for small aerodynamic vehicles that operate over the flight envelope, $0.2 < M < 3$ and altitude between sea level and 30,000 ft. Further, our analysis of the simulation results indicate that the PDE is a high thrust to weight ratio device with a specific fuel consumption on the order of one pound per hour per pound fuel. The predicted performance places the PDE propulsion concept in a strongly competitive position compared with present day small turbojets. The PDE concept has the added attractiveness of rapid variable thrust control, no moving parts and the potential for low cost manufacturing. Finally, the PDE concept is scalable over a wide range of engine sizes and thrust levels. For example, it is theoretically possible to produce PDE engines on the order of one to several inches in diameter and thrusts on the order of pounds, as well as devices which provide thousands of pounds thrust.

The parametric studies that we have carried out to date were possible due to the development of a new generation of CFD tools that have allowed us to accurately simulate the details of the complex nonlinear time dependent processes. A brief description of the CFD methods employed in our studies is given in section 3.

The purpose of the present paper is: (1) to report

the most recent studies of a full simulation of the operation of the PDE with a generic missile configuration cruising at supersonic speeds, (2) to report the results of a parametric-scaling study of the thrust produced as a function of the variation of a given engine configuration with respect to engine size.

The present paper is organized as follows: Section 2 gives, for completeness, a brief description of the PDE concept, Section 3 describes briefly the CFD methods used in our most recent studies, Section 4 gives the results of the parametric-scaling study and, Section 5 describes the simulations of the complete flow around a generic missile configuration powered by a PDE, Section 6 gives our summary and conclusions.

2. The Pulsed Detonation Engine Concept

A detonation process, due to the very high rate of reaction, leads to a propulsion concept in which the constant volume process can be fully realized. In detonative combustion, the strong shock wave, which is part of the detonation wave, acts like a valve between the detonation products and fresh charge. The speed of the detonation wave is about two orders of magnitude higher than the speed of a typical deflagration. This allows the design of propulsion engines with a very high power density. Each detonation has to be initiated separately by a fully controlled ignition device, with a wide range of variable cycle frequencies. There is no theoretical restriction on the range of operating frequencies; they are uncoupled from acoustical chamber resonance. This is very important feature of the constant volume detonation process that differentiates it from the process occurring in a pulse-jet;^{4,5} the pulse jet cycle is tuned to the acoustical resonances of the combustion chamber. This leads to a lack of scalability for the pulse jet concept.

A physical restriction dictating the range of detonation frequency arises from the rate at which the fuel/air mixture can be introduced into the detonation chamber. This also means that a device based on a detonative combustion cycle can be scaled and its operating parameters can be modified for a range of required output conditions. There have been numerous attempts to take

advantage of detonative combustion for engine applications. The most recent and successful of these attempts was carried out at the Naval Postgraduate School (NPS) by Helman et al.¹ During this study, several fundamentally new elements were introduced to the concept distinguishing the NPS research device from previous studies. First, it is important to note that the NPS experimental apparatus was the first successful self aspirating air breathing detonation device. Intermittent detonation frequencies of 25 Hz were obtained. This frequency was in phase with the fuel mixture injection through timed fuel valve opening and spark ignition. The feasibility of intermittent injection was established. Pressure measurements showed conclusively that a detonation process occurred at the frequency chosen for fuel injection. Further, self aspiration was shown to be effective. Finally, the effectiveness of a primary detonation as a driver for the main detonation was clearly demonstrated. Although the NPS studies were abbreviated, many of the technical issues considered to be essential for efficient intermittent detonation propulsion were addressed with positive results.

The generic device we consider here is a small engine shown in Figure 1. Figure 1 shows a schematic of the basic detonation chamber attached to the aft end of a generic aerodynamic vehicle. The combustible gas mixture is injected at the closed end of the detonation chamber and a detonation wave propagates through the mixture. The size of the engine suggests a small payload or aerodynamic vehicle, but the concept can be extended to larger payloads simply by scaling up the size of the detonation chamber and possibly combining a number of chambers into one larger engine.

A key issue in the pulsed detonation engine concept is the design of the main detonation chamber. The detonation chamber geometry determines the propulsion efficiency and the duration of the cycle (frequency of detonations). Since the fresh charge for the generic engine is supplied from the external flow field, the efficiency of the engine depends on the interaction of the surrounding flow with the internal flow dynamics. The range of the physical process requiring simulation in order to model the complex flow phenomena associated with the detonation engine performance is very broad. A partial list is:

1. Initiation and propagation of the detonation wave inside the chamber,
2. Expansion of the detonation products from the chamber into the air stream around the chamber at flight Mach numbers,
3. Fresh air intake from the surrounding air into the chamber.
4. The flow pattern inside the chamber during post-exhaust pressure buildup which determines the

- strategy for mixing the next detonation charge,
5. Strong mutual interaction between the flow inside the chamber and surrounding the engine.

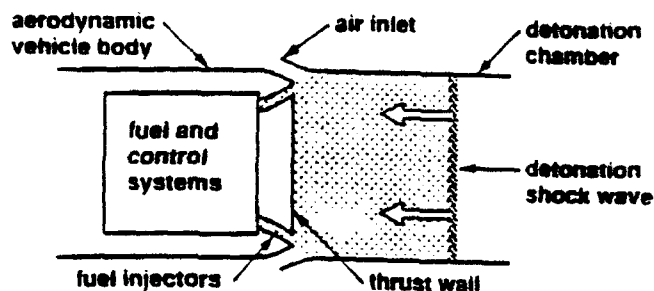


Figure 1. Schematic of the generic PDE showing detonation chamber, inlet, detonation wave, fuel injectors and position relative to an aerodynamic vehicle.

All of these processes are interdependent, and interaction and timing are crucial to engine efficiency. Thus, unlike simulations of steady state engines, the phenomena described above can not be evaluated independently. The need to resolve the flow regime inside the chamber accounting for nozzles, air inlets etc., and at the same time resolve the flow outside and surrounding the engine, where the flow regime varies from high subsonic, locally transonic and supersonic, makes it a challenging computational problem.

The single most important issue is to determine the timing of the air intake for the fresh charge leading to repetitive detonations. It is sufficient to assume inviscid flow for the purpose of simulating the expansion of the detonation products and fresh air intake. The assumption of inviscid flow makes the task of numerically simulating the PDE flow phenomena somewhat easier than if a fully viscous flow model were employed. For the size of the generic device studied in this work the effects of viscous boundary layers are negligible with the exception of possible boundary layer effects on the valve and inlet geometries discussed subsequently.

3. Computational Methods used in the Studies

The basic computational tool that was used for our studies is the FUGGS (Fast Unstructured Grid Second Order Godunov Solver) code, described in detail in Refs. 6,7. This code provides a method for solving the Euler equations of gasdynamics on unstructured grids with arbitrary connectivity. The formulation is based on a second order Godunov method⁸. The use of a data structure with only one level of indirectness leads to an easily

vectorized and parallelized code with a low level of overhead in memory requirement and high computational efficiency. The performance and accuracy of the algorithm has been tested for a very wide range of Mach numbers and geometrical situations, and has demonstrated robustness without the need for any adjustable parameters. The algorithm can either be triangle or vertex based; experience with the method has shown that extremely low levels of artificial viscosity can be achieved using the triangle based version of the method.

A new method of direct dynamic refinement of unstructured grids has been developed, (Ref. 6), and allows an automatic adaptation of the grid to the region of the moving detonation wave inside the PDE geometry. This refinement guarantees that the associated highly inhomogeneous pressure and density contours of the detonation wave are accurately tracked in the simulation. This is an important ingredient in our simulations, since the main component of the detonation process contributing to the thrust generated by the PDE is the total kinetic energy of the wave. Use of the new refinement scheme has more accurately describe the moving detonation wave behavior. These new results concern non-planar wave evolution and, as pointed out in Section 4, may be a factor in controlling the magnitude of the generated thrust.

4. Scaling Study of the PDE

We have shown in our previous study that in the Pulsed Detonation Engines, thrust is primarily produced by the unsteady interaction of shock wave generated by the propagating detonation wave and the thrust wall of the detonation chamber. This interaction will be nonlinear and scalability of the engine will greatly depend on the extent of nonlinearity. For example, for the engine geometry shown in Figure 1, the engine volume can be increased just by elongating the wall of the detonation chamber. If the area of the thrust wall in Figure 1 remains the same and the composition of the detonation mixture does not change, the increase in the detonation chamber length will result in longer duration of the interaction between the shock wave and the thrust wall. This simple situation poses a question concerning the relationship between the increase in PDE thrust and increase in its volume. This is very practical issue in scaling up the size of the engine, since increase in the detonation chamber diameter will eventually result in difficulty generating a planar detonation front, leading to loss of engine efficiency.

To study this aspect of the detonation engine scalability we have conducted a set of numerical simulations for the engine geometry very similar to these shown in Figure 1. The detonation chamber diameter was kept constant at 8cm and its length varied from 8cm to 16cm.

The main objective of our study is to determine how the thrust produced by the detonation engine increases when the engine length doubles and the rest of the engine parameters will remain the same. This section describes the results of two simulations for the detonation chamber geometry described above, using a detonation chamber length of 8 cm and 16 cm. The simulation begins at $t = 0$ when the detonation chamber is placed in an external freestream with the Mach number of 0.8. The detonation wave is initiated at the aft end of the detonation chamber. The detonation chamber for these cases includes a simple annular inlet which remains open during operation. The specific fuel chosen for the present simulations is ethylene. The chemical reaction occurring in the ethylene/air detonation process is given by:



The detonability limits of ethylene in air range from 4% to 12% by volume and depend somewhat on temperature and pressure. We assume for the sake of simplicity that the fuel/air ratio is 6% by volume. In contrast with our previous presentations here, as well as in case of supersonic PDE simulation presented in this paper, we have simulated a propagating detonation wave by releasing the energy of detonative combustion in our mixture immediately behind the detonation front. In our simulations we have used the Dynamically Adaptive FUGGS code which we have developed recently. Figures 2a, 2b, and 2c, present three frames of the results for simulation in a 16 cm long detonation chamber. In these figures, results are presented in the form of pressure contour plots. For illustration of the dynamic grid adaptation to the evolving flow pattern, we have plotted the unstructured triangular grid corresponding to the stage at which contour plots are shown. In Figure 2a, pressure contour plots are shown shortly after the detonation wave has been initiated at the aft end of the detonation chamber. We can observe that the shock wave front is planar. The detonation wave velocity is 1800 m/sec and the pressure at the front of the detonation wave is ≈ 20 atm., corresponding to the CJ condition for the ethylene/air mixture. Figure 2b shows the results of the detonation wave reflecting from the thrust wall and the detonation products starting to expand into the flow stream surrounding the detonation chamber. The detonation products expand through the inlet and into the detonation chamber. This simultaneous expansion results in a complicated wave structure which can be observed in Figure 2b. Here we also note that the dynamically adjustable grid closely follows developing wave structures. In Figure 2c, results are shown at the stage when the two main shock waves generated by the PDE cycle have interacted and are about to leave the computational domain. The maximum pressure here dropped to 1.7 atm.

The computational grid follows the shocks and vortices propagating through the computational domain and we can observe the substantially reduced grid density in the regions of relatively monotonic flow. Figure 2 illustrates the level of detail of this complicated flow regime which can be studied with modern CFD methods and algorithms.

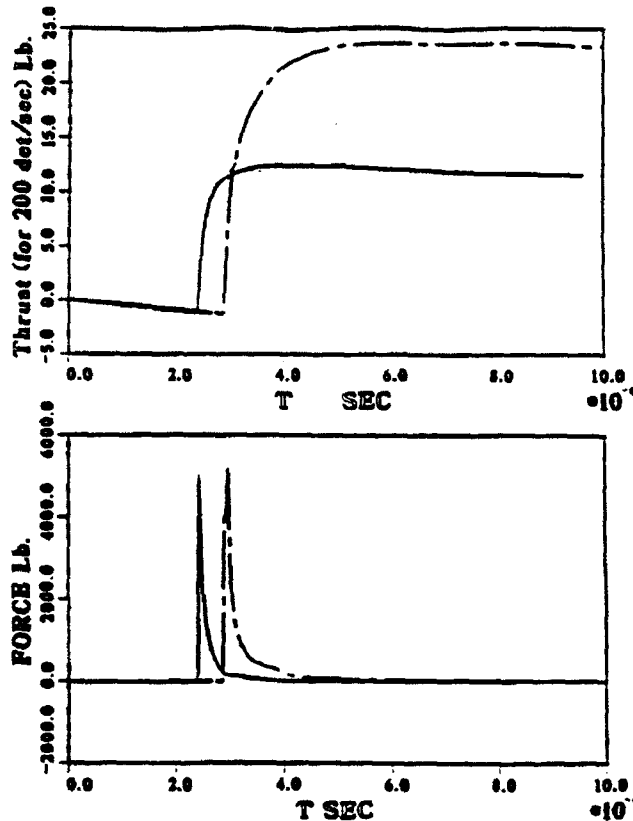


Figure 3. Time averaged thrust and force data from simulation of 8cm (solid lines) and 16cm (dashed lines) detonation chambers, 200 Hz detonation frequency.

In Figure 3 the total force and time averaged thrust generated by the device in the simulations just discussed for 8cm and 16cm long detonation chambers, are shown as a function of time. The time averaged thrust is based on the total time for one cycle defined as 5.0×10^{-3} sec. This time is equivalent to a detonation frequency of 200 Hz. As seen in the figure, initially the force acting on the thrust wall is close to zero. The simulation was run for 2.0×10^4 sec physical time to establish a flow pattern characteristic of the steady nonreactive flow of ambient air around the detonation chamber. At the time 2.0×10^4 sec the detonation wave started to propagate from the aft of the chamber. We can see in Figure 3 that the detonation wave reaches the thrust wall at the time 2.45×10^4 sec (for 8cm case) and 2.9×10^4 sec (for 16cm case), when a very large force of $\approx 5.0 \times 10^3$ lb is felt on the end wall of the detonation chamber. This force is a

result of the high pressure behind the detonation wave. It rapidly decays to virtually zero level in $\approx 0.5 \times 10^{-4}$ sec in the 8cm case and $\approx 1.0 \times 10^{-4}$ sec in the 16cm case. The maximum force produced on the thrust wall is the same in both cases. The increase of the detonation chamber volume is most noticeable in the thrust data. As we can see in Figure 3 the average thrust increases from 12 Lbs in the 8cm chamber case to 24 Lbs in 16cm chamber case. This result shows that the thrust of the detonation chamber will scale linearly with an increase in detonation chamber length when the other parameters are kept constant.

5. Supersonic Missile Simulation

In this section we present the results of a full simulation of a generic supersonic missile powered by a PDE. The purpose of this simulation was to study the requirements placed on the PDE air inlets and internal structures that may be needed to produce a well mixed, uniform flow inside the detonation chamber. In addition, the simulations were carried out on the full vehicle in order to account for all wave drag that a real missile produces; the resulting thrust predictions for the simulations are therefore true net thrust values. We show here the results of a successful geometry that satisfies the requirements of choking flow in the inlet throat and uniform predetonation flow in the chamber produced by means of a grill. The missile geometry and computational grid are shown in figures 4a, 4b, and 4c.

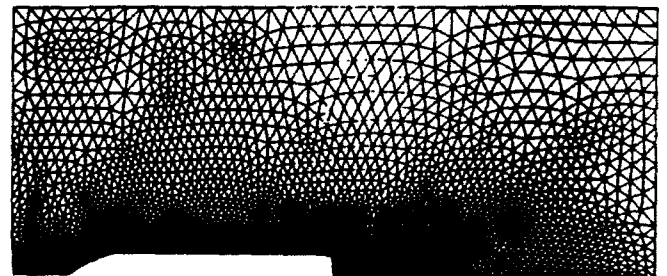


Figure 4a. Unstructured Grid for Missile and Engine Simulation.

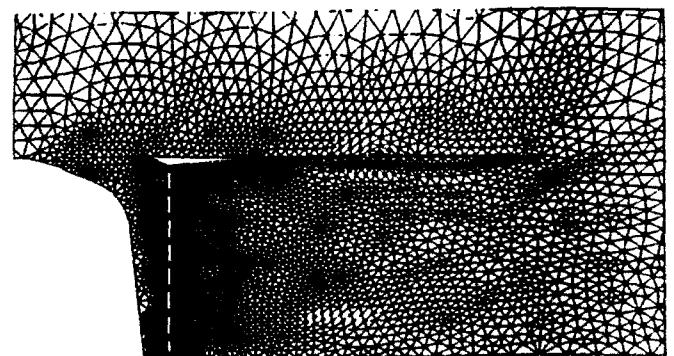


Figure 4b. Grid Detail for Inlet and Manifold.

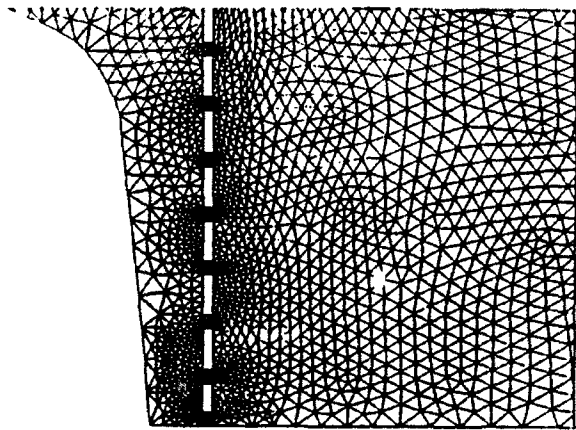


Figure 4c. Grid Detail for Manifold.

Figure 4a shows the main missile body with the PDE covered by the high density of grid points necessary to resolve the details of the PDE chamber, inlets and, grill as shown in the enlarged views of the chamber, figures 4b and 4c.

The simulations were performed by allowing steady subsonic flow conditions to be established in the detonation chamber holding a steady supersonic flow, Mach 2, about the missile. The degree to which this steady and uniform flow can be established in the chamber using the inlet and grill of figure 4 is shown in figure 5. Here the complete flow including the bow shock is shown, figure 5a, as well as an enlarged view of the flow in the vicinity of the inlets showing smaller shocks, figure 5b, and a particle trace showing the streamlines of the uniform chamber flow, figure 5c. When steady flow conditions are reached in the detonation chamber, plane detonation is started at the rear end of the chamber. The detonation then travels towards the inner thrust wall at approximately Mach 4. Figure 6 shows the same sequence of views as figure 5, but with the detonation approximately having travelled halfway to the thrust wall. Notice that the detonation remains more-or-less planar indicating that the flow properties are uniform in the chamber. Figure 7 shows the phenomena associated with the detonation impacting the thrust wall, the high pressure of the detonation wave exhausting from the inlet and particles leaving the chamber through the inlets. The principle results from the simulations of the supersonic missile case are that the use of such a grill structure and inlet shape allow uniform flow to be established before and after detonation in sufficient time that detonation frequencies of 200 cycles per second are obtainable. It is not clear at this time whether such internal grill structures are desirable from the standpoint of structural integrity. This question will be addressed later in planned experimental studies of the PDE.

6. Conclusions

The simulation of the PDE presented in this paper are partial results from an ongoing SAIC research program aimed at development of a practical PDE engine for a wide spectrum of applications including small UAV's and PENAIID missiles among others. The primary focus of the results presented here is the scaling of PDE performance with respect to size variation and the establishment of uniform subsonic flow conditions in the detonation chamber before and after detonation.

The results of the scaling studies described in the text lead to scaling laws that can be used to predict the performance of PDE's over some range of parameters assuming that other parameters are held fixed. For example, holding the external Mach number and basic chamber and inlet geometry fixed suggests that the thrust at constant specific fuel consumption produced by the PDE scales as:

$$Thrust = T_1 * \left(\frac{v}{v_1}\right) * \left(\frac{f}{f_1}\right),$$

where T_1 , (v/v_1) and (f/f_1) are the thrust computed for a chamber of volume v_1 operating at frequency f_1 , the ratio of a new volume to v_1 and the ratio of the new frequency to f_1 respectively. Thus, thrust should scale linearly with the parameter $(v/v_1) * (f/f_1)$ over some range of this parameter. Departure from this linear variation may occur due to the following reasons: First, since volume is proportional to the product of cross-sectional area and length, $v \sim r^2 l$, ($r \sim$ detonation chamber radius, $l \sim$ chamber length) physical limits will be placed on r and l ; if r is too small (less than 1 cm) a detonation will not be sustainable and if l is too small (less than 10 cm) it may be difficult to mix fuel and air effectively. Using the thrust relation established above, we make the following observations. For a PDE device producing 100 pounds thrust at 100 Hz, doubling the frequency and increasing the volume by a factor of 5 yields a thrust level of 1000 pounds. Assuming that the aspect ratio of the chamber (chamber length to radius) is fixed, this would require an engine only 25.5 cm in diameter and 25.5 cm in length. Similarly, scaling the engine down in size to a 5 cm diameter, 5 cm length detonation chamber operating at 100 Hz yields thrust levels of the order of 3.7 pounds. Of course, the derived relation between thrust and $(v/v_1) * (f/f_1)$ cannot be believed over too wide a range of parameters; but, it does serve to point out the

flexibility in scaleup or scaledown permitted by the PDE concept.

We further conclude that the performance computed for PDEs is encouraging from the point of view of thrust, thrust control, simplicity of the device (no moving parts) and specific fuel consumption (SFC). The specific fuel consumption computed from our simulations ($\sim 1\text{Lb/hr./lb}$) is competitive with present day small turbojets (SFCs for small turbojets are in the range of 1.8-2.0 lb./lb.*hr.). Thus, for a given mission and vehicle, a PDE propulsion unit could be more fuel efficient resulting in increased range. Moreover, if the expected thrust control in PDEs is realizable, it may be possible to produce propulsion units that can slow down, loiter and maneuver and finally regain full thrust within the time it takes to increase the detonation frequency.

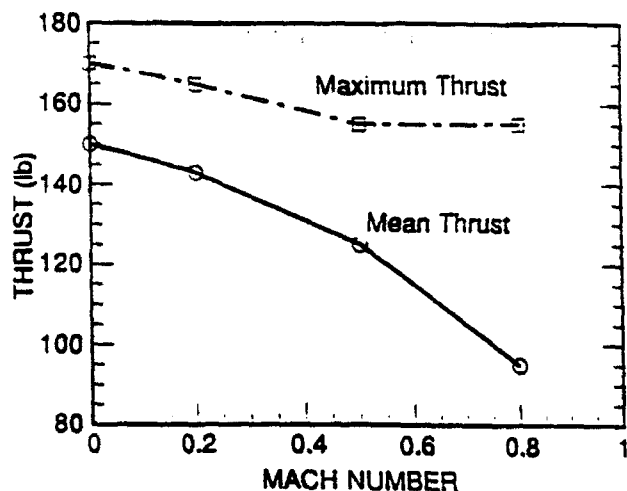


Figure 8. Thrust versus Mach number variation obtained from simulation data.

Another result from the scaling studies is that the thrust data show a dependence on the external flight conditions, e.g. Mach number. The Mach number plays a role in the wave drag that the geometry of the PDE will incur; the details of the valve and inlet configurations figure prominently in the total wave drag.

On the other hand, the simulations showed that the timing of the fresh air refilling required to recharge the chamber for subsequent detonations is a strong function of the details of the valve and inlet geometry, the expansion of the combustion products, the resulting over-expansion of the chamber flow and the external flow regime and interaction of the external flow with the internal flow. For subsonic flight, Mach 0.2-0.9, the fresh air entering the chamber comes from two separate principle flow processes: one comes from the flow through any valve or inlet and the other comes from the self-aspiration or reverse flow from the aft end of the chamber due to strong over-expansion. All these processes are interde-

pendent and, in order to search for a given performance in a given device, requires variation of many parameters. The simulation results obtained to date provide an understanding of the effects caused by variation of the above-mentioned parameters and, with the information available, we are able to conclude that a PDE propulsion unit can be optimized (although no optimization studies were carried out) for a given flight regime. For example, if we consider the simulations obtained for constant (number and inlet) geometry but at Mach numbers 0.8, 0.5, 0.2, and 0.0 respectively, the variation of maximum time averaged thrust and mean thrust as a function of Mach number can be characterized as shown in Figure 8.

The decrease in thrust with Mach number has been described earlier³ to be a result of the increased wave drag produced by the inlet geometry. Optimization of the inlet geometry could help in eliminating a large part of the wave drag. The data contained in Figure 8 could be used to determine the detonation frequency at a given Mach number yielding constant thrust. For a constant thrust level of 90 pounds, the required detonation frequency varies from 84 Hz at $M = 0.0$ to 140 Hz to $M = 0.8$. In a similar fashion, parametric variations of other important aspects of PDE performance, such as minimum time for refill at given Mach number as a function of air inlet opening, can be obtained. In order to find an optimum configuration satisfying given performance over a wide flight regime, a more extensive simulation study will be required. It was mentioned earlier that the simulations presented here were carried out under the assumption of inviscid flow; boundary layer effects were not included. The addition of boundary layers to the PDE engine inlets and valves, the only components where boundary layers will be significant, will lead to increased performance. Roughly the same amount of fresh air will flow into the over-expanded detonation chamber but at a somewhat slower rate and in a pattern that will promote enhanced circulation and hence fuel/air mixing.

A final conclusion can be made concerning the application of PDE's to supersonic vehicles. As shown in the simulations the ability to refill the detonation chamber with fresh air charge is a very strong function of valve and inlet geometry. Refilling may also be somewhat enhanced by the self-aspiration effect, but; to a much less extent than in the subsonic case. The example of supersonic operation discussed in Section 5 shows that care must be taken in design of the inlet or valve configuration. The flow in the chamber must allow for refill and fuel/air mixing. More than likely choked flow conditions will be required at the inlet entrance to the chamber. This could lead to complications in the design of a PDE with simple geometry; choked flow conditions are a function of the external Mach number and a fixed

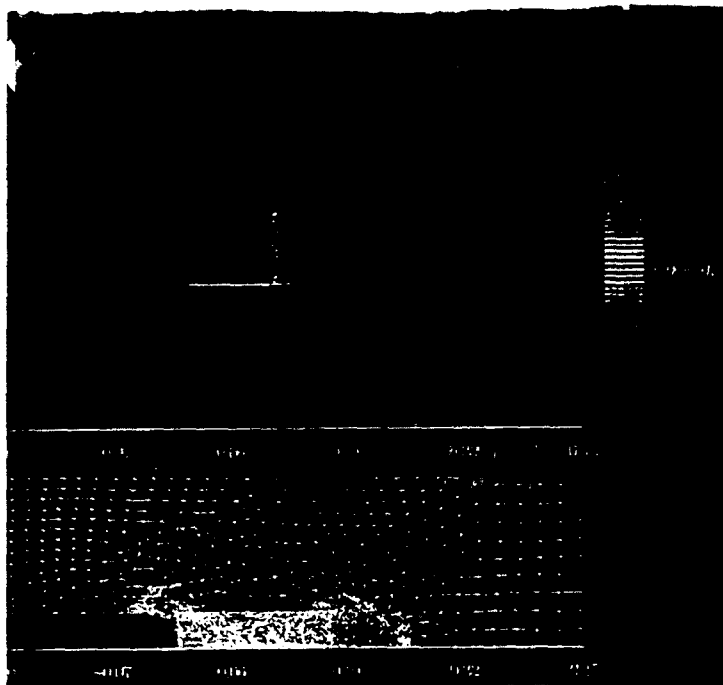
inlet will be optimal only for a small range of the operating envelope. On the other hand, if a given vehicle is to fly at supersonic speeds and is launched at supersonic speeds, this problem may not appear. Further, if the given vehicle is launched at subsonic speeds and a booster is used to bring it up to the required supersonic operating speed, the problem may again not appear. We conclude that the PDE has potential for the supersonic flight regime and it is not excluded that a configuration can be found which will operate over the flight regimes $0.2 < \text{Mach number} < 3$ in a fuel efficient manner.

Acknowledgements

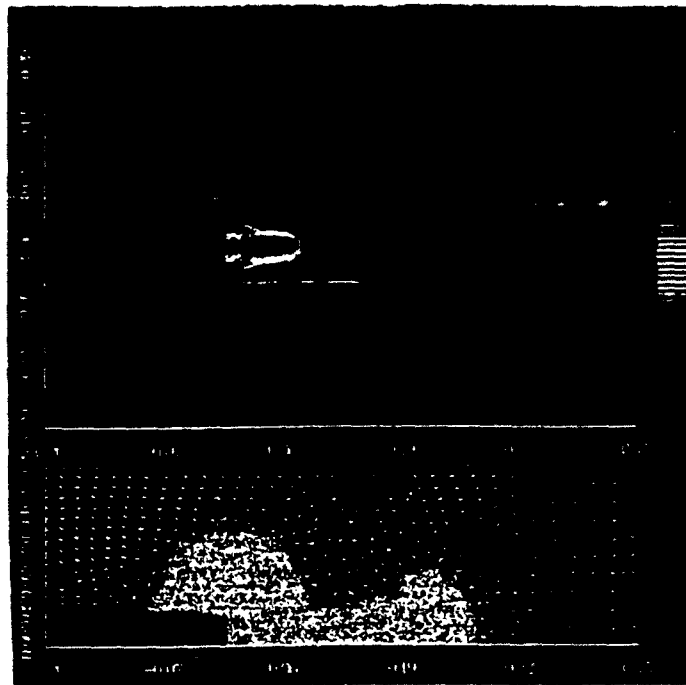
The authors would like to express appreciation to Dr. Adam Drobot of SAIC for helpful suggestions and advice during the course of this work. The work reported on here was partially supported by DARPA under AFOSR contract FY9620-89-(-0087). The authors would like to thank Dr. L. Auslander for his encouragement and interest in this project.

References

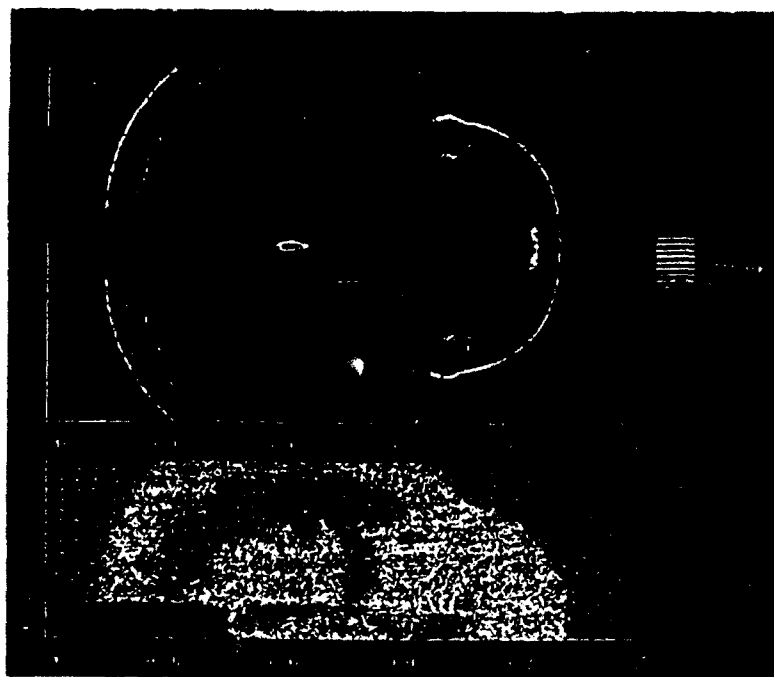
1. Helman, D., Shreeve, R. P., and Eidelman, S., "Detonation Pulse Engine," AIAA-86-1683, 24th Joint Propulsion Conference, Huntsville, 1986.
2. Eidelman, S., W. Grossmann, I. Lottati, "A Review of Propulsion Applications of the Pulsed Detonation Engine Concept," AIAA 89-2466, AIAA/ASME/SAE/ASEE 25th Joint Propulsion Conference, Monterey, CA, July 10-12, 1989 (to be published in AIAA Journal of Propulsion).
3. Eidelman, S., W. Grossmann, and I. Lottati, "Computational Analysis of the Pulsed Detonation Engines and Applications," AIAA 90-0460, 28th Aerospace Sciences Meeting, Reno, NV, Jan 8-11, 1990.
4. Shultz-Grunow, F., "Gas-Dynamic Investigation of the Pulse-Jet Tube," NACA TM-1131, Feb. 1947.
5. Zinn, B. T., Miller, N. Carvelho, J. A., and Daniel B. R., "Pulsating Combustion of Coal in Rijke Type Combustor," 19th International Symposium on Combustion, 1197-1203, 1982.
6. Lottati, I., S. Eidelman, A. Drobot, "A Fast Unstructured Grid Second Order Godunov Solver (FUGGS)," Paper AIAA 90-0699, 28th Aerospace Sciences Meeting, Reno, NV, Jan 8-11, 1990.
7. Lottati, I., S. Eidelman, A. Drobot, "Solution of Euler's Equations on Adaptive Grids Using a FUGGS," to be published in Proceedings of Second International Conference on Free-Lagrange Methods, Held at Jackson Hole, June 1990.
8. Eidelman, S., Collela, P., and Shreeve R. P., "Application of the Godunov Method and Its Second Order Extension to Cascade Flow Modeling," AIAA Journal v. 22, 10 (1984).



a)

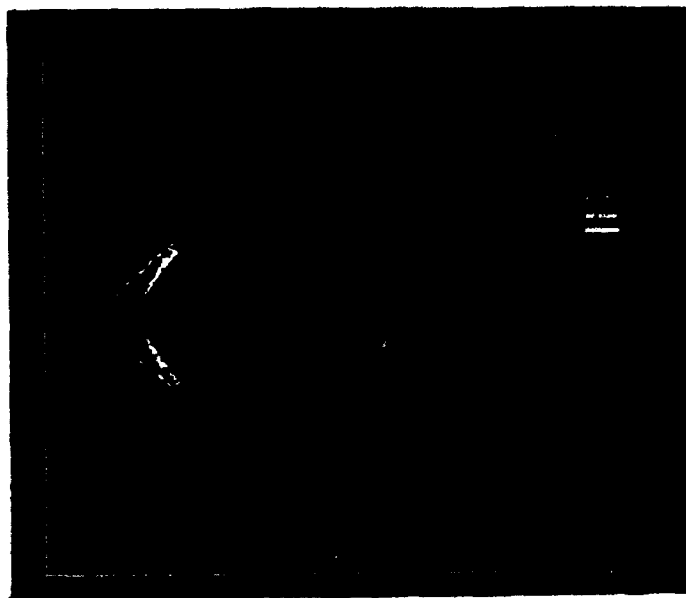


b)

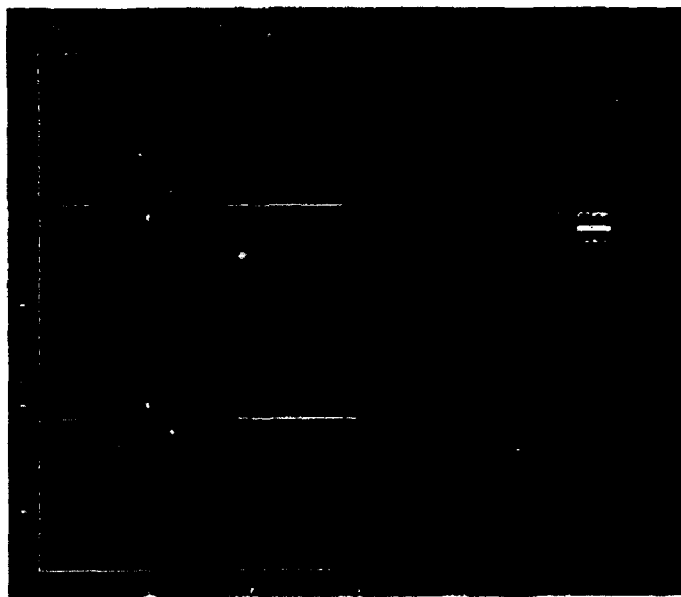


c)

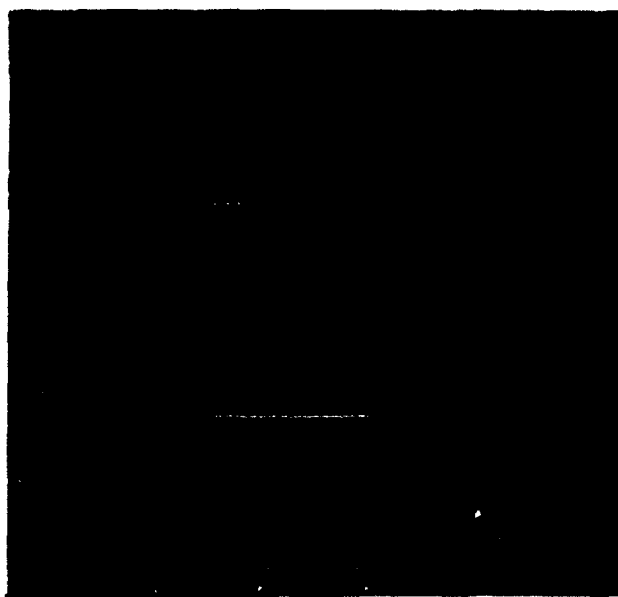
Figure 2. Pressure contours and computational grid for 16 cm long PDE. External flow $M = 0.8$.



a. Pressure Contours. Missile and Engine.

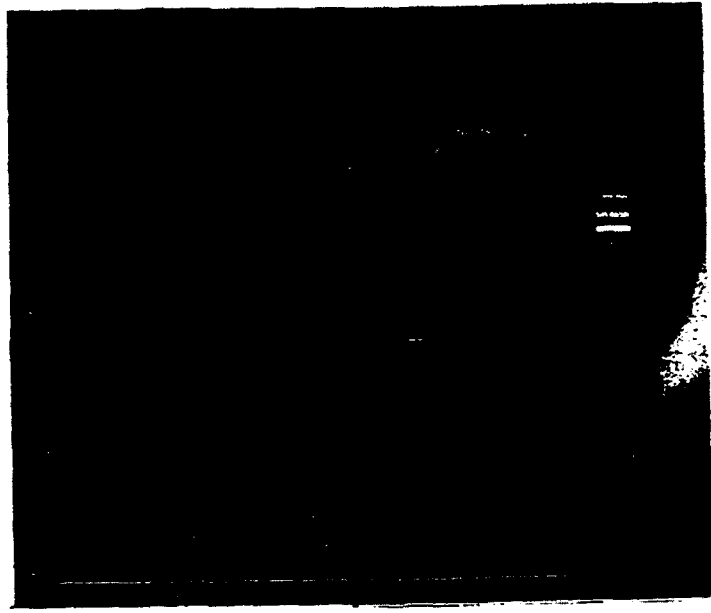


b. Pressure Contours. Detonation Engine.

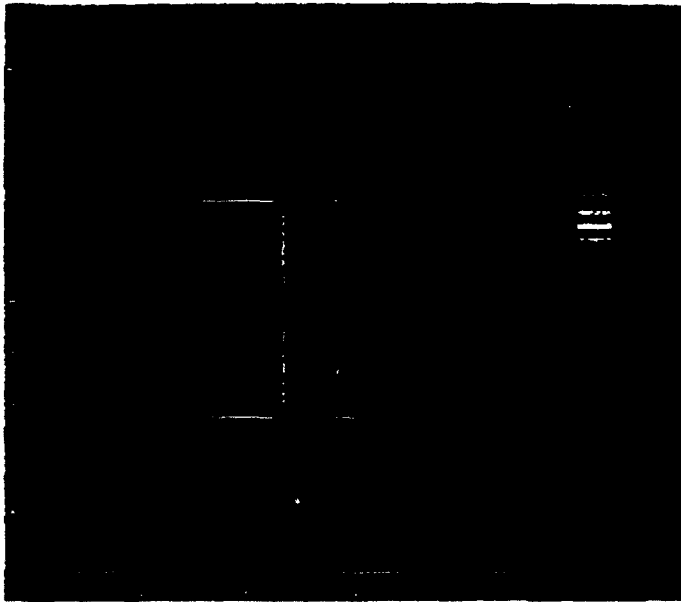


c. Traced Particles. Detonation Engine.

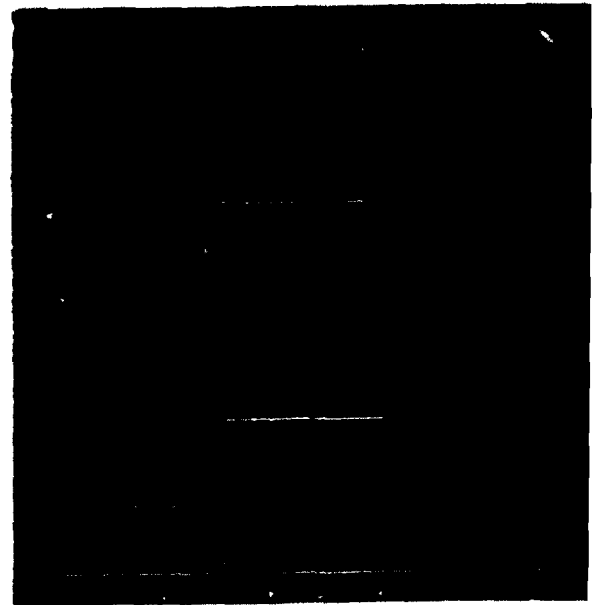
Figure 5. Supersonic missile simulation. Missile speed $M = 2.0$. Time $t = 0.0$.



a. Pressure Contours. Missile and Engine.



b. Pressure Contours. Detonation Engine.

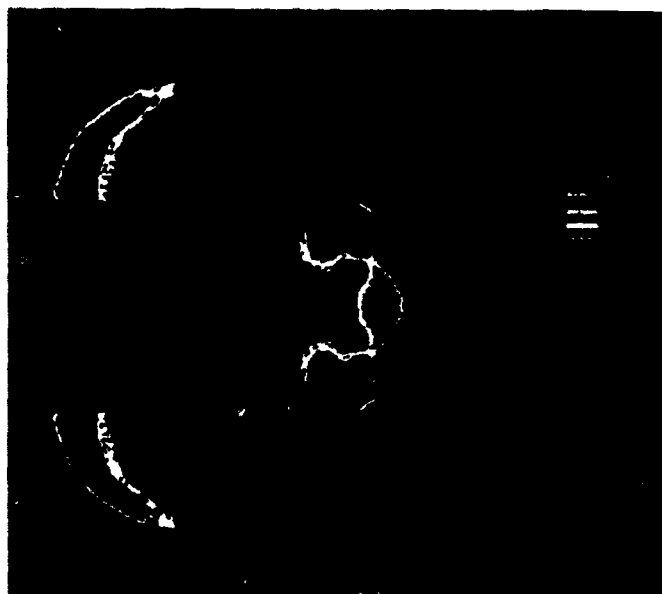


c. Traced Particles. Detonation Engine.

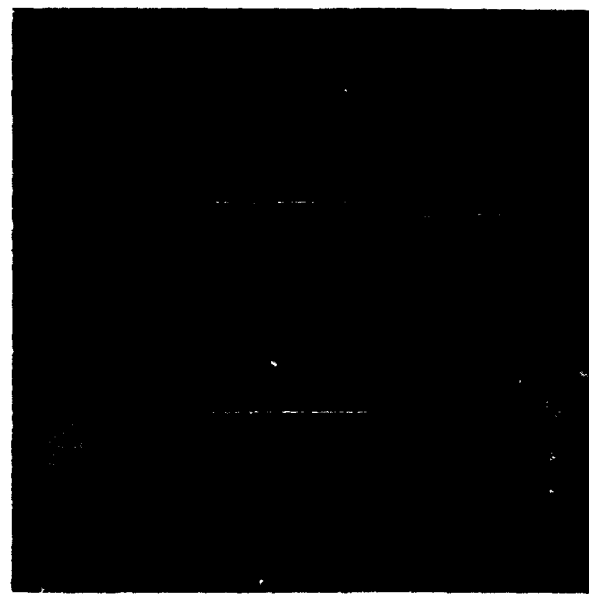
Figure 6. Supersonic missile simulation. Missile speed $M = 2.0$. Time $t = 2.0 \cdot 10^{-5}$ sec.



a. Pressure Contours. Missile and Engine.



b. Pressure Contours. Detonation Engine.



c. Traced Particles. Detonation Engine.

Figure 7. Supersonic missile simulation. Missile speed $M = 2.0$. Time $t = 2.0 \cdot 10^{-4}$ sec.

Plasma enhanced chemical vapor deposition modeling

E. Hyman, K. Tsang, I. Lottati and A. Drobot

Science Applications International Corporation, 1710 Goodridge Drive, McLean, VA 22102 (U.S.A.)

B. Lane*, R. Post and H. Sawin†

Applied Science and Technology, 35 Cabot Road, Woburn, MA 01801 (U.S.A.)

Abstract

We are developing a model to simulate the plasma enhanced chemical vapor deposition (PECVD) of thin diamond films. The emphasis to date has been on the development of stand-alone modules to simulate the microwave-induced time-dependent electric and magnetic fields, the generation and energization of plasma electrons in the discharge, the non-equilibrium hydrocarbon chemistry, and the development of a two-dimensional unstructured mesh hydrodynamics solver capable of simulating flow through geometrically realistic reactors. The coupling of the various modules, and the incorporation of a surface chemistry module for the substrate deposition, into a self consistent reactor model is underway. We present some preliminary results from components of a model 2.45 GHz microwave reactor employing H_2 with 1% CH_4 and operating at a gas pressure of 5.3×10^1 Pa (40 Torr). We have completed an electromagnetic model of the microwave energy deposition in the plasma and calculated the field patterns in the reactor. We have also performed point calculations of the time-dependent electron distribution and of the build-up of atomic hydrogen, the gas temperature, and the resulting generation of CH_3 , C_2H_2 , and other hydrocarbon radicals. We have also completed a fluid simulation of the flow through the reactor using unstructured mesh techniques. The results we discuss in this paper indicate that careful treatment of non-equilibrium processes in PECVD reactors as well as accurate representation of reactor geometry are essential to a useful simulation capability.

1. Introduction

The ability to deposit thin diamond films rapidly onto substrates with a high degree of uniformity using the plasma enhanced chemical vapor deposition (PECVD) technique is a high priority technology goal. It is generally recognized that an improved understanding of the microscopic mechanisms in PECVD reactors and of the sensitivity of the various reactor parameters is needed. The important design issues for PECVD reactors are as follows: efficient coupling of microwave energy to the plasma and to the process gas; efficient transport of activated process gas to the wafer or substrate deposition area; efficient use of the injected gas; uniformity of chemically active species flux across the deposition area. It is desirable to avoid reactor designs that have the following: high microwave electric fields in regions away from the desired plasma formation location, leading to plasma discharge near chamber walls or breakdown of dielectric materials; flow patterns which carry activated species to the reactor walls

or out through pumping ports rather than to the wafer; stagnant or circulating flow patterns above the wafer, buffering the wafer from the desired chemically active species.

To understand these issues and to provide input to improved reactor designs we are developing a self-consistent numerical model which simulates each of the essential mechanisms in the PECVD reactor. A physically realistic model requires careful simulation of the electromagnetics, the plasma physics, the neutral gas flow, and the homogeneous and heterogeneous chemistry. Furthermore, the different elementary processes in the reactor are highly interactive; for this reason it is difficult to foresee intuitively the impact of varying one or another reactor parameter. For example, the microwave source induces a complex, geometrically dependent and time varying electric field which ionizes the gas; the resultant build-up of electrons alters the developing electric field distribution. The microwave energy input heats the electrons, and the energetic part of the non-Maxwellian electron energy distribution dissociates the gas, inducing a rise in the gas temperature. The amount of dissociation and heating depends sensitively on the high energy tail of the electron distribution which consequently must be accurately determined. The interaction of the neutral gas with the plasma alters the molecular input stream of H_2 to include a substantial

*Permanent address: MIT, Plasma Fusion Center, Cambridge, MA 02139, U.S.A.

†Permanent address: MIT, Chemical and Electrical Engineering Department, Cambridge, MA 02139, U.S.A.

component of atomic hydrogen, and this, in turn, affects the ionization rate and the electron distribution. The hydrocarbon chemistry is non-equilibrium and both the flux and the spatial distribution of appropriate radicals reaching the wafer are very sensitive to the geometrical configuration of the reactor and to the details of the flow configuration through the reactor.

We have focused to date on the development of modules to simulate the microwave-induced time-dependent electric and magnetic fields, the generation and energization of plasma electrons in the discharge, the evolution of the molecular and atomic hydrogen gas, the non-equilibrium hydrocarbon chemistry, and the development of a two-dimensional unstructured mesh hydrodynamics solver capable of simulating flow through geometrically realistic reactors. The coupling of these modules, and the incorporation of a surface chemistry module for the substrate deposition, into a self consistent reactor model is underway. In the next section we describe in some detail the generation of the microwave field and the transfer of the field energy to the electrons. This is followed by preliminary model results and our conclusions.

2. Microwave field and plasma generation

The absorption of microwaves and the creation of the plasma which transfers energy to the neutral species in the reactor involves the solution of two closely coupled problems. They are (1) the determination of the electromagnetic field patterns in the complex geometry of the reactor and (2) the formation of the electron distribution function. At a pressure of 5.3×10^3 Pa (40 Torr) and a gas temperature in the plasma region greater than 2000 K, the mean free path of an electron with neutrals is approximately 5×10^{-5} m. During the time an electron gains the average electron energy (approximately 2 eV) typical of the reactors we are modeling, it undergoes around 150 collisions and has a mean displacement of approximately 7×10^{-4} m. Thus, to an excellent approximation, the heating of the electrons results from the microwave electric fields which are local to the electron's spatial location.

The electron distribution function satisfies the Boltzmann equation. Because an electron undergoes many collisions as it is heated, the distribution function is nearly isotropic and can be well approximated by the zero and first order terms of a spherical expansion, the latter representing a distortion of the distribution function in the direction of the applied field, oscillating at the microwave frequency ω . The equation for the electron distribution function is

$$\begin{aligned} & \frac{1}{3} \left[\left(\frac{eE_0}{m_e} \right)^2 \frac{1}{v^2} \frac{\partial}{\partial v} \left(v^2 \frac{v_m}{v_m^2 + \omega^2} \frac{\partial F_0}{\partial v} \right) \right. \\ & \quad \left. + v^2 \nabla \cdot \left(\frac{1}{v_m} \nabla F_0 \right) + v \nabla \cdot \left(V \frac{\partial F_0}{\partial v} \right) \right] \\ & = L_i + L_x - \frac{2m_e}{M} \frac{1}{v^2} \frac{\partial}{\partial v} (v^3 v_m F_0) \end{aligned}$$

where E_0 is the amplitude of the electric field, e , m_e , and v are the electron charge, mass, and velocity respectively, v_m is the electron momentum transfer frequency, F_0 is the zero order approximation to the distribution function, V is the bulk fluid velocity, M is the neutral mass, and L_i and L_x are the inelastic loss terms that affect the distribution function respectively via ionization and excitation of rotational, vibrational, and electronic levels. The first term on the left represents the electron velocity diffusion due to the cumulative affect of many small angle scatterings of the electron induced by the oscillating electric field. The next two terms give the affect of the divergence of the diffusive and convective fluxes respectively. The last term on the right gives the energy loss due to elastic collisions.

Below a critical electric field the ionization rate is exceedingly small and hence the electron density and power deposited per unit volume are also small. Above the critical field the ionization rate and power deposition increase rapidly. If the power deposition is kept approximately constant and equal to the power injected into the reactor the electric field will rapidly adjust to a level close to the breakdown value. The power deposited per unit volume scales as $E_0^2 n_e$, where n_e is the electron density; as the electron density rises the electric field drops. The above considerations provide the necessary prescription for determining the time-dependent evolution of the electric field, the electron density, and the electron energy distribution. Using a set of elastic and inelastic cross-sections, the last two parameters define the time-dependent evolution of the fluid, including the build-up of atomic hydrogen and the rise in the gas temperature as the gas dissociates. In addition to H_2 and H, the Boltzmann calculation monitors the evolution of H_2^+ , H_3^+ , H^+ , and H^- and separately tracks each of the three lowest vibrational levels of H_2 . The hydrocarbon chemistry is initiated by energetic electrons, but being trace constituents the hydrocarbons do not significantly affect the electron development. It is useful to take advantage of the separation of time scales inherent in this problem. The electron distribution function is established on a time scale of around 10^{-8} s, the electron density growth occurs over approximately 1 μ s, and the hydrogen dissociation and hydrocarbon chemistry as well as fluid convection and diffusion occur on a millisecond time scale.

3. Results

We present calculations from the modules of the PECVD model that have been constructed and tested. The results obtained are designed to identify important physical mechanisms and to determine the regimes where they are critical. The calculation of the electric and magnetic fields was accomplished using SAIC's MASK code, a general two-dimensional electromagnetic code designed for the study of microwave devices of arbitrary geometrical configuration. The code introduces the electric fields at the input port and allows them to propagate into the reactor, which can include arbitrarily shaped regions of dielectric or conducting bodies. It employs a finite difference representation of the full set of time-dependent Maxwell equations and solves the initial value problem. Figure 1 shows the results of a simulation for a generic reactor in which the plasma is modeled as a spherical shell with a finite conductivity and a finite dielectric constant. Ultimately the plasma model will be replaced by results of Boltzmann calculations over the plasma region. The MASK calculation retains both the field strength and the phase dependence and determines the energy deposition in the target plasma. The results shown are contours of constant field amplitude for the axial (a) and radial (b) electric components in an azimuthally symmetric configuration. The bottom horizontal line is the axis of symmetry. The input

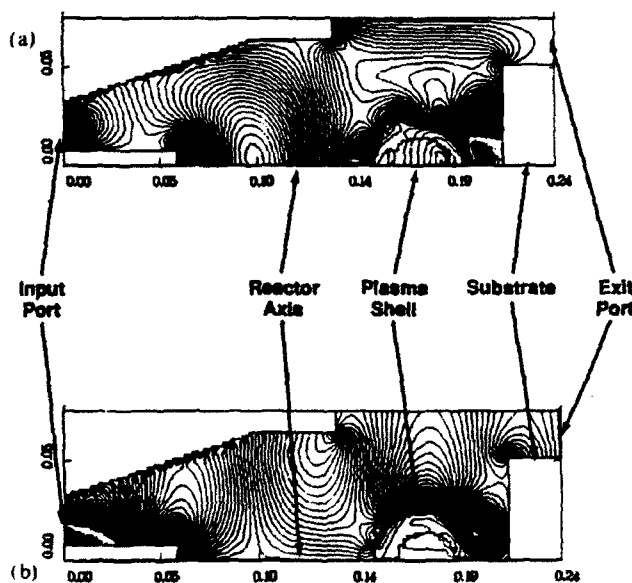


Fig. 1. MASK calculation of (a) the axial and (b) the radial components of the electric field in a model reactor. Shown in the figure are contours of constant amplitude. The reactor axis is the bottom horizontal line. The substrate shelf is at the right and the outlet for the reacting gases is above it. The plasma shell is centered on the axis to the left of the substrate.

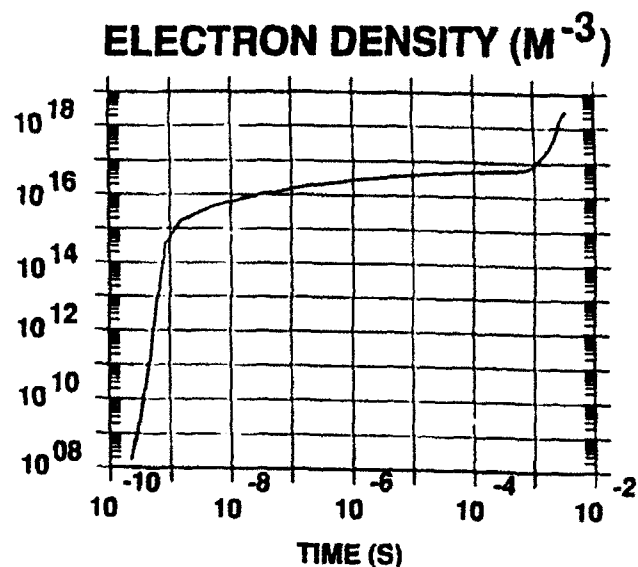
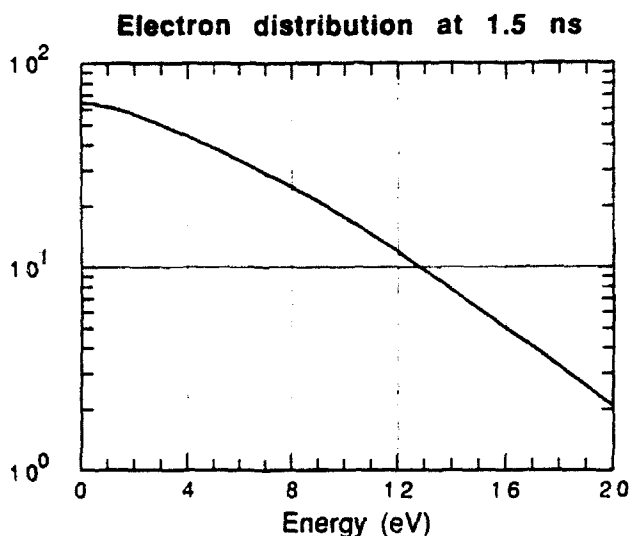


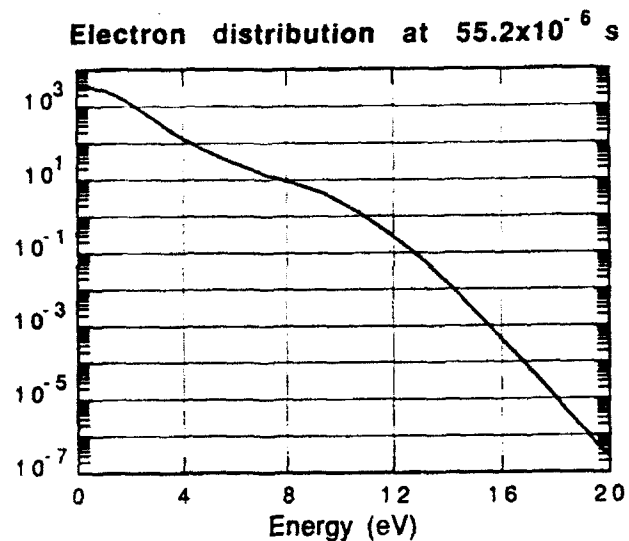
Fig. 2. Time development of the electron density at a point in the reactor of high electric field corresponding to the one-point simulation described in the text.

wave is introduced on the left and propagates into the reactor region through a radially expanding transition region. The substrate is on the right in the figure and immediately above it is the outlet for the reacting gases. The figure indicates regions of high and low field concentration which will provide an important tool for reactor design. This calculation determines the energy deposition in the plasma and, hence, the reactor efficiency. When performed self consistently it also predicts the shape of the plasma region and the subsequent coupling to the hydrodynamic calculation.

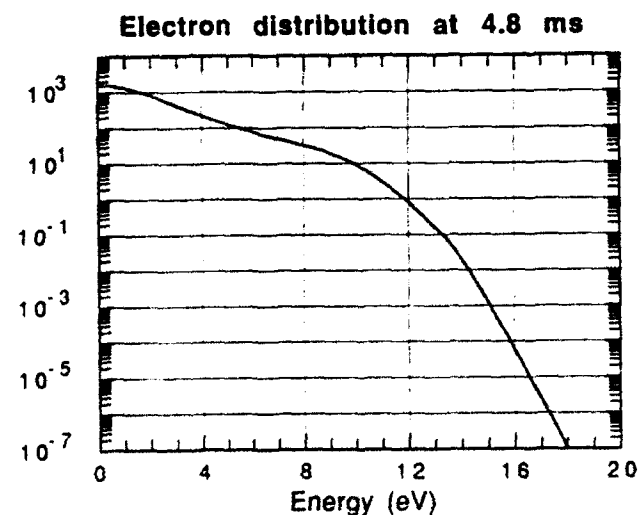
The Boltzmann module calculation simulates the electron and heavy particle evolution at a location within the reactor where the electric field is sufficiently high to create and sustain a plasma. We maintain a constant deposited microwave power and assume the gas pressure is kept constant at 5.3×10^3 Pa (40 Torr). The simulation runs for several milliseconds, beyond which time advection and diffusion effects, not included in this calculation, would become important. The initial conditions are $[H_2] = 1.28 \times 10^{24} \text{ m}^{-3}$, $[CH_4] = 1.28 \times 10^{22} \text{ m}^{-3}$, gas temperature $T = 300$ K. Figure 2 shows the electron density rising rapidly to approximately 10^{15} m^{-3} in around 1 ns, a result of the very energetic electron distribution at early times. Thereafter, it increases nearly two more decades over about 10^{-5} s, the slower increase being reflective of the less energetic electron spectrum as the electrons give up energy to the various inelastic processes. The increase on a millisecond time scale is associated with the conversion of the gas from the molecular to the atomic state which leads to an increasing fraction of atomic ions (which recombine much more slowly than molecular ions) and also causes



(a)



(b)



(c)

an adjustment in the electron distribution function. The evolution of the electron distribution function as determined by the Boltzmann equation is illustrated for these three time regimes in Fig. 3 in which the ordinate scale is arbitrary. In Fig. 3(a) at 1.5 ns we see the very energetic electron spectrum: the average electron energy is around 8 eV. In Fig. 3(b) at $55.2 \mu\text{s}$ the average energy has dropped to 2 eV and in Fig. 3(c) at 4.8 ms when atomic hydrogen predominates, the spectrum has changed again, the average electron energy increasing moderately to about 2.7 eV.

The evolution of the hydrocarbon species is simulated with a chemistry code that uses the output of the Boltzmann code. The reactions used in the hydrocarbon model are listed in Table I along with the constants A , b , and E which determine the rate coefficient k according to $k = AT^b \exp(-E/T)$. The code calculates the rate for each reverse reaction that is not known, using detailed balance. The hydrocarbon chemistry is initiated by the electrons which dissociate H_2 (and also the CH_4), causing the release of chemical energy and heating the gas. Figure 4 shows the gas temperature as a function of time for the simulation described above. In Fig. 5 we show the evolution of 12 hydrocarbon species plus H_2 and H out to 3 ms, at which time the H_2 and H densities are approximately equal. Although the formation of H is initiated by the electron dissociation of H_2 , after about 2.5 ms with rise in temperature, thermal dissociation of H_2 becomes predominant. The build-up of CH_3 due to the dissociation of CH_4 occurs very early (approximately $30 \mu\text{s}$) but it reacts with itself to form C_2H_6 and drops to a local minimum before 1 ms. As the temperature increases, however, the CH_3 recombination reaction rate decreases and CH_3 increases to a new maximum near 2 ms; thereafter it decreases once more as the CH_4 becomes exhausted. Acetylene (C_2H_2) results from the chain of reactions initiated by the formation of C_2H_6 , thence to C_2H_4 and, in turn, to C_2H_3 and finally to C_2H_2 which persists to the end of the simulation. In general, the hydrocarbon species do not have time to reach the equilibrium values that the gas temperature would dictate. Thus, the time between their formation in the plasma and their reaching the substrate determines the densities of the critical radical species reaching the substrate.

Fig. 3. Electron energy distribution for the simulation as in Fig. 2. (a) at a time before inelastic processes reduce the average electron energy; (b) at an intermediate time when the average electron energy is about 2 eV; (c) at a time when dissociation of the H_2 is nearly complete. The ordinate scale is arbitrary.

TABLE I. Hydrocarbon reactions and rate coefficients

Reaction	$A(10^{n(n-1)}(\text{m})^{3(n-1)}\text{s}^{-1})^a$	b	$E(\text{K})$	Range(K)
$\text{H} + \text{H} + \text{H}_2 \rightarrow \text{H}_2 + \text{H}_2$	2.7×10^{-31}	-0.6	0	100-5000
$\text{H}_2 + \text{H}_2 \rightarrow \text{H} + \text{H} + \text{H}_2$	1.5×10^{-9}	0	4.84×10^4	2500-8000
$\text{CH}_4 + \text{H} \rightarrow \text{CH}_3 + \text{H}_2$	3.6×10^{-20}	3.0	4.40×10^3	300-2500
$\text{CH}_3 + \text{H}_2 \rightarrow \text{CH}_4 + \text{H}$	1.1×10^{-21}	3.0	3.90×10^3	300-2500
$\text{CH}_4 + \text{CH}_4 \rightarrow \text{CH}_3 + \text{H} + \text{CH}_4$	$^{(12,9)}_{(18,6)} 3.3 \times 10^{-7}$	0	4.45×10^4	1500-3000
$\text{CH}_3 + \text{H} + \text{CH}_4 \rightarrow \text{CH}_4 + \text{CH}_4$	$^{(12,9)}_{(18,6)} 2.2 \times 10^{-21}$	-3.0	0	300-2500
$\text{CH}_3 + \text{CH}_3 \rightarrow \text{C}_2\text{H}_6 + \text{H}$	1.3×10^{-9}	0	1.34×10^4	1500-3000
$\text{C}_2\text{H}_6 + \text{H} \rightarrow \text{CH}_3 + \text{CH}_3$	5.0×10^{-11}	0	0	300-1500
$\text{CH}_3 + \text{CH}_3 \rightarrow \text{C}_2\text{H}_4 + \text{H}_2$	1.7×10^{-8}	0	1.61×10^4	1500-2500
$\text{CH}_3 + \text{CH}_4 \rightarrow \text{CH}_2 + \text{H} + \text{CH}_4$	$^{(12,9)}_{(18,6)} 1.7 \times 10^{-8}$	0	4.56×10^4	1500-3000
$\text{CH}_2 + \text{H} \rightarrow \text{CH} + \text{H}_2$	6.6×10^{-11}	0	0	300-2500
$\text{CH}_2 + \text{CH}_3 \rightarrow \text{C}_2\text{H}_4 + \text{H}$	6.6×10^{-11}	0	0	300-2500
$\text{C}_2\text{H}_6 + \text{H} \rightarrow \text{C}_2\text{H}_5 + \text{H}_2$	9.0×10^{-22}	3.5	2.62×10^3	300-2000
$\text{C}_2\text{H}_6 + \text{CH}_3 \rightarrow \text{C}_2\text{H}_5 + \text{CH}_4$	9.1×10^{-25}	4.0	4.17×10^3	300-2000
$\text{C}_2\text{H}_6 + \text{CH}_4 \rightarrow \text{CH}_3 + \text{CH}_3 + \text{CH}_4$	$^{(12,9)}_{(18,6)} 1.7 \times 10^{-5}$	0	3.43×10^4	800-2500
$\text{C}_2\text{H}_5 + \text{C}_2\text{H}_5 \rightarrow \text{C}_2\text{H}_4 + \text{C}_2\text{H}_6$	2.3×10^{-12}	0	0	300-1200
$\text{C}_2\text{H}_5 + \text{CH}_4 \rightarrow \text{C}_2\text{H}_4 + \text{H} + \text{CH}_4$	1.7×10^{-7}	0	1.56×10^4	700-1500
$\text{C}_2\text{H}_4 + \text{H} \rightarrow \text{C}_2\text{H}_3 + \text{H}_2$	2.5×10^{-10}	0	5.14×10^3	700-2000
$\text{C}_2\text{H}_4 + \text{CH}_4 \rightarrow \text{C}_2\text{H}_2 + \text{H}_2 + \text{CH}_4$	$^{(12,9)}_{(18,6)} 4.3 \times 10^{-7}$	0	3.99×10^4	1500-2500
$\text{C}_2\text{H}_4 + \text{CH}_4 \rightarrow \text{C}_2\text{H}_3 + \text{H} + \text{CH}_4$	$^{(12,9)}_{(18,6)} 4.3 \times 10^{-7}$	0	4.86×10^4	1500-2500
$\text{C}_2\text{H}_4 + \text{CH}_3 \rightarrow \text{C}_2\text{H}_3 + \text{CH}_4$	7.0×10^{-13}	0	5.59×10^3	300-1000
$\text{C}_2\text{H}_3 + \text{H} \rightarrow \text{C}_2\text{H}_2 + \text{H}_2$	3.3×10^{-11}	0	0	300-2500
$\text{C}_2\text{H}_3 + \text{CH}_4 \rightarrow \text{C}_2\text{H}_2 + \text{H} + \text{CH}_4$	$^{(12,9)}_{(18,6)} 5.0 \times 10^{-9}$	0	1.61×10^4	500-2500
$\text{C}_2\text{H}_2 + \text{H} + \text{CH}_4 \rightarrow \text{C}_2\text{H}_3 + \text{CH}_4$	$^{(12,9)}_{(18,6)} 1.11 \times 10^{-30}$	0	3.5×10^2	300-500
$\text{C}_2\text{H}_2 + \text{H} \rightarrow \text{C}_2\text{H} + \text{H}_2$	1.0×10^{-10}	0	1.19×10^4	300-3000
$\text{C}_2\text{H} + \text{H}_2 \rightarrow \text{C}_2\text{H}_2 + \text{H}$	2.5×10^{-11}	0	1.56×10^3	300-3000
$\text{C}_2\text{H}_2 + \text{CH}_2 \rightarrow \text{C}_1\text{H}_3 + \text{H}$	3.0×10^{-12}	0	0	> 298
$\text{C}_2\text{H}_2 + \text{C}_2\text{H} \rightarrow \text{C}_4\text{H}_2 + \text{H}$	5.8×10^{-11}	0	0	300-2500
$\text{C}_2\text{H}_2 + \text{CH}_4 \rightarrow \text{C}_2\text{H} + \text{H} + \text{CH}_4$	$^{(2,9)}_{(18,6)} 6.6 \times 10^{-8}$	0	5.38×10^4	1500-3500

^a n denotes the number of reactants.

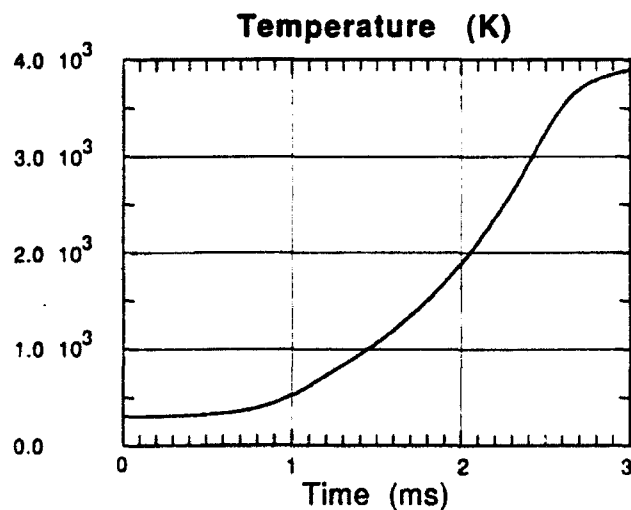
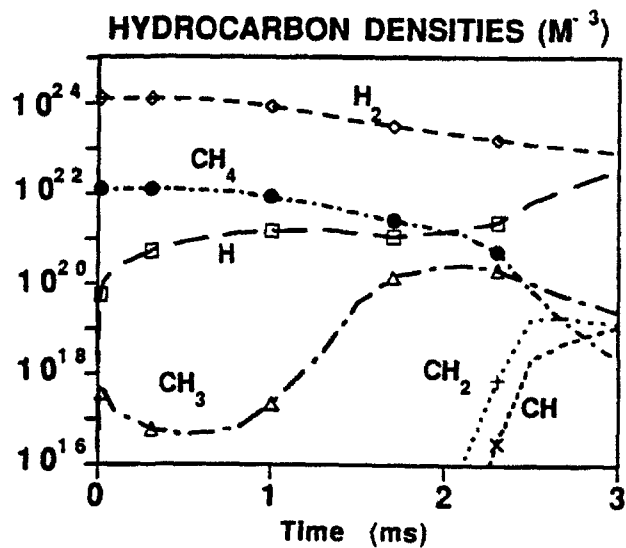


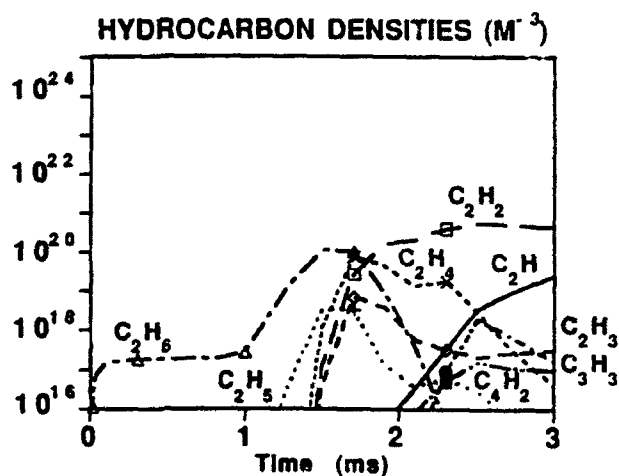
Fig. 4. Evolution of the gas temperature for the simulation as in Fig. 2.

Finally, we present preliminary results of a fluid simulation of a generic PECVD reactor, using SAIC's FUGG code which is capable of performing fluid calculations over arbitrarily complex geometries. The code

employs an unstructured grid allowing extremely fine resolution in critical areas while employing coarser gridding in regions where quantities vary slowly. The code was designed for the study of flow problems dominated by convection and is presently being modified to incorporate thermal conduction and viscosity effects. In Fig. 6 we show two examples of the code's triangular gridding capability. In both cases a cross-section of the azimuthally symmetric model reactor is shown, where the left vertical boundary represents the reactor axis. Three gas inlet ports are modeled allowing the gas to enter at the top (in Fig. 6(a) the plenum region above the inlet ports is also modeled). The gas exits through the horizontal boundary at the bottom right. The substrate wafer is represented in Fig. 6(a) by the left half of the lower horizontal boundary and in Fig. 6(b) by the shelf at the lower left. In Fig. 6(a) the variable gridding capability is clearly illustrated and, in particular, the fine gridding needed in the inlet ports is shown. Figure 7 shows results of a fluid calculation for the reactor of Fig. 6(b) in which hydrogen gas enters at 50 m s^{-1} at a pressure of $5.3 \times 10^3 \text{ Pa}$ (40 Torr). A heating source of 1.5 kW over a spherical volume of radius 0.035 m, centered on the reactor axis and mid-

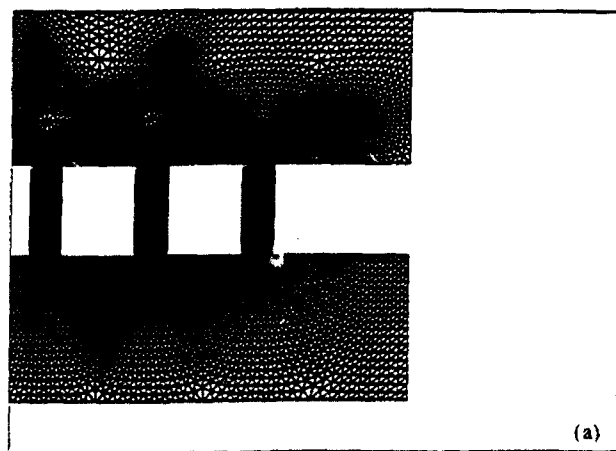


(a)

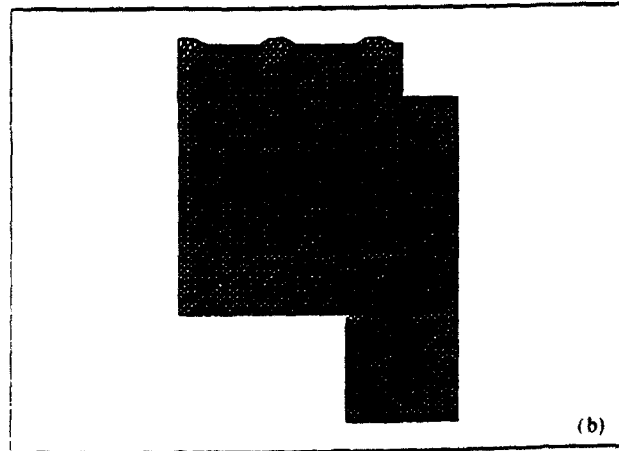


(b)

Fig. 5. Evolution of the hydrocarbon densities for the simulation as in Fig. 2.



(a)



(b)

Fig. 6. Examples of the FUGG code's unstructured gridding capability for two model reactors ((a) and (b)).

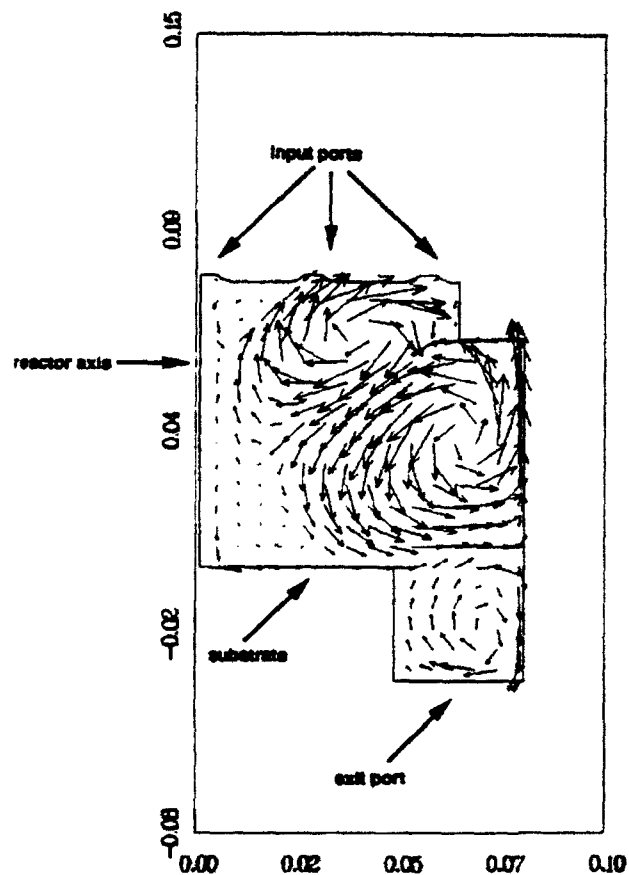


Fig. 7. Velocity field for reactor of Fig. 6(b) in which H_2 enters at a velocity of 50 m s^{-1} at a pressure of $5.3 \times 10^1 \text{ Pa}$ (40 Torr). A heating source of 1.5 kW to simulate the effect of the plasma is centered on the axis between the inlet port and the water in a spherical volume of radius 0.035 m. The ordinate and abscissa dimensions are in meters.

way between the inlet port and the wafer, is included to simulate approximately the effect of the plasma source. Shown are velocity vectors for the flow 2.6 ms after the plasma is turned on. Also calculated but not shown are the pressure, density, and temperature fields. While conclusions should be tempered because of the current lack of inclusion of thermal effects in the code and because the results represent a transient pre-steady-state stage, the effects of buoyancy are apparent. The complex vortex flows seen suggest this reactor configuration would be very poor for efficient diamond deposition.

4. Discussion and conclusions

We have presented results of a model under development that will permit the simulation of PECVD reactors of arbitrary geometry. The model will be an important tool providing better understanding of the microscopic processes occurring within the reactor, permit parameter studies to identify those parameters which critically affect both the rate of deposition and the uniformity of the deposition over the wafer surface, and ultimately enable the design of improved reactors. We have identified several critical elements in the modeling effort that need to be treated carefully if simulation results are to be meaningful. First, the electromagnetic fields which initiate the plasma formation need to be determined in the realistic reactor geometry, including

effects of all metallic, dielectric, and insulator elements actually present, to ensure that the fields are high in the desired plasma formation region but not elsewhere. Second, the coupling of the fields to the plasma electrons, to determine accurately both the time development of the electron density and their energy distribution, is most important for determining the evolution of the rate of hydrogen dissociation and the rise in the gas temperature. This, in turn, critically determines the non-equilibrium hydrocarbon chemistry development, a third area that needs to be carefully modeled. Finally, the flow of hydrocarbon radicals to the wafer is very sensitive to the reactor's geometrical configuration, its thermal properties, and the location of the plasma relative to the wafer. In conclusion, our results emphasize the highly non-equilibrium and coupled nature of PECVD reactor processes and the strong influence of reactor geometry. A numerical simulation that is useful must address all such issues.

Acknowledgment

This research was supported by the U.S. Army Missile Command and sponsored by the Defense Advanced Research Projects Agency (DARPA) under contract DAAH01-90-C-0279 and by the Independent Research and Development Program of Science Applications International Corporation (SAIC).

Nonlinear signal processing using integration of fluid dynamics equations

by

S. Eidelman, W. Grossmann and A. Friedman*
Science Applications International Corporation
1710 Goodridge Drive
McLean, Virginia 22102

1. INTRODUCTION

Very recently, there have been exploratory efforts in image processing based on nonlinear methods.^[1] These efforts involve systems of nonlinear hyperbolic partial differential equations in combination with local wave representation, such as wavelets, for signal enhancement.^[2,3,4] Techniques based on Kalman filtering for feature extraction from complex time-evolving scenes, as well as neural network approaches to image analysis and feature identification, can also be shown to involve nonlinear PDE analogies. The use of nonlinear methods, however, is largely unexplored and may provide another level of improvement for image processing.

If the purpose of an image enhancement process is to highlight the edges of an image, then the technique used in the frequency domain is usually highpass filtering. An image can be blurred, however, by attenuating the high-frequency component of its Fourier transform. Since edges and other abrupt changes in the gray levels are associated with high-frequency components, image sharpening can be achieved in the frequency domain by a highpass filtering process, which attenuates the low-frequency without disturbing high-frequency information in the Fourier transform. The primary problem with this technique is that an ideal discontinuity will have an infinite spectrum of frequencies associated with it. When filtering is applied, some frequencies are cut off, leading to a loss of some edges in an image.

It is interesting to observe that in the field of Computational Fluid Dynamics (CFD) similar problems exist in simulating flows with discontinuities. The problem of simulating flows with discontinuities is less forgiving, since an incorrect calculation usually leads to a complete distortion of the flow field. This has led CFD scientists to develop sophisticated algorithms that identify and preserve discontinuities while integrating the flow field in the computational domain. In the image domain, sharpening is usually done by differentiation. The most commonly used methods involve the use of either gradients or second derivatives of the pixel information. Central differencing is usually used to calculate the derivatives. CFD research has shown that this strategy will lead in many cases to a smearing of the flow discontinuities (analog of the image edges in image enhancement).

Here, we describe a new and unique image sharpening method based on computational techniques developed for CFD. Our preliminary experience with this method shows its capability for nonlinear enhancement of image edges as well as deconvolution of an image with random noise. This indicates a potential application for image deconvolution from sparse and noisy data resulting from measurements of backscattered laser-speckle intensity.

*Present address: Brookhaven National Lab., Apton, NY 11973

2. THE CFD IMAGE ENHANCEMENT TECHNIQUE

Considerable attention has been devoted to the development of numerical methods and algorithms for Computational Fluid Dynamics during the last thirty years. In recent years, however, our understanding of numerical algorithms for a particular class of problems in gas dynamics described by the Euler equations has become more complete. The main numerical difficulty in solving inviscid compressible flows described by Euler equations is the occurrence of features that, in the inviscid approximation, are discontinuous and even in the presence of viscosity are too small to be resolved on an affordable computational mesh. These flow discontinuities in which the fluid state jumps across shock waves or contact surfaces are extremely important in fluid simulations. Most of the efforts in developing numerical techniques in fluid dynamics over the last twenty years were devoted to accurate simulations of these discontinuities. Initially, naive numerical methods that used a formal finite difference representation of the conservation equations on a computational grid were employed. That led to disastrous results, smearing of the discontinuities, and spurious oscillations. Subsequently, sophisticated nonlinear techniques, which allowed accurate simulations of complex discontinuities without smearing and ringing, were developed. These new methods also satisfy a very demanding criteria for robustness and allow simulation of the wide range of flow problems without adjustment or tuning of the numerical technique.

The numerical methods that allow high accuracy resolution of flow discontinuities are so-called TVD (Total Variation Diminishing) methods. The Second Order Godunov Method is one of the most successful numerical techniques developed for this purpose. In Figure 1, an example is given of a solution using the Second Order Godunov Method for a complicated case of multiple shock waves,^[5] illustrating the ability of this method to capture and simulate sharp discontinuities.

The Second Order Godunov Method was developed based on an understanding of the phenomenology of signal propagation in the gasdynamical system. The numerical algorithm implementing this method is not analytic and is based on a set of steps that can be considered as wave filters. These filters are designed to not smear the discontinuity (edge), suppress the spurious oscillations, and propagate the relevant signals through the system. The following algorithmic steps are performed to advance the solution for a single iteration in the Second Order Godunov Method:

1. Local Extrapolation
2. Monotonicity Constraint
3. Characteristics Constraint
4. Riemann Problem Solution
5. Integration

It is interesting to note that most of these steps have an analog in conventional image processing methods. Here, we will give an explanation of the function of each algorithmic step of the Second Order Godunov Method and where applicable, will point to its possible analog in conventional signal processing techniques.

Step 1 consists of extrapolation of the values in the computational grid (pixel) cell to the edges of the cell. Linear or nonlinear extrapolation can be used. This step is analogous to the standard edge sharpening techniques used in image processing, with one important difference: the extrapolation is done not for the value itself but for its flux (change of value across cell boundary).

Step 2 includes a monotonicity constraint for the values at the cells' edges. This is analogous to the nonlinear technique of the locally monotonic regression^[6] only recently introduced for signal processing.

Step 3 subjects the values at the edges to the constraints derived from a solution of one dimensional characteristics. This step assures that the values at the edges have not been extrapolated from directions inconsistent with the characteristic solutions. This prevents extrapolation as well as smearing or overshoot of the discontinuities. For the image processing application, this can be regarded as a form of automatic edge detection step where the shock waves are associated with the edges of an image.

Step 4 uses an exact solution of the system of the gas dynamic equations for calculation of the flux values based on the extrapolated values of the parameters at the left and right side of the edges. This step has no analogy in image processing. However, since the analytical solution includes discontinuities, an exact calculation of the flux at the edge location is allowed, even if this flux is calculated through a discontinuity.

Step 5 consists of finite volume integration of the system of conservation laws. Here, the image is effectively treated as a flow field; the flux integration serves as a smoothing filter from the image perspective.

Application of these steps can be considered as the application of a unique filter stack with proven properties of discontinuity preservation and robustness. Below we illustrate uses of this technique for practical problems of image processing that exemplify the feasibility and advantages of this approach.

The use of image analogies for image processing is not new. One widely applied technique treats an image as a potential field where the image potential acts as a force on the edges that are represented as elastic curves with some elastic properties.^[3] Our approach, as stated, involves an application of a technique developed for gas dynamic problems for image deconvolution. Although this technique is very new, an analysis of the basic steps presented above and our experience with its application for image deconvolution show that this nonlinear algorithm has considerable potential for edge enhancement and filtering of extremely noisy signals.

3. IMAGE ENHANCEMENT BY THE SECOND ORDER GODUNOV METHOD

The field of gray scale intensity of an image can be translated into a flow field. To every image pixel we add a corresponding cell of the computational domain with values of the gas dynamical parameters proportional to the values of the gray scale. Since there are at least five gasdynamical parameters that can be defined in every cell of the computational domain (pressure, density, two velocity components and γ) and only one parameter in the image domain, cell mapping is not unique. Our understanding of the basic gasdynamical processes plays a major role in completing the analogy. Appropriate mapping of the image gray scale intensity into a flow field creates conditions favorable for the formation or enhancement of field discontinuities. For example, a shock wave reflecting from a wall or a contact surface can increase in strength, or two colliding flow streams will produce a contact surface that will become stronger in time. If we have a numerical technique to resolve these discontinuities accurately, then with successive numerical integration of the flow field, the discontinuities will sharpen as the solution evolves in time. Then by inverse mapping of the flow field to the image gray scale field, we can reconstruct an enhanced image. Below we give some examples of practical application of this technique.

3.1. Edge sharpening for a sinusoidal distribution

In Figure 2 results are given for edge definition of a one dimensional signal. The original sinusoidal signal is shown in Figure 2a. This example was chosen to test the ability of our technique to identify the edges of an image where the signal strength has deteriorated in the vicinity of the

edges, producing a gradual (instead of sharp) increase in the gray scale intensity. We observe that application of our technique results in significant sharpening of the edges, even after 15 or more iterations.

In Figure 3 random noise has been added to the sinusoidal signal shown in Figure 2a. The level of random noise addition corresponds to 10% of the maximum intensity of the original signal. The original signal with the random noise is shown in Figure 3a. In Figures 3b, 3c, 3d we observe successive noise filtering and edge enhancement with application of our algorithm for 15, 30, and 45 iterations correspondingly. We see that the edges of the final processed signal are located at exactly the same position as shown in Figure 2d for the uncontaminated signal.

Figure 4 illustrates the application of our algorithm to the signal that has been contaminated with 50% addition of random noise. Significant noise filtering occurs after 15 iterations and edge definition at the exact original locations after 45 iterations.

In Figure 5 the results are shown for a signal with 100% random noise added. Here again the signal is quickly filtered and the edges are picked up exactly at the correct locations.

3.2. Edge sharpening for a two dimensional image

Figure 3 contains a picture of Washington, DC taken from a Russian satellite. Digital representation of this picture had 150 dots per inch resolution. A fragment of the picture shown in Figure 6 is represented on an evenly spaced 400×360 grid. We take the gray scale pixel information of this picture and convert the data into initial conditions for a gasdynamic problem by assigning the values of pressure and density in the computational domain directly proportional to the values of the pixels on the gray scale. Now the gasdynamic problem is defined and we can solve it using our high resolution Second Order Godunov Method. In Figures 7a, 7b, and 7c results in the pixel plane are shown after three, six, and nine iterations respectively in the gasdynamic domain. By "iteration," we mean that the flow solver integration algorithm was applied to the given flow field, or in this case, the pressure and density data derived from the initial picture. Even after three iterations, the picture is significantly sharper and continues to improve with more iterations.

A more detailed examination of the sharpening effect can be obtained by looking at the one-dimensional cross section of the picture plane. In Figure 8, an arbitrary cross section of the original picture shown in Figure 6 is given. For clarity we show only the first fourth of the actual pixels in the cross section. We can see here that this particular cross section contains a multitude of sharp edges expressed only by three or four points. Further sharpening of these edges by a standard differentiation technique will lead to significant smearing of a number of the discontinuities. In Figure 9, the same cross section is shown after three iterations with the Second Order Godunov solver. Significant enhancement of all the sharp edges is evident. The process of enhancement can be followed in Figures 6b, 6c and 6d corresponding to six, nine, and twelve iterations. Continuous improvement in the definition of edges can be observed.

In Figures 10, 11a, 11b, 11c, and 11d, we demonstrate the ability of the current nonlinear PDE methodology to enhance simultaneously the high and low frequency features of an image. The amplitudes of both short and long wavelengths are simultaneously enhanced. However, as seen in the circled area, long wavelength features that retain one grid-cell discontinuities exhibit interesting behavior in that the cell-specific discontinuity, which appears in Figure 10, disappears in Figures 11b and 11c, but reappears in Figure 11d. The long wavelength definition continues to be enhanced in Figures 11a-11d. The origin of this behavior is presently unknown.

3.3. Application to Medical Imaging

Images of internal organs obtained with a Gamma Camera are usually of marginal quality and need significant post-processing to be useful for medical diagnostics. This is especially true if multiple pictures are taken of moving parts of the body, such as the heart, with low pixel resolution. In this section, we will demonstrate the application of our CFD technique for deconvolution of Gamma Camera images obtained during medical examinations.

Shown in Figure 12 is an image of the human heart produced by the staff of the Georgetown University Hospital, Department of Nuclear Medicine, using a Siemens Gamma Camera. This image contains a sequence of 64x64 pixel frames showing the heart at a sequence of time intervals. This plane image, originally recorded in 256 shades of gray scale, is presented here in 64 shades of gray. In Figures 12b, 12c and 12d the deconvoluted image is shown after 6, 12 and 18 processing iterations by our nonlinear technique. We observe in these figures a significant improvement in the image quality over the images in Figure 12a. Some of the diffuse edges in Figure 12a are clearly pronounced in Figures 12c and 12d. We have also applied our CFD technique to the Gamma Camera images of the brain and liver and have found a significant image deconvolution and edge enhancement.

4. CONCLUSIONS

The CFD technique described here for nonlinear signal and image processing is based on numerical techniques developed for Computational Fluid Dynamics, namely, the Second Order Godunov Method. We have demonstrated the application of this numerical method to signal processing, resulting in significant signal deconvolution and edge enhancement effects. Our preliminary analysis has shown that the Second Order Godunov Method, when applied to the gray scale intensity field of an image, is equivalent to an application of a unique filter stack. This filter stack has automatic edge detection, noise reduction and edge enhancement properties. We have demonstrated this nonconventional technique for the system of gas dynamic equations, where the Second Order Godunov Method assures high accuracy resolution of the flow discontinuities that are analogous to the edges in the image field. However, the same methodology can be applied to a reduced set of nonlinear hyperbolic partial differential equations, which will result in a significant optimization of the proposed technique.

5. REFERENCES

1. Nonlinear Image Processing, Proceedings of SPIE, Feb. 1990, Santa Clara, CA.
2. Osher, S. and L. Rudin, "Feature-Oriented Image Enhancement Using Shock Fielders," *Siam J. Numer. Anal.*, Vol. 27, pp. 919-940, NY, 1990.
3. Samadani, R. et al., "A Computer Vision System for Automatically Finding the Auroral Oval from Satellite Images," Image Processing Algorithms and Techniques. Proceedings of SPIE, 1244, 68-75, Feb. 1990.
4. Greene, R. R. et al., "Process and Apparatus for the Automatic Detection and Extraction of Features in Images and Displays," U.S. Patent 4,906,340, Mar. 1990.
5. Collela, P. and P. Woodward, "Piecewise Parabolic Method (PPM) for Gasdynamic Simulations." *J. Comp. Phys.*, Vol. 54, 174-201, 1984.
6. Restrepo, A. and A.C. Bovik, "Statistical Optimality of Locally Monotonic Regression." Non-linear Image Processing, Proceedings of SPIE, 1247, 89-99, Feb. 1990.

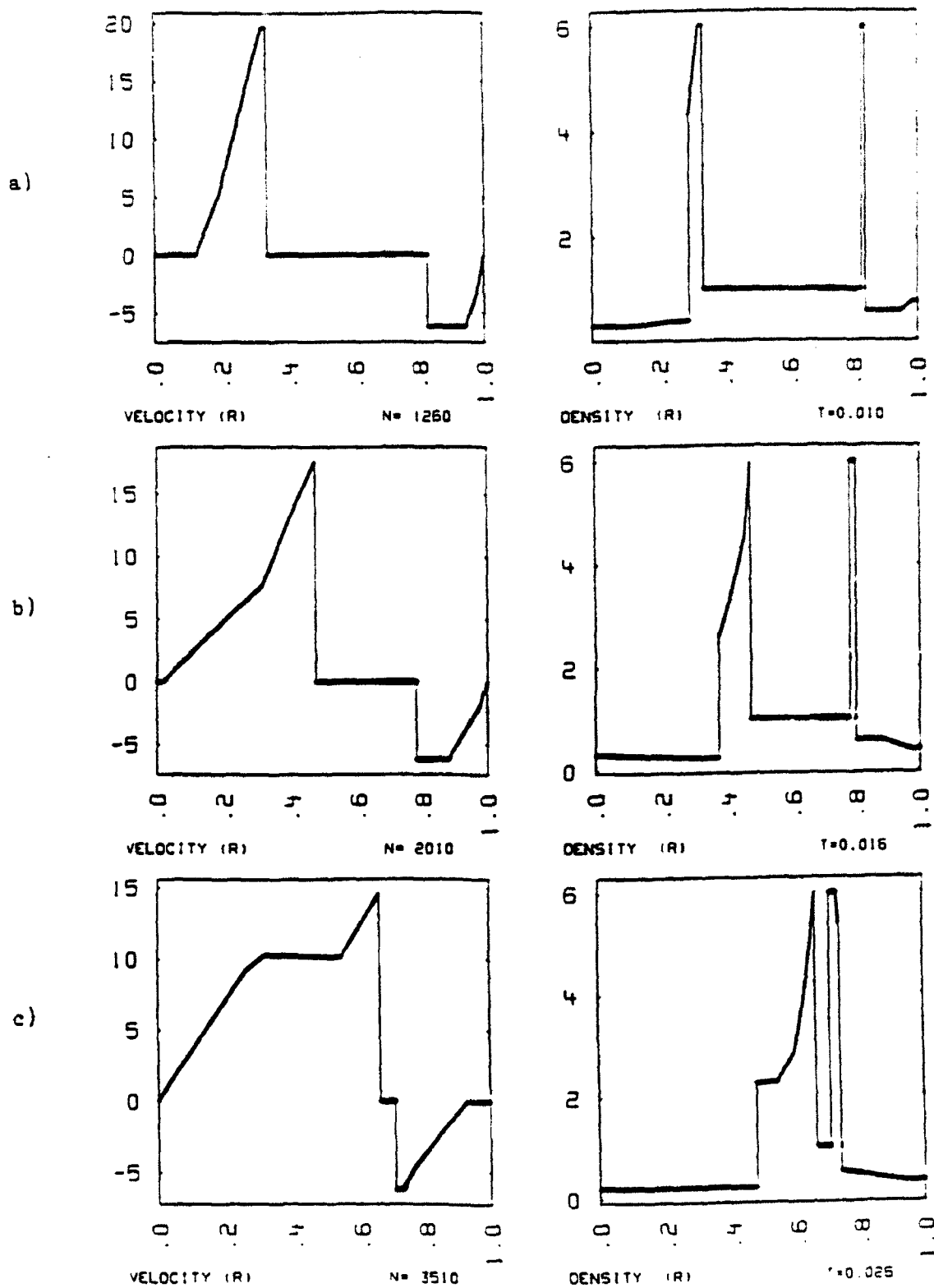


Fig. 1. High resolution of flow discontinuities obtained with the Second Order Godunov Method.

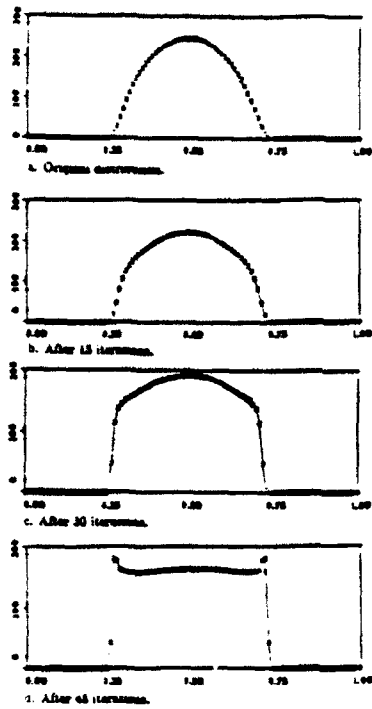


Fig. 2. Edge enhancement for a sinusoidal distribution without noise.

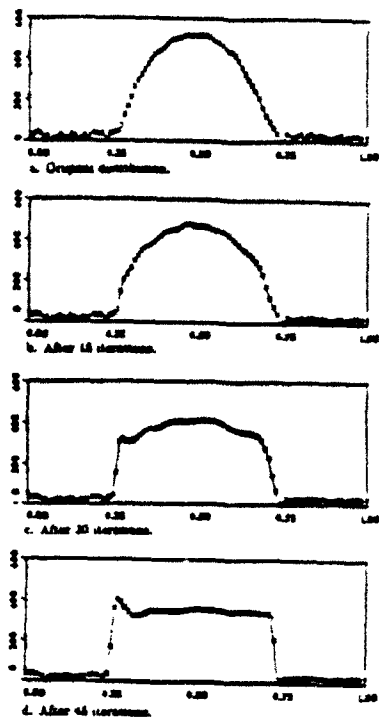


Fig. 3. Edge enhancement for a sinusoidal distribution with 10% intensity random noise.

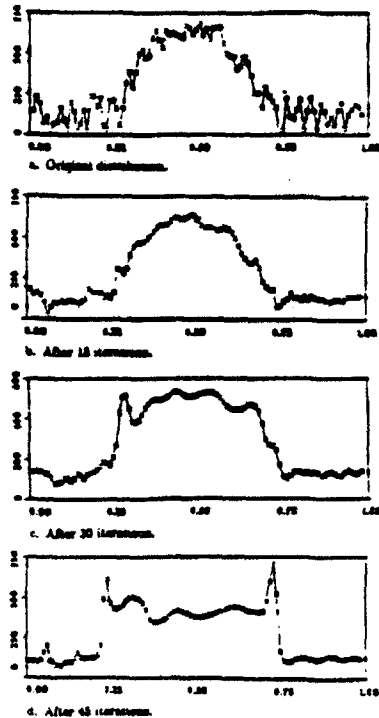


Fig. 4. Edge enhancement for a sinusoidal distribution with 50% intensity random noise.

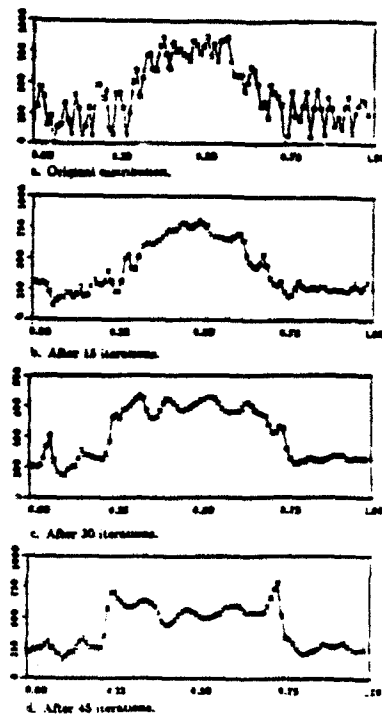


Fig. 5. Edge enhancement for a sinusoidal distribution with 100% intensity random noise.

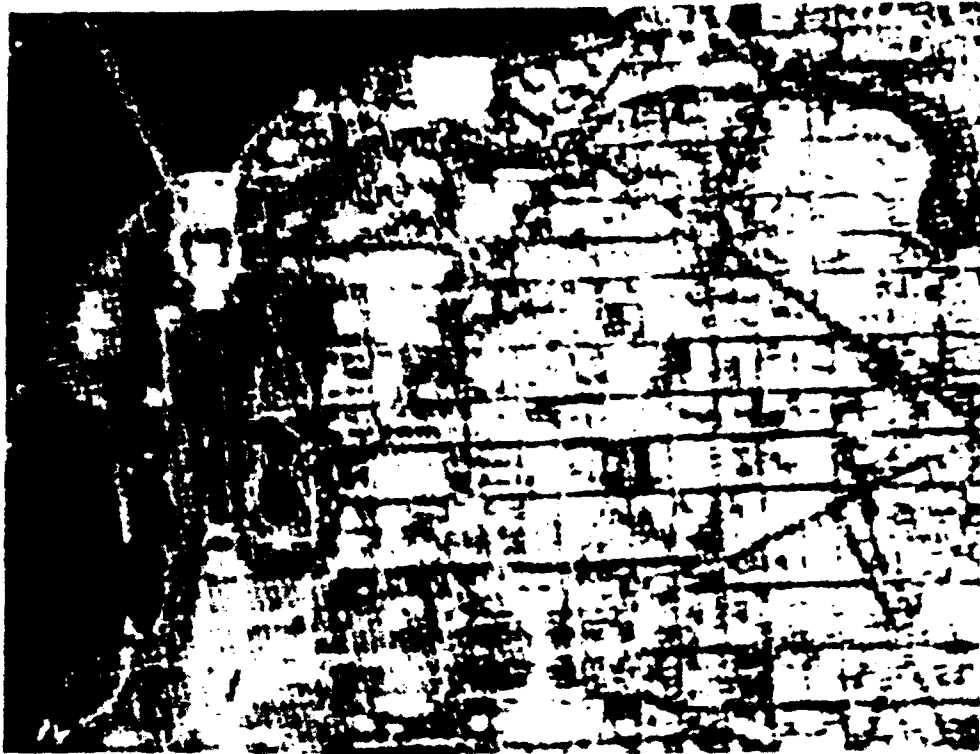


Fig. 6. The original satellite photograph of Washington, DC with resolution reduced to 150 dots/inch.

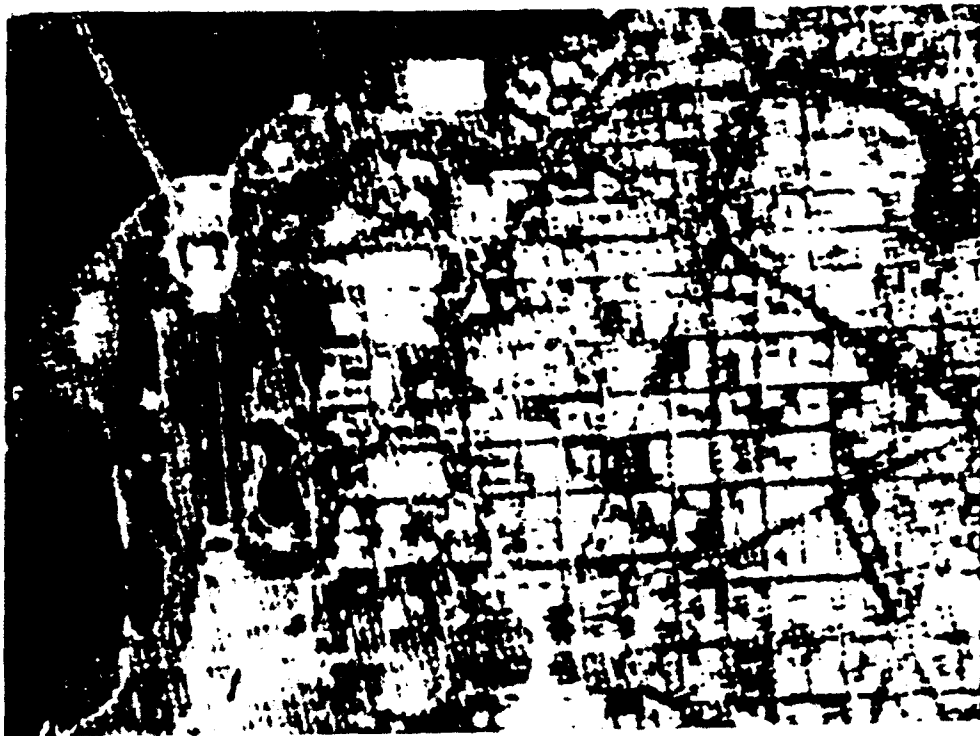


Fig. 7a. The sharpened picture after the Godunov solver has been used. After three iterations. Note the details that appeared on the Potomac. These details are barely visible even on the original high resolution photograph.



Fig. 7b. After six iterations.

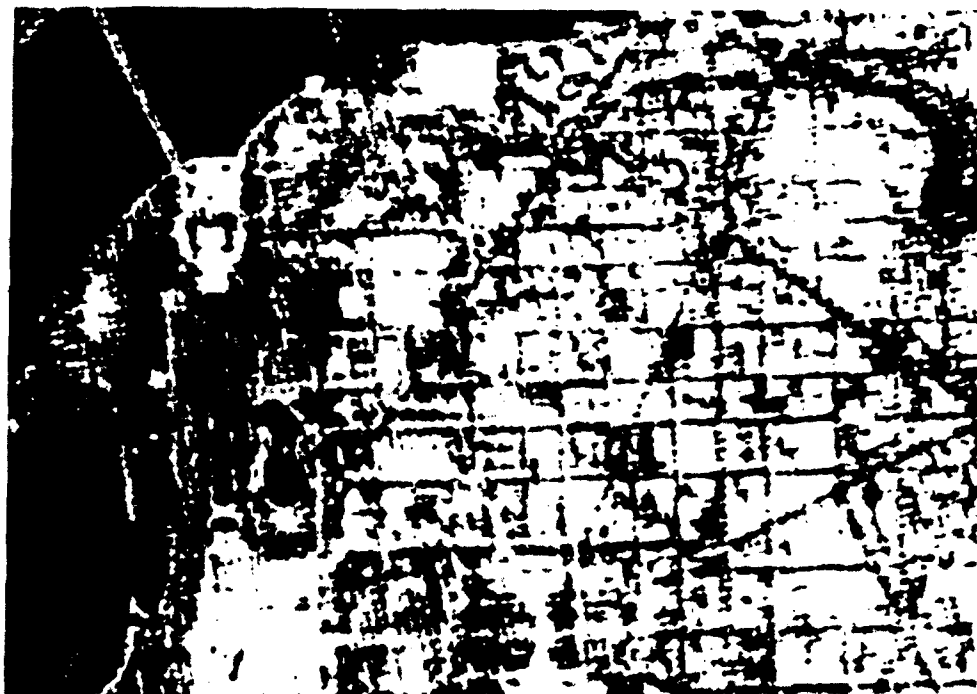


Fig. 7c. After nine iterations.

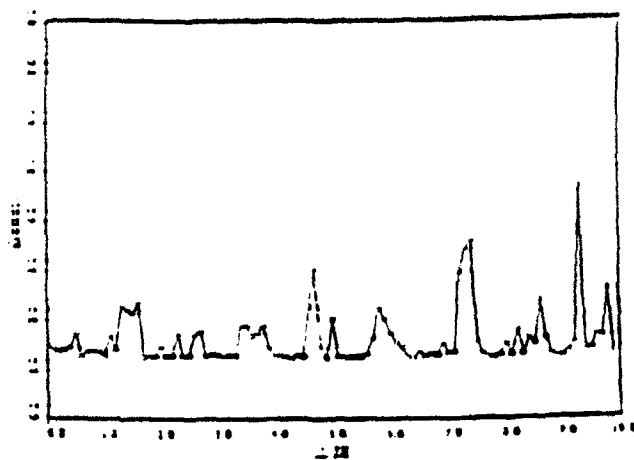


Fig. 8. Gray scale density of a cross section of the original image.

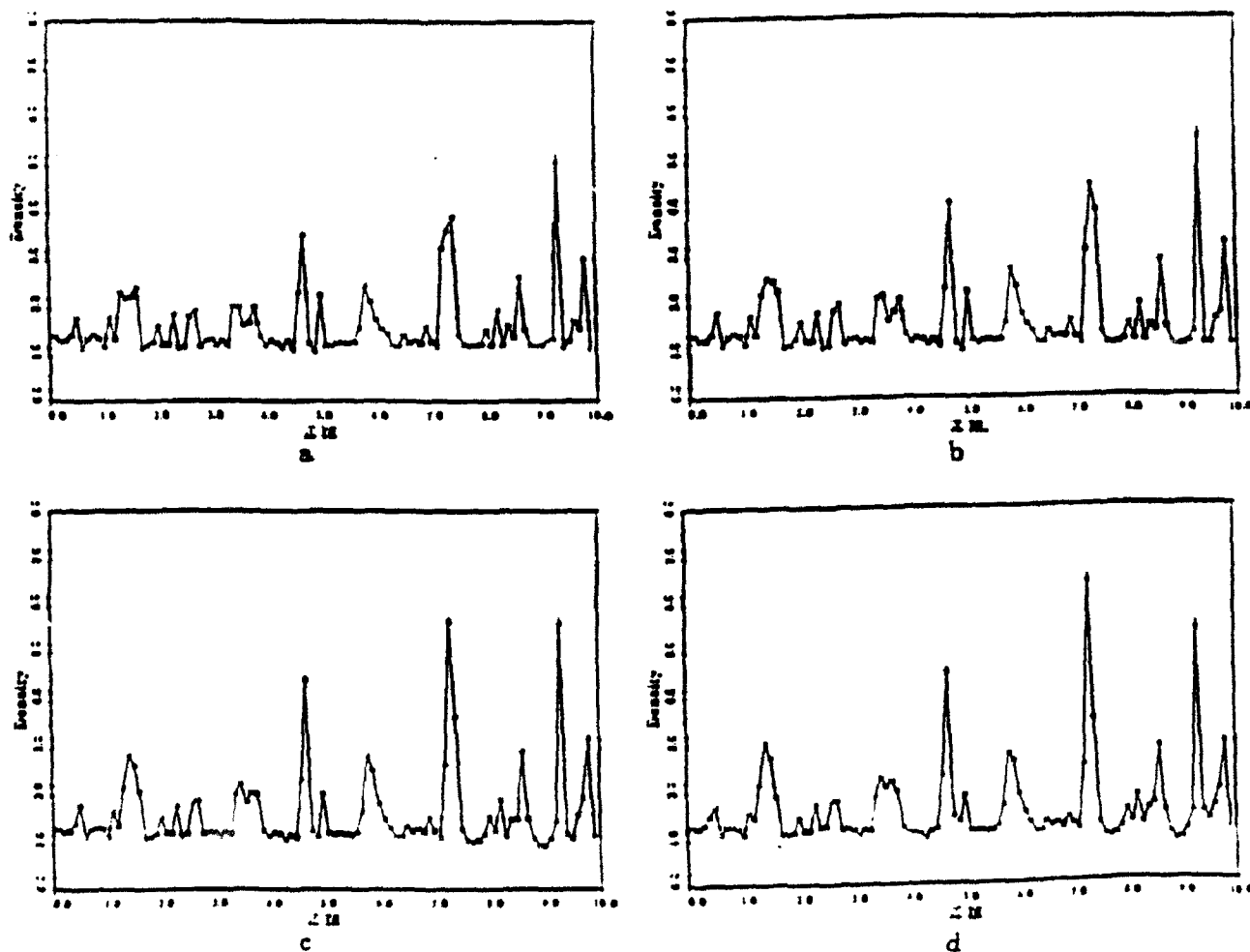


Fig. 9. Gray scale density of the CFD processed image: (a) after 3 iterations; (b) after 6 iterations; (c) after 9 iterations; (d) after 12 iterations.

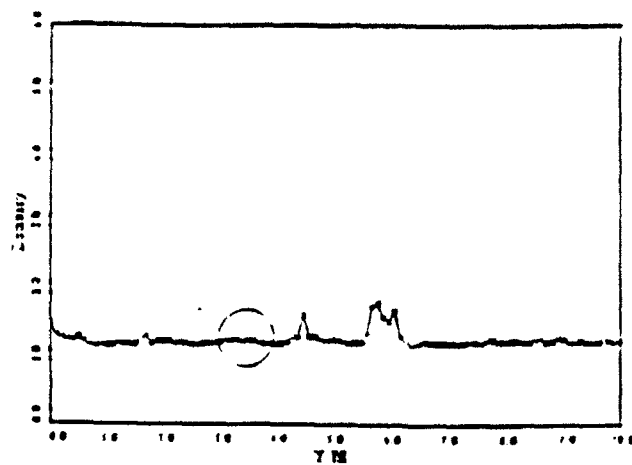
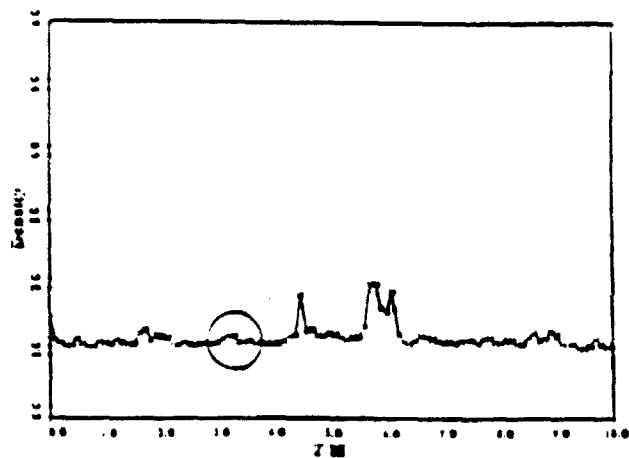
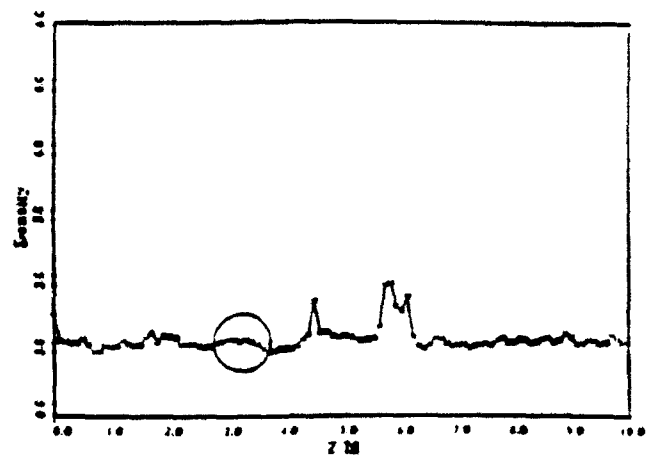


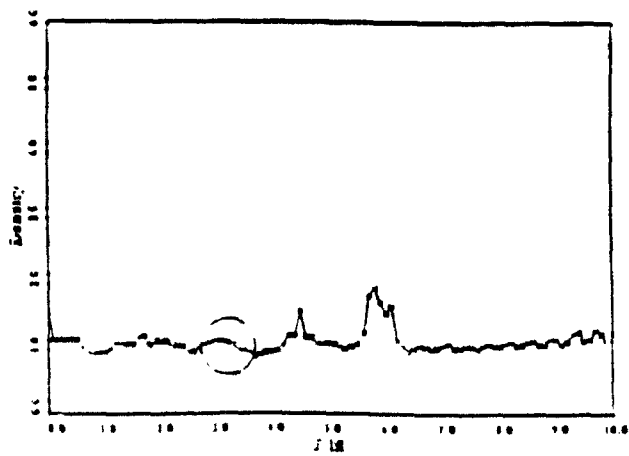
Fig. 10. Gray scale density of a cross section of the original image.



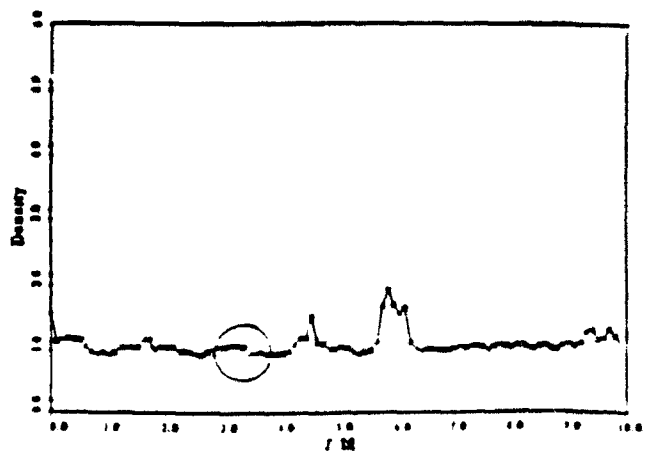
a



b



c



d

Fig. 11. Gray scale density of the CFD processed image: (a) after 3 iterations; (b) after 6 iterations; (c) after 9 iterations; (d) after 12 iterations.



(a)

(b)

(c)

(d)

Fig.12. Image of human heart taken by a Siemens Gamma Camera. (a) Original image 64x64 pixels per frame; (b) Image after six processing iterations; (c) Image after 12 processing iterations; (d) Image after 18 processing iterations.

Review of Propulsion Applications and Numerical Simulations of the Pulsed Detonation Engine Concept

S. Eidelman,* W. Grossmann,† and I. Lottati*

Science Applications International Corporation, McLean, Virginia 22102

Here we review experimental and computational studies of the pulsed detonation engine concept (PDEC) and present results of our recent numerical study of this concept. The PDEC was proposed in the early 1940s for small engine applications; however, its potential was never realized due to a complicated, unsteady operation regime. In this study, we demonstrate the use of current advances in numerical simulation for the analysis of the PDEC. The high-thrust/engine volume ratio obtained in our simulations demonstrates promising potential of the pulsed detonation engine concept.

Introduction

EARLY developments of engine technology leading to practical propulsion engines were almost completely associated with steady-state engine concepts. Unsteady concepts, which initially appeared promising, never evolved from the conceptual state and have remained for the most part unexplored. The early work in unsteady propulsion suffered from a lack of appropriate analytical and design tools, a condition which seriously impeded the advancement of the unsteady concepts to a practical stage.

In this paper, we review the historical development of unsteady propulsion by concentrating on the particular concept of the intermittent detonation engine, and discuss current research activities in this area. A review of the literature¹⁻²⁴ reveals that a significant body of experimental and theoretical research exists in the area of unsteady propulsion. However, this research has not been extended to the point where a conclusive quantitative comparison can be made between impulsive engine concepts and steady-state concepts. For example, the analysis given in Refs. 8-11 of the performance of a detonation engine concept includes neither frequency dependence nor analysis of losses due to multicycle operation. A new generation of analytical and computational tools exists today and allows us to revisit and analyze such issues with a high degree of confidence. Numerical simulation has developed to the state where it can now provide time-dependent two- and three-dimensional modeling of complex internal flow processes^{20,24,25} and will eventually result in tools for systematically analyzing and optimizing engineering design. In addition to a review of applications of the pulsed detonation engine concept (PDEC), we will report results of a numerical study of an air-breathing detonation engine. This study was performed using new unsteady computational fluid dynamics (CFD) tools that we will also describe.

Our paper is structured as follows: 1) historical review of the pulsed detonation development efforts; 2) description of the basic phenomenology of the air-breathing pulsed detonation engine concept; 3) description of the mathematical formulation and new numerical scheme used to simulate the problem; 4) discussion of the simulation results; and 5) conclusions.

Historical Review

Constant-Volume Combustion

From the very early development of jet-propulsion engines, it was known that an engine based on a constant-volume combustion process achieves higher thermodynamics efficiency than a constant pressure engine. This follows from a thermodynamic analysis of the engine cycle.¹

Constant-volume combustion was used in gas turbine engines at the beginning of this century, and the first gas turbine engines in commercial use were based on the constant-volume cycle. Jet-propulsion engines were one of the applications of the constant volume cycle (or explosion cycle) which was explored in the late 1940s.² Although the explosion cycle operates at a larger pressure variation in the combustion chamber than in a pulse jet,^{3,4} the cycle actually realized in these engines was not a fully constant-volume one since the combustion chamber was open-ended.² In Ref. 2, the maximum pressure ratio measured in an explosion cycle engine was 3:1, whereas the pressure ratio for the same mixture under the assumption of a constant-volume cycle would be 8:1. Also, this engine was limited by the available frequency of cycles, which in turn was limited by the reaction rate. A simple calculation² showed that if the combustion time could be reduced in this engine from 0.006-0.003 s, the thrust per pound of mixture would increase 100%. Thus, the explosion-cycle engine has two main disadvantages:

1) Constrained volume combustion (as distinguished from constant-volume combustion) does not take full advantage of the pressure rise characteristic of the constant-volume combustion process.

2) The frequency of the explosion cycle is limited by the reaction rate, which is only slightly higher than the deflagrative combustion rate.

The main advantage of the constant-pressure cycle is that it leads to engine configurations with the steady-state processes of injection of the fuel and oxidizer, combustion of the mixture, and expansion of the combustion products. These stages can be easily identified and the engine designer can optimize them on the basis of relatively simple steady-state considerations.

At the same time, an engine based on constant-volume combustion will have an intermittent mode of operation, which may complicate its design and optimization. We are interested in the question of whether this complication is worth the potential gains in engine efficiency.

Pulsed Detonation Engine as an Ultimate Constant-Volume Combustion Concept

The detonation process, due to the very high rate of reaction, permits construction of a propulsion engine in which the

Presented as Paper 89-2446 at the AIAA/ASME/ASCE/SAE 25th Joint Propulsion Conference, Monterey, CA, July 10-12, 1989; received Sept. 8, 1989; revision received April 12, 1990. This paper is declared a work of the U.S. Government and is not subject to copyright protection in the United States.

*Senior Scientist, Applied Physics Operation, Member AIAA.

†Chief Scientist, Applied Physics Operation, Member AIAA.

constant-volume process can be fully realized. In detonative combustion, the strong shock wave, which is part of the detonation wave, acts like a valve between the detonation products and the fresh charge. The speed of the detonation wave is about two orders of magnitude higher than the speed of a typical deflagration. This allows the design of propulsion engines with a very high power density. Usually, each detonation is initiated separately by a fully controlled ignition device, and the cycle frequency can be changed over a wide range of values. There is only an upper limit for the detonation cycle frequency. This limit is determined by the time it takes to refill the detonation chamber with the fresh combustible mixture. This in turn will depend on chamber geometry and the external flow parameters. In our study, we have established that detonation frequencies of 200–250 Hz appear to be feasible. At the same time, the same PDEC engine can operate at very low detonation frequency with thrust almost linearly proportional to the frequency. This also means that a device based on a detonative combustion cycle can be scaled, and its operating parameters can be modified for a range of required output conditions. There have been numerous attempts to take advantage of detonative combustion for engine applications. In the following, we give a description of the most relevant past experimental and analytical studies of the detonation engine concept.

Hoffmann's Report

The first reported work on intermittent detonation is attributed to Hoffmann⁵ in 1940. He operated an intermittent detonation test stand with acetylene-oxygen and benzene-oxygen mixtures. The addition of water vapor was used to prevent the highly sensitive acetylene-oxygen mixture from premature detonation. Hoffmann⁵ indicated the importance of the spark plug location in reference to tube length and diffuser length. It was found that a continuous injection of the combustible mixture leads to only a narrow range of ignition frequencies that will produce an intermittent detonation cycle. These frequencies are governed by the time required for the mixture to reach the igniter, the time of transition from deflagration to detonation, and the time of expansion of the detonation products. Hoffmann attempted to find the optimum cycle frequency experimentally. It was discovered that detonation-tube firing occurred at lower frequencies than the spark-plug energizing frequencies, indicating that the injection flow rate and ignition were out of phase. World War II prevented further work by Hoffmann and co-workers.

Nicholls' Experiments

A substantial effort in intermittent detonation engine research was done by a group headed by Nicholls⁶⁻¹⁰ of the University of Michigan beginning in the early 1950s. The most relevant work concerns a set of experiments carried out in a

6-ft-long detonation tube.⁹ The schematics of the detonation-tube experiments test rig used by Nicholls and co-workers are shown in Fig. 1. The detonation tube was constructed from a 1-in.-i.d. stainless-steel tube. The fuel and oxidizer were injected under pressure from the left end of the tube and ignited at the 10-in. distance downstream. The tube was mounted on a pendulum platform that was suspended by support wires. Thrust for single detonations was measured by detecting tube (platform) movement relative to a stationary pointer. For multicycle detonations, thrust measurement was achieved by mounting the thrust end of the tube to the free end of the cantilever beam. In addition to direct thrust measurements, the temperature on the inner wall of the detonation tube was measured.

Fuel mixtures of hydrogen/oxygen, hydrogen/air, acetylene-oxygen, and acetylene-air were used. The gaseous oxidizer and fuel were continuously injected at the closed end wall of the detonation tube and three fixed flow rates were used. Under these conditions, the only parameters that could be varied were the fuel/oxidizer ratio and frequency of ignition. A maximum gross thrust of ≈ 3.2 lb was measured in hydrogen/air mixture at the frequency of ≈ 30 detonations/s. The most promising results were demonstrated for the hydrogen/air mixture, where a fuel specific impulse of $I_{sp} = 2100$ s was reached. The maximum frequency of detonations obtained in all experiments was 35 Hz. The temperature measurements on the inner wall showed that for the highest frequency of detonations the temperature did not exceed 800°F.

In their later work,⁸⁻¹⁰ the University of Michigan group concentrated on development of the rotating detonation wave rocket motor. No further work on the pulsed detonation cycle was pursued.

Krzycki's Experiments

In a setup somewhat similar to Nicholls', Krzycki¹¹ performed an experimental investigation of intermittent detonations with frequencies up to 60 cps. An attempt was also made to analyze the basic phenomena using unsteady gas dynamic theory. Krzycki's attempt to analyze the basic phenomena relied on wave diagrams to trace characteristics, assumptions of isentropic flow for detonation and expansion, and incompressible flow for mixture injection processes. The most convincing data from the experiments are the measurement of thrust for a range of initiation frequencies and mixture flow rates. Unfortunately, no direct pressure measurement in the device are reported so that only indirect evidence exists of the nature of the process observed.

The basic test stand used by Krzycki is very similar to that used by Nicholls et al.⁶ The length of the detonation tube and internal diameter were exactly the same as those in Nicholls' experiments. A propane/air mixture was continuously injected through reversed-flow diffuser for better mixing and

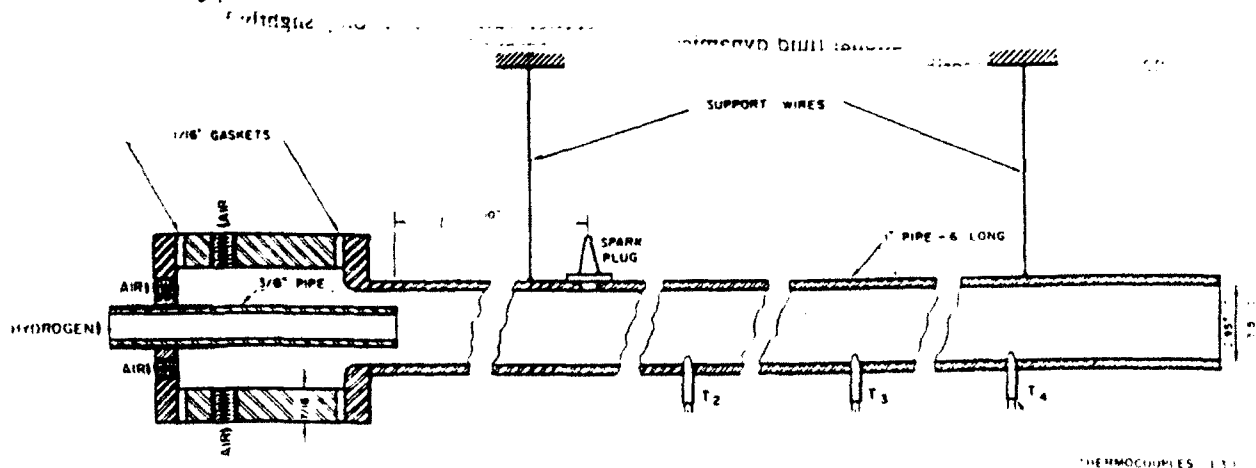


Fig. 1 Detonation tube used in experiments by Nicholls et al.

ignited at the 25-cm distance from the injection point by an automobile spark plug. The spark frequency was varied from 1-60 Hz. The spark plug power output was varied inversely with the initiation frequency and at the frequency of 60 Hz was only 0.65 J. This fact alone eliminated the possibility of direct initiation of the detonation wave by the spark and consequently all of the experiments were performed in the region dominated by transition from deflagration to detonation. According to experimental data and theory,¹² for direct initiation of a mixture of propane/air at the detonability limits, an energy release on the order of 10^6 J is required. Thus, the required deflagration-detonation transition region length would have been prohibitively large for the propane/air mixture. It follows that in all of the experiments a substantial part of the process was deflagrative. This resulted in low efficiency and negligible thrust. Krzycki repeated the experiments of Nicholls using exactly the same size detonation tube and basically the same rates of injection of the detonatable mixture. Krzycki's experimental results are very well-documented, giving enough information to deduce a clear picture of the physical processes occurring in the tube. A conclusion, arrived at by the author, was that thrust was possible from such a device but practical applications did not appear promising. It is unfortunate that, possibly based on Krzycki's extensive but misleading results, all experimental work related to the pulsed detonation engine concept stopped at this time.

Work Reported in Russian Sources on Pulse Detonation Devices

A review of the Russian literature has not uncovered work concerning applications of pulsed detonation devices to propulsion. However, there are numerous reports of applications of such devices for producing nitrogen oxide¹³ (an idea proposed in the 1940s by Zeldovich to use detonation for binding nitrogen directly from air to produce fertilizers) and as rock crushing devices.¹⁴

Korovin et al.¹³ provide a most interesting account of the operation of a commercial detonation reactor. The main objective of this study was to examine the efficiency of thermal oxidation of nitrogen in an intermittent detonative process as well as an assessment of such technological issues as the fatigue of the reactor parts exposed to the intermittent detonation waves over a prolonged time. The reactor consisted of a tube with an inner diameter of 16 mm and length 1.3 m joined by a conical diffuser to a second tube with an inner diameter of 70 mm and length 3 m. The entire detonation reactor was submerged in running water. The detonation mixture was introduced at the end wall of the small tube. Methane, oxygen, and nitrogen comprised the mixture composition and the mixture ratios were varied during the continuous operation of the reactor. The detonation wave velocity was measured directly by piezoelectric sensors placed in the small and large tubes. The detonation initiation frequency in the reactor was 2-16 Hz. It is reported that the apparatus operated without significant changes for 2000 h.

Smirnov and Boichenko¹⁴ studied intermittent detonations of a gasoline/air mixtures in a 3-m-long and 22-mm-i.d. tube operating in the 6-8 Hz ignition frequency range. The main motivation of this work was to improve the efficiency of a commercial rock-crushing apparatus based on intermittent detonations of the gasoline/air mixtures.¹⁵ The authors investigated the dependence of the length of the transitional region from deflagration to detonation on the initial temperature of the mixture.

As a result of the information contained in the Soviet reports, it can be concluded that reliable commercial devices based on intermittent detonations can be constructed and operated.

Development of the Blast Propulsion System at JPL

Back,¹⁶ Varsi et al.,¹⁷ Kim et al.,¹⁸ and Back et al.¹⁹ at the Jet Propulsion Laboratory (JPL) studied the feasibility of a rocket thruster powered by intermittent detonations of solid

explosive. The main application foreseen by the authors is propulsion in dense or high-pressure atmospheres of certain solar system planets. The JPL work was directed at very specific applications; however, the studies¹⁷⁻¹⁹ addressed some key issues of devices using unsteady processes such as propulsion efficiency. The JPL studies have important implications to pulsed detonation propulsion systems.

Reference 19 gives the basic description of the test stand used. In this work, a data sheet type C explosive was detonated inside a small detonation chamber attached to nozzles of various length and geometry. The nozzles, complete with firing plug, were mounted in a containment vessel that could be pressurized with the mixture of various inert gases from vacuum to 70 atm. The apparatus measured directly the thrust generated by single detonations of a small amount of solid explosive charge expanding into conical or straight nozzles. Thrust and specific impulse were measured by a pendulum balance system.

Results obtained from an extensive experimental study of the explosively driven rocket have led to the following conclusions. First, rockets with long nozzles show increasing specific impulse with increasing ambient pressure in carbon dioxide and nitrogen. Short nozzles, on the other hand, show that specific impulse is independent of ambient pressure. Most importantly, most of the experiments obtained a relatively high specific impulse of 250 s and larger. This result is all the more striking since the detonation of a solid explosive yields a relatively low energy release of approximately 1000 cal/g compared with 3000 cal/g obtained in hydrogen/oxygen combustion. Thus, it can be concluded that the total losses in a thruster based on unsteady expansion are not prohibitive and, in principle, very efficient propulsion systems operating on intermittent detonations are possible.

Detonation Engine Studies at Naval Postgraduate School

A modest exploratory study of a propulsion device utilizing detonation phenomena was conducted at the Naval Postgraduate School (NPS).²⁰⁻²³ During this study, several fundamentally new elements were introduced to the concept distinguishing the new device from previous ones.

First, it is important to note that the experimental apparatus constructed by Helman et al.²² showed the first successful self-aspirating, air-breathing detonation device. Intermittent detonation frequencies of 25 Hz were obtained. This frequency was in phase with the fuel-mixture injection through timed fuel-valve opening and spark discharge. The feasibility of intermittent injection was established. Pressure measurements showed conclusively that a detonation process occurred at the frequency chosen for fuel injection. Furthermore, self-aspiration was shown to be effective. Finally, the effectiveness of a primary detonation as a driver for the main detonation was clearly demonstrated. Although the NPS studies were abbreviated, many of the technical issues considered to be essential for efficient intermittent detonation propulsion were addressed with positive results.

Simulations of Pulsed Detonation Engine Cycle at NASA Ames Research Center

Recently Camblier and Adelman²⁴ carried out numerical simulations of a pulsed detonation engine cycle taking into account finite-rate chemistry. Unfortunately, the simulations were restricted to a quasi-one-dimensional model. The configuration considered had a 6-cm-i.d., 50-cm-long main chamber that was attached to a 43-cm-long diverging nozzle. It was assumed that a stoichiometric mixture of hydrogen/air at 3.0 atm is injected from an inlet on the closed end wall of the detonation chamber. Under these conditions, Camblier and Adelman estimated a large range of possible detonation frequencies of engine operation up to 667 Hz. The origin of this estimate is not clear from their work since, according to their simulations, the detonation, expansion, and fresh charge fill requires 2.5 ms. This value leads to a maximum frequency of

400 Hz. The simulated engine performance yielded a large average thrust of ≈ 4000 N and an unusually high specific impulse of 6507 s. These simulations were the first to demonstrate the use of modern CFD methods to address the technical issues associated with unsteady pulsed detonation concepts.

In the remaining sections, we discuss a particular propulsion concept based on the results of the experiments of Helman et al.²² and describe a computational study of its performance characteristics. The unsteady numerical scheme used for the study made use of unique simulation techniques; the key ingredients of these techniques are also described.

Generic Pulsed Detonation Engine

The generic device we consider here is a small cylindrical engine, 15 cm long and 15 cm in diameter. The combustible gas mixture is injected at the closed end of the detonation chamber and a detonation wave propagates through the mixture. The size of the engine suggests a small payload, but the concept can be extended to larger payloads simply by scaling up the size of the detonation chamber and possibly combining a number of engines into one large propulsion engine. A key issue in the pulsed detonation engine concept is the design of the main detonation chamber. The detonation chamber geometry determines the propulsion efficiency and the duration of the cycle (frequency of detonations). Since the fresh charge for the generic engine is supplied from the external flowfield, the efficiency of the engine depends on the interaction of the surrounding flow with the internal flow dynamics. The range of the physical processes requiring simulation in order to model the complex flow phenomena associated with the detonation engine performance is very broad. A partial list is as follows:

- 1) Initiation and propagation of the detonation wave inside the chamber.
- 2) Expansion of the detonation products from the chamber into the airstream around the chamber at flight Mach numbers.
- 3) Reverse flow from the surrounding air into chamber resulting from overexpansion of the detonation products.
- 4) Pressure buildup in the chamber due to reverse flow. The flow pattern inside the chamber during postexhaust pressure buildup determines the strategy for mixing the next detonation charge.
- 5) Strong mutual interaction between the flow processes inside the chamber and flow around the engine.

All of these processes are interdependent and their timing is crucial to the engine efficiency. Thus, unlike simulations of steady-state engines, the phenomena described above cannot be evaluated independently.

The need to resolve the flow inside the chamber accounting for nozzles, air inlets, etc., and at the same time resolve the flow around the engine, where the flow regime varies from high subsonic, locally transonic, and supersonic, makes it a challenging computational problem.

The main issue is to determine the timing of the air intake for the fresh gas charge. It is sufficient to assume inviscid flow for the purpose of simulating the expansion of the detonation products and fresh gas intake. In the following, we present the first results of an inviscid simulation of the detonation cycle in a cylindrical chamber. First, we describe our computational method for solving the time-dependent Euler equations used in the study.

Unsteady Euler Solver

A new second-order algorithm for solving the Euler equations on an unstructured grid was used in our study of the detonation concept. The approach is based on first- and second-order Godunov methods. The method leads to an extremely efficient and fast flow solver that is fully vectorized and easily lends itself to parallelization. The low memory requirements and speed of the method are due to the use of a unique data structure.

Until recently most CFD simulations were carried out with logically structured grids. Vectorization and/or parallelization did not present a problem. The increased need for simulation of flow phenomena in the vicinity of complex geometrical bodies and surfaces has led to the development of CFD codes for logically unstructured grids. The most successful of these unstructured grid codes are based on finite elements or finite volume methods. For an unstructured grid in two dimensions, the computational domain is usually covered by triangles, and the indices of the arrays containing the values of the hydrodynamic flow quantities are not related directly to the actual geometric location of a node. The calculations performed on unstructured grids evolve around the elemental grid shape (e.g., the triangle for two-dimensional problems), and there is no obvious pattern to the order in which the local integrations should be performed. Explicit integration of hydrodynamic problems on an unstructured grid requires that a logical substructure should be created which identifies the locations in the global arrays of all of the local quantities necessary for the integration of one element. This usually results in a large price in computational efficiency, in memory requirements, and in code complexity. As a consequence, vectorization for the conventional unstructured grid methods has concentrated on rearrangement of the data structure in a manner such that these locally centered data structures appear as global arrays. This can be done to some extent using machine dependent gather-scatter operations.^{25,26} Additional optimization can be achieved using localization and search algorithms. However, these methods are complex and result in marginal improvement. Most optimized unstructured codes to date run considerably slower and require an order of magnitude more memory per grid cell than their structured counterparts. Parallelization of the conventional unstructured codes is even more difficult, and there is very little experience with unstructured codes on massively parallel computers.

The method we have developed overcomes these difficulties and results in codes with speed and memory requirements comparable to those found in structured grid codes. Moreover, the ability to construct grids with arbitrary resolution leads to a flexibility in dealing with complex geometries not attainable with structured grids. The essence of the method is based on an independent flux calculation across the edges of a dual baricentric grid, followed by node integration. This approach is order independent. Below we give the essential details of our algorithm; a complete description follows later.

Basic Integration Algorithm

We begin by describing the first-order Godunov method for the system of two-dimensional (axisymmetric) Euler equations written in conservation law form as

$$\frac{\partial Q}{\partial t} + \frac{\partial F}{\partial x} + \frac{\partial G}{\partial r} = -\frac{1}{r} C \quad (1)$$

where

$$Q = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ e \end{bmatrix}, \quad F = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ (e + p)u \end{bmatrix}$$

$$G = \begin{bmatrix} \rho v \\ \rho vu \\ \rho v^2 + p \\ (e + p)v \end{bmatrix}, \quad C = \begin{bmatrix} \rho v \\ \rho vu \\ \rho v^2 \\ (e + p)v \end{bmatrix}$$

Here u and v are the x and r velocity vector components, p the pressure, ρ the density, and e the total energy of the fluid per unit volume. It is assumed that a mixed (initial conditions, boundary conditions) problem is properly posed for the set,

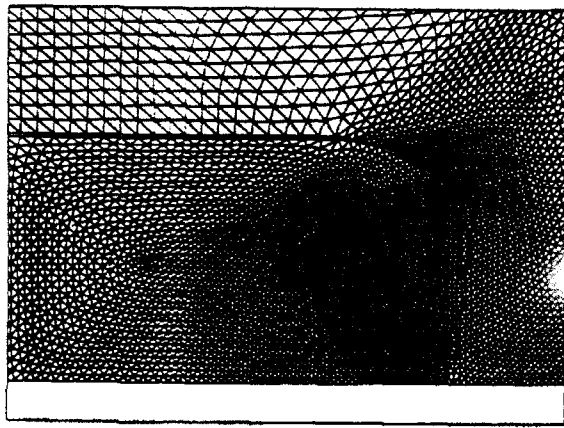


Fig. 2a Computational domain and grid used in simulation of PDEC operation.

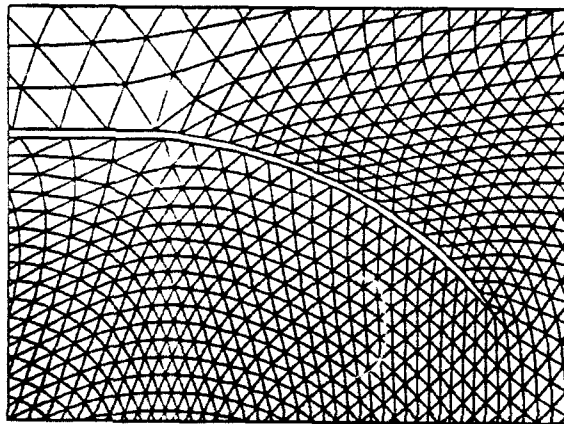


Fig. 2b Enlargement of computational grid in the vicinity of the PDEC nozzle.

Eq. (1), and that an initial distribution of the fluid parameters is given at $t = 0$ and some boundary conditions defining a unique solution are specified on the boundary of the computational domain.

We look for a solution of the system of equations represented by Eq. (2) in the computational domain covered by an unstructured grid. As an example, Fig. 2a shows the unstructured triangular grid used in the pulsed detonation engine simulation. Here most of the computational effort is committed to the resolution of the flow inside the engine detonation chamber and in the immediate vicinity of the nozzle. In Fig. 2b, an enlargement of the nozzle region is shown, illustrating the ability to represent geometry of arbitrary complexity and with localized resolution.

Figure 3 displays a fragment of the computational domain with the corresponding dual grid. The secondary or dual grid is formed by connecting the baricenters of the primary mesh, thus forming finite polygons around the primary vertices.

We have found, as have others,²⁷ that the best practical representation of the integration volume is obtained when the dual grid is formed by connecting baricenters of the triangles. Integration by the Godunov method²⁸ can be divided into two basic steps: 1) calculation of the fluxes at the edges of the secondary grid using solutions of a set of one-dimensional Riemann problems; and 2) integration of the system of partial differential equations, which amounts to addition of all of the fluxes for every polygon at a particular time step.

To define the fluxes for the grid shown in Fig. 3 at every edge of the main grid, it is necessary to solve the corresponding Riemann problem. For example, to define the flux at the edge ab , we solve the Riemann problem between points A and B . The solution of this problem is in coordinates local to the

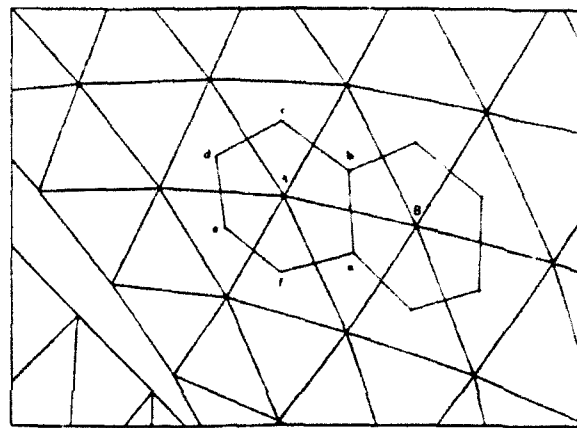


Fig. 3 The primary (triangles) and secondary (polygons) unstructured grids.

edge of the dual grid ab so that the tangential component of velocity will be directed along this edge (ab). Implementation of our approach requires maintaining strict consistency when defining the "left" and "right" states for the Riemann problems at the edges ab , bc , cd , de , ef , and fa . For this reason, we define not only the location of the vertices and lengths of the edges but also the direction of the edges with respect to the primary grid. For the clockwise integration pattern in the same polygon, point A will be the "right" state for all of the Riemann problems related to this point, and the neighbor will represent the "left" side of the diaphragm.

It is easy to see that the flux calculation is based on information at only two nodes and requires single geometrical parameters defining the edge of the secondary grid that dissects the line connecting the two points. Thus, we can calculate all of the values needed for flux calculation in one loop over all edges of the primary grid without any details related to the geometrical structures that these edges form. This in turn assures parallelization or vectorization of the algorithm for the bulk of the calculations involving the Riemann solver that provides the first-order flux. The only procedure not readily parallelizable is the integration of the fluxes for the flow variables at the vertices of the grid. Here we use the "edge coloring" technique that allows us to split the flux addition loop into seven or eight loops for edges of different color. Each of these loops is usually large enough not to impair vectorization. At this stage, all of the fluxes are added with their correct sign corresponding to the chosen direction of integration within the cell. The amount of calculation required here is minimal since the fluxes are known and need only to be multiplied at each time step by a simple factor and added to the vertex quantity.

Second-Order Integration Algorithm

The second-order solver is constructed along lines similar to that of the first-order method. At each cell edge, the Riemann problem is solved for some specified pair of left and right conditions. The solution to this Riemann problem is then used in the calculation of fluxes that are added later to advance to the next integration step. The extension to second-order is achieved by using extrapolation in space and time to obtain time-centered left and right-limiting values as inputs for the Riemann problem. The basic implementation of the method of calculation of second-order accurate fluxes is fundamentally the same as for one-dimensional cases. The only difference is in the method of obtaining linear extrapolation of the flow variables as a first guess of their value at the edges of the dual grid. To obtain the first guess, we need to know the gradient of some gasdynamical parameter U at the vertices of the primary mesh. The value of ∇U can be evaluated by using

a linear path integral along the edges, which delineates the finite volume associated with the vertex. For vertex *A* in Fig. 3,

$$\int_A \nabla U dA = \oint U n dl \quad (2)$$

where integration along the path *l* in this case is equivalent to integration along the edges *ab*, *bc*, *cd*, *de*, *ef*, and *fa*. Knowing the gradient of the gasdynamic parameter in the volume related to vertex *A* will allow us to extrapolate the values of this parameter at any location within the volume. This permits us to evaluate the first guess for *U* at the edges of the dual grid. The final implementation of the second-order algorithm has been described previously.

A schematic flowchart of the basic steps of the second-order algorithm implementation is shown in Fig. 4.

Simulations of the Generic Pulsed Detonation Engine

In this section, we present sample results of simulations of the generic PDE device using the numerical code described in the preceding section. In Fig. 2a, the computational domain containing the PDE main detonation chamber is shown covered with the unstructured grid. In our sample simulation, we have chosen a small ≈ 15 -cm-long and ≈ 15 -cm-i.d. cylindrical chamber with a small converging nozzle. This geometry is one of a number of the geometries we have analyzed in a parametric study whose goal was to evaluate and optimize a typical PDEC device. The device shown in Fig. 1a does not represent the optimum and is given here to illustrate our methodology. We consider a situation when the PDEC serves as a main thruster for a vehicle traveling in air with the velocity of $M = 0.9$ and located at the aft end of the vehicle. The main objectives of the simulations presented here are as follows:

- 1) To find the maximum cycle frequency. This is determined by the time required from detonation, exhaust of combustion products, and intake of fresh charge for the next detonation.
- 2) To calculate the thrust produced during each cycle and the integrated thrust as a function of time.

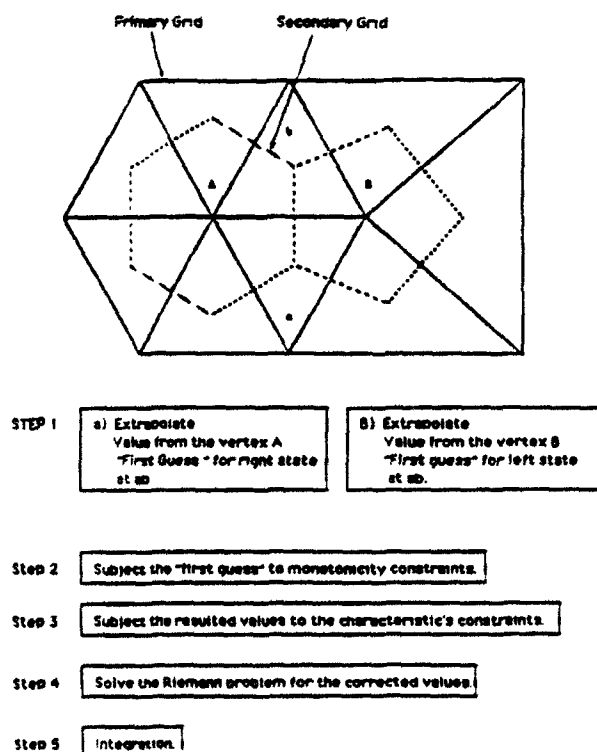


Fig. 4 Grid schematic and outline of steps for second-order Godunov method.

The simulation begins at $t = 0$ when we assume an ideal detonation process has taken place in a stoichiometric propane/air mixture. Initially the detonation wave has traveled from the open aft end of the chamber toward the interior with a maximum velocity of 1800 m/s and maximum pressure of 20×10^5 Pa. The distribution of pressure, velocity, and density of the detonation wave is defined through the self-similar solution for a planar detonation wave. These distributions are shown schematically in Fig. 5. The wave was directed toward the interior of the chamber to capture the kinetic energy of the wave and to prolong exposure of the inner chamber walls to the high pressure. In Fig. 6, simulation results are shown at time $t = 0.19$ ms in the form of pressure contours and particle paths from different locations inside and outside the detonation chamber. From the pressure contour plots, we observe that the shock reflection from the inner wall has taken place and detonation products are expanding into the ambient airstream. The flow inside the chamber is choked due to the converging nozzle and the maximum pressure behind the shock is ≈ 8 atm. The pressure inside the chamber is less than 3 atm. The strong expansion of the detonation products into the ambient airstream produces a shock wave with a spherical-like front rapidly decaying in strength. As a result of the interaction of the expanding detonation products with the external flow, a large toroidal vortex is created. The vortex is carried away quickly from the chamber by the external flow and by its own flow momentum.

In Fig. 6a, we also show trajectories of the particles introduced inside the chamber and just above the nozzle. Examination of these trajectories allows us to follow the dynamics of the chamber evacuation and refill. In order to track the detonation products, we initially place marker particles inside the chamber at three cross sections in clusters of four distributed normally to the detonation chamber axis. Each particle has a different color; however, particles in the same cluster have the same shade of color. At the three chosen cross sections, we have designated shades of red, yellow, and green for the particles located correspondingly at the left end, center, and beginning of the nozzle cross sections of the chamber. The movement of these particles is shown by connecting them with a continuous line beginning with particle location at $t = 0$ to the present time. In Fig. 6a, we observe that at time $t = 0.19$ ms all particles originally in the nozzle cross section and three of the particles originally in the midsection have left the detonation chamber. However, particles originally introduced on the inner wall of the chamber have only advanced to the nozzle region.

We use a different technique for observing the motion of the ambient gas outside the chamber. Here a cluster of seven

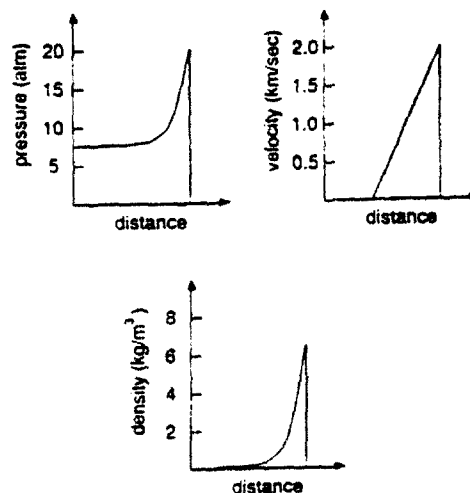


Fig. 5 Distribution of gasdynamic parameters behind the detonation wave according to a one-dimensional self-similar solution.

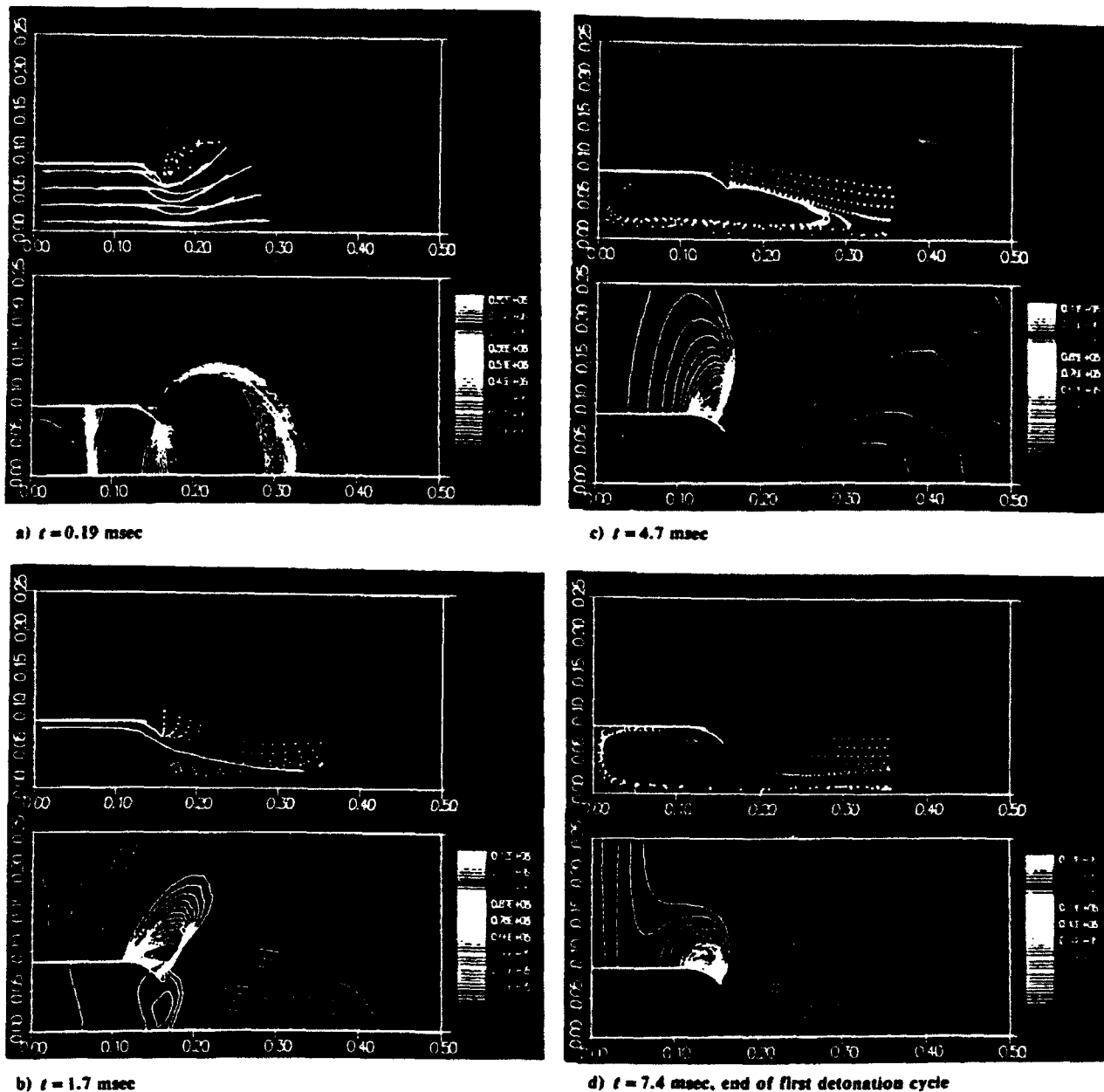


Fig. 6 Pressure contours and particle paths for various times during the PDEC simulation.

particles is introduced every $0.05 \mu\text{s}$ in the external flow above the nozzle. All such particles are traced as they move with the flow until they leave the computational domain. At any given time only the current location of the particle is displayed, and since the particles are introduced periodically with time there are a large number of particles to trace. We assign a color to every cluster of external particles to keep track of the time when they were introduced in the calculation. The colors vary from magenta, for those particles introduced early in calculation, to blue, for those introduced shortly before the end of a detonation cycle. In Fig. 6a, corresponding to very early times, only one cluster of external particles is visible. This cluster was introduced at $t = 0$ and is tracking the expanding flow of the detonation products.

In Fig. 6b, the simulation results are shown for $t = 1.7 \text{ ms}$. The pressure contours show that a shock wave develops at the external edge of the nozzle as a result of a strong expansion of the Mach 0.9 external flow. As a result of overexpansion of the detonation products, the pressure inside the detonation chamber is lower than the ambient pressure, causing the shock

to be located lower on the external surface of the nozzle. The external flow about the chamber has a stagnation point on the axis of symmetry downstream at $\approx 25 \text{ cm}$. At this time, it is evident from the particles' trajectories that most of the detonation products have left the chamber. Figure 6b shows one continuous trace of the particles originating at the back wall of the detonation chamber having advanced well ahead of the stagnation point in the external flow.

The marker particles released outside and just above the nozzle's exit show two distinct flow paths. One path takes the flow past the stagnation point to the right of the detonation chamber; this flow path is marked by the four upper particle traces. Another flow path is marked by three lower particle paths released close to the nozzle surface and is deflected toward the detonation chamber exit. Figure 5b shows particles marking this deflected stream approaching the detonation chamber nozzle. The magenta color of these particles indicates they were released at $\approx 0.5 \text{ ms}$.

Figure 6c corresponds to the simulation time $t = 4.7 \text{ ms}$. The pressure inside the chamber has risen $\approx 1 \text{ atm}$. Higher

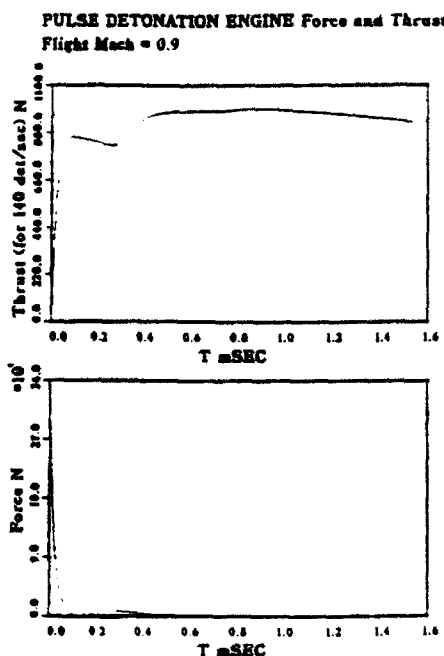


Fig. 7 Thrust and force generated by PDEC as a function of time.

pressure at the chamber exit has resulted in the shock standing on the external surface of the nozzle to move upward. The particles marking the movement of fresh air into the chamber show these to be well inside with some reflecting from the end wall giving a second stagnation point for the reversed fresh airflow.

Figure 6d corresponds to the end of the first cycle when the detonation chamber is filled with fresh charge and ready for the next detonation. In this figure, the particles' paths indicate that the chamber refills in a pattern suitable for fast mixing of the fuel-air mixture. We conjecture then that fuel injection along the chamber axis will promote fast fuel-air mixing. We can see in Fig. 6d that further injection of external air inside the chamber stopped, and from that point on the mixture composition in the chamber will be fixed.

In Fig. 7, the total force and time-averaged thrust generated by the device in the simulations discussed previously are shown as a function of time. The time-averaged thrust is based on the total time for one cycle. As seen in Fig. 7, initially a very large force of $\approx 1.5 \times 10^5$ kg is felt on the end wall of the detonation chamber. This is a result of the inwardly moving detonation wave used in our simulation. Very early during the sequence, this wave reflects from the left wall of the detonation chamber briefly generating a large force. This force rapidly decays and at $t \approx 0.1$ ms changes sign due to interaction of the strong shock wave with the converging nozzle. This effect is noticeable in the thrust data; the average thrust decreases somewhat after reaching levels of ≈ 1980 N. The shock partially reflects from the converging nozzle walls and generates a wave moving to the left wall. The reflected wave thereafter generates positive thrust from $t \approx 0.3$ ms. Finally, thrust levels reach the maximum of ≈ 2200 N and then decay slowly as a result of the cross-sectional drag force. The simulations predict that to sustain this level of thrust will require a detonation frequency of about 150 Hz. All simulations were performed on a Stellar workstation.

Conclusions

The main intent of the present study was to carry out a review of the relevant literature in the area of detonation propulsion, to assess the state of the art, and to recommend future research based on our findings. We have reviewed the literature and presented our summary in the first section of this paper. Our initial conclusion from the review is that there

is a substantial body of evidence leading toward the possibility of producing propulsion engines with significant thrust levels based on an intermittent detonation.

Most of the historical attempts at producing thrust based on the intermittent detonation cycle were carried out with the same basic experimental setup; namely, a long straight detonation tube employing forced fuel injection at the closed tube end. We have discussed the many reasons why such a device cannot take proper advantage of the physical processes associated with detonation.

The experiments performed at the Naval Postgraduate School using a self-aspirating mode of operation for a pulsed detonation thruster produced very useful results which, upon further examination, provide us with a route toward practical propulsion engines of variable thrust levels that are both controllable and scalable.

We have explored some of the implications of the possible applications of the self-aspirating detonation engine concept and have developed a suitable numerical simulation code to be used as a design, analysis, and evaluation tool. In fact, the preliminary analysis of a candidate detonation chamber flow was shown to be dominated completely by unsteady gasdynamics. An attempt to understand the flow properties based on any steady-state model or one-dimensional unsteady analytical model will miss such important aspects as fuel-air mixing and shock reflection from internal geometrical obstacle such as the converging nozzle. The unsteady simulation code developed during the course of our study is a necessary tool that we plan to use in a study leading to a feasible prototype engine design realizing the full potential of the intermittent detonation process.

Acknowledgments

The work reported here was supported by the Defense Advanced Research Projects Agency under Contract N66001-88-D-0088. The authors express appreciation to Adam Drobot and Aharon Friedman for helpful suggestions and advice.

References

- Stodola, A., *Steam and Gas Turbines*, McGraw-Hill, New York, 1927.
- Zipkin, M. A., and Lewis, G. W., "Analytical and Experimental Performance of an Explosion-Cycle Combustion Chamber of a Jet Propulsion Engine," NACA TN-1702, Sept. 1948.
- Shultz-Grunow, F., "Gas-Dynamic Investigation of the Pulse-Jet Tube," NACA TM-1131, Feb. 1947.
- Zinn, B. T., Miller, N., Carvalho, J. A., and Daniel, B. R., "Pulsating Combustion of Coal in Rijke Type Combustor," *19th International Symposium on Combustion*, Combustion Inst., Pittsburgh, PA, 1982, pp. 1197-1203.
- Hoffmann, N., "Reaction Propulsion by Intermittent Detonative Combustion," Ministry of Supply, Volkenrode Translation, 1940.
- Nicholls, J. A., Wilkinson, H. R., and Morrison, R. B., "Intermittent Detonation as a Thrust-Producing Mechanism," *Jet Propulsion*, Vol. 27, 1957, pp. 534-541.
- Dunlap, R., Brehm, R. L., and Nicholls, J. A., "A Preliminary Study of the Application of Steady State Detonative Combustion of a Reaction Engine," *Jet Propulsion*, Vol. 28, 1958, pp. 451-456.
- Nicholls, J. A., Gullen, R. E., and Ragland, K. W., "Feasibility Studies of a Rotating Detonation Wave Rocket Motor," *Journal of Spacecraft and Rockets*, Vol. 3, 1966, pp. 893-898.
- Adamson, T. C., and Olsson, G. R., "Performance Analysis of a Rotating Detonation Wave Rocket Engine," *Astronautica Acta*, Vol. 13, 1967, pp. 405-415.
- Shen, P. I., and Adamson, T. C., "Theoretical Analysis of a Rotating Two-Phase Detonation in Liquid Rocket Motors," *Astronautica Acta*, Vol. 17, 1972, pp. 715-728.
- Krzycki, L. J., "Performance Characteristics of an Intermittent Detonation Device," U. S. Naval Ordnance Test Station, China Lake, CA., Navweps Rept. 7655, 1962.
- Matsui, H., and Lee, J. H., "On the Measure of the Relative Detonation Hazards of Gaseous Fuel-Oxygen and Air Mixtures," *Seventeenth Symposium (International) on Combustion*, Combustion Inst., Pittsburgh, PA, 1978, pp. 1269-1280.
- Korovin, L. N., Losev, A., Ruban, S. G., and Smekhov, G. D.,

"Combustion of Natural Gas in a Commercial Detonation Reactor," *Fizika Gor. Vzryva*, Vol. 17, No. 3, 1981, p. 86.

¹⁴Smirnov, N. N., and Boichenko, A. P., "Transition from Deflagration to Detonation in Gasoline-Air Mixtures," *Fizika Gor. Vzryva*, Vol. 22, No. 2, 1986, pp. 65-67.

¹⁵Lobanov, D. P., Fonbershtein, E. G., and Ekomasov, S. P., "Detonation of Gasoline-Air Mixtures in Small Diameter Tubes," *Fizika Gor. Vzryva*, Vol. 12, No. 3, 1976, p. 446.

¹⁶Back, L. H., "Application of Blast Wave Theory to Explosive Propulsion," *Acta Astronautica*, Vol. 2, No. 5/6, 1975, pp. 391-407.

¹⁷Varsi, G., Back, L. H., and Kim, K., "Blast Wave in a Nozzle for Propulsion Applications," *Acta Astronautica*, Vol. 3, 1976, pp. 141-156.

¹⁸Kim, K., Varsi, G., and Back, L. H., "Blast Wave Analysis for Detonation Propulsion," *AIAA Journal*, Vol. 10, No. 10, 1977, pp. 1500-1502.

¹⁹Back, L. H., Dowler, W. L., and Varsi, G., "Detonation Propulsion Experiments and Theory," *AIAA Journal*, Vol. 21, No. 10, 1983, pp. 1418-1427.

²⁰Eidelman, S., and Shreeve, R. P., "Numerical Modeling of the Nonsteady Thrust Produced by Intermittent Pressure Rise in a Diverging Channel," American Society of Mechanical Engineers, *Multidimensional Fluid Transient*, FED-Vol. 18, 1984, p. 77.

²¹Eidelman, S., "Rotary Detonation Engine," U.S. Patent 4 741 154, 1988.

²²Helman, D., Shreeve, R. P., and Eidelman, S., "Detonation Pulse Engine," AIAA Paper 86-1683, June 1986.

²³Monks, S. A., "Preliminary Assessment of a Rotary Detonation Engine Concept," M.Sc. Thesis, Naval Postgraduate School, Monterey, CA, Sept. 1983.

²⁴Camblier, T. L., and Adelman, N. G., "Preliminary Numerical Simulations of a Pulsed Detonation Wave Engine," AIAA Paper 88-2960, Aug. 1988.

²⁵Lohner, R., Morgan, K., and Zienkiewicz, D. C., "Finite-Element Methods for High Speed Flows," AIAA Paper 85-1531, July

1985.

²⁶Lohner, R., and Morgan, K., "Improved Adaptive Refinement Strategies for Finite-Element Aerodynamic Computations," AIAA Paper 86-0499, Jan. 1986.

²⁷Barth, T. J., and Jespersen, D. C., "The Design and Application of Upwind Schemes on Unstructured Meshes," AIAA Paper 89-0366, Jan. 1989.

²⁸Eidelman, S., Collela, P., and Shreeve, R. P., "Application of the Godunov Method and Its Second-Order Extension to Cascade Flow Modelling," *AIAA Journal*, Vol. 22, No. 1, 1984, p. 10.

²⁹Baker, T. J., "Developments and Trends in Three-Dimensional Mesh Generations," Transonic Symposium, NASA Langley Research Center, Hampton, VA, 1988.

³⁰Jameson, A., Baker, T. J., and Weatherill, N. P., "Calculation of Inviscid Transonic Flow Over a Complete Aircraft," AIAA Paper 86-0103, Jan. 1986.

³¹Greengard, L., and Rokhlin, V., "A Fast Algorithm for Particle Simulations," *Journal of Computational Physics*, Vol. 73, 1987, pp. 325-348.

³²Eidelman, S., Collela, P., and Shreeve, R. P., "Application of the Godunov Method and Its Second Order Extension to Cascade Flow Modeling," *AIAA Journal*, Vol. 22, No. 1, 1984, p. 10.

³³van Leer, B., "Towards the Ultimate Conservative Difference Scheme. V. A Second Order Sequel to Godunov's Method," *Journal of Computational Physics*, Vol. 32, 1979, pp. 101-136.

³⁴Collela, P., and Woodward, P., "The Piecewise Parabolic Method (PPM) for Gasdynamical Simulations," *Journal of Computational Physics*, Vol. 54, 1984, pp. 174-201.

³⁵Barth, T. J., and Jespersen, D. C., "The Design and Application of Upwind Schemes on Unstructured Meshes," AIAA Paper 89-0366, Jan. 1989.

³⁶Glaz, H. M., Collela, P., Glass, I. I., and Deschambault, R. L., "A Detailed Numerical, Graphical, and Experimental Study of Oblique Shock Wave Reflections," Defense Nuclear Agency, DNA-TR-86-365, 1986.

Numerical and analytical study of transverse supersonic flow over a flat cone

D.L. Book¹, S. Eidelman², I. Lottati² and X. Yang²

¹ Code 4405, Naval Research Laboratory, Washington, DC 20375

² Science Applications International, McLean, VA 22102

Received February 8, 1991; accepted March 21, 1991

Abstract. Quasisteady supersonic flow over a flat cone on a plane surface is studied. A formula is derived for the angle through which the flow lines turn at the cone. The results are used to justify the use of two-dimensional simulations of the flow. Peak pressures and total impulses are obtained numerically for various cone angles.

Key words: Cone, Euler equation, Mach reflection, CFD, Supersonic flow

1. Introduction

The purpose of this study is to determine the maximum pressure on the surface of a flat cone (one for which the height is much less than the diameter) in the quasi-uniform flow behind a strong blast wave propagating at right angles to the axis of the cone. If the cone is small compared with the radius of the blast wave, the undisturbed flow is approximately rectilinear. First, the blast wave passes over the cone and an unsteady load builds up on the surface. In general the shock will undergo Mach reflection over at least part of the surface of the cone, the extent depending on the cone angle α , the adiabatic index γ , and the Mach number M . After a short transitional stage the cone will then be subject to the quasisteady supersonic flow field behind the blast front. (For a strong blast wave in air, the pressure drops to one-half the peak value about one-tenth of the way back from the front toward the origin of the blast (Sedov 1959). Thus, if the blast center is located ~ 100 radii from the cone, the pressure arriving at the cone is reduced to half its initial value ~ 10 radii behind the front.) The post-shock flow velocity varies on the same scale. We would like to find whether the pressure on the cone reaches its maximum during

the quasisteady or the unsteady regime of the flow and determine its magnitude.

The cone, shown schematically in Fig. 1, is located on a plane surface. We take its axis to be normal to the surface, and we model the front of the spherical blast wave as a planar shock wave propagating normally to this axis. This is a reasonable approximation when the distance to the blast center is much larger than the radius, i.e., in the same limit for which we can assume that the state behind the front is uniform. The flow over the cone is substantially three-dimensional. The only symmetry is with respect to inversion about the midplane (the plane through the cone axis and parallel to the flow). In the general case in which the cone axis is not normal to the plane, the problem is totally asymmetrical.

Previous studies of the effect of supersonic flows on conical bodies have focused primarily on situations in which the flow is parallel or nearly parallel to the axis of the cone. Those results are applicable to, e.g., the aerodynamical effects associated with the nose cones of re-entry vehicles. In contrast, the problem we are considering may be regarded as an idealized model of the interaction between a blast wave and a ground or shipboard structure. It can also model the flow over a bump or a housing on the skin of a supersonic aircraft or missile. The results may thus be relevant to both damage studies and flight characteristics.

A number of experimental studies related to the problem of oblique supersonic flow over a cone have been carried out, beginning at least three decades ago (Tracy 1963; Damkevala and Zumwalt 1968). Most of this work has dealt with small deviations from axisymmetry, although angles of attack as large as 30° have been studied (Yahalom 1971). Less experimental effort seems to have been devoted to transverse flows (angle of attack equal to 90°). Likewise, theoretical studies by Goman and Davydov (1975) and Gusarov et al. (1979) have concentrated on small deviations from conical shapes in axisymmetric flow. Numerical simulations have been car-

ried out at large angles of attack (30° – 50°) by Fletcher and Holt (1976). These results are useful, and the same techniques can be applied to transverse flows, but they have definite limitations. At the Mach numbers investigated ($M \lesssim 6$ – 8) the flow is strongly conditioned by the presence of a viscous boundary layer. This necessitates solution of the Navier-Stokes equations instead of the Euler equations, and it may be necessary to incorporate a turbulence model as well. In three dimensions it is difficult to obtain good resolution even for inviscid flow; the presence of a thin boundary layer makes the problem even more formidable.

In the next section we determine the streamlines associated with transverse flow over a cone in the Newtonian approximation, i.e., assuming that the streamlines follow the contours of the body surface. We show that for a flat cone the streamlines deviate very little from the vertical plane in which they were propagating before reaching the cone. We use this result to argue that the flow over the cone can be accurately modeled by treating each cross section made by a vertical plane separately, i.e., by solving a series of two-dimensional problems. In the section following that, we describe the results of such calculations. For this purpose we use an Euler code, which is only valid at low flow Mach numbers ($M \lesssim 5$). In our calculations the shock Mach number equals 25, but the Mach number of the flow in the heated region behind the shock is ~ 3 , so we are justified in ignoring viscous effects. At higher values of M our results are at least indicative and can be expected to yield accurate values of the peak pressures on the cone. (Of course the reduction of the problem to two dimensions is a consequence of the cone geometry and would be equally useful for Navier-Stokes applications.) We show that these results can be combined to draw a picture of most of the flow field. In the final section we summarize our conclusions.

2. Streamline trajectories

If the flow deflected by a solid object remains supersonic after deflection, the angle between the flow direction and the surface determines the flow parameters behind the shock for given inlet flow parameters. For shock Mach numbers $M \geq 10$, the shock angle is small and the deflected flow on the upwind side closely follows the form of the deflecting object. The streamlines are determined by the condition that the angle through which they are deflected be as small as possible. We would like to analyze how this deflection angle varies on the surface of the cone shown in Fig. 1. Based on this analysis we can estimate most of the characteristics of steady supersonic flow directed transversely toward the cone.

The equation for the frustrum of a cone with the geometrical center of its base located at the center of coordinates (see Fig. 1) is

$$(x^2 + y^2) \tan^2 \alpha = (z - h)^2, \quad (1)$$

where h is the height of the cone and α is the angle between the side of the cone and the base. The angle

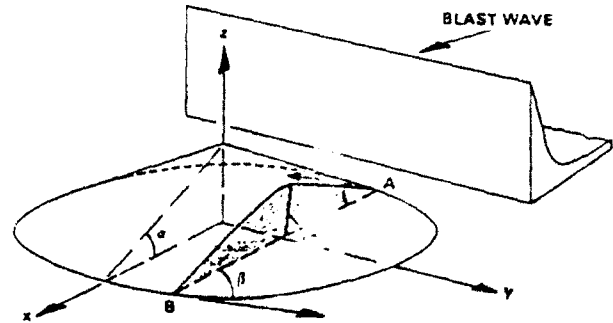


Fig. 1. Schematic of the model. The z axis coincides with the axis of the cone and the flow is taken in the direction of the positive x axis. The angles α , β , γ , and ω are defined in the text

δ between the propagation direction \mathbf{n} (taken to be the positive x -direction) and the deflected streamline at the leading edge of the cone, which determines the shock strength, is bounded above by the angles between \mathbf{n} and the conic sections in the x - y and x - z planes.

The cross sections of the cone parallel to the x - y plane are circles given by

$$x^2 + y^2 = (z_0 - h)^2 / \tan^2 \alpha \equiv r^2, \quad (2)$$

where z_0 is constant for each particular cross section and r is the radius of the circle. The angle β between \mathbf{n} and the tangent to this circle at the point with ordinate y is given by

$$\tan \beta = \frac{\partial y}{\partial x} = -\frac{x}{(r^2 - x^2)^{1/2}} = -\frac{(r^2 - y^2)^{1/2}}{y}. \quad (3)$$

The sign is chosen so that positive values of β correspond to negative values of x (i.e., the upwind side).

The cross sections parallel to the x - z plane form hyperbolas on the surface of the cone. The equation for this family of curves is

$$\frac{(z - h)^2}{\tan^2 \alpha} - x^2 = y_0^2, \quad (4)$$

where y_0 is constant for each particular cross section. The angle γ between \mathbf{n} and the tangent to this hyperbola is given by

$$\begin{aligned} \tan \gamma &= \frac{\partial z}{\partial x} = \tan \alpha / (1 + y_0^2/x^2)^{1/2} \\ &= (1 - y_0^2/r^2)^{1/2} \tan \alpha \end{aligned} \quad (5)$$

Let us examine now how $\tan \beta$ and $\tan \gamma$ vary on the intersection of the cone with the x - y plane when y changes from 0 to $\pm R$, where $r = R = h/\tan \alpha$. From (3), $\tan \beta$ approaches ∞ and 0, i.e., $\beta = 90^\circ$ and $\beta = 0^\circ$, in the limits $y \rightarrow 0$ and $y \rightarrow \pm R$, respectively. From (5), $\tan \gamma$ approaches $\tan \alpha$ and 0 in the same limits, corresponding to $\gamma = \alpha$ and $\gamma = 0^\circ$.

Comparing (3) and (5), we readily conclude that for $\tan \alpha < 1$

$$\begin{aligned} \tan \gamma &< \tan \beta, & 0 < y < R; \\ \tan \gamma &= \tan \beta = 0, & y = R, \end{aligned} \quad (6)$$

and for $\tan \alpha > 1$

$$\begin{aligned} \tan \gamma &< \tan \beta, & 0 < y < R/\tan \alpha; \\ \tan \gamma &= \tan \beta, & y = R/\tan \alpha; \\ \tan \gamma &> \tan \beta, & R/\tan \alpha < y < R; \\ \tan \gamma &= \tan \beta = 0, & y = R. \end{aligned} \quad (7)$$

Thus, at any point with $x \leq 0$ on the cone specified by (1) for $\tan \alpha < 1$, the propagation vector \mathbf{n} makes a smaller angle with the cone in the cross section parallel to the x - z plane than in the one parallel to the x - y plane. For supersonic flow over the cone shown in Fig. 1, condition (6) implies that the velocity vectors behind the shock front in the region of compression of the flow will always be directed over the cone and not around it.

Now we consider intermediate cross sections of the cone, obtainable by rotating through an angle ω about the line AB defined by the intersection of the x - y plane and a plane parallel to the x - z plane. We would like to find the minimum angle between \mathbf{n} and the tangent in these cross sections when ω varies from 0° (cross section parallel to the x - y plane) to 90° (cross section parallel to the x - z plane).

This family of cross sections is defined by (1) together with the equation of the cross-section plane,

$$z - Z = (y - Y) \tan \omega. \quad (8)$$

where ω is the angle between the cross-section plane. We restrict ourselves to points lying in the x - y plane as shown in Fig. 1, since the bow shock produced by the interaction between the flow and the cone will either be attached at this point or will stand off slightly ahead of the cone. The coordinates of the point where the flow encounters the cone are $X = (R^2 - Y^2)^{1/2}$, Y , and $Z = 0$. The tangent line is the intersection of this plane and the plane tangent to the cone at (X, Y, Z) . The equation of the latter is obtained from (1):

$$(xX + yY) \tan^2 \alpha + h(z - h) = 0. \quad (9)$$

Solving (8) and (9) simultaneously yields the equations describing the tangent line:

$$z = (y - Y) \tan \omega = \frac{-(x - X)X \tan^2 \alpha \tan \omega}{h \tan \omega + Y \tan^2 \alpha}. \quad (10)$$

The angle δ between this line and \mathbf{n} is given by

$$\begin{aligned} \tan \delta &= \frac{[(y - Y)^2 + z^2]^{1/2}}{x - X} \\ &= \frac{(h^2 - Y^2 \tan^2 \alpha)^{1/2} \tan \alpha}{h \sin \omega + Y \tan^2 \alpha \cos \omega} \equiv f(\omega). \end{aligned} \quad (11)$$

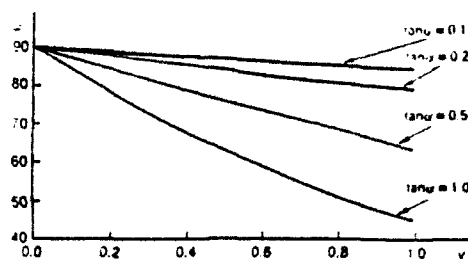


Fig. 2. Value of angle ω which minimizes deflection angle as a function of Y/R

Now we look for the extrema of $f(\omega)$ when ω varies from 0° to 90° :

$$\frac{df}{d\omega} = \frac{\tan \delta (Y \tan^2 \alpha \sin \omega - h \cos \omega)}{h \sin \omega + Y \tan^2 \alpha \cos \omega}, \quad (12)$$

which vanishes only for

$$\tan \omega = \frac{h}{Y \tan^2 \alpha} \equiv \tan \omega_{\min}. \quad (13)$$

It is easy to show that (13) defines the minimum of f . Substituting (13) into (11), we find

$$\tan \delta_{\min} = \frac{(h^2 - Y^2 \tan^2 \alpha)^{1/2} \tan \alpha}{(h^2 + Y^2 \tan^4 \alpha)^{1/2}}, \quad (14)$$

which determines the angle δ_{\min} through which the streamline of the supersonic flow behind the shock wave turns as a function of Y .

From (13) we see that $\tan \omega > 1$ holds for $\tan \alpha < 1$, since $Y \leq h/\tan \alpha$. Figure 2 shows how ω_{\min} changes when Y varies from 0 to R for various values of $\tan \alpha$. This figure implies that for flat cones the supersonic flow will be almost parallel to the x - z plane.

Another way to reach the same conclusion is by finding the maximum y -displacement of a streamline from its original trajectory. This occurs for $x = 0$, when the trajectory reaches its highest point on the surface of the cone. The tangent to the cone at $x = 0$, i.e., the section of the cone in the y - z plane, is described by

$$y \tan \alpha + z = h. \quad (15)$$

Solving this equation together with

$$z = (y - Y) \tan \omega_{\min} = \frac{h(y - Y)}{Y \tan^2 \alpha}, \quad (16)$$

we obtain

$$y = \frac{hY \sec^2 \alpha}{h + Y \tan^2 \alpha} \equiv y_0. \quad (17)$$

Maximizing $\Delta y = y_0 - Y$ with respect to Y , we find that the largest value of Δy occurs for

$$Y = \frac{h \cos^2 \alpha}{\sin \alpha (1 + \cos \alpha)} = \frac{R \cos \alpha}{1 + \cos \alpha} \quad (18)$$

and equals

$$(\Delta y)_{\max} = \frac{h \sin \alpha \cos \alpha}{(1 + \cos \alpha)^2} = \frac{h \tan^2 \alpha / 2}{\tan \alpha} \quad (19)$$

For $\alpha \ll 1$ we have $Y_{\max} \approx R/2$ and $(\Delta y)_{\max} \approx h\alpha/4$.

In summary, the vertical deflection of the streamlines over a cone of small edge angle α is of order α , while the horizontal deflection is of order α^2 . For $\alpha \lesssim 0.1$ we see that the flow over the cone deviates from the vertical plane in which it starts out by an amount 0.01. Thus the compression region of supersonic flow over flat cones can be calculated accurately by modeling the flow over separate cross sections of the cone made by planes parallel to the x - z plane. A wedge, being two-dimensional, is easier to model than a cone. For this reason we carried out several calculations of the interaction between blast waves and wedges, described in the next section.

We can also conclude that the maximum change in the direction of a streamline for such cones will be α . If the shock undergoes regular reflection, uniform supersonic flow over a wedge with base angle α gives an upper bound for the pressure on the cone. Where Mach reflection occurs it is necessary to model the transient regime, as the pressure peaks associated with the Mach stem and the contact surface could conceivably be larger.

3. Numerical modeling

Let us consider a cone with $\alpha = 10^\circ$ ($\tan \alpha = 0.176$) at the base. According to the analysis in the preceding section of a transversely directed supersonic flow over a cone, an upper limit can be obtained by modeling the same supersonic flow over a wedge with opening angle α .

Here we present the result obtained by numerically solving the equations for the flow over the wedge when it is loaded by a passing blast wave. For the simulation we used the Fast Unstructured-Grid Second-Order Godunov Solver, described by Eidelman and Lottati (1990). This code, which is based on a second-order Godunov method (Eidelman et al. 1984), provides a method for solving the Euler equations of gasdynamics on unstructured grids with arbitrary connectivity. The use of a data structure with only one level of indirectness leads to an easily vectorized and parallelized code with low memory requirements and high computational efficiency. The algorithm has been tested for performance and accuracy over a wide range of Mach numbers and geometrical situations, and has demonstrated robustness without the need for any adjustable parameters. It can be implemented in either a triangle- or vertex-based form; experience with the method has shown that extremely low levels of artificial viscosity can be achieved using the triangle-based version of the method. Direct dynamic refinement of the grid (Eidelman and Lottati 1990) allows automatic adaptation to the front of the moving blast wave. This refinement guarantees that the associated highly inhomogeneous pressure and density features are accurately tracked.

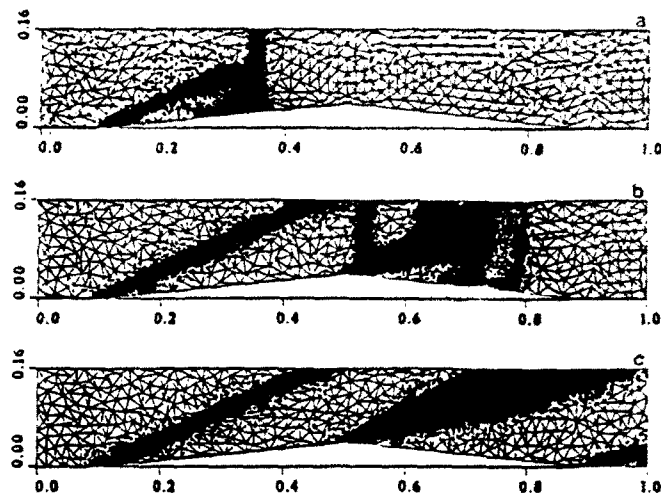


Fig. 3. Unstructured grid for $\alpha = 10^\circ$ at times (a) $35 \mu\text{s}$, (b) $55 \mu\text{s}$, and (c) $130 \mu\text{s}$, associated with the density and pressure contour plots of Figs. 4 and 5. Distances are in meters. These reproductions are unable to resolve the smallest triangles, which show up as dark regions roughly coincident with the locations of gasdynamic discontinuities

For the initial conditions in the computational domain we assume air at standard temperature and pressure. At $t = 0$, a strong ($M = 25$) blast wave, propagating to the right, is located at the left boundary. We assume that the blast wave is "square" and that conditions at the left boundary of the computational domain remain constant for the whole time of the simulation. A constant value of $\gamma = 1.2$ was used (appropriate to flow behind shocks with this value of M on account of real-gas effects). For these values of α , γ , and M , shock tube measurements described by Glass (1987) of diffraction over a wedge indicate that double Mach reflection should occur.

Figure 3 shows the computational grid at various times t : (a) at $t = 35 \mu\text{s}$, shortly before the blast front reaches the apex of the wedge, located at a horizontal distance $l = 1 \text{ m}$ from the corner; (b) at $t = 55 \mu\text{s}$, just after it passes the apex; and (c) at $t = 130 \mu\text{s}$, after the leading shock has exited from the computational domain and a quasisteady state has developed. The highly refined portions of the grid follow shock fronts, contact discontinuities, etc. The numbers of vertices shown are 4166, 11785, and 10959, respectively, reflecting the complexity of the corresponding states, i.e., the amount of structure in the gasdynamic processes present.

Figure 4 shows contours of density scaled by the ambient density $\rho_0 = 1.29 \text{ kg m}^{-3}$ at the same times as in Fig. 3. In the first frame the flow is still identical with that for a shock reflecting from a single wedge with opening angle α , and therefore is evolving self-similarly (Glass 1987). The first Mach stem and incident shock are clearly defined. The associated contact surface is barely discernible, both because the contour levels are bunched near the much larger jumps at the shocks and because at very high Mach numbers the slip line is found quite close

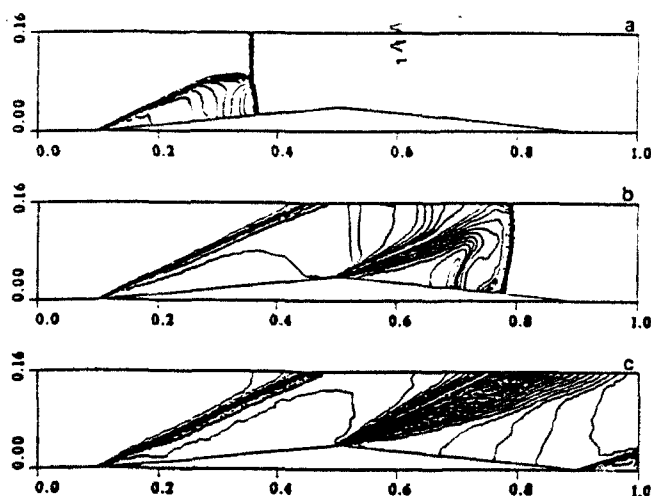


Fig. 4. Scaled density contours for $\alpha = 10^\circ$ at times (a) $35 \mu\text{s}$, (b) $55 \mu\text{s}$, and (c) $130 \mu\text{s}$. Thirty-five contour levels are plotted, with ρ/ρ_0 varying from 1.0 to 13.5. They are concentrated at the large density jumps in the strong shocks. This causes the shocks to be emphasized more than the contact discontinuity, where the density change is relatively small. The structure of the Mach reflection is discernible only in the earliest frame. In the final frame the flow has become essentially steady

to the Mach stem (Glaz et al. 1985). In Fig. 4b the front has passed the apex and the evolution is no longer self-similar. The flow behind the front expands through an expansion fan attached to the corner. Also clearly visible is the recompression shock two-thirds of the way from the corner to the front. This shock, which serves to reconcile the high pressures in the region following the Mach stem with the lower values appropriate to the expanded flow downstream from the corner, is propagating backward but is being swept to the right by the strong flow behind the leading shock. The triple points have moved far above the cone and no longer appear on the grid. Note that the supersonic outflow boundary condition imposed at the top of the mesh allows material and waves to pass out of the system without reflecting and without causing other signals to propagate back inside. Figure 4c depicts the flow at late times, when transients have essentially disappeared. The only gasdynamic features visible are shocks at the leading and trailing edges and the expansion fan.

Figure 5 shows traces of the static and dynamic pressure scaled by the ambient pressure $p_0 = 101.3 \text{ kPa}$ along the top surface of the wedge as functions of the horizontal distance x in meters at the three specified times. Ahead of the blast these quantities are at ambient levels, they rise sharply when the shock sweeps past, fluctuate, and finally reach their asymptotic values. Note that, as is seen experimentally in shock-tube studies (Glass 1987), the pressure on the surface of the wedge is highest at the leading edge. It is also important to notice that, although these traces exhibit considerable structure (especially the static pressure), the maximum values of the pressure and density for the transient stages are

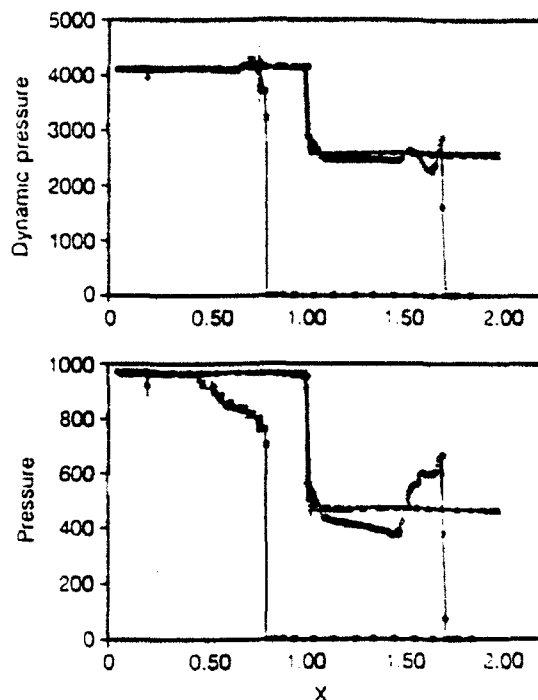


Fig. 5. Scaled dynamic and static pressure on the wedge surface in the case $\alpha = 10^\circ$ as functions of the distance for times $35 \mu\text{s}$ (\square), $55 \mu\text{s}$ (\circ), and $130 \mu\text{s}$ (\triangle). Ahead of the shock these quantities have their ambient values; far behind the shock they become essentially steady

always smaller than those in the quasisteady flow regime. At the same time, values of the Mach number in the transitional stage can be higher than in the quasisteady state. For our case, however, the maximum Mach number is at most 10% higher than the steady-state value. This shows that the maximum force is applied to any point on the surface of the wedge in the quasisteady state.

Figure 6 shows as functions of time the drag and lift coefficients, defined by

$$C_D = \frac{\int p_{\parallel} dx}{\rho_{\infty} u_{\infty}^2 l} \quad (20)$$

and

$$C_L = \frac{\int p_{\perp} dx}{\rho_{\infty} u_{\infty}^2 l} \quad (21)$$

Here $p_{\parallel} = p \cos \theta$ and $p_{\perp} = p \sin \theta$ are the horizontal and vertical components of the pressure in terms of the angle θ between the normal to the surface and the x axis, the integrals are carried out over the surface of the wedge, and ρ_{∞} and u_{∞} are the density behind the undisturbed shock front. The lift grows monotonically, but the drag first rises, then drops to its quasisteady value. The decrease results from the increase in pressure on the trailing side of the wedge when the shocked air reaches that side.

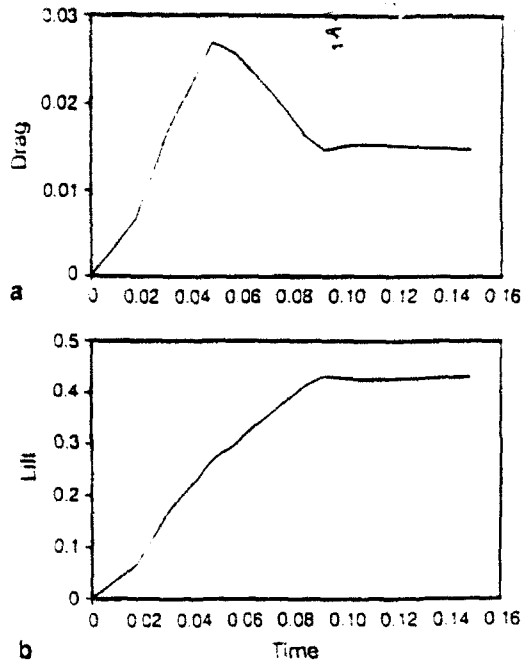


Fig. 6. Lift and drag coefficients for $\alpha = 10^\circ$ as functions of time

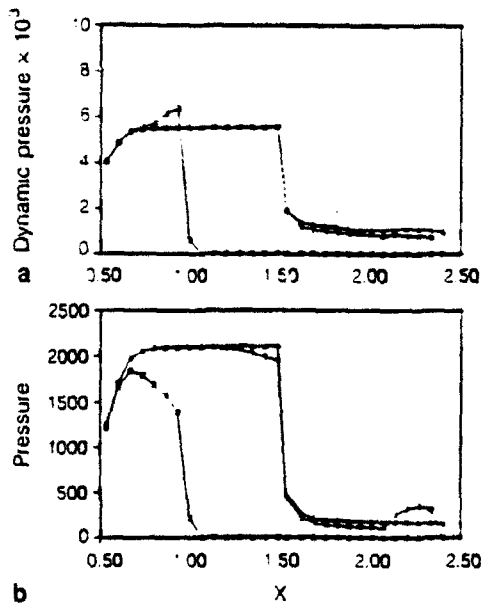


Fig. 7. Scaled dynamic and static pressure on the wedge surface in the case $\alpha = 20^\circ$ as functions of the distance for times $34 \mu s$ (\square), $94 \mu s$ (\circ), and $168 \mu s$ (\triangle)

To learn how sensitive the flow is to the wedge angle, we carried out a second calculation with $\alpha = 20^\circ$ ($\tan \alpha = 0.364$) and the other parameters unchanged. This calculation was done with a coarser grid than the previous one, with triangles about a factor of three larger. Most of the features resembled those of the first case. For example, the traces of the static and dynamic

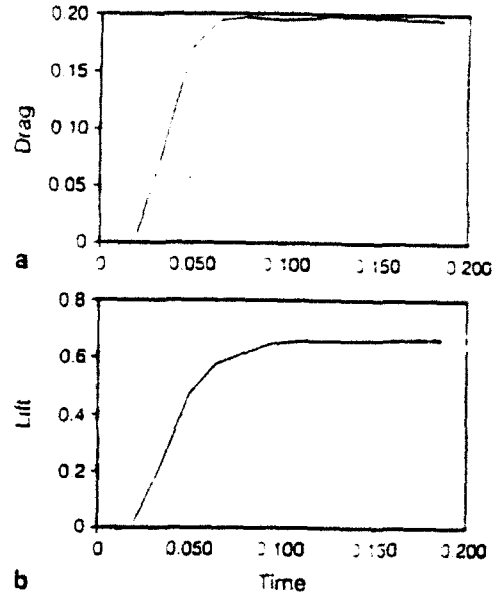


Fig. 8. Lift and drag coefficients for $\alpha = 20^\circ$ as functions of time (in milliseconds)

pressure along the top surface, shown in Fig. 7, are qualitatively similar to those for $\alpha = 10^\circ$. One difference is that the drag rises monotonically as a function of time (Fig. 8), rather than decreasing after the shock has passed. This is because the expansion fan attached to the top of the cone is stronger and a low-pressure "bubble" forms on the lee side.

We carried out additional calculations with other values of the parameters. As long as α was small and M was large the results resembled those discussed above. They are not described here, since in no case did the transient pressures exceed those in the quasisteady state, nor were the features in the flow qualitatively different.

4. Conclusions

From the foregoing treatment it is clear that the same modeling technique can be used to determine the pressure distribution in cross sections other than the mid-plane. So long as $|y|$ does not approach R , the deflection is mostly vertical. The corresponding profile is now a hyperbola, but it differs noticeably from a wedge only near the top. The principal difference is in the expansion wave at the top, which becomes broader than the centered rarefaction wave seen above. For larger values of y the cross section of minimum deflection, found by solving (1) and (8) together, is more rounded at the top but the leading edge of this hyperbolic "wedge" has a smaller angle. By combining pressure distributions at several representative values of y we can find the pressure loading over the entire cone. The picture breaks down only at the lateral extremities of the cone ($|y| \sim R$).

On the basis of qualitative arguments and numerical simulation, our study of the flow resulting from a blast wave propagating transversely over a cone leads to the following conclusions:

1. Flow over a cone with small base angle can be accurately simulated by individually modeling the two-dimensional flows over cross sections of the cone made by vertical planes perpendicular to the shock front.
2. The maximum load on the cone can be calculated from the solution of the flow over the cross section determined by the plane through the cone axis (Fig. 1).
3. In this solution the pressure attains its maximum as a function of time in the quasisteady supersonic regime established after the front has passed.

Acknowledgements. This work was sponsored in part by the Defense Nuclear Agency. The work of the first author was supported by the Naval Research Laboratory.

References

- Damkevala RJ, Zumwalt GW (1968) Technique for studying interactions between a body moving at supersonic speeds and blast waves approaching obliquely. *Rev Sci Instr* 39:1254-1256
- Eidelman S, Colella P, Shreeve RP (1984) Application of the Godunov method and its second-order extension to cascade flow modeling. *AIAA J* 22:1604-1615
- Eidelman S, Lottati I (1990) Reflection of the triple point of Mach reflection in planar and axisymmetric converging channels. In: Reichenbach H. (Ed) *Proc 9th Mach Reflection Symp Freiburg FGR*
- Fletcher CJ, Holt M. (1976) Supersonic viscous flow over cones at large angles of attack. *J Fluid Mech* 74:561-591
- Glass II (1987) Some aspects of shock-wave research. *AIAA J* 25:214-228
- Glaz HM, Colella P, Glass II, Deschambault RL (1985) A numerical study of oblique shock-wave reflections with experimental comparisons. *Proc R Soc London A398:117-140*
- Goman OG, Davydov VI (1975) Determination of aerodynamic characteristics of cone with arbitrary small surface deviations. *Izv VUZ Aviats Tekh* 18:58-62. [Transl Soviet Aeronautics 18(4):44-48]
- Gusarov AA, Dvoretckij, VM, Ivanov, MYa, Levin VA, Chernyi, GG (1979) Theoretical and experimental investigation of the aerodynamic characteristics of three-dimensional bodies. *Izv Akad Nauk SSSR Mekh Zhidkosti i Gaza* No 3, 97-102. [Transl Fluid Dynamics 14:402-406]
- Sedov LI (1959) *Similarity and dimensional methods in mechanics*. Academic Press, New York
- Tracy RR (1963) Hypersonic flow over a yawed cone. Memo 69 Calif Inst Tech Graduate Aeronautical Labs
- Yahalom R (1971) An experimental investigation of supersonic flow past yawed cones. Rept AS-71-2 Univ Calif Berkeley

This article was processed using Springer-Verlag T_EX Shock Waves macro package 1990.



AIAA 92-0346

**Detonation Wave Propagation in Variable
Density Multi-Phase Layers**

S. Eidelman and X. Yang

Science Applications International Corporation
McLean, VA 22102

**30th Aerospace Sciences
Meeting & Exhibit**
January 6-9, 1992 / Reno, NV

Detonation Wave Propagation in Variable Density Multi-Phase Layers

Shmuel Eidelman and Xiaolong Yang
Science Applications International Corporation
McLean, VA 22102

ABSTRACT

A mathematical model is presented describing a physical system of detonation waves propagating in a solid particle/air mixture with a wide range of solid phase concentrations. The mathematical model was solved numerically using the Second Order Godunov method, and numerical solutions were validated for detonation waves propagating in mixtures with concentrations of solid phase from 0.75 kg/m^3 to 1000 kg/m^3 . Numerical solution was obtained for detonation waves propagating in a system consisting of layers of explosive powder with substantial variation in particle density between the layers. The study revealed a specific detonation front structure that is dependent on the thickness of the layers and their energetic content. The dynamics of lateral initiation of the adjacent layers and the structure of detonation waves in this system were investigated. Results are given for detonation of clouds having a small concentration of particles and a ground layer in which solid particle densities are three orders of magnitude larger than in the cloud.

1. INTRODUCTION

It is of considerable practical interest to study diffraction and transmission of the detonation waves into bounding layers of explosives. When combustible particles are intentionally or unintentionally dispersed into the air, the resulting mixture can be detonable. Formation of this potentially explosive dust environment and the properties of its detonation are of significant practical interest in view of its destructive or creative effects. The experimental and theoretical study of these phenomena until now has addressed only homogeneous particle/oxidizer mixtures. However, intentional or accidental processes of the explosive dust dispersion will always lead to inhomogeneous particle density distribution. Some industrial methods of explosive forming rely on detonation of explosive powder. This powder can be deposited as a thin layer over the surface area of the forming metal, with some remaining concentration in the vicinity of the layer. Also a multi-layer system can be formed from several layers of condensed explosives of different density. The structure of the detonation waves, phenomenology of its initiation, and propagation in these environments, are the main subjects of this paper.

When the detonation wave is generated in a homogeneous mixture by a "direct initiation," it starts with a strong blast wave from the initiating charge. As the blast wave decays, combustion of the reactive mixture behind its shock front starts to have a larger role in support of the shock wave motion. When the initial explosion energy exceeds some critical value, transition to steady state detonation occurs.⁽¹⁻⁴⁾ In explosive dust mixtures with a nonuniform distribution of particle density, the initiation dynamics is significantly more complicated. The critical initiation energy sufficient for one of the explosive particle density regions is not necessarily adequate for other regions. Also, when there is a sig-

nificant variation in density between the different layers (regions) of the mixture, steady detonation in one layer can result in an overdriven detonation in an adjacent layer. Liu et al.⁵ has studied experimentally a system of gaseous layers and lateral interactions for gaseous detonations. Our paper demonstrates that the phenomenology of these interactions is somewhat different from these experimental studies of multi-layer detonations in gases. This is primarily because the energy content of adjacent layers in a typical multi-gas layer experiment⁵ varies by a factor of less than two, whereas the energy content in explosive dust/air mixtures can vary by several orders of magnitude.

In this paper we use detailed numerical simulation to study the initiation dynamics and propagation phenomenology for a general case of explosive dust dispersion. We will consider particle density variation from 1000 kg/m^3 in the ground layer to 0.75 kg/m^3 for the upper edges of the cloud. The effects of variation of the cloud density on detonation wave parameters will be examined for different cases of cloud particle density distribution. When possible, the results of computer simulations are validated in comparison with experimental and theoretical studies.

Section 2 of this paper describes a mathematical model that includes governing conservation equations for two phases and the constitutive laws, as well as the model for a particle gas interaction, combustion and equation-of-state for gas phase. The numerical integration technique for solving the mathematical model will be also outlined. In Section 3, we present our numerical simulation results. We first validate our model by comparing one-dimensional detonation wave simulation with available experimental results. We then give the two-dimensional simulation for detonation wave propagation in combustible particle/air mixtures with variable parti-

cle density distribution. Concluding remarks are given in Section 4.

2. THE MATHEMATICAL MODEL AND THE NUMERICAL SOLUTION

The mathematical model consists of conservation governing equations and constitutive laws that provide closure for the model. The basic formulation adopted here follows the two-phase fluid dynamics model presented in the text by Kuo⁷. The approach assumes that there are two distinct continua, one for gas and one for solid particles, each moving at its own velocity through its own control volume. The sum of these two volumes represents an average mixture volume. With these assumptions, distinct equations for continuity, momentum and energy are written for each phase. The interaction effects between the two phases are accounted as the source terms on the right hand side of the governing equation. The following is a short description of the two phase flow model used in our study, with conservation equations written in Eulerian form for two-dimensional flow in Cartesian coordinates.

Conservation Equations

Continuity of gaseous phase

$$\frac{\partial \rho_1}{\partial t} + \frac{\partial(\rho_1 u_g)}{\partial x} + \frac{\partial(\rho_1 v_g)}{\partial y} = \Gamma; \quad (2.1)$$

Continuity of solid particle phase

$$\frac{\partial \rho_2}{\partial t} + \frac{\partial(\rho_2 u_p)}{\partial x} + \frac{\partial(\rho_2 v_p)}{\partial y} = -\Gamma; \quad (2.2)$$

Conservation of momentum of gaseous phase in x-direction

$$\frac{\partial(\rho_1 u_g)}{\partial t} + \frac{\partial(\rho_1 u_g^2 + \phi p_g)}{\partial x} + \frac{\partial(\rho_1 u_g v_g)}{\partial y} = -F_x + \Gamma u_p; \quad (2.3)$$

Conservation of momentum of solid particle phase in y-direction

$$\frac{\partial(\rho_1 v_g)}{\partial t} + \frac{\partial(\rho_1 u_g v_g)}{\partial x} + \frac{\partial(\rho_1 v_g^2 + \phi p_g)}{\partial y} = -F_y + \Gamma v_p; \quad (2.4)$$

Conservation of momentum of solid particle phase in x-direction

$$\frac{\partial(\rho_2 u_p)}{\partial t} + \frac{\partial(\rho_2 u_p^2)}{\partial x} + \frac{\partial(\rho_2 v_p u_p)}{\partial y} = F_x - \Gamma u_p; \quad (2.5)$$

Conservation of momentum of solid particle phase in y-direction

$$\frac{\partial(\rho_2 v_p)}{\partial t} + \frac{\partial(\rho_2 u_p v_p)}{\partial x} + \frac{\partial(\rho_2 v_p^2)}{\partial y} = F_y - \Gamma v_p; \quad (2.6)$$

Conservation of energy of gas phase

$$\frac{\partial(\rho_1 E_{gT})}{\partial t} + \frac{\partial(\rho_1 u_g E_{gT} + u_g \phi p_g)}{\partial x} + \frac{\partial(\rho_1 v_g E_{gT} + v_g \phi p_g)}{\partial y}$$

$$\Gamma \left(\frac{u_p^2 + v_p^2}{2} + E_{chem} + C_s \bar{T}_p \right) - (F_x u_p + F_y v_p) - \dot{Q}; \quad (2.7)$$

Conservation of energy of solid particle phase

$$\frac{\partial(\rho_2 E_{pT})}{\partial t} + \frac{\partial(\rho_2 E_{p1} u_p)}{\partial x} + \frac{\partial(\rho_2 E_{p1} v_p)}{\partial y} = \dot{Q} + (F_x u_p + F_y v_p)$$

$$- \Gamma \left(\frac{u_p^2 + v_p^2}{2} + E_{chem} + C_s \bar{T}_p \right); \quad (2.8)$$

Conservation of number density of solid particle

$$\frac{\partial N_p}{\partial t} + \frac{\partial(N_p u_p)}{\partial x} + \frac{\partial(N_p v_p)}{\partial y} = 0. \quad (2.9)$$

In the above equations, $\phi = 1 - \frac{N_p M_p}{\rho_p}$, $\rho_1 = \phi \rho_g$, $\rho_2 = (1 - \phi) \rho_p$, where N_p and M_p are the number density and mass of each particle, respectively, and ρ_g and ρ_p are the material density of gas and particle densities, respectively. u_g, v_g, p_g are gas phase x-velocity, y-velocity and pressure, respectively; u_p, v_p, \bar{T}_p are x-velocity, y-velocity and average temperature of particle, respectively. C_s is specific heat of solid particle and E_{chem} is chemical energy of solid phase, Γ is the rate of phase change from solid to gas and \dot{Q} is heat transfer between the two phases; F_x, F_y are the drag force between the two phases in x and y directions, respectively.

Equations (2.2) and (2.7) are linked through the relation $\rho_2 = n M_p$. In the case of a reactive solid phase, M_p decreases due to combustion. The mass of a single particle at any point can be obtained from $M_p = \rho_2(x, y) / n(x, y)$, and the diameter of a particle at any spatial location is $D(x, y) = \{6 M_p(x, y) / \pi \rho_p\}^{1/3}$. The total internal energy of gaseous phase

$$E_{gT} = E_g + \frac{1}{2}(u_g^2 + v_g^2) \quad \text{and} \quad E_g = E_g(p_g, \rho_g) \quad (2.10)$$

where $E_g(p_g, \rho_g)$ is the equation-of-state for the gas phase, which will be discussed later.

The total internal energy of solid particle phase is

$$E_{pt} = E_p + \frac{1}{2}(v_p^2 + v_g^2) \quad \text{and} \quad E_p = E_{chem} + C_p \bar{T}_p \quad (2.11)$$

In order to close the above system of conservation equations, it is necessary to define certain criteria and interaction laws between the two phases, which include mass generation rate, Γ , drag force between particles and gas, F_x, F_y and the interphase heat transfer rate \dot{Q} . The model for particle and gas interaction and particle combustion that results in the constitutive relation for the conservation equations, is explained in detail in the next subsection.

Model for a Particle Gas Interaction and Combustion

Presently, the physics of the energy release mechanisms in solid particles/air mixtures is not clearly understood. This can be attributed to the obvious difficulties of making a direct non-obtrusive measurement in the optically thick environment typical for this system. In the experimental and theoretical work done for the grain dust detonation conditions,⁷ it was demonstrated that the volatile components released by the particle heated behind the shock front play a major role in determining the detonability limits of the mixture. Eidelman and Burcat⁸ successfully applied a combination of fast evaporation and aerodynamic shattering mechanisms to simulate a two-phase detonation process.

The chemical processes of a single particle combustion, which mainly occur in the gaseous phase, are significantly faster than the physical processes of particle gasification or disintegration. Thus, in the multi-phase mixtures, the rate of energy release will be mostly determined by physics of particle disintegration. It is very difficult to describe the details of particle disintegration in the complex environment prevalent behind the shock or detonation wave. For example, Reinecke and Waldman⁹ defined five different disintegration regimes for a relatively simple environment of water droplets passing through a weak shock. Fortunately, in most cases of multi-phase detonation, only the main features of the particle disintegration dynamics need to be captured to describe the phenomena. For example, Eidelman and Burcat,¹⁰ using simple models for particle evaporation and shattering, obtained simulation results that compared very favorably with experimental data. Because of our inability to resolve the particle disintegration problem in all its complexity, the validation of the model against known experimental data is essential.

In this paper we consider solid particles consisting of explosive material. Explosive material contains fuel and oxidizer in a passive state at low temperature; however, when the temperature rises the fuel and oxidizer

react, leading to detonation or combustion. The initiation of reaction for explosives will occur at relatively low temperature. For example, TNT will detonate when heated to the temperature¹¹ of 570°C. Only particles larger than a critical detonation size can detonate directly when initiated by a shock wave. We consider here particles smaller than 4mm in diameter that will not detonate when heated, but will burn when the temperature on the particle surface reaches a critical value. Since the heat conduction inside the explosive material is relatively slow, the process of particle heating needs to be resolved in detail. Our simulations numerically solve the temperature field in the particles at every step of numerical integration of the global conservation equations. The explosive particle combustion model examined in this paper assumes that the fraction of the particle that reaches the critical temperature will burn instantaneously.

Energy transfer by convection and conduction is simulated by solving the unsteady heat conduction equation in each computational cell at each time step. Assuming a particle's temperature T_p to be a function of time and radial position only, the unsteady heat conduction equation may be transformed to:

$$\frac{d^2 w}{dr^2} = \frac{1}{\alpha} \frac{dw}{dt} \quad (2.12)$$

subject to the boundary conditions:

$$w = 0 \quad \text{at} \quad r = 0, \quad t > 0$$

$$k_s \frac{dw}{dr} = \left(h - \frac{1}{R}\right) w = hRT_g \quad \text{at} \quad r = R, \quad t > 0 \quad (2.13)$$

where:

- $w(r, t) = rT_p(r, t)$
- r = radial position
- $T(r, t)$ = temperature
- R = particle radius
- T_g = temperature of surrounding gas
- k_s = thermal conductivity of particle
- h = convective heat transfer coefficient.

The Nusselt number, used to find h , is given by an empirical relation provided by Drake.¹² The gas viscosity is found from Sutherland's Law. The gas thermal conductivity is calculated by assuming a constant Prandtl number. Lastly, the boiling temperature at a given pressure is found from the Clapeyron-Clausius equation, assuming: 1) constant latent enthalpy of phase change, 2) the vapor obeys the ideal equation of state, and 3) the specific volume of the solid/liquid is negligible compared to that of the vapor. A critical temperature is also employed to serve as an upper limit to the boiling point, regardless of pressure.

Equation (2.13) with boundary condition (2.14) can be numerically integrated using either implicit or explicit schemes.

Since the particle radius, R , will become very small due to evaporation, the implicit Crank-Nicolson algorithm is used because of its stability properties and its second order temporal and spatial accuracy. Using the Crank-Nicolson scheme to predict the particle temperature profiles at times t_1 and t_2 permits easy calculation of the total energy exchange, Q between t_1 and t_2 due to convection and conduction.

Knowledge of the particle temperature profile also allows us to determine Γ , the rate of phase change from solid particle to gas. Once any point at a radial location $0 \leq r \leq R$ has a temperature exceeding the boiling temperature, the entire mass between r and R is transferred to the gas phase in one time step. In so doing, an energy equal to the product of the mass lost and the particle intrinsic energy is transferred by the particle to the gas.

The interphase drag force (F_x, F_y) is determined from the experimental drag for a sphere, as presented by Schlichting¹³.

$$F_x = \left(\frac{\pi}{8}\right) N_p \rho_g C_D |V_g - V_p| (u_g - u_p) R^2 \quad (2.14)$$

where

$$C_D = \begin{cases} \frac{24}{Re} \left(1 + \frac{Re^{2/3}}{6}\right) & \text{for } Re < 1000; \\ 0.44 & \text{for } Re > 1000, \end{cases} \quad (2.15)$$

and $Re = \frac{2R|V - V_p|}{\mu_g}$, R is radius of particle and μ_g is gas viscosity at temperature of $T_{film} = \frac{1}{2}(T_g + \bar{T}_p)$. Similarly, the formulae for F_y is

$$F_y = \frac{\pi}{8} N_p \rho_g C_D |v_g - v_p| (v_g - v_p) R^2. \quad (2.16)$$

Equation of State for Detonation Products

To close the system of governing equations, one needs a constitutive relation between density, pressure, temperature and energy for gas phase, which is an equation-of-state. This study uses the Becker-Kistiakowsky-Wilson (BKW) equation-of-state^{14,15} that is,

$$p_g V_g / \bar{R} T_g = 1 + z e^{bz}, \quad (2.17)$$

where V_g = volume of gas phase
 p_g = pressure of gas phase
 T_g = temperature of gas phase

\bar{R} = universal gas constant

$$z = k/F_g(T + \Theta)^a \quad k = K \sum_i X_i k_i$$

with empirical constants a, b, K, Θ and k_i . The constants k_i , one for each molecular species, are co-volumes. The co-volumes are multiplied by their mole fraction of species, X_i , and are added to find an effective volume for a mixture. For a particular explosive, if we know the composition of detonation products and a, b, Θ, K , and all k_i 's can be found in Ref. 15.

The internal energy is determined by thermodynamics relation

$$\left(\frac{\partial E_g}{\partial V_g}\right)_T = T_g \left(\frac{\partial p_g}{\partial T_g}\right)_V - p_g. \quad (2.18)$$

Integration of this equation for a fixed composition of the detonation products will allow us to calculate the energy of the detonation products as a function of temperature and volume. For each component, its thermodynamic properties as functions of temperature were calculated from the NASA tables compiled by Gordon and McBride¹⁶.

The BKW equation-of-state is the most common and well calibrated of those equations-of-state used to calculate the properties of detonation products. The detailed discussion and review of the BKW equation-of-state can be found in Ref. 15.

Numerical Method of Solutions

The system of partial differential equations described in the previous paragraph is integrated numerically. The Second Order Godunov method is used for the integration of the subsystem of equations describing flow of gaseous phase material. This method is described in Ref. 17. In the following, we will elaborate only on some specifics of its application to simulations of detonation products. The subsystem of equations describing the flow of particles is integrated using a simple upwind integration. This is done because our mathematical model neglects pressure of interparticle interaction and that prevents formulation of a Second Order Godunov scheme for particles.

The physical system under study will have concentrations of solid explosive powder ranging from 1000 kg/m³ near the ground to 0.75 kg/m³ or less in the cloud. Detonation of this mixture will create detonation products with effective γ ranging from 3 to 1.1. To describe the flow of detonation products, we use the BKW equation-of-state described above. Since the Second Order Godunov method uses primitive variables to calculate Riemann problems at the edges of the cells, its implementation for non-ideal EOS is difficult. In our simulations, we have resolved this problem by using direct and inverse equations-of-state. After integrating a system of gas conservation laws, we use the direct BKW

equation-of-state to calculate pressure, gamma and temperature as functions of thermal energy, density, and mixture composition. After this step, we have a complete set of parameters allowing calculation of the fluxes in the Second Order Godunov method as well as interaction of the multi-phase processes. The "inverse" EOS calculates internal energy as a function of density, pressure and mixture composition. In our code we use the "inverse" EOS to calculate the fluxes of conserved variables after calculation of the flux of primitive variables.

For the multi-phase system under study, $dx=dy=1\text{mm}$ was used to allow explicit integration of the gasdynamic and physical processes of evaporation and heat release. When a mismatch occurred between the physical and gasdynamical characteristic times, the time step was adjusted by some fraction to assure stability. However, this did not result in a significantly smaller time step as compared with that calculated by CFL criteria. For larger cell sizes, this approach is impractical. Recently we implemented a scheme in which multi-phase processes are calculated implicitly; however, this will be reported elsewhere.

The numerical method is implemented in a code named MPHASE, which is fully vectorized and supported by number of graphics and diagnostics codes.

3. RESULTS

Model Validation for One-Dimensional Detonation Wave Problem

The main advantage of our particle combustion model is its description of the phenomenology of detonation for a wide range of explosive particle sizes and densities. We will demonstrate this capability on a set of one-dimensional test problems. For these test problems, we simulated the initiation and propagation of the detonation waves in a shock tube-like setting, where the explosive particles are distributed uniformly through the shock tube volume.

Results of these simulations are summarized in Table 1, which shows detonation wave velocity, peak pressure, and peak density given as a function of the average density of the solid explosive. Here the explosive two-phase mixture is composed from RDX particles and air, where RDX particle concentration varies from 0.75 kg/m^3 to 1000 kg/m^3 . This concentration variation covers the whole range of solid explosive concentrations of interest to our problem. The simulations performed with the MPHASE code were compared with the experimental results,^{15,16} and the calculations presented in Ref. 19 were done with the TIGER code.

From Table I, it is clear that our simulation results compare favorably with other simulation results and experimental data. The maximum deviation between our results and referenced results is no greater than 15% for the entire range of explosives densities. Considering that

our results were obtained with a single model for particle combustion applied to the extreme range of densities, our model gives an excellent prediction of the detonation wave parameters.

Two-Dimensional Simulation Results

In our two-dimensional simulations, we first study the dynamic of the lateral initiation in a simple system formed by two layers of explosive with different concentrations of the explosive powder in the layers. These layers of explosive will be considered confined in a rectangular shock tube with rigid walls. The schematics of the set up for a typical simulation of this type are shown in Figure 1. The detonation wave is initiated in the lower layer, and its propagation through the shock tube causes lateral initiation of the adjacent layer. In one of the test cases, both layers are initiated simultaneously with a planar front.

First we simulated initiation and propagation of the detonation in a system of two layers of detonable RDX powder/air mixture contained in a rectangular channel 4 cm wide and 35 cm long. The lower layer has an RDX powder concentration of $800\frac{\text{kg}}{\text{M}^3}$ and occupies half of the channel width, and the upper layer of the channel has a mixture concentration of $200\frac{\text{kg}}{\text{M}^3}$. Detonation is initiated in the lower layer by a planar front that is propagating from left to right. In Figures 2a:2f, results of this simulation are shown in the form of pressure contours on a logarithmic scale in MPa for a sequence of time frames. In these figures, we can follow the evolution of the lateral initiation and formation of the detonation wave structure in this system.

In Figure 2a, contour plots are shown at time $t=0$, which corresponds to the beginning of the simulation and depicts initial conditions of the planar wave in the lower layer. This initial wave causes lateral initiation of the upper layer through an oblique detonation front shown in Figure 2b at $t=9 \times 10^{-6}$ sec. The oblique front reflects from the upper wall of the channel, and in Figure 2c we observe that the wave pattern indicates it is a single Mach reflection. The Mach stem is very short at this point. In Figure 2d, the pressure contours are shown at the time $t=31 \times 10^{-6}$ sec. Here the Mach stem is clearly visible and the reflected shock has reached the lower wall of the channel. The Mach stem will continue to grow and the triple point will propagate towards the high density layer. In Figure 2e, the simulation results are shown at $t=52 \times 10^{-6}$ sec when detonation wave complex has reached steady state propagation regime. The triple point has reached the interface between the two layer and is unable to continue propagation downwards due to the high level of pressure and density in the lower layer. Also at this stage of the detonation wave propagation, the reflected shock has reached the upper wall

of the channel. In Figure 2f, the simulation results are shown at $t=64 \times 10^{-6}$ sec. Here the structure of the detonation front is basically unchanged from the previous picture, except for an additional reflection from the upper wall of the channel. The detonation wave parameters are also unchanged from the previous time frame, indicating that the detonation wave in this two layer system has reached steady state.

To validate that the detonation waves complex observed in above reported simulation is not a function of the initial conditions, we simulated a test case in which all problem parameters, except the initiation wave, are the same as in the previous case. The initiation is done by a single planar wave that starts propagating simultaneously in both layers of the explosive. In Figures 3a:3e, results for this simulation are shown in the form of pressure contours for a sequence of time frames. The initial conditions are shown in Figure 3a. Here we can observe a planar front impinging simultaneously on both layers of explosive in the channel. At first, this front propagates some distance planarly, as observed in Figure 3b. However, a significant difference in the explosive powder density quickly leads to formation of the oblique front in the upper layer, as shown in Figure 3c. As in the previous case, the oblique front reflects from the upper wall in the single Mach reflection shown in Figure 3d. And as in the previous case, the triple point of the Mach stem propagates downward to the interfaces between the layers to form the stable wave pattern shown in Figure 3e. The parameters of the detonation waves and the structure of the detonation wave complex are identical to those observed in the previous case, which proves that it is not a function of the initial conditions, but physical conditions of the layers.

We studied the effects of the channel walls using a system that included a 2cm thick lower layer of high density ($800 \frac{\text{kg}}{\text{m}^3}$) RDX powder and a 10cm thick upper layer of low density ($200 \frac{\text{kg}}{\text{m}^3}$) RDX powder. The results of this simulation are shown as pressure contours on a logarithmic scale in Figures 4a:4d. Figure 4a shows the initial conditions. In Figure 4b, we can see at the time $t = 25 \times 10^{-6}$ a planar detonation wave is propagating through the lower layer and an oblique wave is propagating through the upper layer. In Figure 4c, the detonation wave is shown at the time $t=41 \times 10^{-6}$ from the initiation. Here the oblique wave is reflecting from the upper wall; however, it is distinct from the previous cases because only a regular reflection pattern is formed. This is due to the shallow angle of incidence of the detonation wave, that corresponds to the large wedge angles in classical reflection problems. Figure 4d shows the results of the simulation at $t = 52 \times 10^{-6}$. Here we can observe the same regular reflection pattern as in the previous stage; however, the incidence angle of the oblique

wave in the upper layer is increasing. Thus, if this trend continues, later in the detonation wave evolution we will see the formation of the Mach reflection pattern, as we have in previous cases.

We have also examined propagation of the detonation wave in the system shown in Figure 5 that corresponds to the situation where the upper layer is not confined by the channel wall. Here the computational domain is 25cm \times 25cm in size. The explosive powder density is distributed according to the 4th power law of vertical distance, starting from the ground where the density is 860 kg/m^3 , to 1.2cm, where the density is 0.75 kg/m^3 . From this point to 25cm height, the density is constant and equal to 0.75 kg/m^3 . The density distribution in the direction of the "x" axis is uniform. The boundary conditions for the computational domain shown in Figure 5 are specified as follows: solid wall along the "x" axis; symmetry conditions along the "y" axis; supersonic outflow for upper boundary and at the right of the computational domain. The mixture consists of RDX powder and air at ambient conditions and it is assumed to be quiescent at the time of initiation.

The simulation starts at $t=0$ when the mixture is initiated at the lower left corner of the computational domain, as shown in Figure 5. The energy released by the initiating explosion leads to formation of the detonation wave propagating through the multi-phase media. Figure 6a shows pressure contours for the propagating detonation wave at the time of $t=12 \times 10^{-6}$ msec after initiation. Here the pressure contour levels are shown on logarithmic scale in MPa. The maximum pressure value of 7940 MPa is observed in the layer of condensed explosive located near the ground. The pressure in this layer is two to three orders of magnitude higher than the pressure behind the detonation wave in the 0.75 kg/m^3 RDX cloud and air, which is located above the distance of 1.2cm from the ground. Figure 6a demonstrates that the detonation wave in the cloud is overdriven, since the pressure behind the shock continuously rises and reaches its maximum in the layer. From this figure, we also observe that the overdriven wave propagates faster in the cloud than in the layer. This is explained by the fact that it is easier to compress air that is very lightly loaded with particles and located above the ground layer, than it is to compress air heavily loaded with a particle mixture near the ground. It is interesting to note a discontinuous pressure change between the yellow contours and the light blue and green contours behind the detonation front. This discontinuity is over-emphasized by the presentation of contour lines on the logarithmic scale; however, further examination of our simulation results indicates this feature is real and is similar in nature to barrel shocks observed for strong jets. It is different in nature from the triple shock structures described above.

In Figure 6b, gas phase density contours are shown for the time $t = 12 \times 10^{-6}$ sec. Here the contour lines are distributed on logarithmic scale. The main features of the shock wave structure are very similar to those observed in the pressure contours figure. Here we see that a jet of high density gases reflects from the center of symmetry axis, creating a contact discontinuity that we will observe at a later time. The barrel shock is clearly visible in this figure. In Figure 6c, the particle density contour plots are shown for $t=12 \times 10^{-6}$ sec. The contour levels in this figure are given on the logarithmic scale and the initial deposition of the explosive material in the ground layer of the computational domain can be clearly observed. The black contour lines delineate the beginning and the end of the reaction zone in the cloud. To the left of these contours lies an area with combustion products and to the right unburned particles in the cloud. Here we can see that the reaction zone length is of the order of 1cm.

Figure 6d shows pressure contours for the same simulation for the time $t = 55 \times 10^{-6}$ sec, just before the detonation wave leaves the computational domain. In this figure, we see that the global structure of the wave did change slightly from Figure 6a. We observe that the barrel shock wave is fully developed and has a half ellipse shape. The detonation wave in the cloud is still overdriven; however, part of the shock wave front that propagates vertically weakened as it got further away from the detonation front in the layer. In Figure 6e, gas temperature contours are shown at $t = 55 \times 10^{-6}$ sec. In this case, it is interesting to note that the highest temperatures are observed behind the front of the overdriven detonation wave in the cloud, in the immediate vicinity of the upper strata of the layer. Very high temperatures in this region can be explained by the high pressure generated by the detonation of the explosive material in the layer and by relatively low density of strata of the cloud in the immediate vicinity to the layer. Here, as in the pressure contours graph, the area of barrel shock can be clearly identified.

We also observe in Figure 6 a clear development of two detonation fronts, one moving vertically in the cloud and another moving horizontally in the layer. Because the energy density of the explosive powder in the layer is about three orders of magnitude larger than that in the cloud, the vertical parts of the front represent overdriven detonation waves in the cloud. Even though the vertical front has slowed down compared with the horizontal front, its speed and parameters far exceed those typical for detonation waves in a cloud. In fact, the self-sustained detonation regime in the cloud will develop at the distance of about three meters from the layer. The area of the front close to the detonation wave in the layer will remain hot and overdriven, since it is located very

close to detonation front in the layer. In Figure 6f, particle density contours are shown on a logarithmic scale. We can clearly observe the reaction zone delineated by black contour lines. In this case, the reaction zone length in the cloud is about 1cm. Consistent with the gradual transition from overdriven to self-sustained detonation, the reaction zone length is larger for the vertical part of the detonation front. The detonation wave velocity observed in our simulation is approximately 4048 m/sec, which is significantly lower than the detonation wave velocity observed in RDX with a density of 860 kg/m^3 (see Table 1), which is the highest density in the ground layer. This can be explained by the high gradient of particle density distribution in the layer, where the density drops rapidly from 860 kg/m^3 at the bottom of the layer to 0.75 kg/m^3 at the top strata of the layer at 12 mm above the ground.

4. CONCLUSIONS

We have presented a mathematical model and numerical solution for the simulation of initiation and propagation of the detonation waves in multi-phase mixtures consisting of solid combustible particles and gas. Using this model, we studied detonations in mixtures of solid RDX particles and air for the purpose of examining the effects of wide variation in particle density distribution on the dynamics and structure of detonation waves. We considered a physical system of layers of explosive RDX powder confined in a channel and studied initiation and propagation of the detonation waves in this system. This study revealed a specific structure of the detonation front that is dependent on the thickness of the layers and their energetic content. We showed that for the system consisting of two layers of the same thickness but of vastly different powder density, a Mach stem reflection occurs that propagates to the interface between the layers and helps create a stable detonation front. However, formation of the Mach stem reflection will be a strong function of the relative thickness of the layer; in one of the simulated examples, only a regular reflection would form in the simulation time frame.

For the system consisting of a solid particle cloud in air and a layer of high particle density near the ground, our simulations have revealed a specific detonation front shape with a characteristic precursor of the blast front in the strata immediately above the layer. This feature of the detonation front can be explained by the fact that the energy released in the detonation wave in the ground layer produces a faster shock wave in the dilute cloud than in these heavily loaded with solid particles strata of the ground layer. However, these structures were not observed experimentally, and more studies are needed to examine their parameters.

The maximum pressure affecting the ground was di-

rectly related to the maximum particle density in the lower strata of the layer. However, the detonation front velocity for the fourth power distribution case was considerably lower than calculated for a one-dimensional case with 860 kg/m^3 particle density, reflecting the significant effect of two-dimensional expansion. Existence of the high density strata at the bottom of the ground layer in the fourth power case significantly increased the maximum pressure at the ground, and produced higher detonation wave velocity.

Using a variable density layer, one can reach a combination of pressure and velocity conditions outside of Chapman-Jougett limitations. The range of conditions that can be obtained in the variable density system and its parametrics of that system needs a more systematic study. In this article, we introduced only the mathematical formulation and numerical simulation method validated for the range of conditions of interest. In addition, we have given some examples of its application for two-dimensional simulations. However, this methodology should be linked to an experimental study for a more in-depth analysis of the phenomenology discussed here.

REFERENCES

- Eidelman, S., Timnat, Y.M., and Burcat, A., (1976). "The Problem of a Strong Point Explosion in a Combustible Medium," 6th Symp. on Detonation, Coronado, CA, Office of Naval Research, 590.
- Burcat, A., Eidelman, S., and Manheimer-Timnat, Y., (1978). "The Evolution of a Shock Wave Generated by a Point Explosion in a Combustible Medium," Symp. of High Dynamic Pressures (H.D.P.), Paris, 347.
- Oved, Y., Eidelman, S., and Burcat, A., (1978). "The Propagation of Blasts from Solid Explosives to Two-Phase Medium," Propellants and Explosives, 3, 105.
- Eidelman, S., and Burcat, A., (1980). "The Evolution of a Detonation Wave in a Cloud of Fuel Droplets; Part I, Influence of the Igniting Explosion," AIAA Journal, 18, 1103.
- Liu, J.C., Kauffman, C.W., and Sichel, M., (1990). "The Lateral Interaction of Detonating and Detonable Mixtures," (Private communication).
- Kuo, K., (1990). "Principles of Combustion," John Wiley and Sons, Inc.
- Kauffman, C.W., et al., (1979). "Shock Wave Initiated Combustion of Grain Dust," Symposium on Grain Dust, Manhattan, KS.
- Eidelman, S., and Burcat, A., (1981). "Numerical Solution of a Non-Steady Blast Wave Propagation in Two-Phase ('Separated Flow') Reactive Medium," J. Comput. Physics, 39, 456.
- Reinecke, W.G., and Waldman, G.D., (1975). "Shock Layer Shattering of Cloud Drops in Reentry Flight," AIAA Paper 75-152.
- Eidelman, S., and Burcat, A., (1980). "The Mechanism of Detonation Wave Enhancement in a Two-Phase Combustible Medium," 18th Symposium on Combustion, The Combustion Institute, Waterloo, Ontario, Canada.
- "Engineering Design Handbook, Explosives Series, Properties of Explosives of Military Interest," AMC Pamphlet, AMCP 706-7177, 1971.
- Drake, R.M., Jr., (1961). "Discussions on G.C. Vliet and G. Leppert: Forced Convection Heat Transfer from an Isothermal Sphere to Water," Journal of Heat Transfer, 83, 170.
- Schlichting, H., (1983). "Boundary Layer Theory," 7th ed. McGraw-Hill.
- Cowan, R.D., and Fickett, W., (1956). "Calculation of the Detonation Products of Solid Explosives with the Kistiakowsky-Wilson Equation of State," Journal of Chemical Physics, 24, 932.
- Mader, C.L., (1979). "Numerical Modeling of Detonation," University of California Press, Ltd. London, England.
- Gordon, S., and McBride, B.J., "Computer Program for Calculations of Complex Chemical Equilibrium Compositions, Rocket Performance, Incident and Reflected Shocks and C-J Detonations," NASA SP-273, 1976 Revision.
- Eidelman, S., Collela, P., and Shreeve, R.P., (1984). "Application of the Godunov Method and Its Second Order Extension to Cascade Flow Modelling," AIAA Journal, 22, 10.
- Stanukovitch, K.P., (1975). "Physics of Explosion" (in Russian), Nauka.
- Wiedermann, A., (1990). "An Evaluation of Bimodal Layer Loading Effects," IITRI Report, February.

D[m/sec] - Detonation wave velocity,
 P_{CJ} [Pa] - Pressure at Chapman-Jouguet Point
 P_p [Pa] - Peak pressure; ρ_p [kg/m³] - Peak density

RDX Density (kg/m ³)	Parameters	Present Calculation	Expt'l Ref. 1	Tiger Calculation Ref. 2	BKW Calculation Ref. 1	Soviet Experiments Ref. 3
1000 kg/m ³	D	6155	5981		6128	
	P_{CJ}	1.220×10^{10}			1.08×10^{10}	1.00×10^{10}
	P_p	2.57×10^{10}				
	ρ_p	1936				
860 kg/m ³	D	6031		5900		
	P_{CJ}	0.986×10^{10}		0.88×10^{10}		0.82×10^{10}
	P_p	1.95×10^{10}				
	ρ_p	1722				
466 kg/m ³	D	4800		4500		
	P_{CJ}	0.379×10^{10}		0.30×10^{10}	0.3×10^{10}	
	P_p	0.625×10^{10}				
	ρ_p	924				
250 kg/m ³	D	4049		3600		
	P_{CJ}	0.2478×10^{10}		0.13×10^{10}		
	P_p	0.4538×10^{10}				
	ρ_p	552				
100 kg/m ³	D	3495				
	P_{CJ}	0.5013×10^9				
	P_p	0.7658×10^9				
	ρ_p	220				
0.75 kg/m ³	D	1622	1410 ^a	1870 ^a		
	P_{CJ}	0.25×10^7	0.284×10^{7a}	0.26×10^{7a}		
	P_p	0.484×10^7				
	ρ_p	8				

Ref. 1 - Mader, C., "Numerical Modeling of Detonation." (University of California Press, Ltd., 1979), p. 47.
 Ref. 2 - Wiedermann, A., "An Evaluation of Bimodal Layer Loading Effects," IITRI Report, Feb., 1990.
 Ref. 3 - Stankovitch, K.P., "Physics of Explosion" (in Russian), Nauka, 1975.

Table 1. One Dimensional Validation Result.

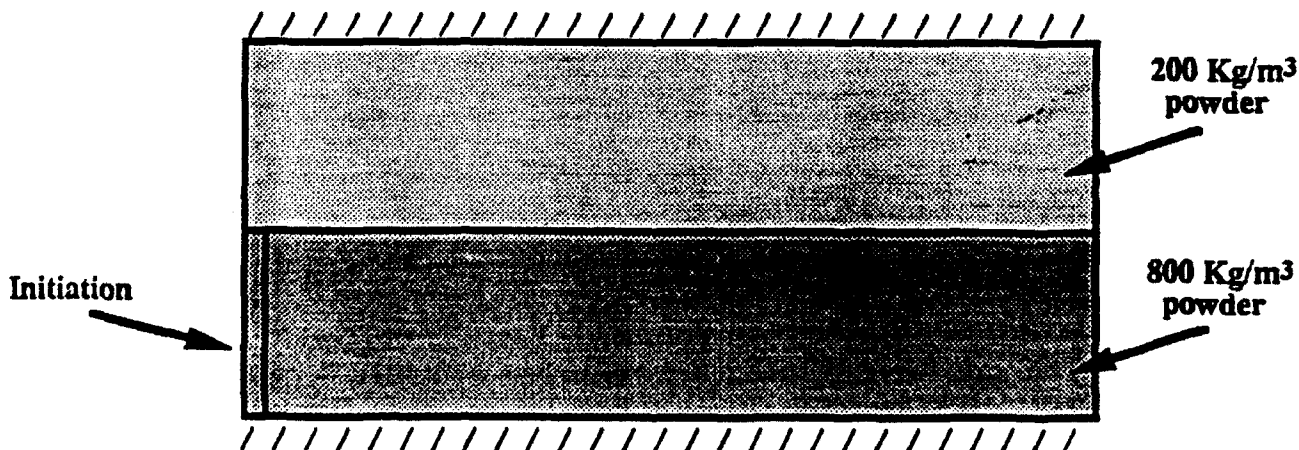


Figure 1. Setup for the two-layer detonation simulation problem.

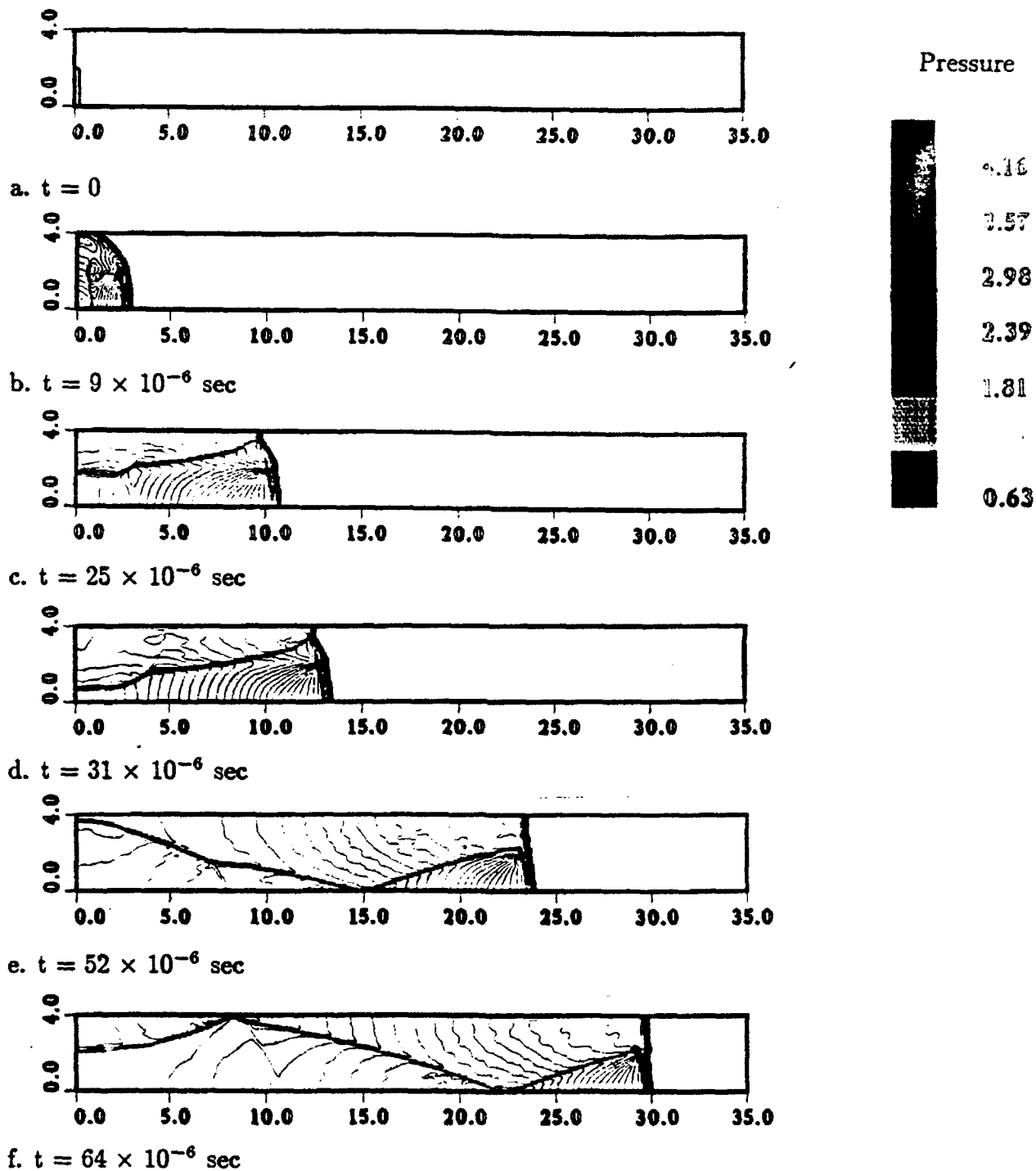


Figure 2. Initiation and propagation of the detonation wave in a two layers system. Only lower layer is initiated. Pressure contours.

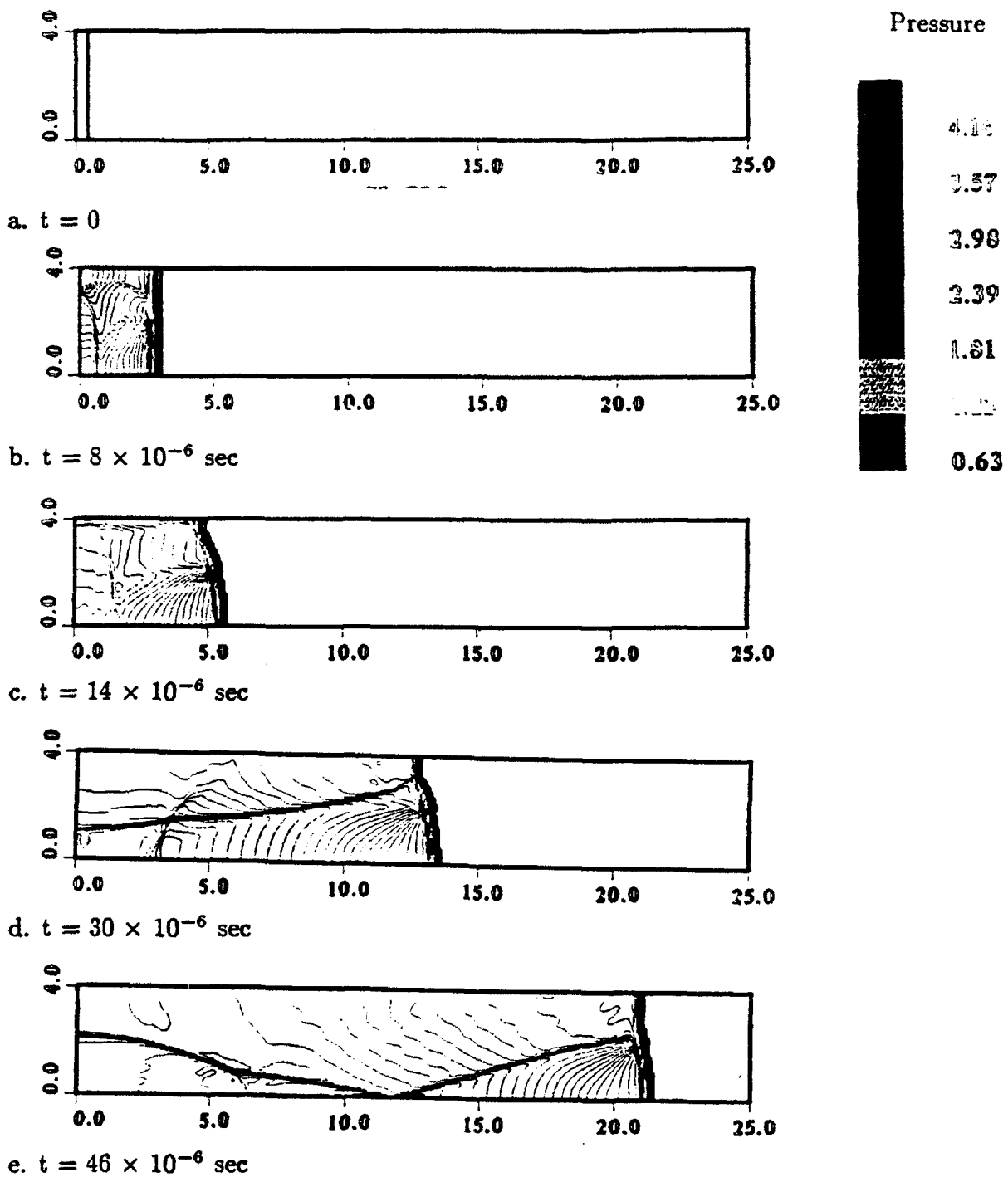


Figure 3. Initiation and propagation of the detonation wave in a two layers system. Both layers are initiated. Pressure contours.

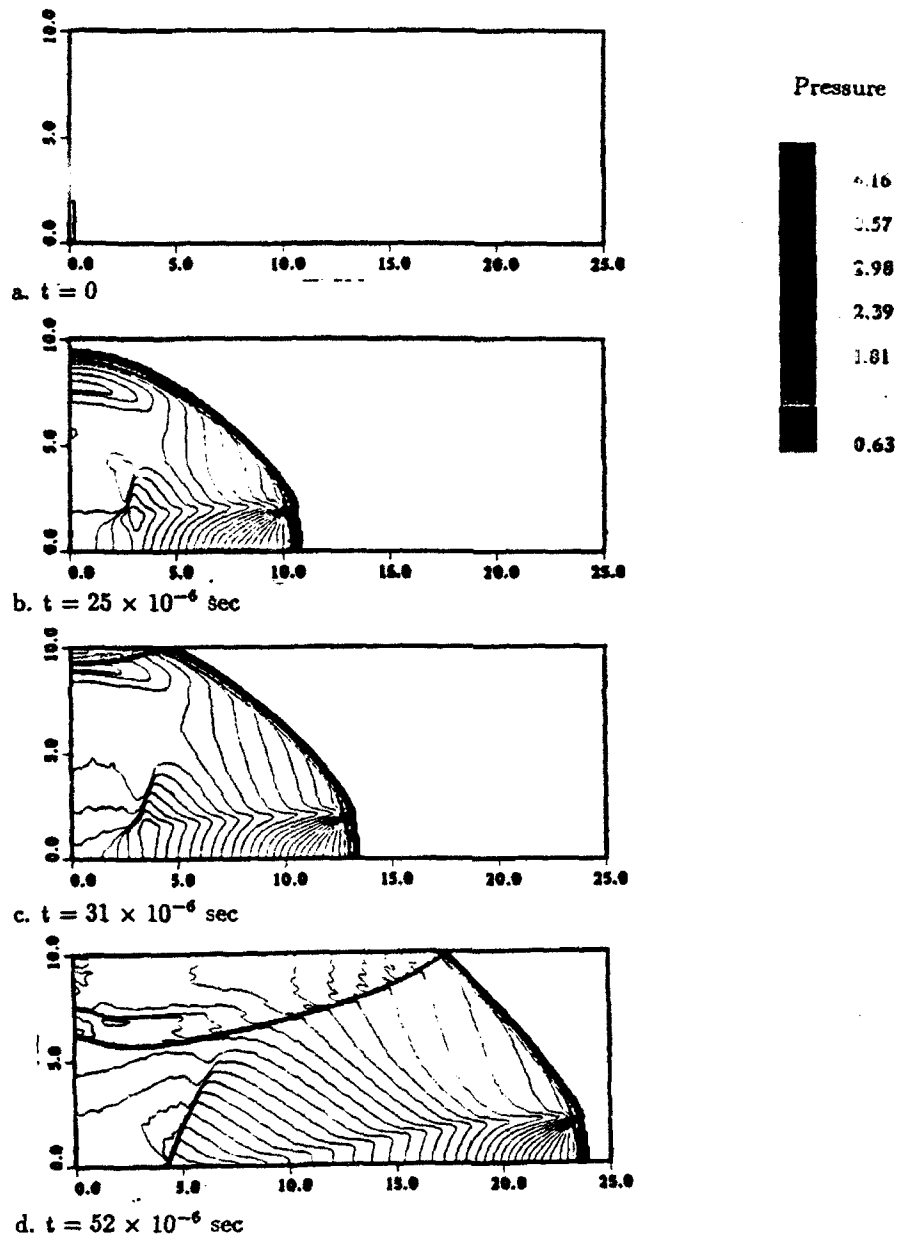


Figure 4. Propagation of the detonation wave in a system with different thickness of explosive layers. Pressure contours.

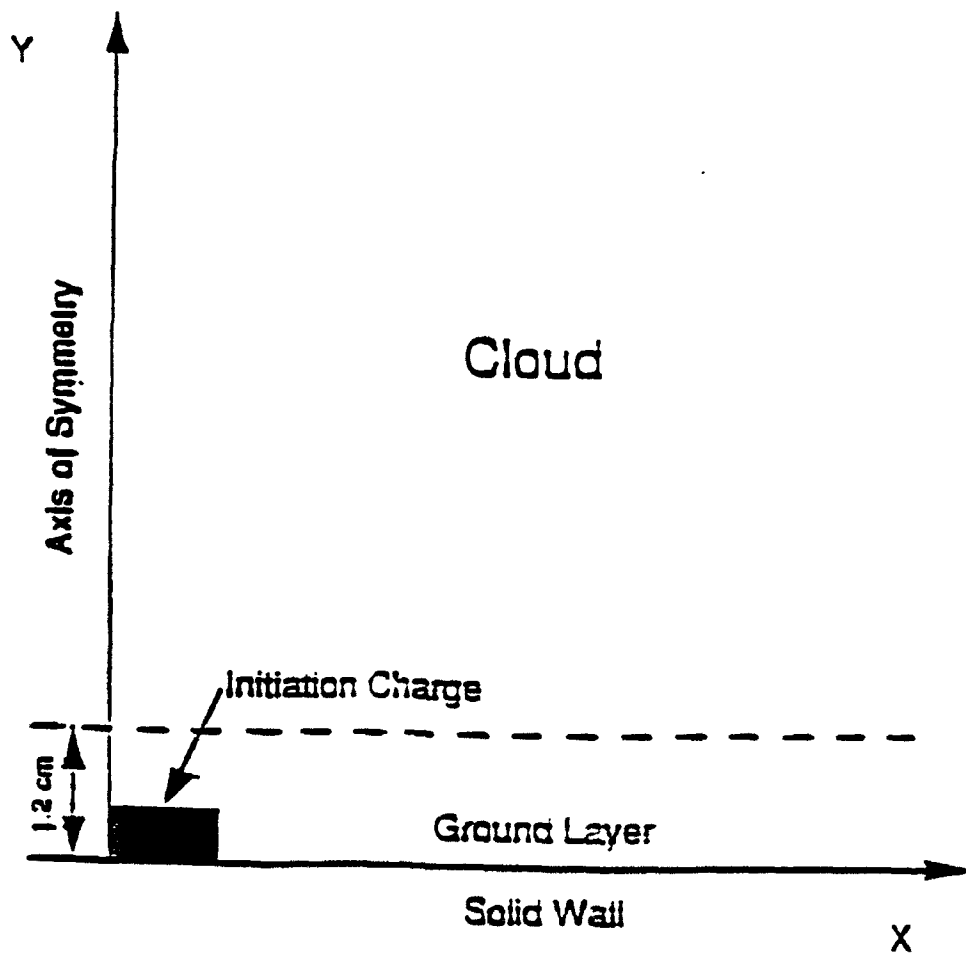


Figure 5. Computational domain and boundary conditions.

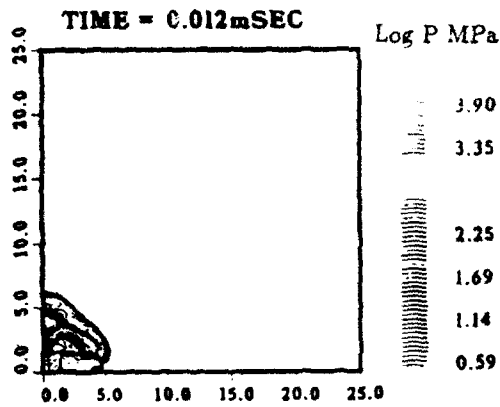


Fig. 6a. Pressure.

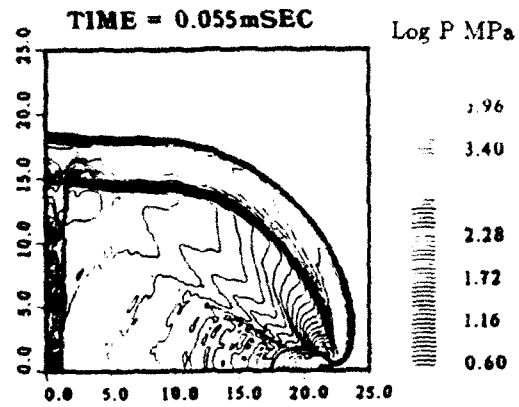


Fig. 6d. Pressure.

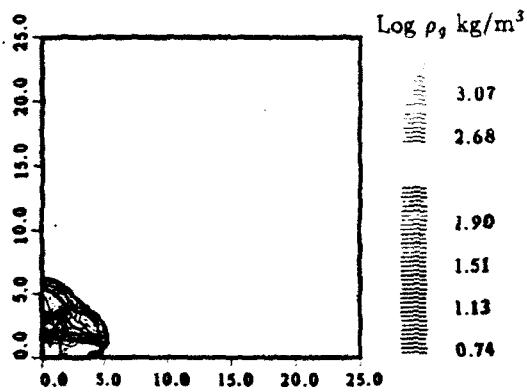


Fig. 6b. Gas Density.

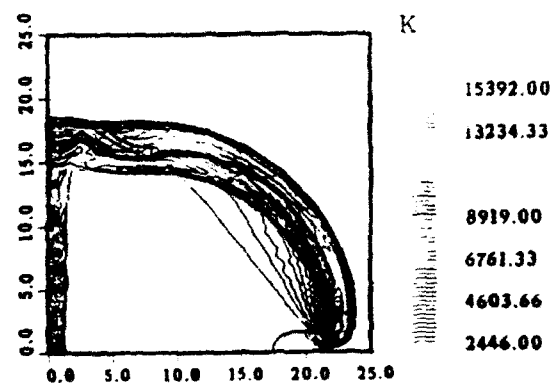


Fig. 6e. Temperature.

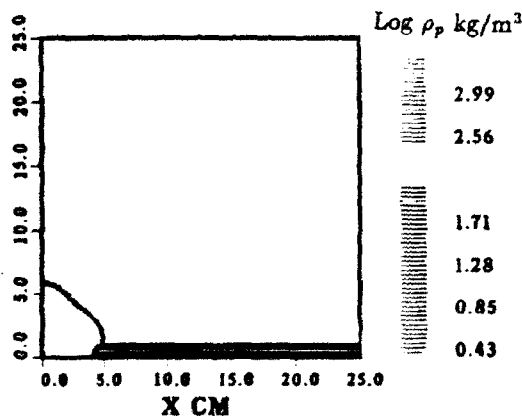


Fig. 6c. Particle Density.

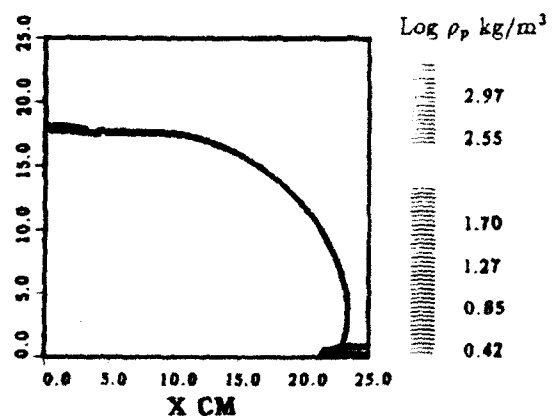


Fig. 6f. Particle Density.

Figure 6. Fourth power layer distribution. Maximum density in the layer 800 kg/m^3 . Density in the cloud 0.75 kg/m^3 . Time 0.012 msec and 0.055 msec after initiation.



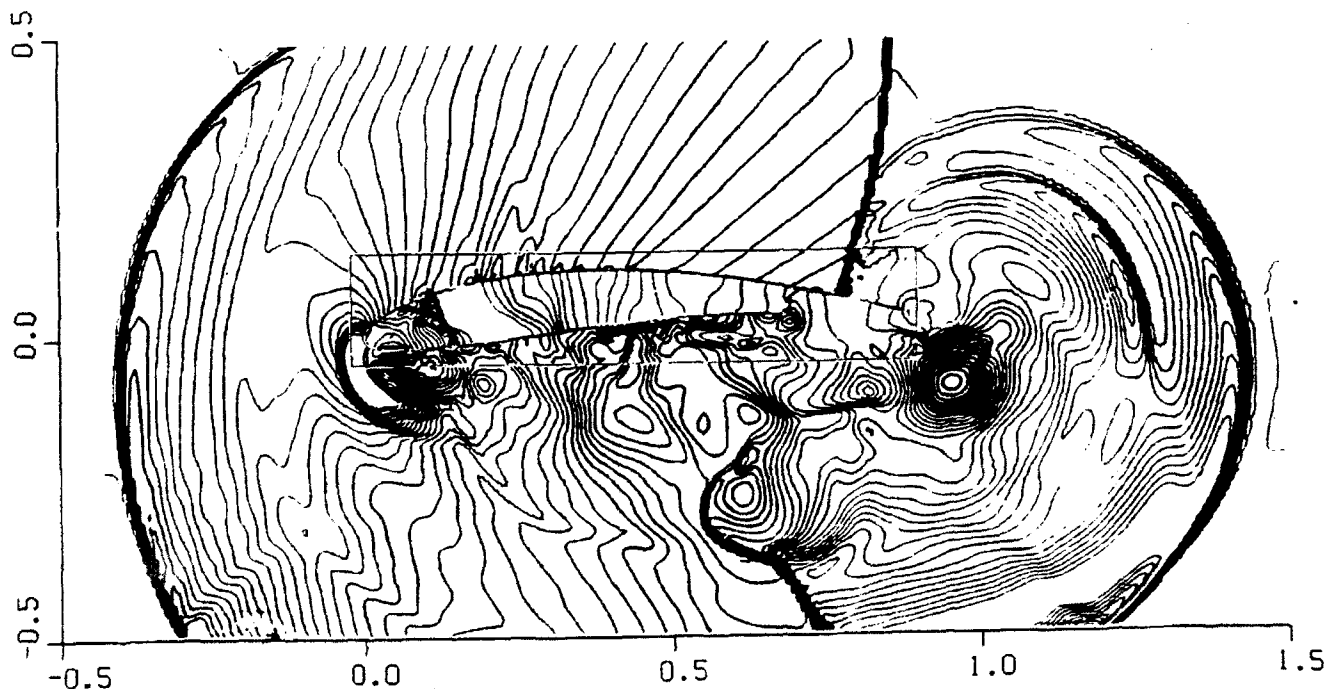
AIAA 92-0392

**A Parametric Study of the Air-Breathing
Pulsed Detonation Engine**

S. Eidelman, I. Lottati and W. Grossmann

Science Applications International Corporation

McLean, VA 22102



**30th Aerospace Sciences
Meeting & Exhibit**
January 6-9, 1992 / Reno, NV

A PARAMETRIC STUDY OF AIRBREATHING PULSED DETONATION ENGINE

S. Eidelman, I. Lottati, and W. Grossmann

Applied Physics Operation
Science Applications International Corporation
1710 Goodridge Drive, McLean, Virginia 22102

Abstract

The airbreathing Pulsed Detonation Engine (PDE) is analyzed by direct simulations of its cycle using Computational Fluid Dynamics. We describe a new CFD methodology of composite structured/unstructured grids, which is used for detailed analysis of the PDE performance. This performance is analyzed for a unique engine geometry in which the PDE is located in a wing section. Examination of the key processes in the PDE device shows that the largest portion of its thrust is produced during the very short time interval when the detonation wave reflects from the thrust wall, and that detonation cycle frequency up to 200Hz is feasible. We conclude that the PDE type devices can compete with small diameter turbojet engines in performance characteristics while surpassing them in simplicity of design, flexibility of geometrical configuration, and price.

1. Introduction

Our first reports on the airbreathing Pulsed Detonation Engine (PDE) concept¹⁻⁵ described a systematic series of parametric studies of the PDE via computational fluid dynamics (CFD). They also detailed an analysis of engine performance over a wide range of flight regimes, including subsonic and supersonic flows and physical geometries with various nozzle and air inlets. Additionally, static table top experiments¹ demonstrated that the principle of pulsed or repetitive detonation can be successfully applied. To date, our results indicate that practical engines for certain vehicles can be conceptualized and designed with the information that has already been generated from the studies. Specifically, our studies have shown that the PDE is an excellent candidate for the primary propulsion source for small aerodynamic vehicles that operate over the flight envelope, $0.2 < M < 2.0$. Further, our analysis of the simulation results indicates that the PDE is a high thrust-to-weight ratio device. The predicted performance places the PDE propulsion concept in a strongly competitive position compared with present day small turbojets. The PDE concept has the added attractiveness of rapid variable thrust control, no moving parts and the potential for low cost manufacturing. The PDE concept is scalable over a wide range of engine sizes and thrust levels.⁴ For example, it is theoretically possible to produce PDE engines on the order of one to several inches in diameter and thrusts on the order of pounds, as well as devices that provide thousands of pounds thrust. One of the unique features of the PDE that will be explored in this paper is its geometric flexibility. All the configurations of the engine that we have examined in previous papers had an axisymmetric geometry. However, the PDE concept allows a

tremendous flexibility in engine geometry. In this paper we will investigate the possibility of fitting a PDE detonation chamber into a section of a conventional wing. One of the obvious advantages of this design is reduction of the drag and weight penalty; other advantages can be associated with stealth quality of the Wing-PDE design.

The parametric studies to date were made possible by the development of a new generation of CFD tools. These tools have allowed us to accurately simulate the details of the complex nonlinear time dependent processes. In this article, we used a new algorithm implemented on a composite structured/unstructured grid. This algorithm combines the flexibility of describing complicated geometries characteristic of the unstructured triangular grids with the computational efficiency of the structured grids. A brief description of the CFD methods employed in our studies is given in Section 3.

2. The Pulsed Detonation Engine Concept

A detonation process, due to the very high rate of reaction, leads to a propulsion concept in which the constant volume process can be fully realized. In detonative combustion, the strong shock wave, which is part of the detonation wave, acts like a valve between the detonation products and fresh charge. The speed of the detonation wave is about two orders of magnitude higher than the speed of a typical deflagration. This allows the design of propulsion engines with a very high power density. Each detonation has to be initiated separately by a fully controlled ignition device with a wide range of variable cycle frequencies. A physical restriction dictating the range of detonation frequency arises from the rate at which the fuel/air mixture can be introduced into the detonation chamber. This also means that a device based on a detonative combustion cycle can be scaled and its operating

parameters can be modified for a range of required output conditions.

There have been numerous attempts to take advantage of detonative combustion for engine applications,^{6,7,8} the most recent and successful which was carried out at the Naval Postgraduate School¹ (NPS) by Helman et al. During this study, several fundamentally new elements were introduced to the concept that distinguished the NPS research device from previous studies. First, it is important to note that the NPS experimental apparatus was the first successful self-aspirating air breathing detonation device. Intermittent detonation frequencies of 25 Hz were obtained, which was in phase with the fuel mixture injection through the timed fuel valve opening and spark ignition. The feasibility of intermittent injection was established. Pressure measurements showed conclusively that a detonation process occurred at the frequency chosen for fuel injection. Further, self-aspiration was shown to be effective. Finally, the effectiveness of a primary detonation as a driver for the main detonation was clearly demonstrated. Although the NPS studies were abbreviated, many of the technical issues considered to be essential for efficient intermittent detonation propulsion were addressed with positive results.

The generic device we considered in our previous studies²⁻⁵ is a small engine shown in Figure 1, which is a schematic of the basic detonation chamber attached to the aft end of a generic aerodynamic vehicle. In the current study, we considered a Wing-PDE configuration that will be described below; however, for the sake of simplicity we will describe the basics of the PDE concept using the illustration in Figure 1. For the engine configuration shown in this figure, the combustible gas mixture is injected at the closed end of the detonation chamber and a detonation wave, initiated at the aft end of the detonation chamber, propagates through the mixture. The main portion of the thrust is produced by the detonation wave in a very short period of time as it impinges on the thrust wall. After the detonation wave has reflected from the thrust wall, the detonation products will vent from the volume of the detonation chamber through the open aft end of the chamber and air inlets shown in Figure 1. Then the chamber volume will be filled with the fresh combustible gas mixture and the process will be repeated with the frequency of 100 to 200Hz. A key issue in the pulsed detonation engine concept is the design of the main detonation chamber. The detonation chamber geometry determines the propulsion efficiency and the duration of the cycle (frequency of detonations). Since the fresh charge for the generic engine is supplied from the external flow field, the efficiency of the engine depends on the interaction of the surrounding flow with the internal flow dynamics. The range of the physical pro-

cesses requiring simulation in order to model the complex flow phenomena associated with the detonation engine performance is very broad. These processes include 1) initiation and propagation of the detonation wave inside the chamber; 2) expansion of the detonation products from the chamber into the air stream around the chamber at flight Mach numbers; 3) fresh air intake from the surrounding air into the chamber; 4) the flow pattern inside the chamber during post-exhaust pressure buildup which determines the strategy for mixing the next detonation charge; and 5) strong mutual interaction between the flow inside the chamber and surrounding the engine.

All of these processes are interdependent, and interaction and timing are crucial to engine efficiency. Thus unlike simulations of steady state engines, the phenomena described above cannot be evaluated independently. The need to resolve the flow regime inside the chamber and account for nozzles, air inlets, etc., and at the same time resolve the flow outside and surrounding the engine where the flow regime varies from high subsonic locally transonic and supersonic, makes it a challenging computational problem.

The single most important issue is to determine the timing of the air intake and mixing of the fresh charge leading to repetitive detonations. It is sufficient to assume inviscid flow for the purpose of simulating the expansion of the detonation products and fresh air intake. This assumption makes the numerical simulation of the PDE flow phenomena somewhat easier than using a fully viscous flow model. For the size of the generic device studied in this work, the effects of viscous boundary layers are negligible, with the exception of possible boundary layer effects on the valve and inlet geometries discussed subsequently.

3. Computational Method Used in the Study

The basic computational tool used for our studies is the AUGUST (Adaptive Unstructured Godunov Upwind Second Order on Triangular Grids) code, described in detail by Lottati et al.^{9,10} This code provides a method for solving the Euler equations of gas dynamics on unstructured grids with arbitrary connectivity. The formulation is based on a second order Godunov method.¹¹ For the current study, the AUGUST code has been implemented on a composite structured/unstructured grid. The combined structured/unstructured method is a much more efficient approach to domain decomposition than the separate application of each method. In the following discussion, we show that the results of applying this technique to the complex problem of the external/internal reactive flow typical for the PDE engine show complex wave patterns propagating seamlessly through interfaces between structured/unstructured grids without reflections or distortions. This new approach provides ultimate flexibility

in domain decomposition with maximum code efficiency.

Introduction

Structured rectangular grids allow the construction of numerical algorithms that perform an efficient and accurate integration of fluid conservation equations. The efficiency of these schemes results from the extremely low storage overhead needed for domain decomposition and the efficient and compact indexing that also defines domain connectivity. These two factors allow code construction based on a structured domain decomposition that can be highly vectorized and parallelized. Integration in physical space on orthogonal and uniform grids produces the highest possible accuracy of the numerical algorithms. The disadvantage of structured rectangular grids is that they cannot be used for decomposition of computational domains with complex geometries.

The early developers of computational methods realized that, for many important applications of Computational Fluid Dynamics (CFD), it is unacceptable to describe curved boundaries of the computational domain using the stair-step approximation available with the rectangular domain decomposition technique. The techniques of boundary-fitted coordinates were developed to overcome this difficulty. With these techniques, the computational domain is decomposed on quadrilaterals that can be fitted to the curved domain. The solution is then obtained in the physical space using the geometrical information defining the quadrilaterals, or in the computational coordinate system that is obtained by transformation of the original domain into a rectangular domain. The advantage of this technique is that it employs the same indexing method as the rectangular structured domain decomposition methods that also serve to define domain connectivity. The boundary fitted coordinated approach leads to efficient codes, with approximately a 4:1 penalty in terms of memory requirement per cell as compared with rectangular domain decomposition. However, this approach is somewhat restricted in its domain decomposition capability, since distortion or large size variations of the quadrilaterals in one region of the domain lead to unwanted distortions or increased resolution in other parts of the domain. An example of this is the case of structured body fitted coordinates that are used for simulations of flows over a profile with sharp trailing edges. In this case, increased resolution in the vicinity of the trailing edge leads to increased resolution in the whole row of elements connected to the trailing edge elements.

The most effective methods of domain decomposition developed to overcome this disadvantage are those using unstructured triangular grids. These methods were developed to cope with very complex computational domains. The unstructured grid method, while efficient and powerful in domain decomposition, results in codes

that must store large quantities of information defining the grid geometry and connectivity, and have large computational and storage overheads. As a rule, an unstructured grid code requires greater storage by a factor of 10, and will run about 20 times slower when compared on a per cell per iteration basis with a structured rectangular code.

Unstructured grid methods are used to their best advantage when combined with grid adaptivity. This feature usually allows dynamic decomposition of the computational domain subregions, thus leading to an order of magnitude reduction in the number of cells for some problems, as compared to the unstructured grid without this adaptive capability. However, this advantage is highly dependent on the problem solved. Adaptive unstructured grids have an advantage over the unadaptive unstructured domain decomposition if the area of high resolution domain decomposition is less than one tenth of the global area of the computational domain. This explains the fact that while the adaptive unstructured method may be extremely effective for solutions with multiple shock waves in complex geometries, it becomes extremely inefficient when high resolution is needed in a substantial area of the computational domain.

Our approach to domain decomposition combines the structured and unstructured methods for achieving better efficiency and accuracy. Using this method, structured rectangular grids are used to cover most of the computational domain, and unstructured triangular grids are used only to patch between the rectangular grids (Figure 2), or to conform to the curved boundaries of the computational domain (Figure 3). In these figures, an unstructured triangular grid is used to decompose the regions of the computational domain that have a simple geometry.

Our paper will illustrate the performance gains achieved from the use of this composite grid decomposition approach. We apply the Second Order Godunov method¹¹ to solve the Euler equations on both structured and unstructured sections of the grid.

Mathematical Model and Integration Algorithm

We consider a system of two-dimensional Euler equations written in conservation law form as:

$$\frac{\partial U}{\partial t} + \frac{\partial F}{\partial x} + \frac{\partial G}{\partial y} = 0 \quad (1)$$

where

$$U = \begin{vmatrix} \rho \\ \rho u \\ \rho v \\ \rho e \end{vmatrix}, \quad F = \begin{vmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ u(e + p) \end{vmatrix}, \quad G = \begin{vmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ v(e + p) \end{vmatrix}$$

Here u, v are the x, y velocity vector components, p is the pressure, ρ is the density and e is total energy of the fluid. We assume that the fluid is an ideal gas and the pressure is given by the equation-of-state.

$$p = (\gamma - 1)\left(e - \frac{\rho}{2}(u^2 + v^2)\right) \quad (2)$$

where γ is the ratio of specific heats and typically taken as 1.4 for air. It is assumed that an initial distribution of the fluid parameters is given at $t = 0$, and the boundary conditions defining a unique solution are specified for the computational domain.

The system of governing equations in Eq. (1) can be written as

$$\frac{\partial U}{\partial t} + \nabla \cdot Q = 0 \quad (3)$$

where Q represents the convective flux vector. By integrating Eq. (3) over space and using Gauss' theorem, the following expression is obtained

$$\frac{\partial}{\partial t} \int_{\Omega} U dA + \oint_{\partial\Omega} Q dl = 0 \quad (4)$$

where $dl = n d\mathcal{L}$, n is the unit normal vector in the outward direction, and $d\mathcal{L}$ is a unit length on the boundary of the domain. The variable Ω is the domain of computation and $\partial\Omega$ is the circumference boundary of this domain.

Equation (4) can be discretized for each element (cell) in the domain

$$\frac{(U_i^{n+1} - U_i^n)}{\Delta t} A_i = \sum_{j=1}^M Q_j^n n_j \Delta L_j \quad (5)$$

where A_i is the area of the cell; Δt is the marching time step; U_i^{n+1} and U_i^n are the primitive variables at the center of the cell at time n and at the update $n = 1$ time step; Q_j is the value of the fluxes across the M boundaries on the circumference of the cell where n_j is the unit normal vector to the boundary edge j , and ΔL_j is the length of the boundary edge j . The fluxes Q_j^n are computed by applying the Second Order Godunov algorithm, and Eq. (5) is used to update the physical primitive variables U_i according to computed fluxes for each marching time step Δt . The marching time step is subjected to the CFL (Courant-Fredrichs-Lewy) constraint.

We seek a solution to the system of Eq. (1) in the computational domain, which is decomposed in part into triangles with arbitrary connectivity and in part into rectangles using a logically structured grid. We use the advantage of the unstructured grid¹²⁻¹⁵ to describe the curved boundary of the computational domain and areas that need increased local resolution; this covers 10% of

the total computational domain. The structured grid occupies the remaining 90% of the computational domain in our example. The numerical technique for solving Euler's equation on an unstructured grid is described in Refs. 9-10, and the technique for the structured grid is described in Ref. 11. These numerical techniques apply some of the ideas that were introduced in Refs. 17-18. The structured and unstructured codes apply the center-based formulation, i.e., the primitive variables are defined in the center of the cell, which makes the cell the integration volume, while the fluxes are computed across the edges of the cell. The basic algorithmic steps of the Second Order Godunov method can be defined as follows:

1. Find the value of the gradient at the baricenter of the cell for each gas dynamic parameter U_i ;
2. Find the interpolated values of U at the edges of the cell using the gradient values;
3. Limit these interpolated values based on the monotonicity condition;
4. Subject the projected values to the characteristic's constraints;
5. Solve the Riemann problem by applying the projected values at the two sides of the edges;
6. Update the gas dynamic parameter U according to the conservation equations (1) applying to the fluxes computed and the current time step.

As was advocated in Ref. 9, we prefer the triangle center-based over the vertex-based version of the code. For the same unstructured grid, a triangle-based algorithm will result in smaller control volumes than a vertex-based. In addition, for the Second Order Godunov solver, implementation of the boundary conditions is more straightforward and accurate for the center-based algorithm than in the vertex-based. These two factors, along with the effects of grid connectivity, strongly affect the algorithm accuracy and performance, and are the main reasons for the superiority of the center-based version over the vertex version.

4. Results for Wing-PDE configuration

All of our previous studies considered axisymmetric configurations of the PDE devices. However, because PDE does not have rotating parts, it allows another degree of flexibility that enables us to configure the PDE devices in other than axisymmetric geometries. To illustrate this, we used the inner volume of a section of the wing as a detonation chamber for a PDE device. The schematic of the Wing-PDE geometry considered in this study is shown in Figure 4. We assume that the wing is located in a subsonic air flow stream with $M = 0.8$. The particular wing shape used is the Gastelow cusped supercritical airfoil.¹² Two significant modifications of the original Gastelow airfoil geometry, provision for an inlet

at the leading edge and an outlet nozzle at the trailing edge, allow its use as a PDE device.

In Figure 5, the cross section of the Wing-PDE geometry is shown in the computational domain that is decomposed into structured rectangular and unstructured triangular grids. For clarity, we show only every sixth point of the grid used in simulation. In our simulations we have used a structured grid with 255×131 nodes and an unstructured grid with 7229 nodes. The area covered by the unstructured grid is about 10% of the total area of the computational domain. It is obvious from Figure 5 that the unstructured grid is used in the regions of the computational domain having complex geometry, i.e., wing external and internal surfaces, inlet, and nozzle. The structured rectangular grid is used to cover the rest of the computational domain. As mentioned previously, this method of domain decomposition leads to the most efficient use of computer resources. Our results demonstrate that flow propagates through the interfaces between the triangular unstructured and rectangular structured sections seamlessly.

First, we have to examine the flow pattern for the steady state flow regime of the Wing-PDE device shown in Figure 5. This will also establish the reference values of the aerodynamic drag and lift for this configuration. In Figure 6a, the results are shown in form of the pressure contours for the converged steady state solution for the Wing-PDE configuration in $M=0.8$ external flow stream at zero angle of attack. We can observe in Figure 6a a very complex internal/external flow pattern around Wing-PDE geometry. In addition to the shock wave near the trailing edge on the upper surface of the wing, we can observe two additional shock waves. One is created by the flow exiting from the inner volume of the wing through the nozzle at the trailing edge, and another is created at the flow inlet located under the leading edge. The air flow enters the inner volume of the wing through the inlet and creates a complex flow field with an average pressure of ≈ 1.0 atm. It is easy to improve the flow uniformity in the inner volume of the inlet geometry and geometry of the inner surfaces. However, these aspects of the Wing-PDE design will be considered in future studies; for the purposes of this paper, we examine only the main features of the Wing-PDE configuration. The air flow in the inner volume of the wing create considerable drag. By integrating the pressure over the inner and outer surface of the Wing-PDE configuration, we have calculated the basic air dynamic characteristics of this profile at $M = 0.8$ flow. The following values for the steady state flow:

Lift: $C_l = 0.18$; Drag: $C_d = -0.138$; Pitching Moment: $C_m = 0.034$.

We have assumed that at $t = 0$, the inner volume of the wing is filled with a detonable gas mixture. The

detonation wave is initiated at the aft end of the inner volume of the wing by a planar front. The fuel chosen for these simulations was ethylene. The detonability limits of ethylene in air range from 4% to 12% concentrations by volume, and depend somewhat on temperature and pressure. We assume for the sake of simplicity that the fuel/air ratio is 6% by volume.

In Figure 6b, the pressure contours are shown at $t = 1.18 \times 10^{-4}$ sec. The propagation of the detonation front is planar. However, because of the curved inner walls of the wing, the detonation front reflects from the wall surfaces and the maximum pressure in the reflected waves reach 36.6 atm. However, this level of pressure is observed in a very small area of the detonation front where reflected or colliding transverse waves can cause a local maximum. The detonation wave velocity for this mixture is about 1800 m/sec.

In Figure 6c, the pressure contours are shown at the time $t = 5.24 \times 10^{-4}$ sec, shortly after the detonation front has reflected from the inner surface of the leading edge. Here the maximum pressure was dropped to 12.1 atm, the reflected shock is moving in the direction of the trailing edge, and the expansion of the detonation products through the inlet was created a semicircular shock wave that propagates in the opposite direction to the external flow stream. In Figure 6d at the time $t = 9.5 \times 10^{-4}$ sec, the reflected wave reaches the nozzle at the trailing edge, and expansion of the detonation products through this nozzle creates an additional shock wave that expands in the direction of the flow stream. When the original reflected shock has reached the converging area at the trailing edge, it will partially reflect and send a shock wave towards the inner surface of the leading edge. In Figure 6e, the pressure contours are shown at $t = 1.39 \times 10^{-3}$ sec. Here the shock waves created by the detonation products emitting from the inlet and nozzle of the Wing-PDE device collide, creating a complex flow pattern with two triple point shocks, a vortex at the trailing edge and a complex system of waves propagating through the inner volume of the wing. The maximum pressure observed in Figure 6e at the wave shock wave interaction is 3.2 atm. It is important to note that the numerical method simulates the flow evolution seamlessly through the structured/unstructured grid interfaces.

In Figure 6f, the simulation results are shown at $t = 5.7 \times 10^{-3}$ sec; this corresponds to the end of one cycle for the Wing-PDE configuration. Here we can observe that the flow pattern is very similar to the one in Figure 6a, except for some vortices propagating in the lower right part of the computational domain. The maximum pressure is reached at the leading edges and has the same values as shown in Figure 6a. The inner volume of the wing has a relatively uniform flow pattern

with an average pressure of 0.83 atm. At this time the gaseous mixture in the inner chamber of the wing will be initiated at the trailing edge and the second cycle will get started.

Examination of the details of the flow pattern resulting from a single detonation not only allows evaluation of the timing between the subsequent detonations but also provides important information for optimization of mixing, detonation products expansion, and other gas-dynamic processes related to operation of the PDE cycle. Performance characteristics of the PDE device can be analyzed by integrating in time the forces exerted by pressure on the inner and outer surfaces of the Wing-PDE device. In Figure 7, results for such an integration of the force parallel to the ground as a function of time are shown. Calculation of this force, taking into account the drag and the thrust resulting from the detonation cycle, yields the net thrust force. Figure 7 gives this force for a linear meter of the wing in pounds. In this figure, we observe that the net thrust force is negative before the detonation is initiated, reaches the value of 4.6×10^5 Lb/M during the reflection of the main detonation front from the inner walls of the wing, and quickly decays to its negative initial values that correspond to the drag of the Wing-PDE configuration in $M=0.8$ ambient flow stream. The positive thrust force is produced by the detonation engine in a very short time interval; $\approx 3.0 \times 10^{-4}$ sec.

The time integral of the force shown in Figure 7 is thrust produced by the PDE device. Because of its intermittent operation, we need to assume the cycle frequency to be able to calculate the net thrust. In Figure 8, the results of thrust force integration are shown in the assumption of 200Hz detonation frequency of the Wing-PDE device. Our analysis above of a single cycle shows that this frequency of operation is feasible. In Figure 8, we observe that the maximum thrust of 5000 lb per linear meter of the wing is achieved in the first $\approx 4.0 \times 10^{-4}$ sec after the detonation wave impinges on the thrust wall. This period of time corresponds to the duration of the positive thrust force shown in Figure 7. After this, the thrust will erode because of drag force to the value of 4000 lb at the end of the cycle. The average thrust for the duration of the cycle is 4250 lb per linear meter of the wing.

One of the advantages of the Wing-PDE configuration is that it will generate lift. Our simulations show that the chosen configuration will produce significant lift even at zero angle of attack because of the flow of detonation products. In Figure 9, the net integrated lift is presented as a function of time in the same format as the net thrust shown in Figure 8. The integrated lift shown in Figure 9 is not a linear function of time, as will be the case for the steady state flow regime. Substantial lift is

generated shortly after the detonation products start to expand into the surrounding flow stream. The average lift generated is about 2250 lb per meter of wing length: this is comparable to the net thrust of 4250 lb. Our estimates indicate that about half of this lift is generated by the detonation products and the other half by the free stream flow through the chamber.

5. Conclusions

We have presented a powerful numerical technique for analysis of nonsteady flow over a complex geometrical configuration in the computational domain decomposed on unstructured triangular and structured rectangular grids. Simulations of the Wing-PDE cycle have demonstrated flexibility and efficiency of this technique of domain decomposition. Numerical results show seamless propagations through structured/unstructured grid interfaces of the multiple shocks, contact discontinuities, vortices, rarefaction waves and other complex flow features.

Use of this powerful numerical technique allowed us to examine the operation cycle and propulsion characteristics of the Wing-PDE device. We demonstrated in this study that in principle, the Wing-PDE device can operate with the 200 Hz cycle frequency producing 4250 lb per linear meter of the wing of the net thrust. We examined the Wing-PDE configuration to illustrate the geometric flexibility of this engine. This is an additional advantage to efficiency,³ scalability,⁴ thrust control,³ simplicity, and low cost of this device discussed in our previous publications.

References

- 1 Helman, D., Shreeve, R.P., and Eidelman, S., (1986), "Detonation Pulse Engine," AIAA-86-1683, 24th Joint Propulsion Conference, Huntsville.
- 2 Eidelman, S., W. Grossmann, I. Lottati, (1989) "A Review of Propulsion Applications of the Pulsed Detonation Engine Concept," AIAA 89-2466, AIAA, July 10-12 (to be published in AIAA Journal of Propulsion), Nov - Dec issue.
- 3 Eidelman, S., W. Grossmann, and I. Lottati, (1990), "Computational Analysis of the Pulsed Detonation Engines and Applications," AIAA 90-0460, 28th Aerospace Sciences Meeting, Reno, NV, Jan 8-11.
- 4 Eidelman, S., W. Grossmann, and I. Lottati, "Air-Breathing Pulsed Detonation Engine Concept; A Numerical Study," AIAA/SAE/ASME/ASEE 26th Joint Propulsion Conference, Orlando, FL, July 16-18, 1990.
- 5 Eidelman, W., W. Grossmann, and I. Lottati, "A Propulsion Device Driven by Reflected Shock Waves," 18th International Symposium on Shock Waves, Sendai, Japan, 1991.
- 6 Hoffman, N., (1940), "Reaction Propulsion by Intermittent Detonative Combustion," Ministry of Sup-

- ply, Volkenrode Translation.
- 7 Nicholls, J.A., Wilkinson, H.R., and Morrison, R.B., (1957), "Intermittent Detonation as a Thrust-Producing Mechanism," *Jet Propulsion*, 27, 534-541.
 - 8 Nicholls, J.A., Gullen, R.E. and Ragland K.W., (1966), "Feasibility Studies of a Rotating Detonation Wave Rocket Motor," *J. of Spacecrafts and Rockets*, 3, 893-896.
 - 9 Lottati, I., S. Eidelman, A. Drobot (1990a), "A Fast Unstructured Grid Second Order Godunov Solver (FUGGS)," Paper AIAA 90-0699, 28th Aerospace Sciences Meeting, Reno, NV, Jan 8-11.
 - 10 Lottati, I., S. Eidelman, A. Drobot (1990b), "Solution of Euler's Equations on Adaptive Grids Using a FUGGS," to be published in Proceedings of Second International Conference on Free-Lagrange Methods, Jackson Hole, WY.
 - 11 Eidelamn, S., Collela, P., and Shreeve R.P., (1984) "Application of the Godunov Method and Its Second Order Extension to Cascade Flow Modeling," *AIAA Journal*, v. 22, 10.
 12. A. Jameson, T.J. Baker and N.P. Weatherill, "Calculation of Inviscid Transonic Flow Over a Complete Aircraft," AIAA 24th Aerospace Sciences Meeting, Reno, NV, AIAA Paper 86-0103, January 1986.
 13. R. Löhner, "Adaptive Remeshing for Transient Problems," *Comp. Meth. Appl. Mech. Eng.* 75, 195-214 (1989).
 14. J. Peraire, M. Vahdati, K. Morgan and O.C. Zienkiewicz, "Adaptive Remeshing for Compressible Flow Computations," *J. Comp. Phys.* 72, 449-466, (1987).
 15. D. Mavriplis, "Accurate Multigrid Solution of the Euler Equations on Unstructured and Adaptive Meshes," AIAA 88-3707 (1988).
 16. I. Lottati and S. Eidelman, "Second Order Godunov Solver on Adaptive Unstructured Grids," Proceeding of the 4th International Symposium on Computational Fluid Dynamics, Davis, CA, September 1991.
 17. B. van Leer, "Towards the Ultimate Conservative Difference Scheme, V.A. Second Order Sequel to Godunov's Method," *J. Comp. Phys.* 32, 101-136 (1979).
 18. P. Collela and P. Woodward, "The Piecewise Parabolic Method (PPM) for Gasdynamic Simulations," *J. Comp. Phys.* 54, 174-201 (1984).
 - 19 Dulikravich, D.S., (1982), Private communication.

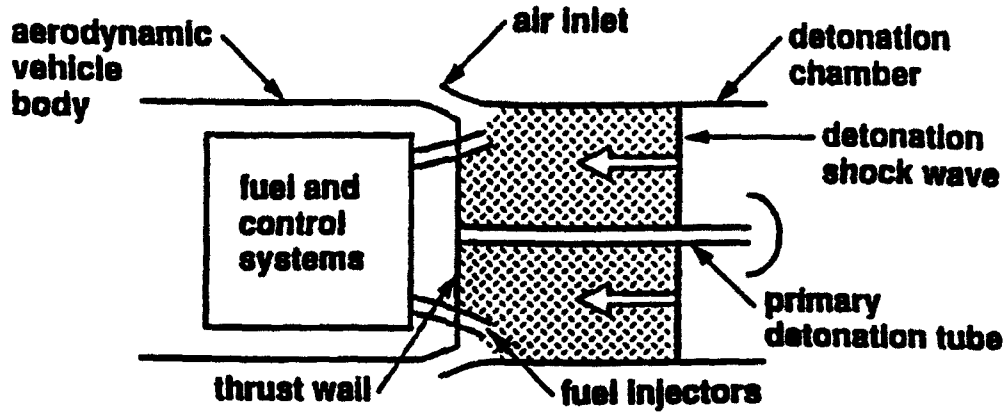


Figure 1. Schematic of the generic PDE showing detonation chamber, inlet, detonation wave, fuel injectors and position relative to an aerodynamic vehicle.

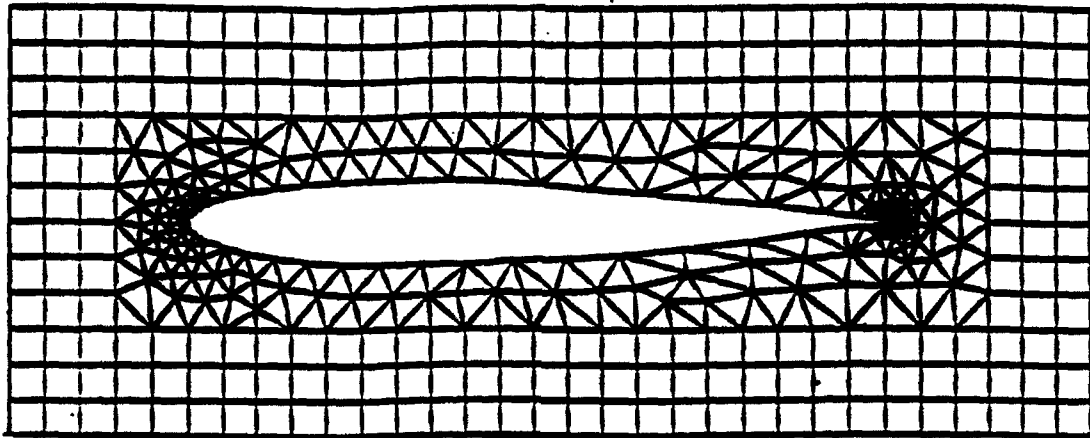


Figure 2. An example of hybrid structured/unstructured domain decomposition.

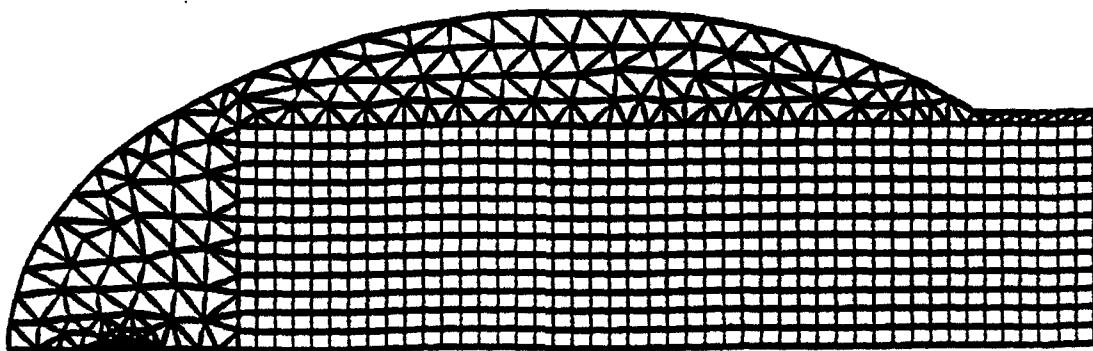


Figure 3. An example of hybrid structured/unstructured domain decomposition.

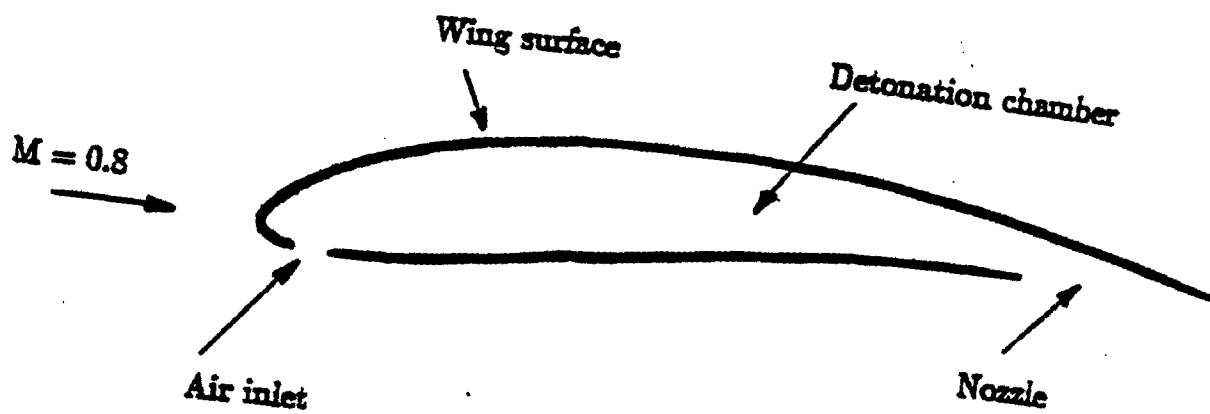


Figure 4. Schematics drawing of the wing-PDE configuration.

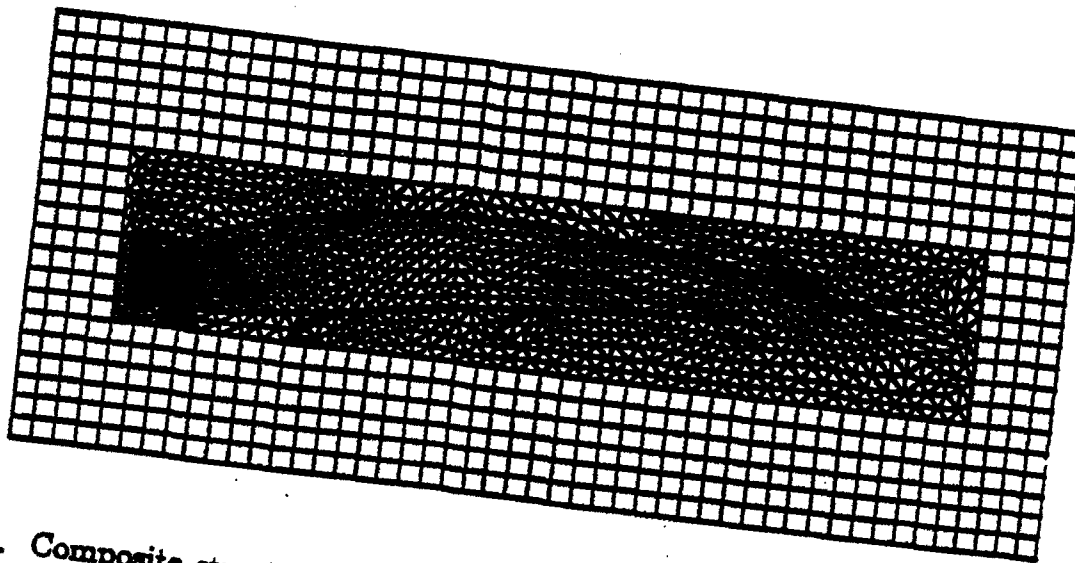


Figure 5. Composite structured/unstructured computational domain for the wing-PDE configuration.

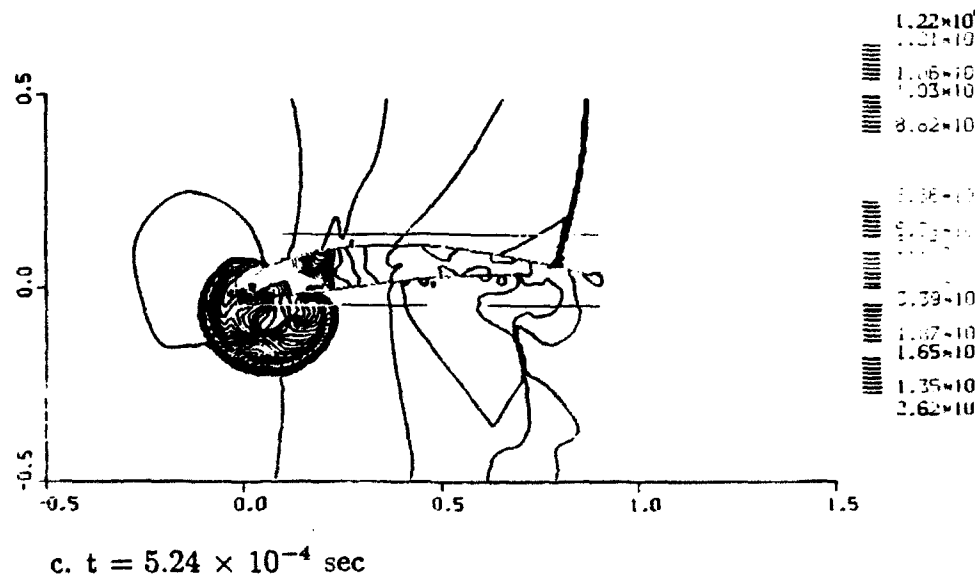
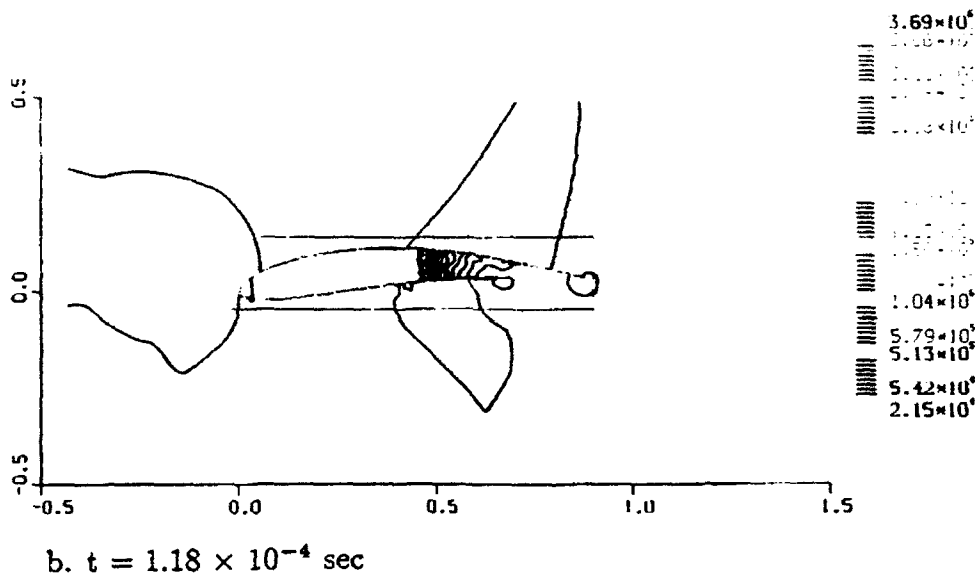
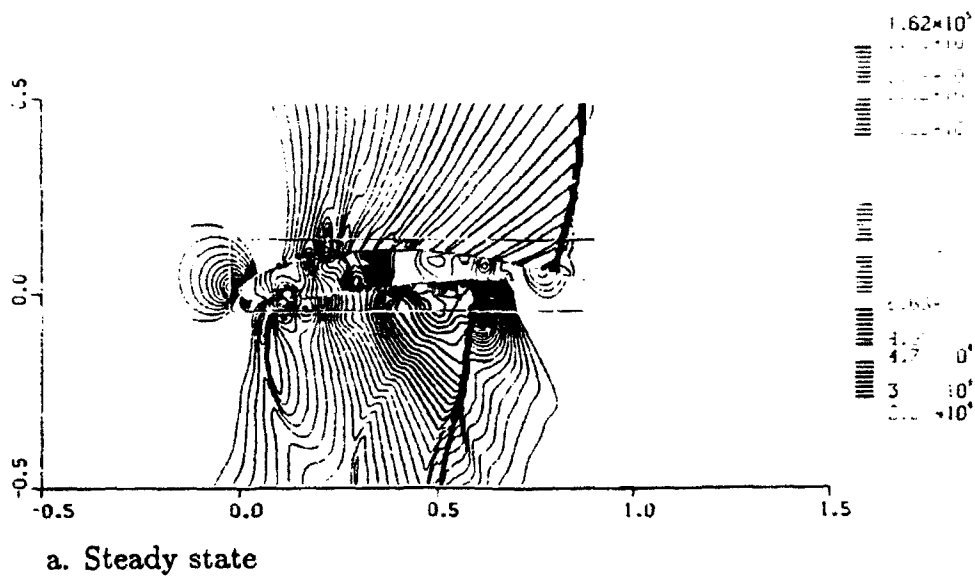
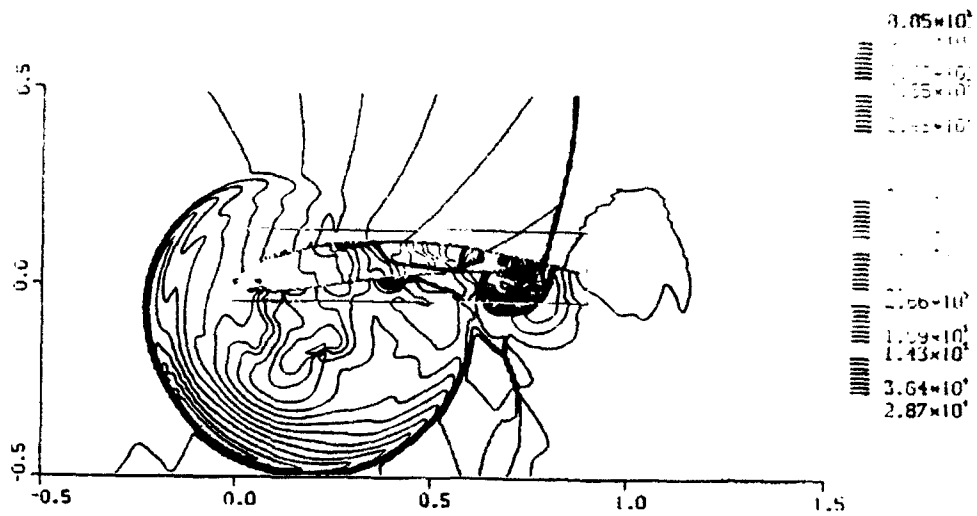
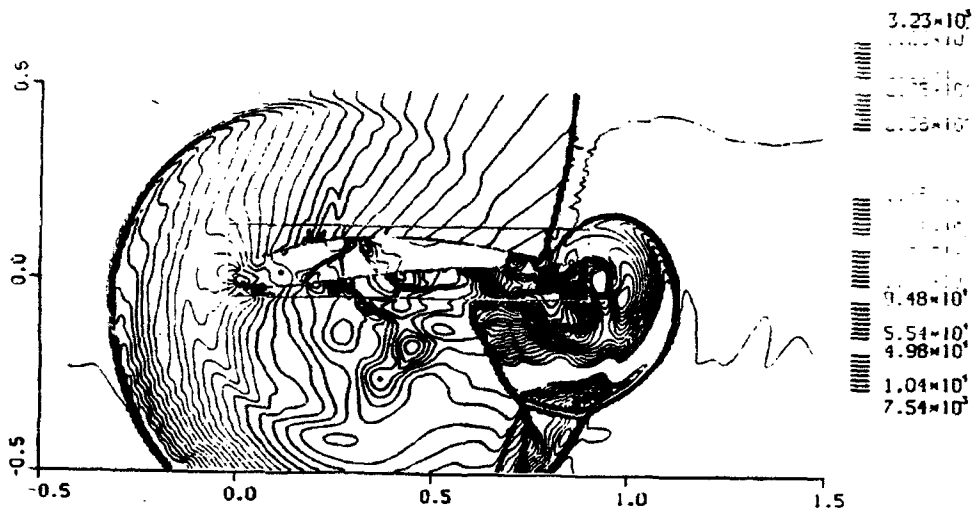


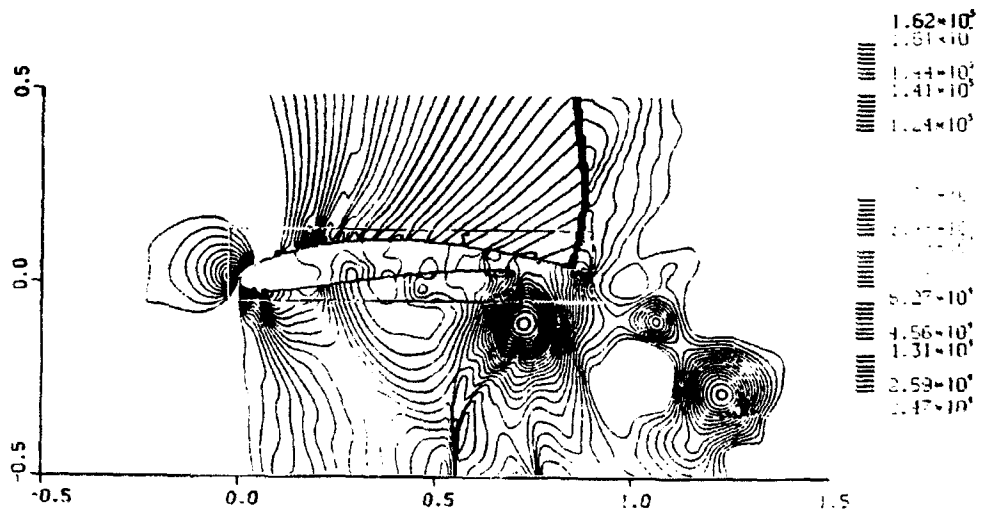
Figure 6. Pressure contours for the various time intervals of the wing-PDE cycle.



d. $t = 9.50 \times 10^{-4}$ sec



e. $t = 1.39 \times 10^{-3}$ sec



f. $t = 5.77 \times 10^{-3}$ sec

Figure 6. Pressure contours for the various time intervals of the wing-PDE cycle (continued).

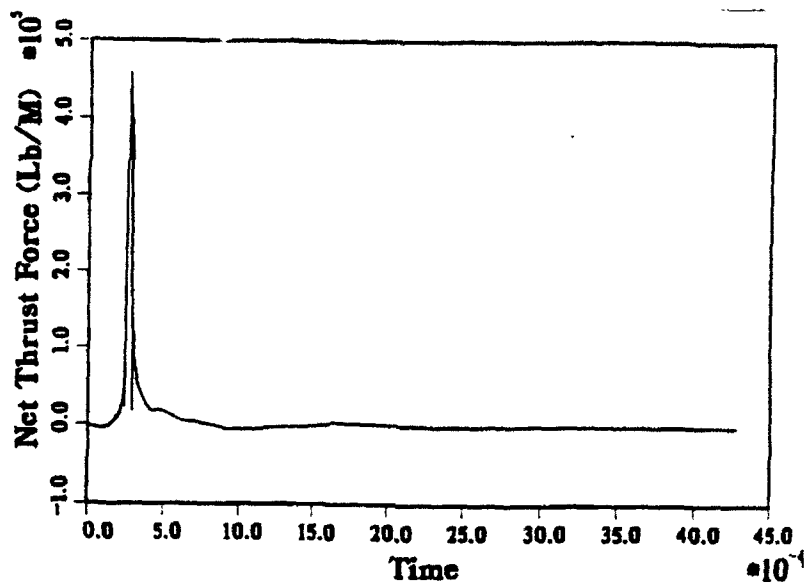


Figure 7. Thrust force as function of time for the wing-PDE device simulation.

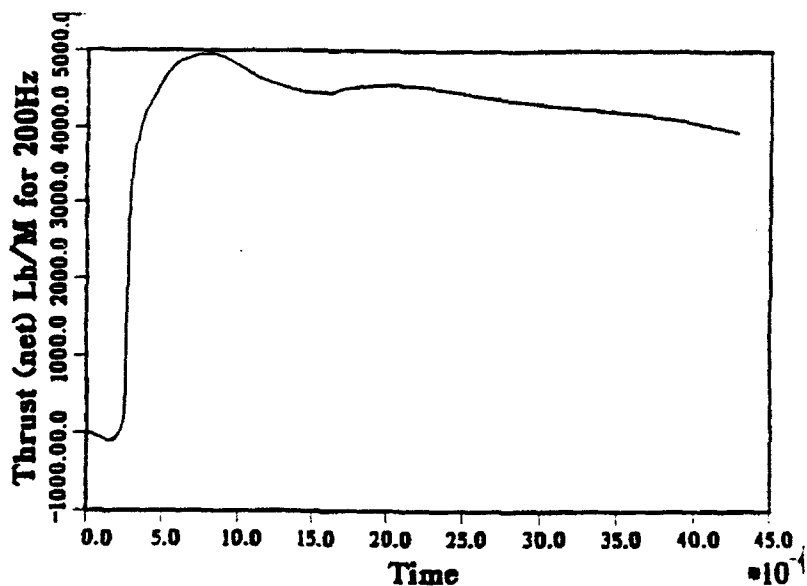


Figure 8. Net integrated thrust for the wing-PDE simulation.

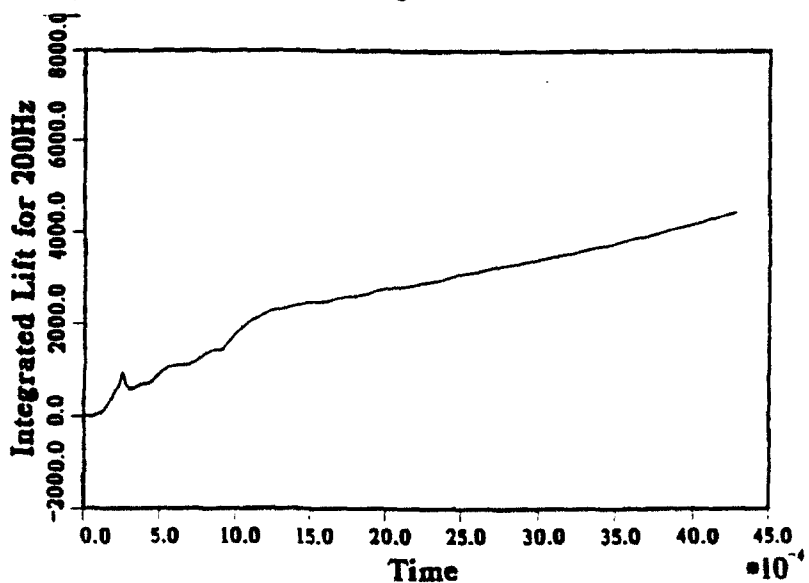


Figure 9. Integrated net lift for the wing PDE simulation.

A Second Order Godunov Scheme on Spatial
Adapted Triangular Grid

Itzhak Lottati and Shmuel Eidelman

Science Applications International Corporation

ABSTRACT

Spatial adaptation procedure for the accurate and efficient solution of unsteady inviscid flow simulation is described. The adaptation procedures were developed and implemented applying a second order Godunov scheme. These procedures involve mesh enrichment/coarsening to either add/remove vertices in high/low gradient regions of the flow, respectively. The goal is to achieve solutions of high spatial accuracy at minimal computational cost. The paper describes a very effective error estimator to detect high/low activity regions of the flow to be enriched or coarsened, respectively. The error estimator is based on total energy and density fluxes into the cell combined with gradient of density. Included in the paper is a detailed description of the direct dynamic refinement method that is used for adaptation. A detailed simulation of a reflection and diffraction of multiple shock waves flowing over a diamond shape wedge is presented and compared with experimental results. The simulated results are shown to be in excellent agreement with the experiment primarily in that all the complicated features of the physics are accurately accounted for and the shock waves, slip lines, vortices are sharply captured.

INTRODUCTION

Considerable progress has been made over the past decade in developing methods for spatial adaptation of the computational meshes based on the numerical solution of the simulated physics. These methods are being developed to produce higher spatial accuracy in such simulation more efficiently. The goal of mesh adaptation is to enrich meshes locally, based on the numerical solution, in order to capture physical features of importance; in contrast to globally fine meshes, this process will minimize computer run times and memory costs. The methods of mesh adaptation can be categorized into three general classes: 1) mesh regeneration, 2) mesh movement, and 3) mesh enrichment.

The idea of mesh regeneration is systematically to identify high/low activity region in the flow and accordingly remesh those regions applying mesh generation code. This is done by assigning criteria for spatial accuracy and number of vertices. This procedure requires a mapping of the "old" flow solution into the "new" generated meshes by using one of the interpolated schemes. For the second method, mesh movement, the number of points in the computational domain remains fixed. The adaptation procedure moves vertices from low activity regions to high gradient regions to achieve a high concentration of vertices to resolve high activity regions. The movement of the points is dictated by forcing functions in the Poisson - equation in the grid generator code. The final method of spatial adaptation is mesh

enrichment. In this method, vertices are added or removed according to the spatial resolution of the physical features in the flow. The advantages of mesh enrichment over regeneration and movement are its higher degree of flexibility in being able to add points where they are needed and to remove points where they are not needed. In our mesh enrichment method, we add points ahead of the shock wave, thus preventing the need of interpolation in the high gradient region for achieving higher accuracy of the results. Adding and removing points are done in monotone/very low activity regions to prevent numerical dissipation.

Lohner⁽¹⁾ has developed procedures to enrich the mesh for transient flow problems locally by subdividing elements in the grid according to specific spatial resolution criteria. The method, referred to as H-refinement, keeps a history of the initial grid (mother grid) and the subdivision of each level (daughter grids). The H-refinement relies heavily on the initial grid as it is subdivided for enrichment and recovered in the coarsening stage. A similar adaptive strategy to Lohner is adopted by Rausch⁽²⁾ et al., but applies a different error estimator and upwind type algorithm for a solver.

In our paper, we describe a Godunov scheme to solve Euler equations on an unstructured adaptive triangle mesh. We discuss the methodology of a cell centered Second Order Godunov scheme applied to a triangular mesh, and the method of Direct Dynamic Refinement that is used for adaptation of the unstructured triangular grid. Simulation and experimental results

are compared for a test case applying the adaptive unstructured grid to a complicated pattern of planar shock wave flow diffraction over a half diamond shape wedge.

SECOND ORDER GODUNOV ALGORITHM ON UNSTRUCTURED GRID

This section describes the implementation of the Second Order Godunov algorithm on a triangular unstructured grid. The algorithm is explicit and is cell-center based.

We consider a system of two-dimensional Euler equations written in conservation law form as:

$$\frac{\partial \bar{U}}{\partial t} + \frac{\partial \bar{F}}{\partial x} + \frac{\partial \bar{G}}{\partial y} = 0 \quad (1)$$

where

$$U = \begin{Bmatrix} \rho \\ \rho u \\ \rho v \\ e \end{Bmatrix}, F = \begin{Bmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ u(e + p) \end{Bmatrix}, G = \begin{Bmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ v(e + p) \end{Bmatrix}.$$

Here u, v are the x, y velocity vector components, p is the pressure, ρ is the density and e is total energy of the fluid. We assume that the fluid is an ideal gas. The total energy of gas is given by the following equation:

$$e = \frac{p}{\gamma - 1} + \frac{\rho}{2}(u^2 + v^2) \quad (2)$$

where γ is the ratio of specific heats. It is assumed that an initial distribution of the fluid parameters is given at $t = 0$, and the boundary conditions defining a unique solution are specified for the computational domain.

The system of governing equation (1) can be written in the following form:

$$\frac{\partial U}{\partial t} + \bar{\nabla} \cdot \bar{Q} = 0 \quad (3)$$

where \bar{Q} represents the convective flux vector. By integrating Eq. (3) over space and using Gauss' theorem, the following expression is obtained

$$\frac{\partial}{\partial t} \int_{\Omega} U dA + \oint_{\partial\Omega} \bar{Q} \cdot d\bar{l} = 0 \quad (4)$$

where $d\bar{l} = \bar{n} dL$, \bar{n} is the unit normal vector in the outward direction, and dL is a unit length on the boundary of the domain. The variable Ω is the domain of computation and $\partial\Omega$ is the circumference boundary of this domain.

Equation (4) can be discretized for each element (cell) of the domain

$$\frac{(U_i^{n+1} - U_i^n)}{\Delta t} A_i = \sum_{j=1}^3 \bar{Q}_j^{n+\frac{1}{2}} \bar{n}_j \Delta L_j \quad (5)$$

where A_i is the area of the cell; Δt is the marching time step; U_i^{n+1} and U_i^n are the primitive variables at the center of the cell at time n and at the update $n + 1$ time step; \bar{Q}_j is the value of the fluxes across the three boundaries edges on the circumference of the cell where \bar{n}_j is the unit normal

vector to the boundary edge j , and ΔL_j is the length of the boundary edge j . Equation (5) is used to update the physical primitive variables U_i according to computed fluxes for each time step Δt . The time step is subjected to the CFL (Courant-Fredrichs-Lewy) constraint.

To obtain a second order spacial accuracy, the gradient of each primitive variable is computed in the baricenter of the cell. This gradient is used to define the projected values of primitive variables at the two sides of the cell's edge, as is shown in Figure 1. The gradient is approximate by a path integral

$$\int_{\Omega} \bar{\nabla} U_i^{cell} dA = \oint_{\partial\Omega} U_j^{edge} d\bar{l}. \quad (6)$$

The notation is similar to the one used for Eq. (5) except the domain Ω is a single cell and U_i^{cell} and U_j^{edge} are values at the baricenter and on the edge respectively. The gradient is estimated as

$$\bar{\nabla} U_i^{cell} = \frac{1}{A} \sum_{j=1}^3 \bar{U}_j^{edge} \bar{n}_j \Delta L_j \quad (7)$$

where \bar{U}_j^{edge} is an average value representing the primitive variable value for edge j .

The gradients that are computed at each baricenter are used to project values for the two sides of each edge by piecewise linear interpolation. The interpolated values are subjected to monotonicity constraints.⁽³⁾ The monotonicity constraint assures that the interpolated values are not creating new

extrema.

The monotonicity limiter algorithm can be written in the following form:

$$U_{projected}^{edge} = U_i^{cell} + \phi \bar{\nabla} U_i \cdot \Delta \bar{r} \quad (8)$$

where $\Delta \bar{r}$ is the vector from the baricenter to the point of intersection of the edge with the line connecting the baricenters of the cells over the two sides of this edge. ϕ is the limiter coefficient that limits the gradient $\bar{\nabla} U_i$.

First, we compute the maximum and minimum values of the primitive variable in the i 's cell and its three neighboring cells that share common edges (see Fig. 1):

$$\left. \begin{aligned} U_{cell}^{max} &= Max(U_k^{cell}) \\ U_{cell}^{min} &= Min(U_k^{cell}) \end{aligned} \right\} k = i, 1, 2, 3 \quad (9)$$

The limiter can be defined as:

$$\phi = Min \{1, \phi_k^{lr}\} \quad k = 1, 2, 3 \quad (10)$$

where superscript lr stands for left and right of the three edges (6 combinations in total). ϕ_k^{lr} is defined by:

$$\phi_k^{lr} = \frac{[1 + Sgn(\Delta U_k^{lr})] \Delta U_{cell}^{max} + [1 - Sgn(\Delta U_k^{lr})] \Delta U_{cell}^{min}}{2(\Delta U_k^{lr})} \quad k = 1, 2, 3 \quad (11)$$

where $\Delta U_k^{!r} = \bar{\nabla} U_i^{!r} \cdot \Delta \bar{r}_k$. and

$$\left. \begin{aligned} \Delta U_{cell}^{\max} &= U_{cell}^{\max} - U_i^{cell} \\ \Delta U_{cell}^{\min} &= U_{cell}^{\min} - U_i^{cell} \end{aligned} \right\} \quad (12)$$

To obtain a second order of accuracy in time and space, we subject the projected values of the left and right side of the cell edge to characteristic constraints following Ref. 4. The one dimensional characteristic predictor is applied to the projected values at half time step $t^n + \frac{\Delta t}{2}$. The characteristic predictor is formulated in the local system of coordinates for the one dimensional Euler equation. We illustrate the implementation of the characteristic predictor in the direction of the unit vector \bar{n}_c . The Euler equations for this direction can be written in the following form:

$$W_t + A(W)W_{nc} = 0 \quad (13)$$

where

$$W = \begin{Bmatrix} \tau \\ u \\ p \end{Bmatrix}; \quad A(W) = \begin{pmatrix} u & -\tau & 0 \\ 0 & u & \tau \\ 0 & \rho c^2 & u \end{pmatrix} \quad (14)$$

where $\tau = \rho^{-1}$, ρ denotes density while u, p are the velocity and pressure. The matrix $A(W)$ has three eigenvectors ($l^\#, r^\#$) (l for left and r for right where $\#$ denote $+, 0, -$) associated with the eigenvalues $\lambda^+ = u + c$, $\lambda^0 = u$, $\lambda^- = u - c$.

An approximation of projected value to an edge accurate to second order in space and time can be written as:

$$\begin{aligned} W_{i+\Delta r}^{n+1/2} &\approx W_i^n + \frac{\Delta t}{2} \frac{\partial W}{\partial t} + \Delta r \frac{\partial W}{\partial r_{nc}} \\ &\approx W_i^n + \left[\Delta r - \frac{\Delta t}{2} A(W_i) \right] \frac{\partial W}{\partial r_{nc}} \end{aligned} \quad (15)$$

An approximation to $W_{i+\Delta r}^{n+1/2}$ can be written as:

$$W_{i+\Delta r}^{n+1/2} = W_i + (\Delta \bar{r}_i - \frac{\Delta t}{2} (M_x M_n) \cdot \bar{n}_c) \bar{\nabla} W_i \quad (16)$$

where

$$(M_x M_n) = \begin{cases} \text{Max}(\lambda_i^+, 0) & \text{for cell left to the edge} \\ \text{Min}(\lambda_i^-, 0) & \text{for cell right to the edge.} \end{cases} \quad (17)$$

The gradients applied in the process of computing the projected values at $t^n + (\Delta t/2)$ are subjected to the monotonicity limiter.

Following the characteristic predictor described above, the full Riemann problem is solved at the edge. The solution of the Riemann problem defines the flux $\bar{Q}^{n+1/2}$ through the edge. The fluxes through the edges of triangles are then integrated (Eq. 5), thus giving an updated value of the variables at t^{n+1} . One of the advantages of the described algorithm is that calculation of the fluxes is done over the largest loop in the system (loop over edges) and can be carried out in the vectorized or parallelized loop. This fact leads to an efficient algorithm.

The algorithm presented is a modification of the algorithm of Ref. 5 which was derived for structured mesh. This algorithm has been applied to simulate a wide range of flow problems and has been found very accurate in predicting the features of the physics. The performance of the algorithm is well documented in Refs. 6-8. The next section, the spatial adaptive procedure, is described in detail. These descriptions include explanations of the error estimator for flow feature detection and the Direct Dynamic Refinement Method used to enrich and coarsen the mesh.

DIRECT DYNAMIC REFINEMENT METHOD FOR ADAPTATION ON AN UNSTRUCTURED TRIANGULAR GRID

The Direct Dynamic Refinement method (DDR) is a new method for adapting unstructured triangular grids during the computational process. As stated, an unstructured grid is very suitable for implementing boundary conditions on complex geometrical shapes as well as the adaptation of the grid, if necessary. The adaptation of the unstructured triangular grid leads to efficient usage of memory resources. The adaptive grid enables the user to capture moving shocks and high gradient flow features with high resolution. The available memory resources can be very efficiently distributed in the computational domain to accommodate the resolution needed to capture features of the physical property of the solution as they are evolved. Dynamic refinement controls the resolution priorities. These priorities can be set according to the physical features that the user wishes to emphasize

in the simulation. The user has control over the accuracy of the physical features resolved in the simulation, without being restricted to the initial grid. The alternative to Direct Dynamic Refinement (DDR) is the hierarchical dynamic refinement (H-refinement) that keeps a history of the initial grid (mother grid) and the subdivision of each level (daughter grids). In the H-refinement method, it is necessary to keep overhead information on the level of each triangle subdivision, and double indirect indexing is needed to keep track of the H-refinement process. As mentioned, the H-refinement relies heavily on the initial grid as it subdivides this grid and returns to it after the passage of the shock.

To minimize the dissipation caused by the interpolation and extrapolation in the refinement and coarsening of the grid, the addition and deletion of point is done in the region where the flow features are smooth. Thus for capturing the shock, the refinement should be applied in the region ahead of the shock. The coarsening of the grid is done in the flow regions where the gradients of the flow parameters are small.

In the present version of AUGUST (Adaptive Unstructured Godunov Upwind Second order Triangular), we implemented an algorithm with multiple criteria for capturing a variety of features that might exist in the physics of the problem to be solved. To identify the location of a moving shock, we use the flux of total energy into triangles. The fluxes entering and leaving triangles are the most accurate physical variables computed by the Godunov

algorithm for solving Euler's equations, and are used to update the physical variables for each time step in each triangle. Supplementary to the flux of energy as an error indicator, we use the flux of total density into triangles and the density gradient. The error indicator is the only sensor that is solely responsible for identifying the area to be refined or coarsened in the computational domain. As such, the error indicator should be sensitive enough to detect physical features that are of interest to the user, such as shock waves, rarefaction waves, slip lines and vortices. The error indicators that are implemented in the code are able to sense very weak slip lines in the presence of strong shock waves. The ability of the error indicators to identify weak physical features in the presence of strong ones, without picking up numerical noises, is essential to the simulation of adaptive grids. As stated, the quality of the results is as good as the error indicators applied. If the error indicators fail to identify the physical feature, this feature probably will be overlooked in the simulated results. It should be noted that the process of applying error indicators for identifying the areas to be adaptively refined or coarsened is an expensive loop that has to check the whole triangles table in the simulation. Thus, the error indicators are applied each 9 to 15 time steps. This process is preceded by application of an algorithm that refines a buffer zone ahead of the features and coarsens the grid after it was moved away. The buffer zone ahead of the feature is identified by using a search pattern of finding the neighbors of the flagged triangles sorted by the error indicators.

We are not applying any physical parameters to identify the zones "ahead."

The refinement algorithm follows several basic steps. The process of adding points to refine the grid locally is done by either adding a new vertex in the baricenter of the triangle or adding a new vertex in the middle of the edge. Adding a new vertex in the baricenter of a triangle is very efficient in the sense that the refinement affects this individual triangle only. We apply this process exclusively for refinement. As a supplement, especially on the boundary, we apply the method of adding a new vertex on an edge. As a complement to adding new vertices, we apply the reconnection/swapping algorithm that flips the diagonal (common edge) of two adjacent triangles to improve the quality of the triangles constructed. Figure 2 displays a chain of those basic steps to illustrate the refinement process. Figure 2a shows the original grid. Figure 2b illustrates a one step scheme refinement in which a new vertex is introduced into a triangular cell forming three cells (two new ones). On the boundary edges, a new vertex is introduced in the middle of those edges to form two cells (one new one). This refinement is followed by reconnection that modifies the grid as demonstrated in Fig. 2c. The process of refinement and reconnection can be continued until the necessary grid resolution is achieved. As an example, another loop of refinement is illustrated in Figs. 2d and 2e. This direct approach to grid refinement provides extreme flexibility in resolving local flow features.

A similar direct approach is applied to grid coarsening. The basic step

in this process is deleting the cells and edges associated with a vertex to be removed, as shown in Fig. 3b. During the second step, this void in the grid is filled with new larger triangles (Fig. 3c) without introducing new vertices. The last step is local reconnection and relaxation as shown in Fig. 3d. The relaxation procedure is a simple relocation of the vertex moved to the center of the polygon surrounding this vertex (only if the polygon is a convex).

The algorithm of direct dynamic refinement proved to be very efficient in refining and coarsening the grid adaptively. The refinement and coarsening followed a short inquiry on the quality and shape of the triangle flagged and its close neighbors. Since we do not keep any history or tree for each triangle, the DDRM algorithm has much less checking to do as compared to the H-refinement algorithm. The vectorization and parallelization of the solver is straightforward.

NUMERICAL RESULTS FOR THE TWO DIMENSIONAL TEST PROBLEM

We have tested the Second Order Godunov algorithm in a variety of flow simulations ranging from the low subsonic to the high hypersonic Mach ⁽⁶⁻⁸⁾ regime. The AUGUST code proved to be very robust and accurate. The results obtained are comparable to or better than those obtained applying leading flow solvers in all of the regimes tested.

To validate our DDRM implemented in the AUGUST code, we simulated the problem of interaction of a Mach 2.85 planar shock wave, propagating

are accurately accounted for and the shock waves, slip lines, vortices are sharply captured.

in a channel with a 45° symmetrical double ramp. Figure 4 shows the experimental interferogram of the problem to be simulated (reproduced). The example that we chose to simulate is most appropriate to test the performance of an adaptive algorithm. The experimental results show a complex flow pattern containing a mix of strong discontinuities, as shock waves, and very weak features such as slip lines, vortices, and rarefaction waves. The error estimator must recognize and flag all these features for refinement. The error estimator should be sensitive enough to identify very weak slip lines without picking up numerical noises present in the simulation. We have simulated the shock wave reflection and diffraction over a 45° corner at the conditions that correspond to the experimental result shown in Fig. 4. Here we present results for several shapes of the flow evolution. The flow in the channel is from left to right. Figure 5 displays density contour plots after the shock passed the apex of the double wedge obstacle. In Fig. 5a, the density contours are overlaid on the grid used at this stage of the evolving flow. For clarity, only the density contours are displayed in Fig. 5b. The grid displayed in Fig. 5a shows how well the adaptation technique follows the high activity region in the flow. The grid is adapting to regions with high pressure gradients and high density gradient. In Fig. 5a, one can observe high quality grid produced by the DDR method. The shock has a relatively thin buffer zone ahead of its front, allowing us to avoid the interpolations related to grid adaptation of the flow variables in the area of high gradient.

The flow features are resolved accurately, and the contact discontinuity and triple point are clearly defined.

Figure 6 shows the density contours at a later time in the same format as in Fig. 5. This figure demonstrates the ability of the DDRM to identify and follow flow features in the computational domain. In this figure we can observe a complicated flow pattern developing as a result of interaction of the rarefaction wave with the complex pattern of shock waves. A recompression shock and a strong vortex that are developed in this time frame are well resolved. We can also observe a slip line originating at the triple point. The adaptation algorithm, as in the previous time frame, follows both shock waves and contact discontinuities.

Figure 7 displays the density contours at the stage comparable to that shown in Fig. 4 for the experimental results. The computed results as displayed in Fig. 7b show a flow pattern similar to the experiment. The slip line and the formation of vertices along it are clearly depicted. The shock and reflected shock as well as the recompression shock are very sharply defined with very low numerical noise. The vortex developed after the compression shock is distinctly displayed. A new reflected shock can be seen developing at the channel wall behind the double wedge.

The results shown in Figs. 5-7 display the ability of the algorithm to simulate a complex transient flow problem on dynamically adapting grid. The error estimates used in our algorithm allow detection of strong and

weak shock waves, conducted discontinuities, vortices or other fronts that need enhanced resolution.

CONCLUSION

The Direct Dynamic Refinement (DDR) method was developed and tested for a challenging problem of reflection and diffraction of a strong shock over a double ramp. For this test problem we have demonstrated that a set of error indicators developed for the DDR allow capturing strong and weak features of the complex wave structure developing in this test case.

The above described algorithms were implemented in the AUGUST code. The AUGUST code was used for a range of subsonic, transonic, and supersonic transient and steady problems. For all these conditions the AUGUST code produced robust results with the error indicators proving to be applicable for all these diverse flow regimes.

ACKNOWLEDGMENT

The work reported here was partially supported by DARPA and AFOSR under Contract #F49620-89-C-0087. The authors would like to thank Col. James Crowley and Dr. A. Nachman for their interest in this project.

REFERENCES

- 1) R. Lohner, "An Adaptive Finite Element Scheme for Transient Problems in CFD," *Comp. Meth. Appl. Mech. Eng.*, Vol. 61, 323-338. 1987.

- 2) R.D. Rausch, J.T. Batina, H.T.Y. Yang "Spatial Adaptation of Unstructured Meshes for Unsteady Aerodynamic Flow Computation." AIAA Journal, Vol 30, No. 5, May 1992, pp. 1243-1251.
- 3) B. Van Leer, "Toward the Ultimate Conservative Difference Scheme. V. A Second Order Sequel to Godunov Method," J. Comp. Phys., 32, 101-136, 1979.
- 4) P. Collela and H.M. Glaz, "Efficient Solution Algorithm for the Riemann Problem for Real Gases," J. Comp. Phys. 59, 264-289, 1985.
- 5) S. Eidelman, P. Collela, and R.P. Shreeve. "Application of the Godunov Method and its Second Order Extension to Cascade Flow Modeling," AIAA Journal 22, 10, 1984.
- 6) I. Lottati, S. Eidelman, and A.T. Drobot, "A Fast Unstructured Grid Second Order Godunov Solver (FUGGS)," AIAA 90-0699, 27th Aerospace Sciences Meeting, Reno, Nevada, 1989.
- 7) I. Lottati, S. Eidelman, and A.T. Drobot. "Solution of Euler's Equations on Adaptive Grids Using a Fast Unstructured Grid Second Order Godunov Solver," Proceedings of the Free Lagrange Conference, Jackson Lake, WY, June 1990.
- 8) I. Lottati and S. Eidelman, "Second Order Godunov Solver on Adaptive Unstructured Triangular Grid," Proceedings of 4th International Symposium on Computational Fluid Dynamics, Davis, CA, September 1991.

9. D.L. Zhang, I.I. Glass, "An Interferometric Investigation of the Diffraction of Planar Shock Waves Over a Half-Diamond Cylinder in Air,"
NTIAS Report No. 322, March 1988.

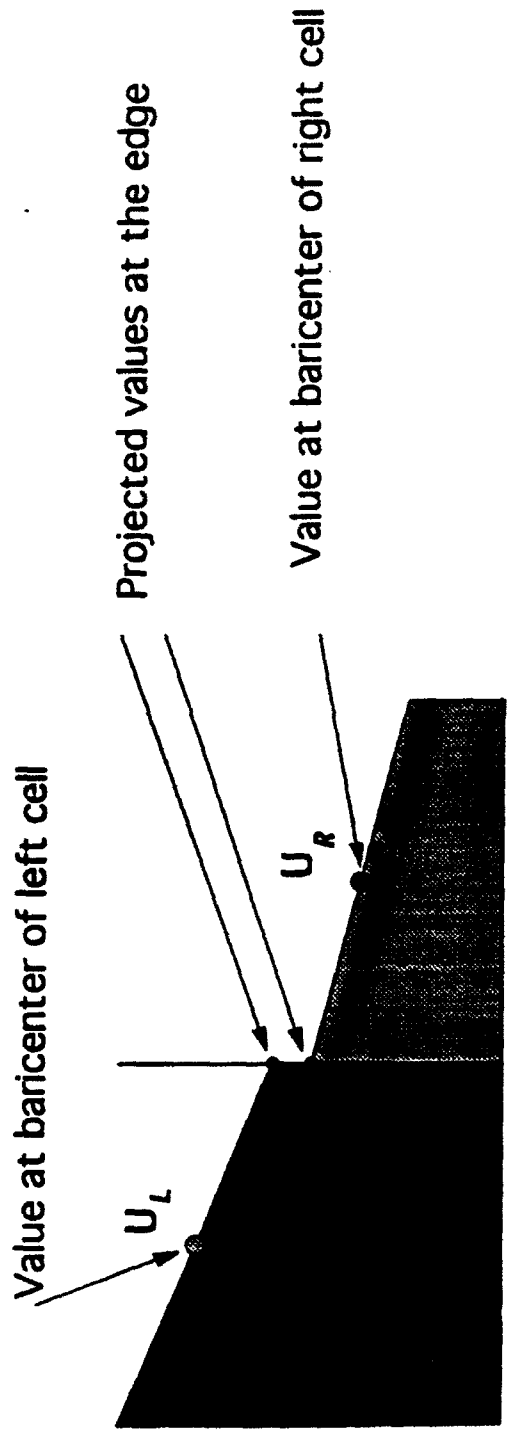
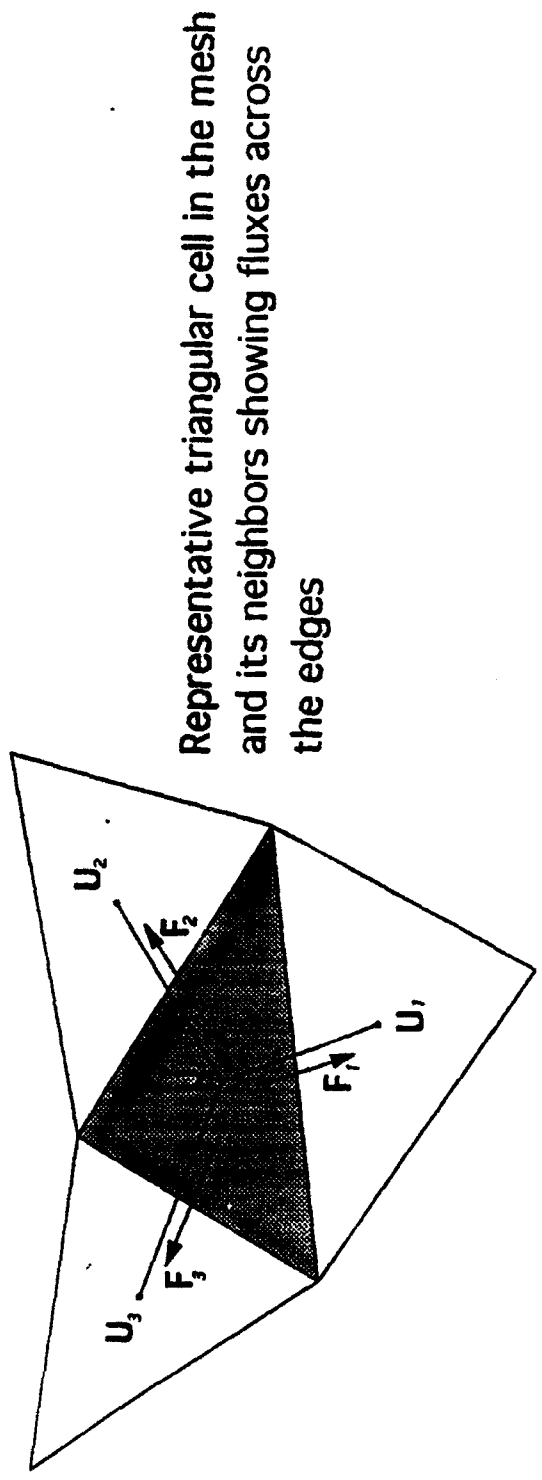
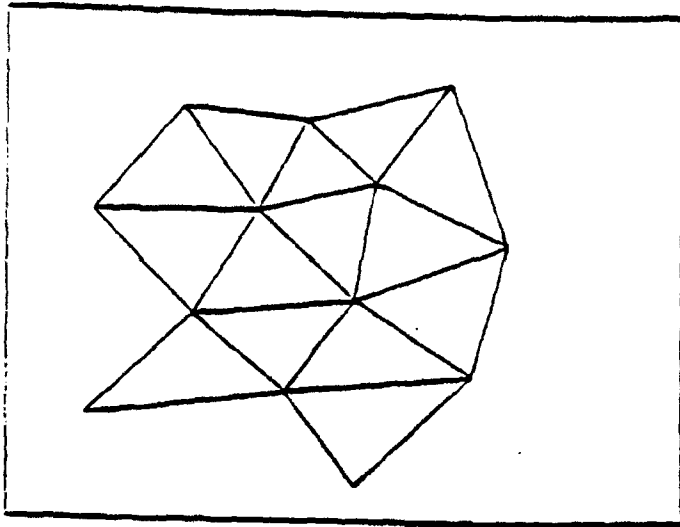
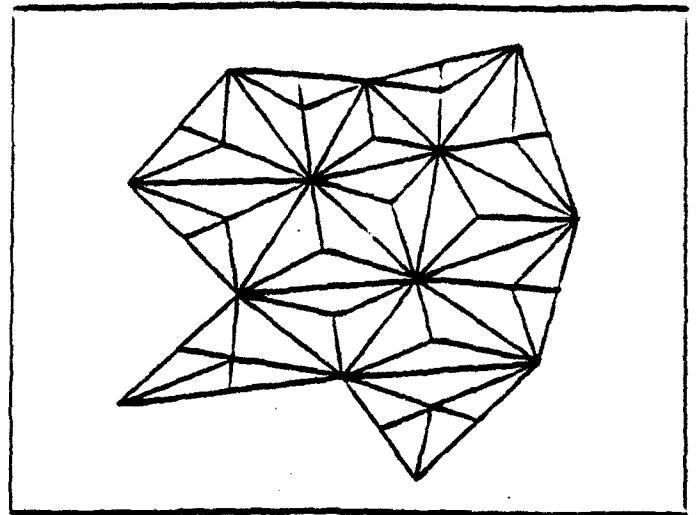


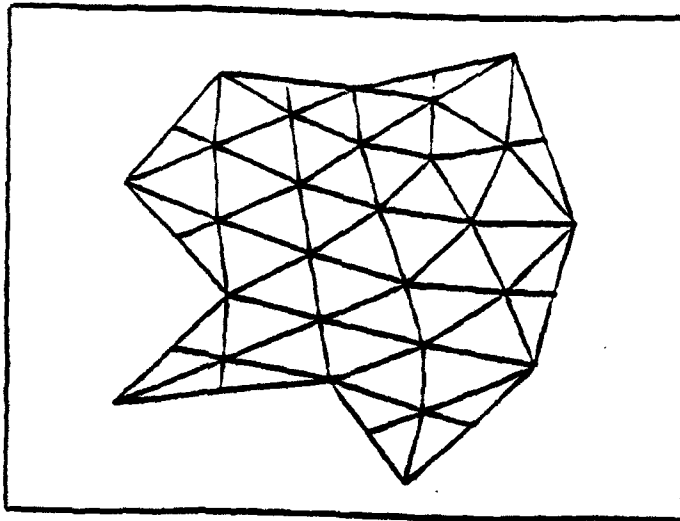
Figure 1. Representative triangular cell in the mesh showing fluxes and projected values.



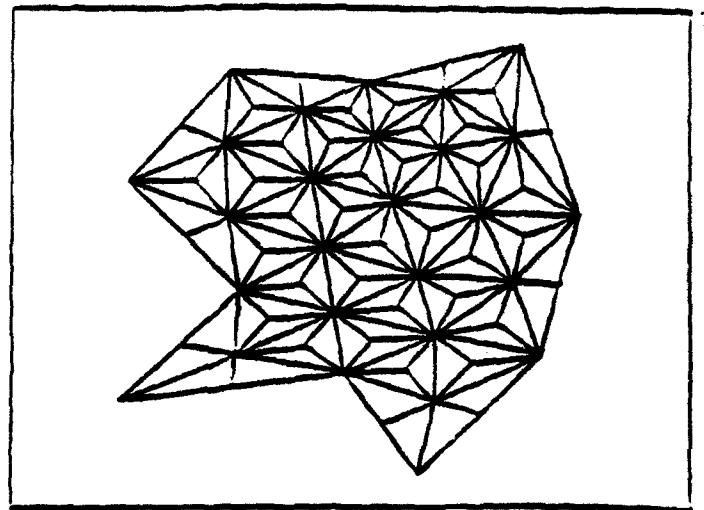
a. Original grid.



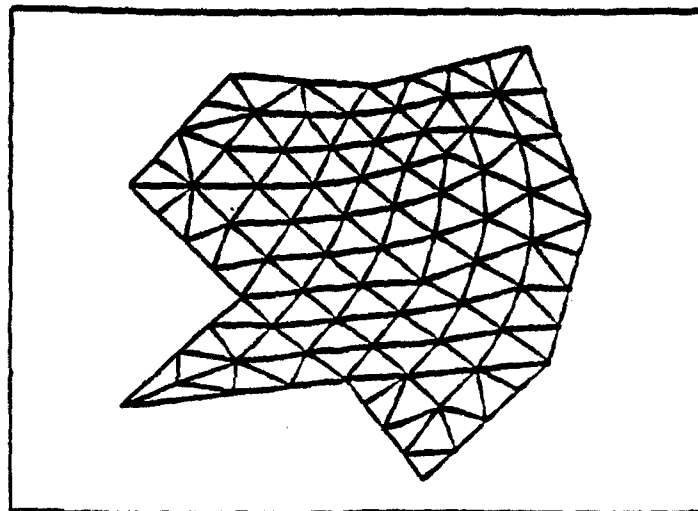
b. Grid after one refinement.



c. Grid after one refinement
and one reconnection.

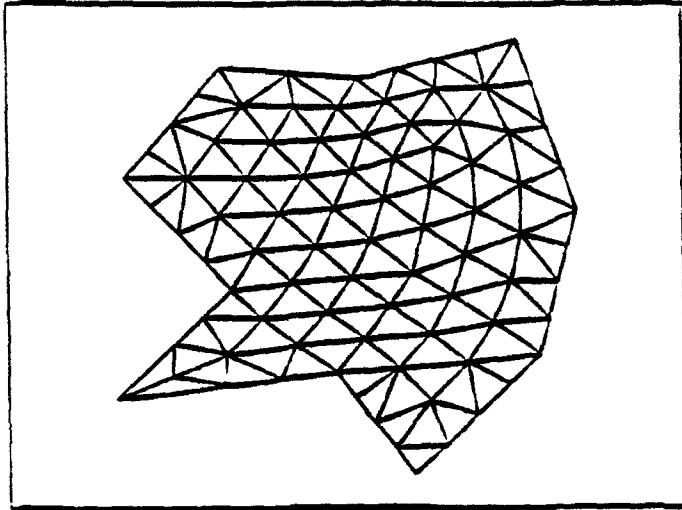


d. Second refinement.

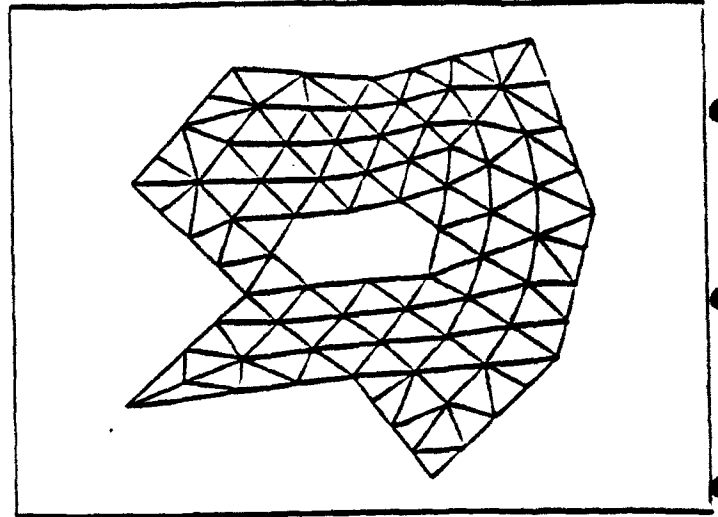


e. Second reconnection.

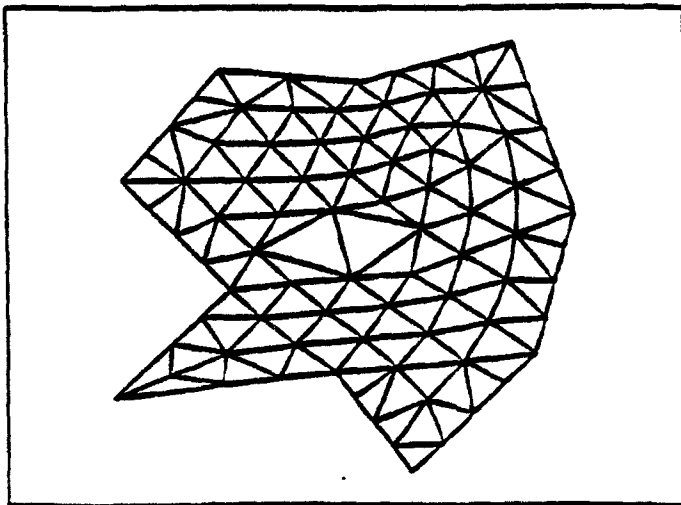
Figure 2. Illustration of the grid refinement process.



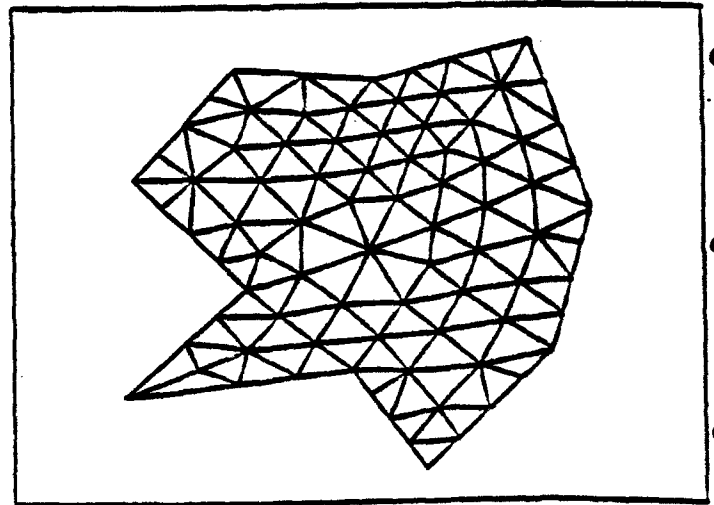
a. Original grid.



b. Point removal.



c. Constructing of new cells.



d. Grid after reconnection and relaxation.

Figure 3. Illustration of the grid coarsening process.

Region	ρ/ρ_0
(0)	1.00
(1)	3.70
a	1.80
b	2.56
c	3.32
d	4.08
e	4.84
f	5.60
g	6.36
h	7.12
i	6.74
j	6.36
k	7.12
l	7.50
m	7.88



Figure 4. An experimental interferogram taken at $96 \mu\text{s}$ after shock wave hits a diamond shaped obstacle, Mach $M_0 = 2.85$.

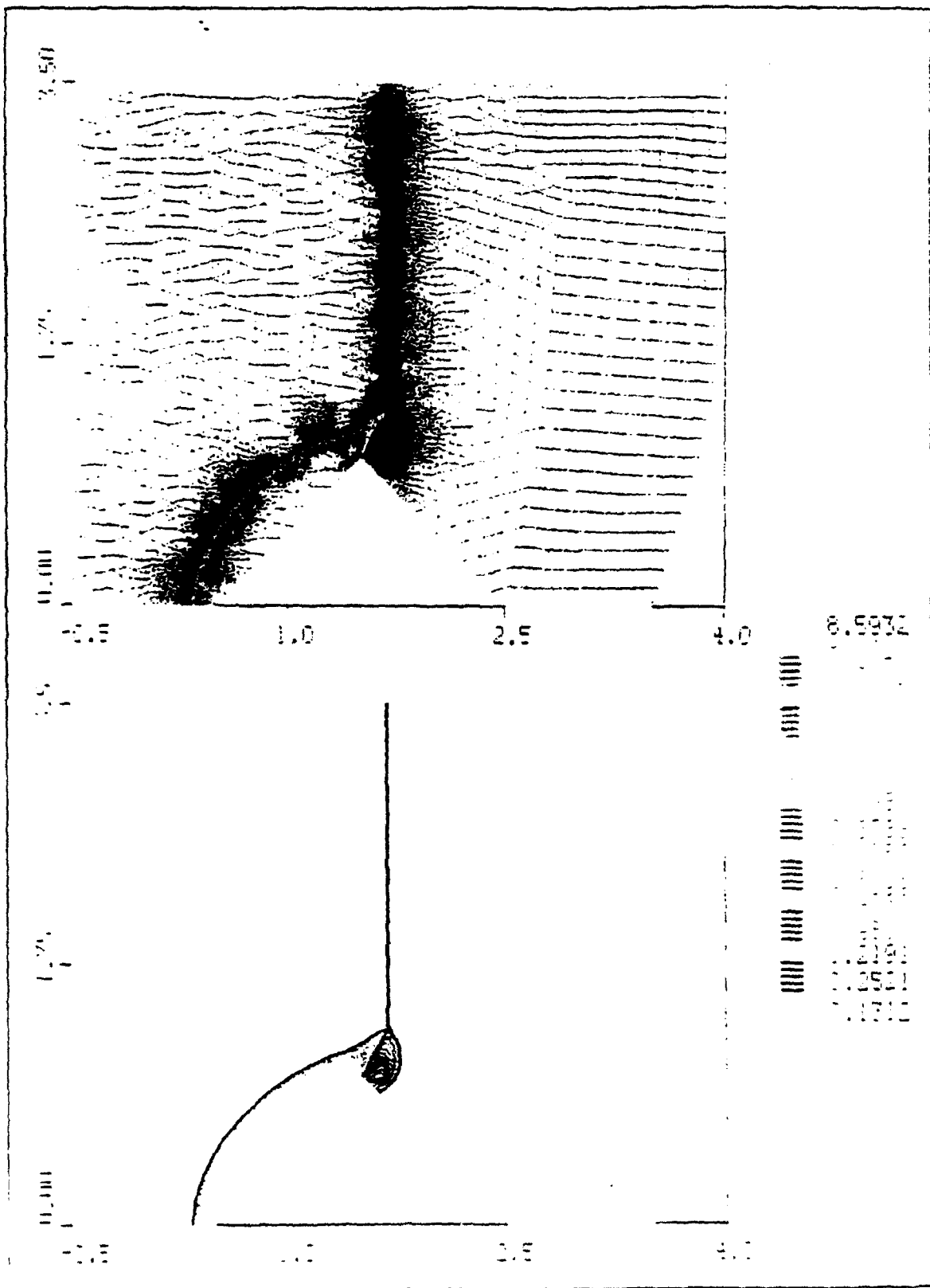


Figure 5. Computed density contours simulating flow identical to the setup of the experiment of Fig. 4. The grid is composed of 21121 vertices.

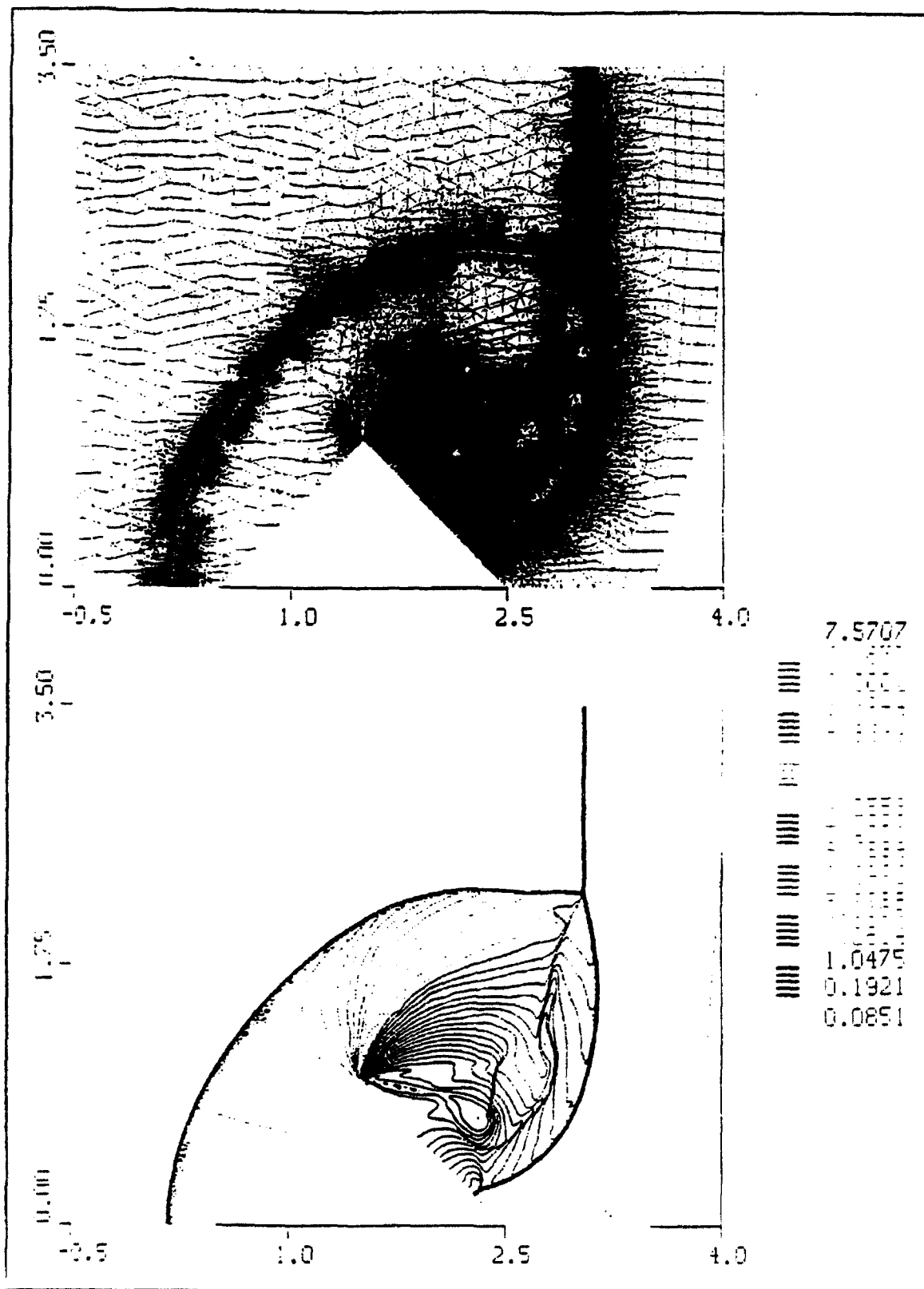


Figure 6. Computed density contours simulating flow identical to the setup of the experiment of Fig. 4. The grid is composed of 65624 vertices.

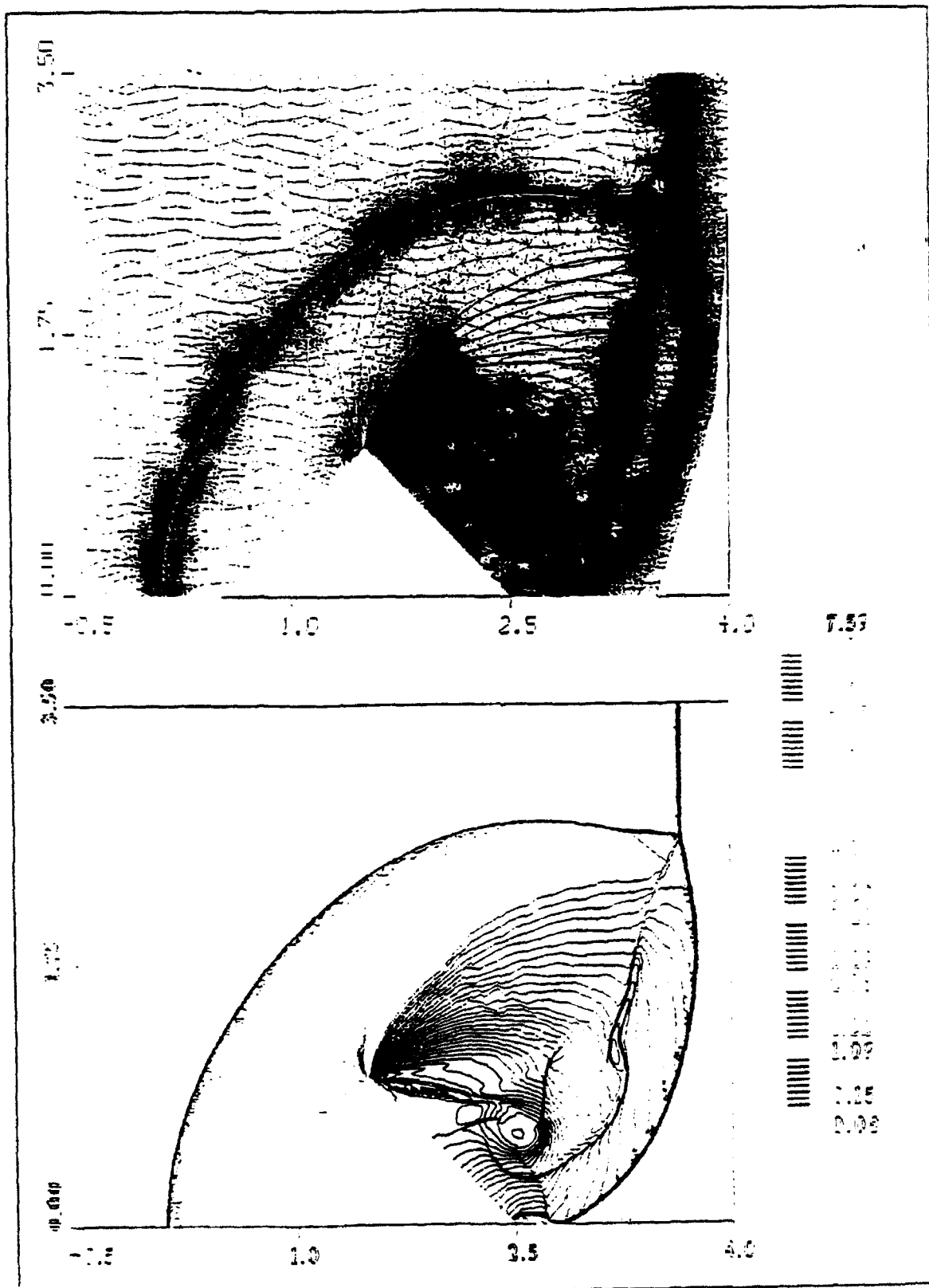


Figure 7. Computed density contours comparable to time of the experimental results shown in Fig. 4. The grid is composed of 79352 vertices.

A Second Order Godunov Scheme on Spatial

Adapted Triangular Grid

Itzhak Lottati and Shmuel Eidelman

Science Applications International Corporation

ABSTRACT

Spatial adaptation procedure for the accurate and efficient solution of unsteady inviscid flow simulation is described. The adaptation procedures were developed and implemented applying a second order Godunov scheme. These procedures involve mesh enrichment/coarsening to either add/remove vertices in high/low gradient regions of the flow, respectively. The goal is to achieve solutions of high spatial accuracy at minimal computational cost. The paper describes a very effective error estimator to detect high/low activity regions of the flow to be enriched or coarsened, respectively. The error estimator is based on total energy and density fluxes into the cell combined with gradient of density. Included in the paper is a detailed description of the direct dynamic refinement method that is used for adaptation. A detailed simulation of a reflection and diffraction of multiple shock waves flowing over a diamond shape wedge is presented and compared with experimental results. The simulated results are shown to be in excellent agreement with the experiment primarily in that all the complicated features of the physics are accurately accounted for and the shock waves, slip lines, vortices are sharply captured.

INTRODUCTION

Considerable progress has been made over the past decade in developing methods for spatial adaptation of the computational meshes based on the numerical solution of the simulated physics. These methods are being developed to produce higher spatial accuracy in such simulation more efficiently. The goal of mesh adaptation is to enrich meshes locally, based on the numerical solution, in order to capture physical features of importance; in contrast to globally fine meshes, this process will minimize computer run times and memory costs. The methods of mesh adaptation can be categorized into three general classes: 1) mesh regeneration, 2) mesh movement, and 3) mesh enrichment.

The idea of mesh regeneration is systematically to identify high/low activity region in the flow and accordingly remesh those regions applying mesh generation code. This is done by assigning criteria for spatial accuracy and number of vertices. This procedure requires a mapping of the "old" flow solution into the "new" generated meshes by using one of the interpolated schemes. For the second method, mesh movement, the number of points in the computational domain remains fixed. The adaptation procedure moves vertices from low activity regions to high gradient regions to achieve a high concentration of vertices to resolve high activity regions. The movement of the points is dictated by forcing functions in the Poisson - equation in the grid generator code. The final method of spatial adaptation is mesh

enrichment. In this method, vertices are added or removed according to the spatial resolution of the physical features in the flow. The advantages of mesh enrichment over regeneration and movement are its higher degree of flexibility in being able to add points where they are needed and to remove points where they are not needed. In our mesh enrichment method, we add points ahead of the shock wave, thus preventing the need of interpolation in the high gradient region for achieving higher accuracy of the results. Adding and removing points are done in monotone/very low activity regions to prevent numerical dissipation.

Lohner⁽¹⁾ has developed procedures to enrich the mesh for transient flow problems locally by subdividing elements in the grid according to specific spatial resolution criteria. The method, referred to as H-refinement, keeps a history of the initial grid (mother grid) and the subdivision of each level (daughter grids). The H-refinement relies heavily on the initial grid as it is subdivided for enrichment and recovered in the coarsening stage. A similar adaptive strategy to Lohner is adopted by Rausch⁽²⁾ et al., but applies a different error estimator and upwind type algorithm for a solver.

In our paper, we describe a Godunov scheme to solve Euler equations on an unstructured adaptive triangle mesh. We discuss the methodology of a cell centered Second Order Godunov scheme applied to a triangular mesh, and the method of Direct Dynamic Refinement that is used for adaptation of the unstructured triangular grid. Simulation and experimental results

are compared for a test case applying the adaptive unstructured grid to a complicated pattern of planar shock wave flow diffraction over a half diamond shape wedge.

SECOND ORDER GODUNOV ALGORITHM ON UNSTRUCTURED GRID

This section describes the implementation of the Second Order Godunov algorithm on a triangular unstructured grid. The algorithm is explicit and is cell-center based.

We consider a system of two-dimensional Euler equations written in conservation law form as:

$$\frac{\partial \bar{U}}{\partial t} + \frac{\partial \bar{F}}{\partial x} + \frac{\partial \bar{G}}{\partial y} = 0 \quad (1)$$

where

$$U = \begin{Bmatrix} \rho \\ \rho u \\ \rho v \\ e \end{Bmatrix}, F = \begin{Bmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ u(e + p) \end{Bmatrix}, G = \begin{Bmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ v(e + p) \end{Bmatrix}.$$

Here u, v are the x, y velocity vector components, p is the pressure, ρ is the density and e is total energy of the fluid. We assume that the fluid is an ideal gas. The total energy of gas is given by the following equation:

$$e = \frac{p}{\gamma - 1} + \frac{\rho}{2}(u^2 + v^2) \quad (2)$$

where γ is the ratio of specific heats. It is assumed that an initial distribution of the fluid parameters is given at $t = 0$, and the boundary conditions defining a unique solution are specified for the computational domain.

The system of governing equation (1) can be written in the following form:

$$\frac{\partial U}{\partial t} + \bar{\nabla} \cdot \bar{Q} = 0 \quad (3)$$

where \bar{Q} represents the convective flux vector. By integrating Eq. (3) over space and using Gauss' theorem, the following expression is obtained

$$\frac{\partial}{\partial t} \int_{\Omega} U dA + \oint_{\partial\Omega} \bar{Q} \cdot d\bar{l} = 0 \quad (4)$$

where $d\bar{l} = \bar{n} d\mathcal{L}$, \bar{n} is the unit normal vector in the outward direction, and $d\mathcal{L}$ is a unit length on the boundary of the domain. The variable Ω is the domain of computation and $\partial\Omega$ is the circumference boundary of this domain.

Equation (4) can be discretized for each element (cell) of the domain

$$\frac{(U_i^{n+1} - U_i^n)}{\Delta t} A_i = \sum_{j=1}^3 \bar{Q}_j^{n+\frac{1}{2}} \bar{n}_j \Delta L_j \quad (5)$$

where A_i is the area of the cell; Δt is the marching time step; U_i^{n+1} and U_i^n are the primitive variables at the center of the cell at time n and at the update $n + 1$ time step; \bar{Q}_j is the value of the fluxes across the three boundaries edges on the circumference of the cell where \bar{n}_j is the unit normal

vector to the boundary edge j , and ΔL_j is the length of the boundary edge j . Equation (5) is used to update the physical primitive variables U_i according to computed fluxes for each time step Δt . The time step is subjected to the CFL (Courant-Fredrichs-Lewy) constraint.

To obtain a second order spacial accuracy, the gradient of each primitive variable is computed in the baricenter of the cell. This gradient is used to define the projected values of primitive variables at the two sides of the cell's edge, as is shown in Figure 1. The gradient is approximate by a path integral

$$\int_{\Omega} \bar{\nabla} U_i^{cell} dA = \oint_{\partial\Omega} U_j^{edge} d\bar{l}. \quad (6)$$

The notation is similar to the one used for Eq. (5) except the domain Ω is a single cell and U_i^{cell} and U_j^{edge} are values at the baricenter and on the edge respectively. The gradient is estimated as

$$\bar{\nabla} U_i^{cell} = \frac{1}{A} \sum_{j=1}^3 \tilde{U}_j^{edge} \bar{n}_j \Delta L_j \quad (7)$$

where \tilde{U}_j^{edge} is an average value representing the primitive variable value for edge j .

The gradients that are computed at each baricenter are used to project values for the two sides of each edge by piecewise linear interpolation. The interpolated values are subjected to monotonicity constraints.⁽³⁾ The monotonicity constraint assures that the interpolated values are not creating new

extrema.

The monotonicity limiter algorithm can be written in the following form:

$$U_{projected}^{edge} = U_i^{cell} + \phi \bar{\nabla} U_i \cdot \Delta \bar{r} \quad (8)$$

where $\Delta \bar{r}$ is the vector from the baricenter to the point of intersection of the edge with the line connecting the baricenters of the cells over the two sides of this edge. ϕ is the limiter coefficient that limits the gradient $\bar{\nabla} U_i$.

First, we compute the maximum and minimum values of the primitive variable in the i 's cell and its three neighboring cells that share common edges (see Fig. 1):

$$\left. \begin{aligned} U_{cell}^{max} &= Max(U_k^{cell}) \\ U_{cell}^{min} &= Min(U_k^{cell}) \end{aligned} \right\} k = i, 1, 2, 3. \quad (9)$$

The limiter can be defined as:

$$\phi = Min \{1, \phi_k^{lr}\} \quad k = 1, 2, 3 \quad (10)$$

where superscript lr stands for left and right of the three edges (6 combinations in total). ϕ_k^{lr} is defined by:

$$\phi_k^{lr} = \frac{[1 + Sgn(\Delta U_k^{lr})] \Delta U_{cell}^{max} + [1 - Sgn(\Delta U_k^{lr})] \Delta U_{cell}^{min}}{2(\Delta U_k^{lr})} \quad k = 1, 2, 3 \quad (11)$$

where $\Delta U_k^{lr} = \bar{\nabla} U_i^{lr} \cdot \Delta \bar{r}_k$. and

$$\left. \begin{aligned} \Delta U_{cell}^{\max} &= U_{cell}^{\max} - U_i^{cell} \\ \Delta U_{cell}^{\min} &= U_{cell}^{\min} - U_i^{cell} \end{aligned} \right\} \quad (12)$$

To obtain a second order of accuracy in time and space, we subject the projected values of the left and right side of the cell edge to characteristic constraints following Ref. 4. The one dimensional characteristic predictor is applied to the projected values at half time step $t^n + \frac{\Delta t}{2}$. The characteristic predictor is formulated in the local system of coordinates for the one dimensional Euler equation. We illustrate the implementation of the characteristic predictor in the direction of the unit vector \bar{n}_c . The Euler equations for this direction can be written in the following form:

$$W_t + A(W)W_{nc} = 0 \quad (13)$$

where

$$W = \begin{Bmatrix} \tau \\ u \\ p \end{Bmatrix}; \quad A(W) = \begin{pmatrix} u & -\tau & 0 \\ 0 & u & \tau \\ 0 & \rho c^2 & u \end{pmatrix} \quad (14)$$

where $\tau = \rho^{-1}$, ρ denotes density while u, p are the velocity and pressure. The matrix $A(W)$ has three eigenvectors ($l^\#, r^\#$) (l for left and r for right where $\#$ denote $+, 0, -$) associated with the eigenvalues $\lambda^+ = u + c$, $\lambda^0 = u$, $\lambda^- = u - c$.

An approximation of projected value to an edge accurate to second order in space and time can be written as:

$$\begin{aligned} W_{i+\Delta r}^{n+1/2} &\approx W_i^n + \frac{\Delta t}{2} \frac{\partial W}{\partial t} + \Delta r \frac{\partial W}{\partial r_{nc}} \\ &\approx W_i^n + \left[\Delta r - \frac{\Delta t}{2} A(W_i) \right] \frac{\partial W}{\partial r_{nc}} \end{aligned} \quad (15)$$

An approximation to $W_{i+\Delta r}^{n+1/2}$ can be written as:

$$W_{i+\Delta r}^{n+1/2} = W_i + (\Delta \bar{r}_i - \frac{\Delta t}{2} (M_x M_n) \cdot \bar{n}_c) \bar{\nabla} W_i \quad (16)$$

where

$$(M_x M_n) = \begin{cases} \text{Max}(\lambda_i^+, 0) & \text{for cell left to the edge} \\ \text{Min}(\lambda_i^-, 0) & \text{for cell right to the edge.} \end{cases} \quad (17)$$

The gradients applied in the process of computing the projected values at $t^n + (\Delta t/2)$ are subjected to the monotonicity limiter.

Following the characteristic predictor described above, the full Riemann problem is solved at the edge. The solution of the Riemann problem defines the flux $\bar{Q}^{n+1/2}$ through the edge. The fluxes through the edges of triangles are then integrated (Eq. 5), thus giving an updated value of the variables at t^{n+1} . One of the advantages of the described algorithm is that calculation of the fluxes is done over the largest loop in the system (loop over edges) and can be carried out in the vectorized or parallelized loop. This fact leads to an efficient algorithm.

The algorithm presented is a modification of the algorithm of Ref. 5 which was derived for structured mesh. This algorithm has been applied to simulate a wide range of flow problems and has been found very accurate in predicting the features of the physics. The performance of the algorithm is well documented in Refs. 6-8. The next section, the spatial adaptive procedure, is described in detail. These descriptions include explanations of the error estimator for flow feature detection and the Direct Dynamic Refinement Method used to enrich and coarsen the mesh.

DIRECT DYNAMIC REFINEMENT METHOD FOR ADAPTATION ON AN UNSTRUCTURED TRIANGULAR GRID

The Direct Dynamic Refinement method (DDR) is a new method for adapting unstructured triangular grids during the computational process. As stated, an unstructured grid is very suitable for implementing boundary conditions on complex geometrical shapes as well as the adaptation of the grid, if necessary. The adaptation of the unstructured triangular grid leads to efficient usage of memory resources. The adaptive grid enables the user to capture moving shocks and high gradient flow features with high resolution. The available memory resources can be very efficiently distributed in the computational domain to accommodate the resolution needed to capture features of the physical property of the solution as they are evolved. Dynamic refinement controls the resolution priorities. These priorities can be set according to the physical features that the user wishes to emphasize

in the simulation. The user has control over the accuracy of the physical features resolved in the simulation, without being restricted to the initial grid. The alternative to Direct Dynamic Refinement (DDR) is the hierarchical dynamic refinement (H-refinement) that keeps a history of the initial grid (mother grid) and the subdivision of each level (daughter grids). In the H-refinement method, it is necessary to keep overhead information on the level of each triangle subdivision, and double indirect indexing is needed to keep track of the H-refinement process. As mentioned, the H-refinement relies heavily on the initial grid as it subdivides this grid and returns to it after the passage of the shock.

To minimize the dissipation caused by the interpolation and extrapolation in the refinement and coarsening of the grid, the addition and deletion of point is done in the region where the flow features are smooth. Thus for capturing the shock, the refinement should be applied in the region ahead of the shock. The coarsening of the grid is done in the flow regions where the gradients of the flow parameters are small.

In the present version of AUGUST (Adaptive Unstructured Godunov Upwind Second order Triangular), we implemented an algorithm with multiple criteria for capturing a variety of features that might exist in the physics of the problem to be solved. To identify the location of a moving shock, we use the flux of total energy into triangles. The fluxes entering and leaving triangles are the most accurate physical variables computed by the Godunov

algorithm for solving Euler's equations, and are used to update the physical variables for each time step in each triangle. Supplementary to the flux of energy as an error indicator, we use the flux of total density into triangles and the density gradient. The error indicator is the only sensor that is solely responsible for identifying the area to be refined or coarsened in the computational domain. As such, the error indicator should be sensitive enough to detect physical features that are of interest to the user, such as shock waves, rarefaction waves, slip lines and vortices. The error indicators that are implemented in the code are able to sense very weak slip lines in the presence of strong shock waves. The ability of the error indicators to identify weak physical features in the presence of strong ones, without picking up numerical noises, is essential to the simulation of adaptive grids. As stated, the quality of the results is as good as the error indicators applied. If the error indicators fail to identify the physical feature, this feature probably will be overlooked in the simulated results. It should be noted that the process of applying error indicators for identifying the areas to be adaptively refined or coarsened is an expensive loop that has to check the whole triangles table in the simulation. Thus, the error indicators are applied each 9 to 15 time steps. This process is preceded by application of an algorithm that refines a buffer zone ahead of the features and coarsens the grid after it was moved away. The buffer zone ahead of the feature is identified by using a search pattern of finding the neighbors of the flagged triangles sorted by the error indicators.

We are not applying any physical parameters to identify the zones "ahead."

The refinement algorithm follows several basic steps. The process of adding points to refine the grid locally is done by either adding a new vertex in the baricenter of the triangle or adding a new vertex in the middle of the edge. Adding a new vertex in the baricenter of a triangle is very efficient in the sense that the refinement affects this individual triangle only. We apply this process exclusively for refinement. As a supplement, especially on the boundary, we apply the method of adding a new vertex on an edge. As a complement to adding new vertices, we apply the reconnection/swapping algorithm that flips the diagonal (common edge) of two adjacent triangles to improve the quality of the triangles constructed. Figure 2 displays a chain of those basic steps to illustrate the refinement process. Figure 2a shows the original grid. Figure 2b illustrates a one step scheme refinement in which a new vertex is introduced into a triangular cell forming three cells (two new ones). On the boundary edges, a new vertex is introduced in the middle of those edges to form two cells (one new one). This refinement is followed by reconnection that modifies the grid as demonstrated in Fig. 2c. The process of refinement and reconnection can be continued until the necessary grid resolution is achieved. As an example, another loop of refinement is illustrated in Figs. 2d and 2e. This direct approach to grid refinement provides extreme flexibility in resolving local flow features.

A similar direct approach is applied to grid coarsening. The basic step

in this process is deleting the cells and edges associated with a vertex to be removed, as shown in Fig. 3b. During the second step, this void in the grid is filled with new larger triangles (Fig. 3c) without introducing new vertices. The last step is local reconnection and relaxation as shown in Fig. 3d. The relaxation procedure is a simple relocation of the vertex moved to the center of the polygon surrounding this vertex (only if the polygon is a convex).

The algorithm of direct dynamic refinement proved to be very efficient in refining and coarsening the grid adaptively. The refinement and coarsening followed a short inquiry on the quality and shape of the triangle flagged and its close neighbors. Since we do not keep any history or tree for each triangle, the DDRM algorithm has much less checking to do as compared to the H-refinement algorithm. The vectorization and parallelization of the solver is straightforward.

NUMERICAL RESULTS FOR THE TWO DIMENSIONAL TEST PROBLEM

We have tested the Second Order Godunov algorithm in a variety of flow simulations ranging from the low subsonic to the high hypersonic Mach ⁽⁶⁻⁸⁾ regime. The AUGUST code proved to be very robust and accurate. The results obtained are comparable to or better than those obtained applying leading flow solvers in all of the regimes tested.

To validate our DDRM implemented in the AUGUST code, we simulated the problem of interaction of a Mach 2.85 planar shock wave, propagating

in a channel with a 45° symmetrical double ramp. Figure 4 shows the experimental interferogram of the problem to be simulated (reproduced). The example that we chose to simulate is most appropriate to test the performance of an adaptive algorithm. The experimental results show a complex flow pattern containing a mix of strong discontinuities, as shock waves, and very weak features such as slip lines, vortices, and rarefaction waves. The error estimator must recognize and flag all these features for refinement. The error estimator should be sensitive enough to identify very weak slip lines without picking up numerical noises present in the simulation. We have simulated the shock wave reflection and diffraction over a 45° corner at the conditions that correspond to the experimental result shown in Fig. 4. Here we present results for several shapes of the flow evolution. The flow in the channel is from left to right. Figure 5 displays density contour plots after the shock passed the apex of the double wedge obstacle. In Fig. 5a, the density contours are overlaid on the grid used at this stage of the evolving flow. For clarity, only the density contours are displayed in Fig. 5b. The grid displayed in Fig. 5a shows how well the adaptation technique follows the high activity region in the flow. The grid is adapting to regions with high pressure gradients and high density gradient. In Fig. 5a, one can observe high quality grid produced by the DDR method. The shock has a relatively thin buffer zone ahead of its front, allowing us to avoid the interpolations related to grid adaptation of the flow variables in the area of high gradient.

The flow features are resolved accurately, and the contact discontinuity and triple point are clearly defined.

Figure 6 shows the density contours at a later time in the same format as in Fig. 5. This figure demonstrates the ability of the DDRM to identify and follow flow features in the computational domain. In this figure we can observe a complicated flow pattern developing as a result of interaction of the rarefaction wave with the complex pattern of shock waves. A recompression shock and a strong vortex that are developed in this time frame are well resolved. We can also observe a slip line originating at the triple point. The adaptation algorithm, as in the previous time frame, follows both shock waves and contact discontinuities.

Figure 7 displays the density contours at the stage comparable to that shown in Fig. 4 for the experimental results. The computed results as displayed in Fig. 7b show a flow pattern similar to the experiment. The slip line and the formation of vortices along it are clearly depicted. The shock and reflected shock as well as the recompression shock are very sharply defined with very low numerical noise. The vortex developed after the compression shock is distinctly displayed. A new reflected shock can be seen developing at the channel wall behind the double wedge.

The results shown in Figs. 5-7 display the ability of the algorithm to simulate a complex transient flow problem on dynamically adapting grid. The error estimates used in our algorithm allow detection of strong and

weak shock waves, conducted discontinuities, vortices or other fronts that need enhanced resolution.

CONCLUSION

The Direct Dynamic Refinement (DDR) method was developed and tested for a challenging problem of reflection and diffraction of a strong shock over a double ramp. For this test problem we have demonstrated that a set of error indicators developed for the DDR allow capturing strong and weak features of the complex wave structure developing in this test case.

The above described algorithms were implemented in the AUGUST code. The AUGUST code was used for a range of subsonic, transonic, and supersonic transient and steady problems. For all these conditions the AUGUST code produced robust results with the error indicators proving to be applicable for all these diverse flow regimes.

ACKNOWLEDGMENT

The work reported here was partially supported by DARPA and AFOSR under Contract #F49620-89-C-0087. The authors would like to thank Col. James Crowley and Dr. A. Nachman for their interest in this project.

REFERENCES

- 1) R. Lohner, "An Adaptive Finite Element Scheme for Transient Problems in CFD," *Comp. Meth. Appl. Mech. Eng.*, Vol. 61, 323-338, 1987.

- 2) R.D. Rausch, J.T. Batina, H.T.Y. Yang "Spatial Adaptation of Unstructured Meshes for Unsteady Aerodynamic Flow Computation." AIAA Journal, Vol 30, No. 5, May 1992, pp. 1243-1251.
- 3) B. Van Leer. "Toward the Ultimate Conservative Difference Scheme. V. A Second Order Sequel to Godunov Method." J. Comp. Phys., 32, 101-136, 1979.
- 4) P. Collela and H.M. Glaz, "Efficient Solution Algorithm for the Riemann Problem for Real Gases," J. Comp. Phys. 59, 264-289, 1985.
- 5) S. Eidelman, P. Collela, and R.P. Shreeve. "Application of the Godunov Method and its Second Order Extension to Cascade Flow Modeling," AIAA Journal 22, 10, 1984.
- 6) I. Lottati, S. Eidelman, and A.T. Drobot, "A Fast Unstructured Grid Second Order Godunov Solver (FUGGS)," AIAA 90-0699, 27th Aerospace Sciences Meeting, Reno, Nevada, 1989.
- 7) I. Lottati, S. Eidelman, and A.T. Drobot, "Solution of Euler's Equations on Adaptive Grids Using a Fast Unstructured Grid Second Order Godunov Solver," Proceedings of the Free Lagrange Conference, Jackson Lake, WY, June 1990.
- 8) I. Lottati and S. Eidelman, "Second Order Godunov Solver on Adaptive Unstructured Triangular Grid," Proceedings of 4th International Symposium on Computational Fluid Dynamics, Davis, CA, September 1991.

9. D.L. Zhang, I.I. Glass. "An Interferometric Investigation of the Diffraction of Planar Shock Waves Over a Half-Diamond Cylinder in Air."
NTIAS Report No. 322, March 1988.

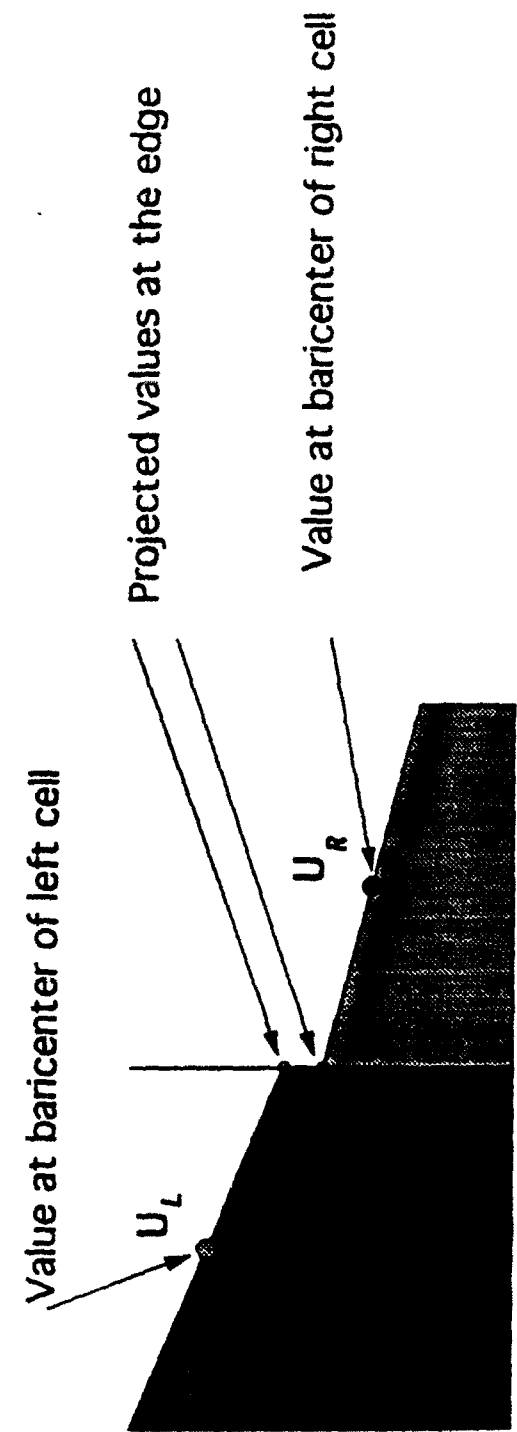
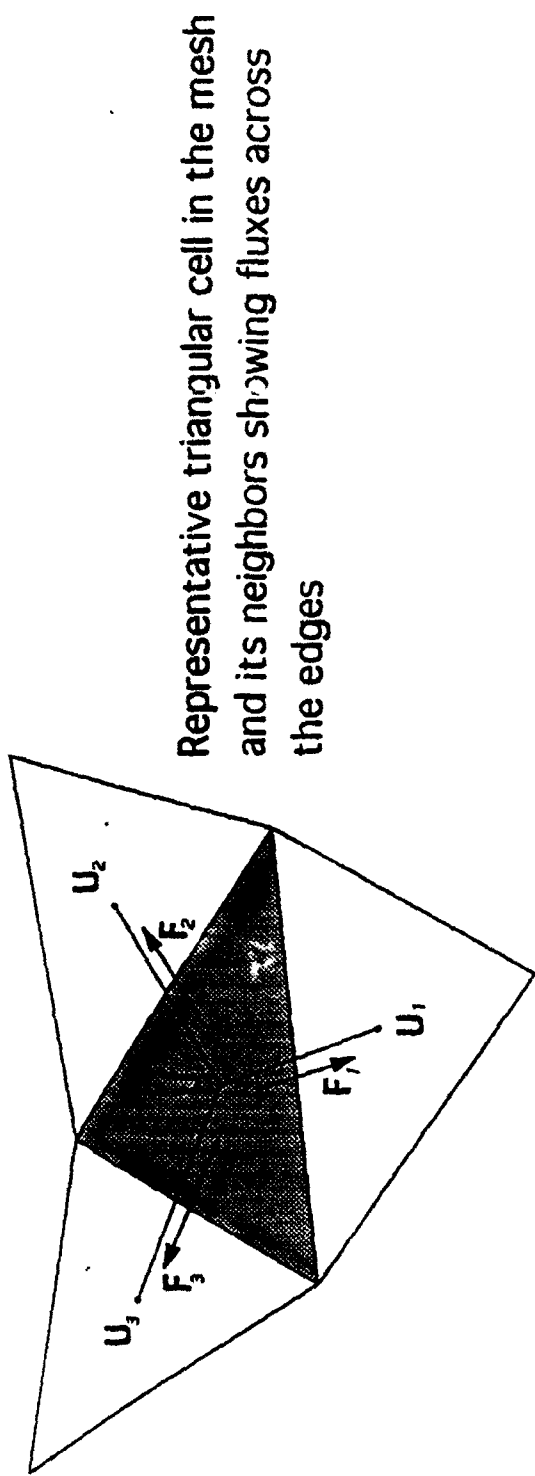
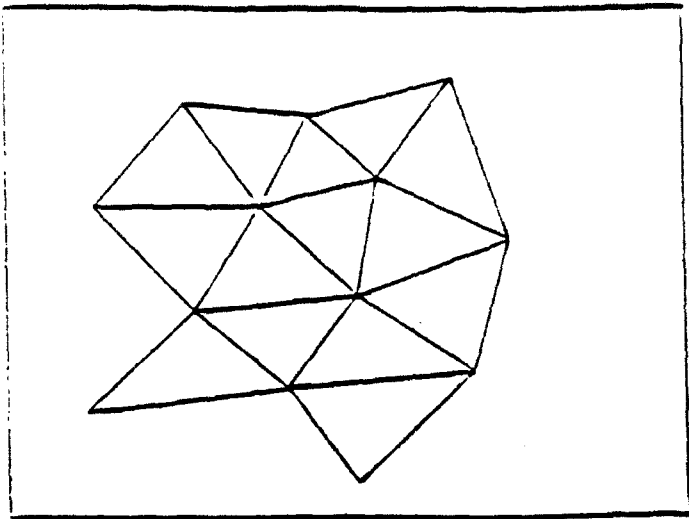
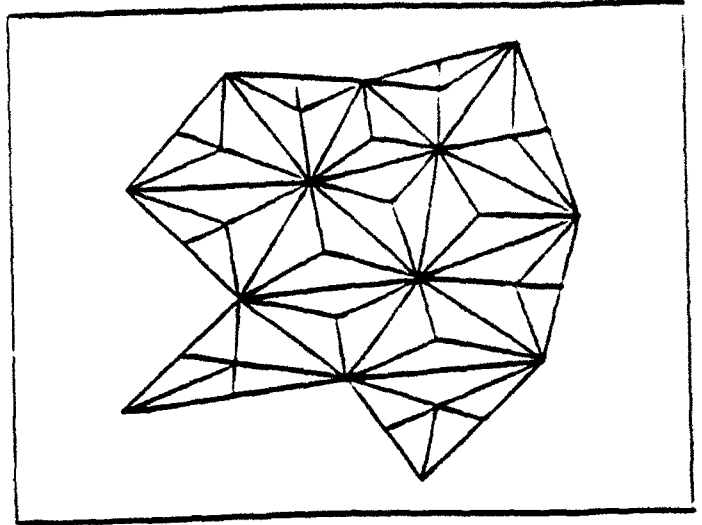


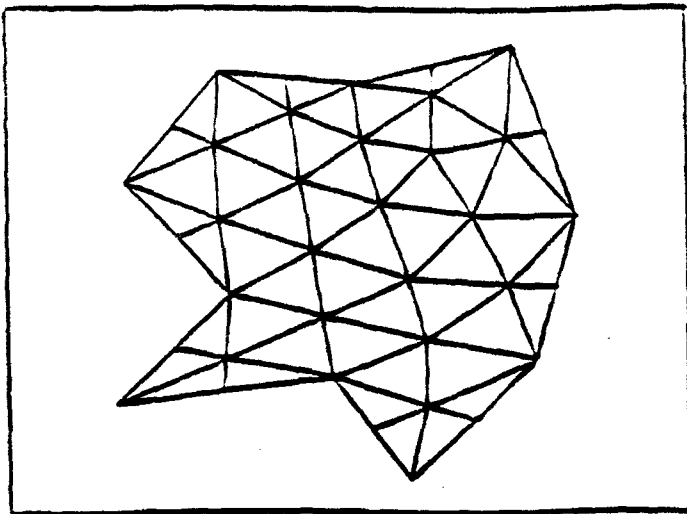
Figure 1. Representative triangular cell in the mesh showing fluxes and projected values.



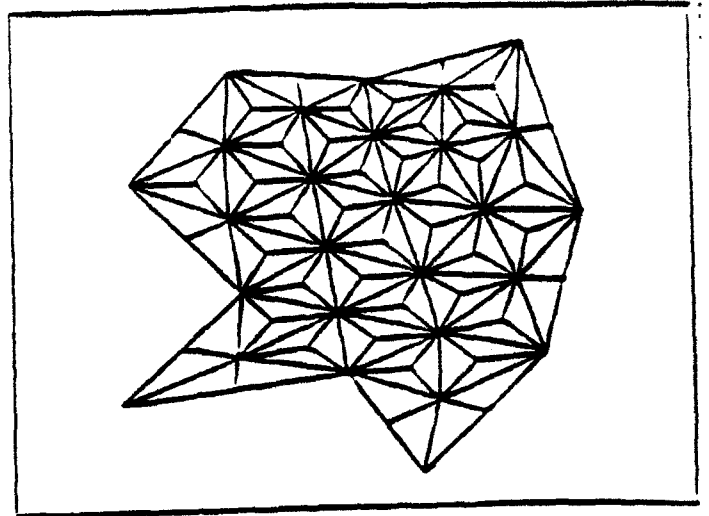
a. Original grid.



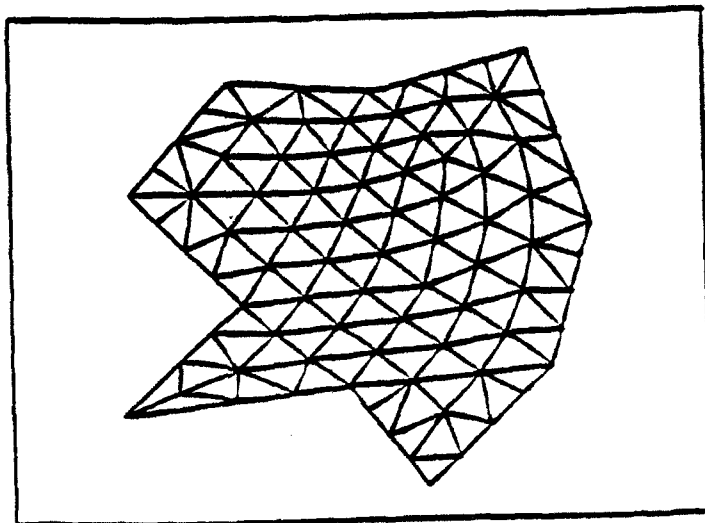
b. Grid after one refinement.



c. Grid after one refinement
and one reconnection.

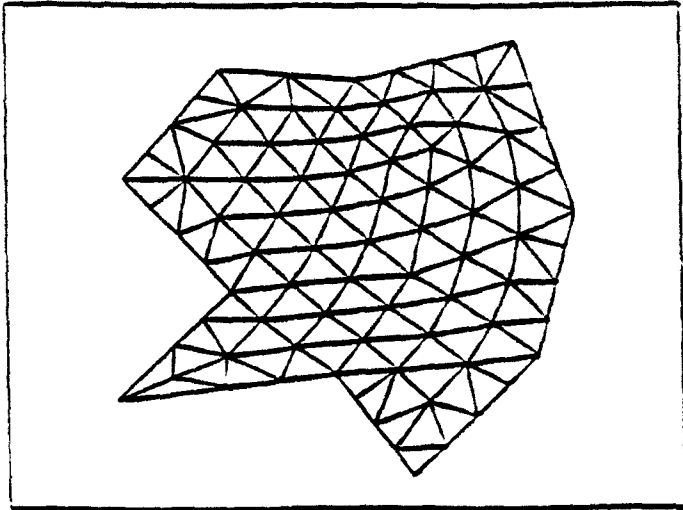


d. Second refinement.

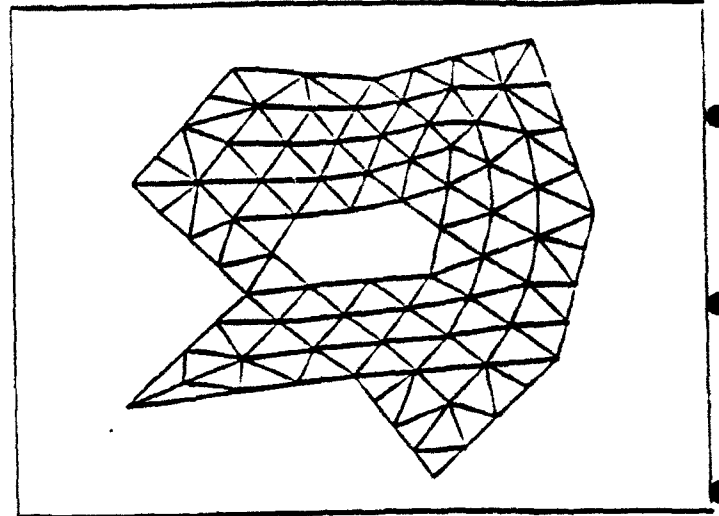


e. Second reconnection.

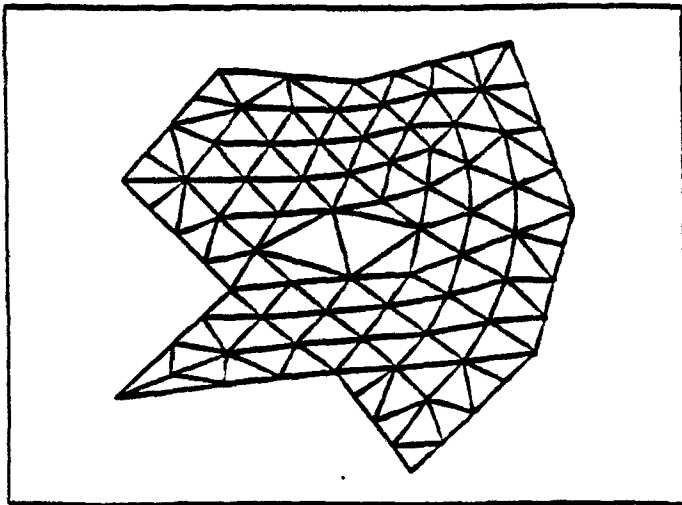
Figure 2. Illustration of the grid refinement process.



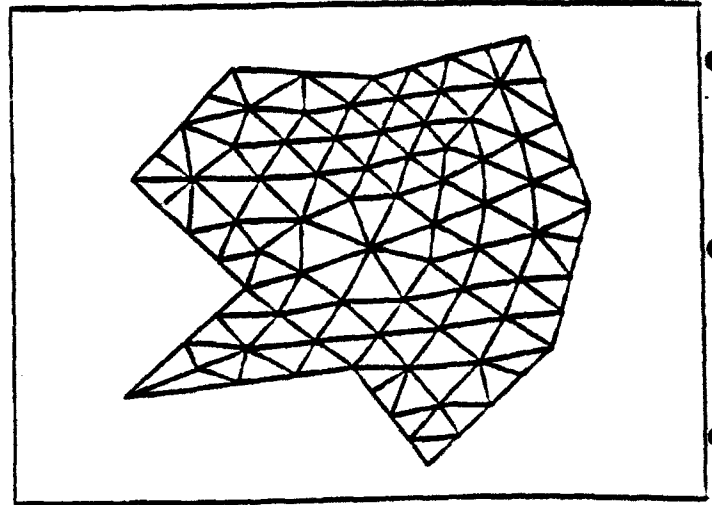
a. Original grid.



b. Point removal.



c. Constructing of new cells.



d. Grid after reconnection and relaxation.

Figure 3. Illustration of the grid coarsening process.

Region	ρ/ρ_0
(0)	1.00
(1)	3.70
a	1.80
b	2.56
c	3.32
d	4.08
e	4.84
f	5.60
g	6.36
h	7.12
i	6.74
j	6.36
k	7.12
l	7.50
m	7.88

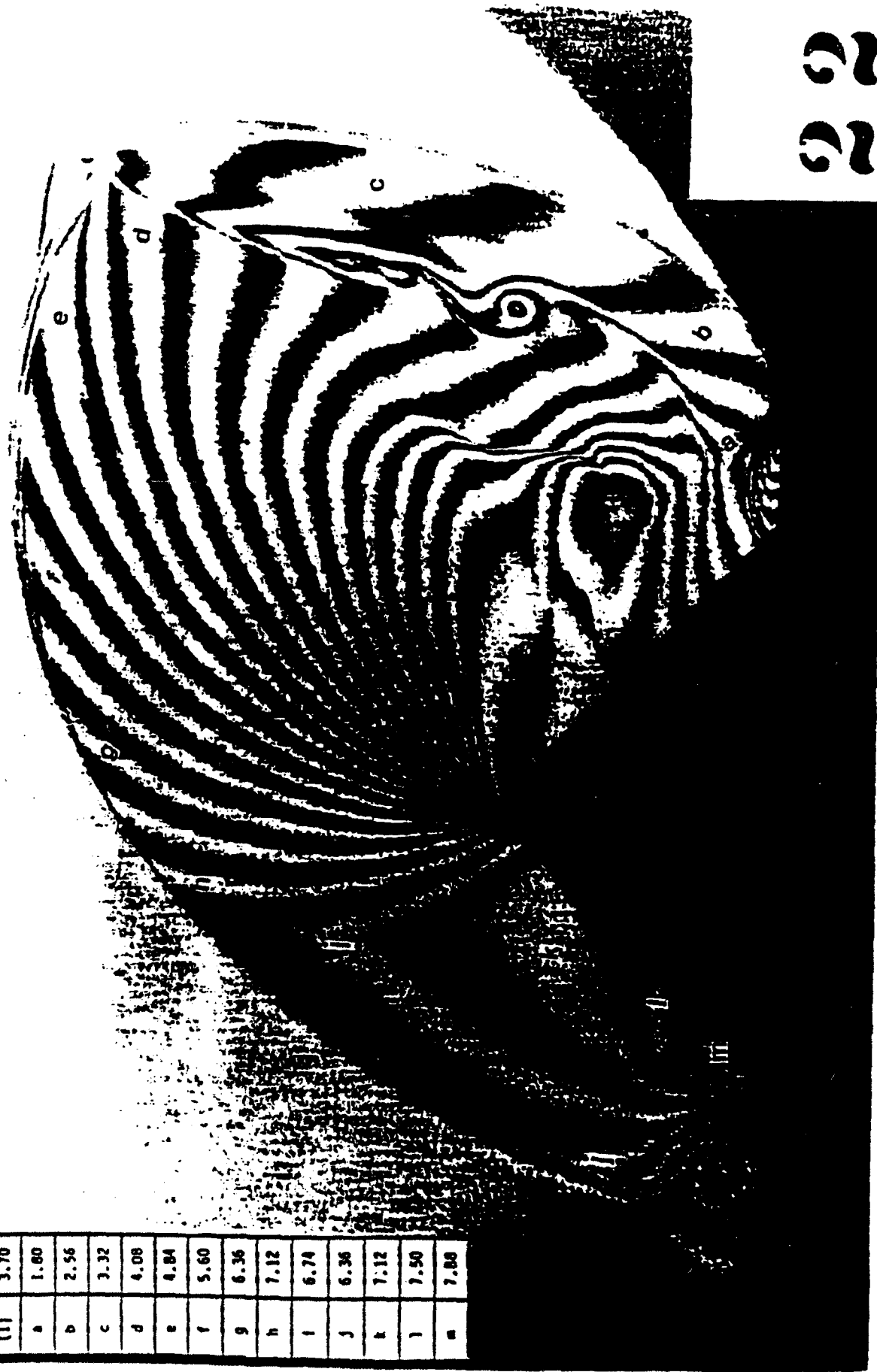


Figure 4. An experimental interferogram taken at $96 \mu\text{s}$ after shock wave hits a diamond shaped obstacle, Mach $M_0 = 2.85$.

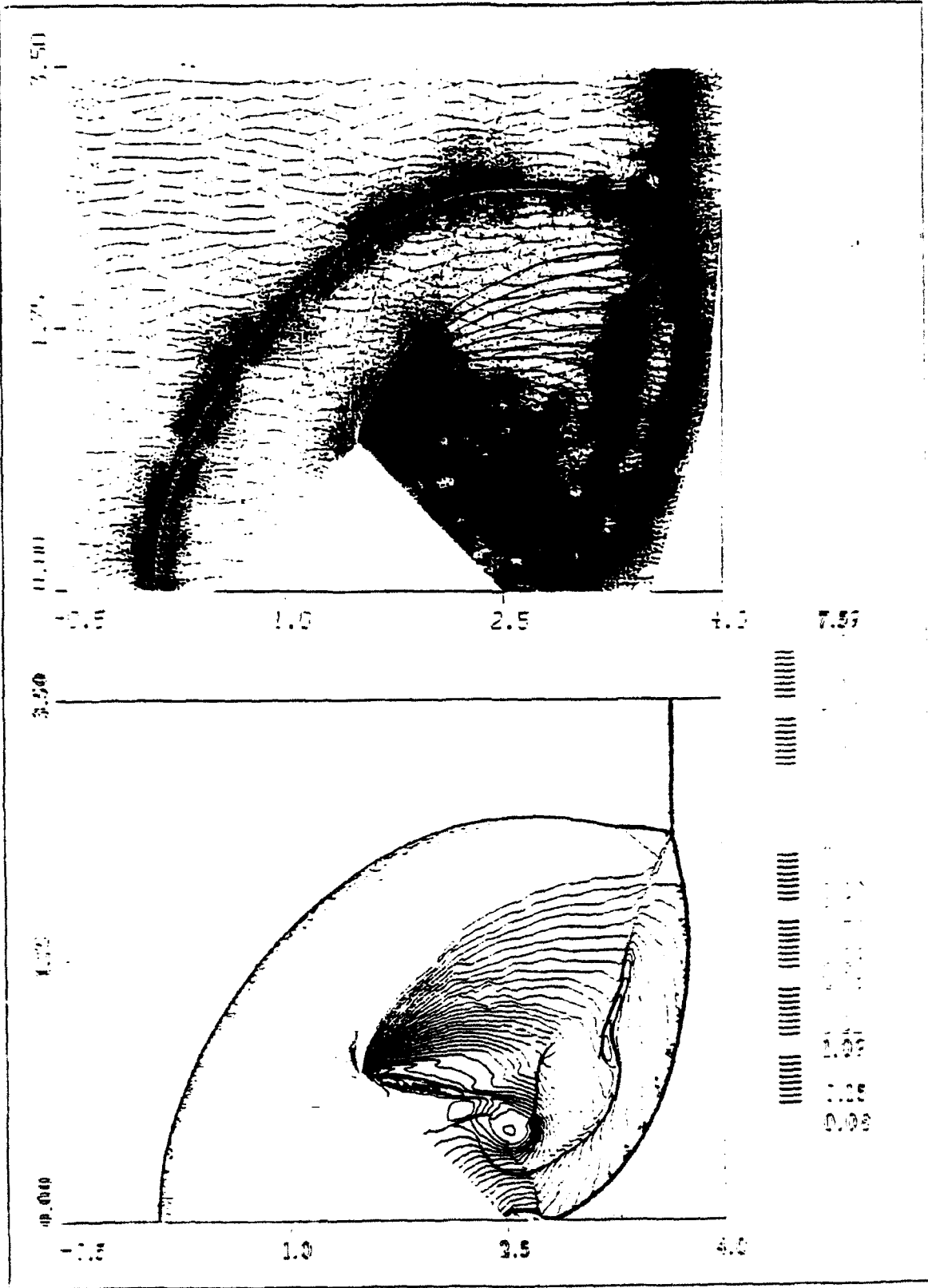


Figure 7. Computed density contours comparable to time of the experimental results shown in Fig. 4. The grid is composed of 79352 vertices.

DECOMPOSITION BY STRUCTURED/UNSTRUCTURED COMPOSITE GRIDS FOR EFFICIENT INTEGRATION IN DOMAINS WITH COMPLEX GEOMETRIES

Itzhak Lottati and Shmuel Eidelman
Applied Physics Operation, MS 2-3-1
Science Applications International Corporation
McLean, VA 22102

Abstract

The Second Order Godunov method has been simultaneously implemented on both unstructured triangular and structured rectangular grids. This combined structured/unstructured method is a much more efficient approach to domain decomposition as compared to the separate application of each method. Application of this new technique to the complex problem of acoustic wave focusing in an ellipsoid reflector has demonstrated its advantages over both structured and unstructured methods of domain decomposition. It has been shown that the complex pattern of acoustic waves propagates seamlessly through structured/unstructured grid interfaces without reflection or distortion. The new approach provides ultimate flexibility in domain decomposition with minimum penalty in terms of memory and CPU requirements, and at the same time capitalizes on the advantages of both structured and unstructured grid methods.

Introduction

Structured rectangular grids allow the construction of numerical algorithms that perform an efficient and accurate integration of fluid conservation equations. The efficiency of these schemes results from the extremely low storage overhead needed for domain decomposition and the efficient and compact indexing that also defines domain connectivity. These two factors allow code construction based on a structured domain decomposition that can be highly vectorized and parallelized. Integration in physical space on orthogonal and uniform grids produces the highest possible accuracy of the numerical algorithms. The disadvantage of structured rectangular grids is that they cannot be used for decomposition of computational domains with complex geometries.

The early developers of computational methods realized that, for many important applications of Computational Fluid Dynamics (CFD), it is unacceptable to describe curved boundaries of the computational domain using the stair-step approximation available with the rectangular domain decomposition technique. To overcome this difficulty, the techniques of boundary-fitted coordinates were developed. With these techniques, the computational domain is decomposed on quadrilaterals that can be fitted to the curved domain. The solution is then obtained in the physical space using the geometrical information defining the quadrilaterals, or in the computational coordinate system that is obtained by transformation of the original domain into a rectangular domain. The advantage of this technique is that it employs the same indexing method as the rectangular structured domain decomposition methods that also serve to define domain connectivity. The boundary fitted coordinates approach leads to efficient codes, with approximately a 4:1 penalty in terms of memory requirement per cell as compared with rectangular domain decomposition. However, this approach is somewhat restricted in its domain decomposition capability, since distortion or large size variations of the quadrilaterals in one region of the domain lead to unwanted distortions or increased resolution in other parts of the domain. An example of this is the case of structured body fitted coordinates that are used for simulations of flows over a profile with sharp trailing edges. In this case, increased resolution in the vicinity of the trailing edge leads to increased resolution in the whole row of elements connected to the trailing edge elements.

The most effective methods of domain decomposition developed to overcome this disadvantage are those using unstructured triangular grids. These methods were developed to cope with very complex computational domains. The unstructured grid method, while efficient and powerful in domain decomposition, results in codes that must store large quantities of information defining the grid geometry and connectivity, and have large computational and storage overheads. As a rule, an unstructured grid code requires greater storage by a factor of 10, and will run about 5 times slower when compared on a per cell per iteration basis with a structured rectangular code.

Unstructured triangular meshes are designed to provide a grid that is fitted to the boundary of complex geometry. The flexibility of the unstructured mesh that allows gridding complex geometry should be weighed against the huge memory requirement needed to define the inter connectivity between the triangles. To cut down on the memory overhead, unstructured grid methods are used to their best advantage when combined with grid adaptivity. This feature usually allows the dynamic reallocation of triangles according to the physics and geometry of the problem solved, which leads to a substantial reduction in the number of cells needed for the domain decomposition. However, this advantage is highly dependent on the problem solved. Adaptive unstructured grids have an advantage over the unadaptive unstructured domain decomposition if the area of high resolution needed is around one-tenth of the global area of the computational domain. As a result, while the adaptive unstructured method may be extremely effective for simulating flow with multiple shock waves in complex geometries, it becomes extremely inefficient when high resolution is needed in a substantial area of the computational domain.

Our approach to domain decomposition combines the structured and unstructured methods for achieving better efficiency and accuracy. Under this method, structured rectangular grids are used to cover most of the computational domain, and unstructured triangular grids are used only to patch between the rectangular grids (Fig. 1), or to conform to the curved boundaries of the computational domain (Fig. 2). In these figures, an unstructured triangular grid is used to accurately define the curved internal or external boundaries and a structured rectangular grid is used to decompose the regions of the computational domain that have a simple geometry.

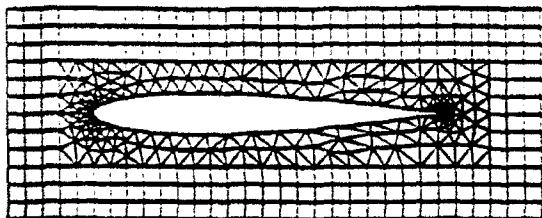


Figure 1. A possible candidate configuration for hybrid structured/unstructured domain decomposition.

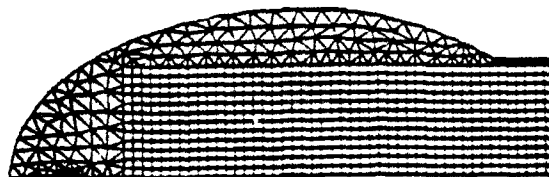


Figure 2. A possible candidate configuration for hybrid structured/unstructured domain decomposition, representing the ellipsoid reflector grid used for the numerical simulation.

Our paper will illustrate the performance gains achieved from the use of this composite grid decomposition approach. We apply the Second Order Godunov method to solve the Euler equations on both structured and unstructured sections of the grid. The challenging problem of acoustic wave focusing in an ellipsoid is used as a test case to confirm the soundness of the approach and to check its performance characteristics and accuracy.

Mathematical Model and Integration Algorithm

We consider a system of two-dimensional Euler equations written in conservation law form as:

$$\frac{\partial U}{\partial t} + \frac{\partial F}{\partial x} + \frac{\partial G}{\partial y} = 0 \quad (1)$$

where

$$U = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ e \end{bmatrix}, \quad F = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ u(e + p) \end{bmatrix}, \quad G = \begin{bmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ v(e + p) \end{bmatrix}$$

Here u, v are the x, y velocity vector components, p is the pressure, ρ is the density and e is total energy of the fluid. We assume that the fluid is an ideal gas and the pressure is given by the equation-of-state

$$p = (\gamma - 1)(e - 0.5\rho(u^2 + v^2)) \quad (2)$$

where γ is the ratio of specific heats and typically taken as 1.4 for air. It is assumed that an initial distribution of the fluid parameters is given at $t = 0$, and the boundary conditions defining a unique solution are specified for the computational domain.

The system of governing equations in Eq. (1) can be written as

$$\frac{\partial U}{\partial t} + \nabla \cdot Q = 0 \quad (3)$$

where Q represents the convective flux vector. By integrating Eq. (3) over space and using Gauss' theorem, the following expression is obtained

$$\frac{\partial}{\partial t} \int_{\Omega} U dA + \oint_{\partial\Omega} Q \cdot dl = 0 \quad (4)$$

where $dl = n dL$, n is the unit normal vector in the outward direction, and dL is a unit length on the boundary of the domain. The variable Ω is the domain of computation and $\partial\Omega$ is the circumference boundary of this domain.

Eq. (4) can be discretized for each element (cell) in the domain

$$\frac{(U_i^{n+1} - U_i^n)}{\Delta t} A_i = \sum_{j=1}^M Q_j^n n_j \Delta L_j \quad (5)$$

where A_i is the area of the cell; Δt is the marching time step; U_i^{n+1} and U_i^n are the primitive variables at the center of the cell at time n and at the update $n+1$ time step; Q_j is the value of the fluxes across the M boundaries on the circumference of the cell where n_j is the unit normal vector to the boundary edge j , and ΔL_j is the length of the boundary edge j . The fluxes Q_j^n are computed applying the Second Order Godunov algorithm, and Eq. (5) is used to update the physical primitive variables U , according to computed fluxes for each marching time step Δt . The marching time step is subjected to the CFL (Courant-Frerichs-Lewy) constraint.

We seek a solution to the system of Eq. (1) in the computational domain, which is decomposed in part into triangles with arbitrary connectivity and in part into rectangles using a logically structured grid. We use the advantage of the unstructured grid (Refs. 1-4) to describe the curved boundary of the computational domain and areas that need increased local resolution. In our example, the unstructured grid covers 10% of the total computational domain while the structured grid occupies the remaining 90%. The numerical technique for solving Euler's equation on an unstructured grid is described in Refs. 5-7, and the technique for the structured grid is described in Ref. 8. These numerical techniques apply some of the ideas that were introduced in Refs. 9-10. The structured and unstructured codes apply the center-based formulation, i.e., the primitive variables are defined in the center of the cell, which makes the cell the integration volume, while the fluxes are computed across the edges of the cell. The basic algorithmic steps of the Second Order Godunov method can be defined as follows:

1. Find the value of the gradient at the baricenter of the cell for each gas dynamic parameter U ;
2. Find the interpolated values of U at the edges of the cell using the gradient values;
3. Limit these interpolated values based on the monotonicity condition (Ref. 9);
4. Subject the projected values to the characteristic's constraints (Ref. 10);
5. Solve the Riemann problem applying the projected values at the two sides of the edges;
6. Update the gas dynamic parameter U according to the conservation equations (1) applying to the fluxes computed and the current time step.

As was advocated in Ref. 7, we prefer the triangle center-based over the vertex-based version of the code. For the same unstructured grid, a triangle-based algorithm will result in smaller control volumes than a vertex-based. In addition, for the Second Order Godunov solver, implementation of the boundary conditions is more straightforward and accurate for the center-based algorithm than in the vertex-based. These two factors, along with the effects of grid connectivity, strongly affect the algorithm accuracy and

performance, and are the main reasons for the superiority of the center-based version over the vertex version.

Sound Wave Focusing in an Ellipsoid Reflector

Research relating to focusing of shock and acoustic waves is of considerable practical interest for application to Extracorporeal Shock Wave Lithotripsy (ESWL). Most of the interest in this area is related to acoustic waves in liquids; however, the basic reflection and focusing mechanisms for a given reflector geometry can be studied in air as well. For our test simulation, we chose a deep reflector shaped like an ellipsoid, which was used for ESWL by Dornier (Ref. 11) and other companies. A schematic of the cross section of this reflector is shown in Fig. 3. Strong acoustic waves are generated in the left focal point of the ellipsoid by an instantaneous release of energy and are refocused at the right focal point. Ideally, focusing should be based on waves of acoustic intensity, since the nonlinear reflections of strong shock waves lead to significant distortions in wave propagation and impair simple geometrical focusing.

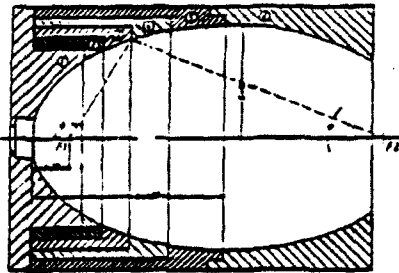


Figure 3. A schematic drawing of the center cross section of the ellipsoid reflector.

Figure 2 shows the computational domain and grid for the ellipsoid reflector example. In order to illustrate the concept of the composite structured/unstructured grid, we have shown only every 1/16 cell of the grid that was actually used for the simulation. In this example, we observe that the structured rectangular grid covers about 90% of the computational domain and the unstructured triangular grid is restricted to the curved surface of the ellipsoid and covers about 10% of the domain. The major axis of the ellipsoid is 150 mm and the minor axis is 90 mm.

The integration in the structured part of the domain is performed using a version of the split Second Order Godunov method described in Ref. 8. For the unstructured triangular grid, we used our implementation of the Second Order Godunov method that includes a compact integration stencil suitable for unstructured grids (Refs. 5-7). In the current implementation, the two sections of the grid communicate through the boundary conditions at their interfaces. According to this, the values in the mirror points at the grid interfaces for the triangular grid are taken from the computational domain of the structured grid and vice versa. These mirror values are used for calculations of the flux at the interface boundaries. For focusing problem simulations, we used 55188 triangles in the unstructured part of the grid and 141312 (736×192) rectangles in the structured part. It should be mentioned that in order to obtain a uniform grid (i.e., the structured and unstructured grids have the same level of refinement), the unstructured portion of the code was run with adaptivity (adding and deleting vertices). This ability enabled us to match the grid resolution based on cell areas in the structured/unstructured grids while computing the results. The initial grid had a very refined grid at the left focal point to initiate accurately the detonation. This area was coarsened later in the simulation by turning on the adaptive capability of the unstructured code.

We used the following initial condition at the time $t = 0$ for the simulation of the acoustic wave focusing:

- Quiescent air in the cavity of the reflector, i.e., Pressure $P_0 = 101350$. Pa and Density $\rho_0 = 1.2 \text{ Kg/m}^3$.
- Blast in the left focal point of the ellipsoid confined in a spherical volume of a radius of $R = 2\text{mm}$. Condition at initial blast area: Pressure $P_b = 45. * P_0$, and Density $\rho_b = 4.5 * \rho_0$.

This definition of the initial conditions guarantees that a weak blast wave will be generated, ensuring that waves of acoustic intensity will be reflected from the wall of the ellipsoid. We examined this particular reflection regime because the blast wave focusing in water occurs in acoustic mode. As it was pointed out in Ref. 11, reflection of even very weak waves in water will lead to considerable deviations from the reflection mode of a pure acoustic wave. However, the purpose of this simulation

is to demonstrate the numerical method and not to study in detail the focusing modes of the ellipsoid reflector. Therefore, we present results for one simulation following conditions outlined above.

In Fig. 4a, the simulation results are shown in the form of pressure contours at the time $t = 1.31 \times 10^{-4}$ sec when the incident shock started its reflection from the reflector wall. Here we can observe that the maximum reflected pressure is no higher than 14% over the ambient pressure, which is consistent with our objective to create weak waves. Figure 4b is an enlargement of the region in the computational domain that contains structure and unstructured grids. We can also observe that the incident wave propagates seamlessly through the interface of the structured and unstructured regions. In Fig. 5, we show pressure contour plots at time $t = 2.09 \times 10^{-4}$ sec. We observe that the interfaces between the two grids carry the information seamlessly.

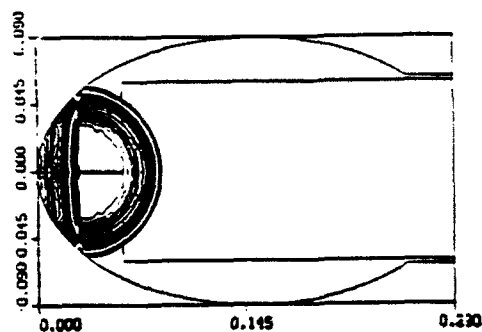


Figure 4a. Pressure contours at time $t = 1.31 \times 10^{-4}$ sec showing the incident wave as reflected from the reflector's wall.

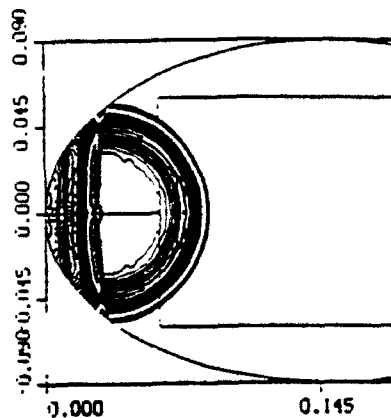


Figure 4b. Blowup of the pressure contours at time $t = 1.31 \times 10^{-4}$ sec showing the matching pressure contours between the structured and the unstructured grid.

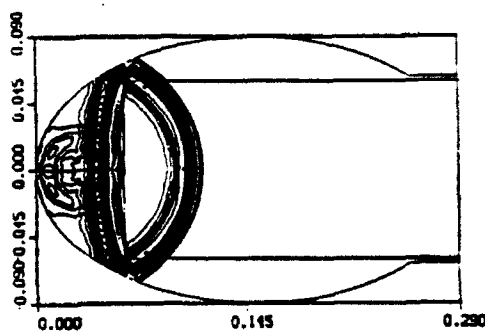


Figure 5. Pressure contours at time $t = 2.09 \times 10^{-4}$ sec showing the incident wave and the reflected wave pattern.

Figure 6 shows the simulation results at time $t = 4.35 \times 10^{-4}$ sec. At this stage, the blast wave front that propagated to the left has undergone full reflection and the reflected wave propagates in the direction of the incident wave to the right. However, the incident and the reflected wave are both of acoustic intensity and they are propagating at the speed of sound. Therefore, the reflected wave will not be able to catch up with the incident wave at this stage of expansion. We can observe in Fig. 7, where the two waves are shown past the ellipsoid centers ($t = 5.41 \times 10^{-4}$ sec), that the distance between these acoustic waves does not change as compared with Fig. 6. The reflected wave has maximum pressure in the vicinity of the axis and its value remains relatively constant (about 1.10×10^5 Pa) through the propagation process. The wave complex at the axis of symmetry consists of the incident acoustic wave front, a reflected wave that has positive followed by negative phases.

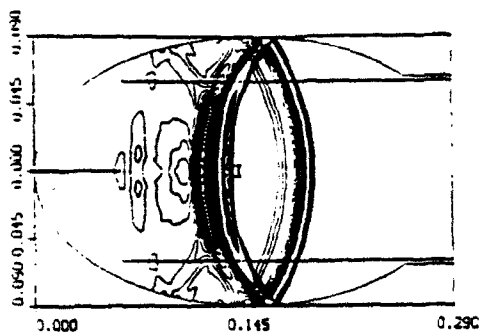


Figure 6. Pressure contours at time $t=4.35 \times 10^{-4}$ sec showing the incident wave and the reflected wave pattern.

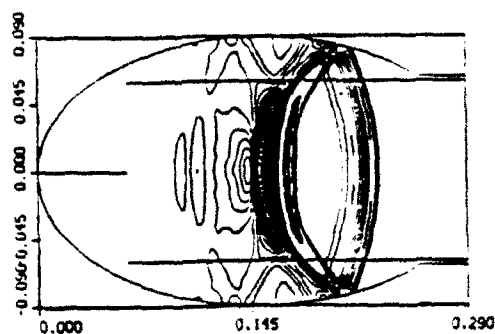


Figure 7. Pressure contours at time $t=5.41 \times 10^{-4}$ sec showing the wave pattern past the center of the ellipsoid.

The enhancement of the reflected wave's amplitude starts gradually when the reflected wave is approaching the second focal point caused by the convergence of the ellipsoid. In Fig. 8, the pressure contours ($t = 8.41 \times 10^{-4}$ sec) are shown at the stage that the maximum focused pressure is obtained in the system. As we can observe in Fig. 8, the incident front has left the computational domain, and the maximum pressure is obtained in small volume in the vicinity of the right focal point. In our simulation, the maximum focused pressure has reached 1.32×10^5 Pa and is located 11 mm to the right of the focal point of the ellipsoid.

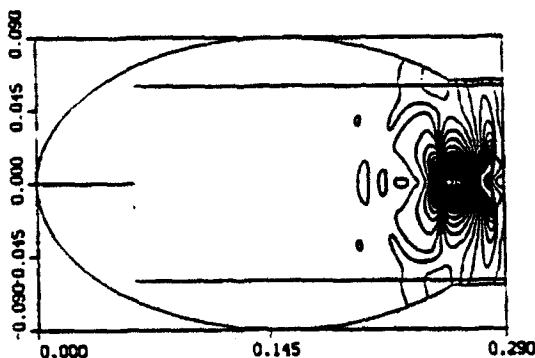


Figure 8. Pressure contours at time $t=8.41 \times 10^{-4}$ sec showing the stage at which the maximum focused pressure is obtained.

In all the figures presented, the method of composite domain decomposition works extremely well, producing seamless solutions at the interfaces. We should mention here that our test problem is particularly sensitive because the main acoustic waves are weak, and any inaccuracy introduced at the grid interfaces would produce a distortion in the phase or in the intensity of the traveling waves that would be a visible disturbance evident in the results needless to mention that an adaptive scheme would have difficulty in simulating this problem due to the weakness of the wave pattern.

Conclusions

A composite method of structured/unstructured domain decomposition is introduced as an efficient technique for dealing with the computational domains of complex geometry. We have simulated a demanding acoustic wave focusing problem and have shown that our approach leads to accurate wave propagation without any reflection or distortion at the structured/unstructured grid interfaces. It should be noted that for the acoustic focusing problem as simulated and presented in this paper, both structured and unstructured methods of domain decomposition can be shown to be inadequate if used separately. The structured method has difficulty describing the curved boundaries of the computational domain, while the unstructured method is totally inefficient in describing phenomena with wide fronts that occupy a large portion of the computational domain. Our hybrid method combines the advantages of structured and unstructured methods of domain decomposition. This hybrid technique combines the efficiency of the unstructured grid to accurately represent curved walls, with the computational and memory efficiency of the structured grid in the majority of the computational domain. We also attribute the quality of the numerical result to the Second Order Godunov method, which allows a consistent, accurate and robust formulation for handling both grids and boundary conditions.

Acknowledgments

The work reported here was partially supported by DARPA and AFOSR under Contract #F49620-S9-C-0087. The authors would like to thank Col. James Crowley and Dr. A. Nachman for their interest in this project.

References

1. A. Jameson, T.J. Baker, and N.P. Weatherill, "Calculation of Inviscid Transonic Flow Over a Complete Aircraft," AIAA 24th Aerospace Sciences Meeting, Reno, NV, AIAA Paper 86-0103, January 1986.
2. R. Löhner, "Adaptive Remeshing for Transient Problems," Comp. Meth. Appl. Mech. Eng. 75, 195-214, 1989.
3. J. Peraire, M. Vahdati, K. Morgan, and O.C. Zienkiewicz, "Adaptive Remeshing for Compressible Flow Computations," J. Comp. Phys. 72, 449-466, 1987.
4. D. Mavriplis, "Accurate Multigrid Solution of the Euler Equations on Unstructured and Adaptive Meshes," AIAA 88-3707, 1988.
5. I. Lottati, S. Eidelman, and A. Drobot, "A Fast Unstructured Grid Second Order Godunov Solver (FUGGS)," 28th Aerospace Sciences Meeting, AIAA-90-0699, Reno, NV, 1990.
6. I. Lottati, S. Eidelman, and A. Drobot, "Solution of Euler's Equations on Adaptive Grids Using a Fast Unstructured Grid Second Order Godunov Solver," Proceeding of the Free Lagrange Conference, Jackson Lake, WY, June 1990.
7. I. Lottati and S. Eidelman, "Second Order Godunov Solver on Adaptive Unstructured Grids," Proceeding of the 4th International Symposium on Computational Fluid Dynamics, Davis, CA, September 1991.
8. S. Eidelman, P. Collela, and R.P. Shreeve, "Application of the Godunov Method and Its Second Order Extension to Cascade Flow Modeling," AIAA Journal 22, 10, 1984.
9. B. van Leer, "Towards the Ultimate Conservative Difference Scheme, V: A Second Order Sequel to Godunov's Method," J. Comp. Phys. 32, 101-136, 1979.
10. P. Collela and P. Woodward, "The Piecewise Parabolic Method (PPM) for Gasdynamic Simulations," J. Comp. Phys. 54, 174-201, 1984.
11. H. Gronig, "Past, Present and Future of the Shock Focusing Research," Proceedings of the International Workshop on Shock Wave Focusing, Sendai, Japan, March 1989.

TWO-PHASE COMPRESSIBLE FLOW COMPUTATION ON ADAPTIVE UNSTRUCTURED GRID USING UPWIND SCHEMES

Xiaolong Yang, Shmuel Eidelman and Itzhak Lottati
Applied Physics Operation, MS 2-3-1
Science Applications International Corporation
McLean, VA 22102

ABSTRACT

A computer program called MPHASE for numerical study of shock wave propagation in a multi-phase, multi-component gas environment is described and applied. The mathematical model of the multi-phase, multi-component system is based on the multi-fluid Eulerian approach. Basically, we consider the two phases (i.e. gas and particle) to be interpenetrating continua: the dynamics of the flow is governed by conservation equations for each phase, and the two phases are coupled by interactive drag force and heat transfer. The code is formulated on unstructured triangular grids.

The numerical solution method is based on the Second Order Godunov Method for the gaseous medium, an upwind integration for the particles, and an implicit integration technique for the gas-particle interaction simulation. In order to produce a solution with high spatial accuracy at minimal computational cost, an adaptive procedure on the unstructured grid is used. The adaptive procedure will automatically enrich the grid by adding points in the high-gradient (or high flow activity) region and by removing points (coarsening the mesh) where they are not needed. This technique allows a detailed study of the complex two-phase shock reflection phenomena, where the effects of momentum and heat exchange between phases will significantly modify the shock structure and shock parameters.

Results will be given from the code validation study for the shock propagation in the dusty gases. The code performance will be illustrated by solving the problem of reflection and diffraction of a plan shock wave over a semicircular cylinder in a dusty gas.

1. THE MATHEMATICAL MODEL AND THE NUMERICAL SOLUTION

Conservation Equations

The mathematical model consists of conservation governing equations and constitutive laws that provide closure for the model. The basic formulation adopted here follows the gas and dilute particle flow dynamics model presented by Soo¹. The following assumptions are used during the derivation of governing equations:

- (1) The gas is air and is assumed to be ideal gas;
- (2) The particles do not undergo a phase change because particles are considered as sand whose phase transition temperature is much higher than the gas temperature considered here;
- (3) The particles are solid spheres of uniform diameter and have a constant material density;
- (4) The volume occupied by the particles is negligible;
- (5) The interaction between particles can be ignored;
- (6) The only force acting on the particles is drag force and the only heat transfer between the two phases is convection. The weight of the solid particles and their buoyancy force are negligibly small compared to the drag force;
- (7) The particles have a constant specific heat and are assumed to have a uniform temperature distribution inside each particle.

Under the above assumptions, distinct equations of continuity, momentum, and energy are written for each phase. The interaction effects between the two phases are listed as the source terms on the righthand side of the governing equation. The two dimensional unsteady conservation equations for the two phases can be written in the vector form in Cartesian coordinates:

$$\frac{\partial U}{\partial t} + \frac{\partial F}{\partial x} + \frac{\partial G}{\partial y} = S. \quad (1)$$

Here U is the vector of conservative variables, F and G are fluxes in x and y direction, respectively, and S is the source term for momentum and heat exchange. The definition of these vectors are:

$$U = \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ e \\ \rho_p \\ \rho_p u_p \\ \rho_p v_p \\ e_p \end{pmatrix}, F = \begin{pmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ u(e + p) \\ \rho_p u_p \\ \rho_p u_p^2 \\ \rho_p u_p v_p \\ u e_p \end{pmatrix}, G = \begin{pmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ v(e + p) \\ \rho_p v_p \\ \rho_p u_p v_p \\ \rho_p v_p^2 \\ v e_p \end{pmatrix}, S = \begin{pmatrix} 0 \\ -f_x \\ -f_y \\ -q - u_p f_x - v_p f_y \\ 0 \\ f_x \\ f_y \\ q + u_p f_x + v_p f_y \end{pmatrix}$$

where ρ, u, v , and e are gas density, velocities, and energy, respectively; ρ_p, u_p, v_p and e_p are particle density, velocities, and energy, respectively; (f_x, f_y) and q denotes drag force components acting on the particles and heat transfer to the particles, respectively. The gas pressure p is related to ρ, u, v and e for by

$$p = (\gamma - 1)[e - 0.5\rho(u^2 + v^2)] \quad (2)$$

where γ is the specific heat ratio. The gas temperature can be found through the equation-of-state for ideal gas

$$p = \rho RT \quad (3)$$

where R is the gas constant.

The particle temperature T_p is calculated through relation

$$e_p = \rho_p c_p T_p + 0.5\rho_p(u_p^2 + v_p^2). \quad (4)$$

The source terms on the righthand side of equation (1) are momentum and heat exchange between gas and particle phases. If we let r_p and ρ_s be the particle radius and material density, respectively, then the drag forces are

$$\begin{pmatrix} f_x \\ f_y \end{pmatrix} = \frac{3}{8} \frac{\rho_p \rho}{\rho_s r_p} C_d [(u - u_p)^2 + (v - v_p)^2]^{1/2} \begin{pmatrix} u - u_p \\ v - v_p \end{pmatrix}. \quad (5)$$

The particle drag coefficient C_d is a function of Reynolds number, Re , which is based on the relative velocity between the gas and particle phases. After testing the drag coefficients given by Sommerfeld² and by Clift et al.³, the following were two adopted:

$$C_d = \frac{24}{Re} (1 + 0.15Re^{0.687}) \text{ for } Re < 800$$

and (5)

$$C_d = \frac{24}{Re} (1 + 0.15Re^{0.687}) + \frac{0.42}{1 + 42500Re^{-1.16}} \text{ for } Re > 800.$$

Here the Reynolds number, Re is defined as

$$Re = \frac{2\rho r_p [(u - u_p)^2 + (v - v_p)^2]^{1/2}}{\mu} \quad (6)$$

Viscosity, μ is calculated at film temperature, namely, $T_f = 0.5(T_p + T)$, and the temperature dependency of the viscosity is evaluated according to Sutherland's law

$$\mu = \mu_r \left(\frac{T}{T_r} \right)^{3/2} \frac{T_r + \Phi}{T + \Phi} \quad (7)$$

where μ_r is the dynamic viscosity of the gaseous phase at the reference temperature and Φ is an effective temperature, called the Sutherland constant.

The rate of heat transfer from gaseous phase to the particle phase is given by

$$Q = \frac{3}{2} \frac{\rho_p}{\rho_g} \frac{\mu C_p}{Pr} Nu (T_g - T_p) \quad (8)$$

where $Pr = \mu c_p / k_g$ is the Prandtl number, and c_p and k_g are the specific heat and thermal conductivity of gas, respectively. The Nusselt number Nu is a function of this Reynolds number and the Prandtl number as given by Drake⁴

$$Nu = \frac{2r_p h}{R} = 2 + 0.459 Re^{0.55} Pr^{0.33} \quad (9)$$

Initial and Boundary Conditions

The geometry of the computational domain is shown in Fig. 1. The initial conditions for gas are $\rho_g = 1.2 \text{ kg/m}^3$ and $p_g = 101.3 \text{ kpa}$, with a coming shock at $x = -0.5$. There are no particles from $-1.0 \leq x \leq 0.0$. From $x \geq 0.0$, particles are initially in thermal and kinematic equilibrium with surrounding gas. The particles that are uniformly distributed in the dusty region have the following parameters for different test problems:

Mass loading, ρ_p : 0.25 kg/m^3 , 0.76 kg/m^3 ;

Mass material density, ρ_s : 2500 kg/m^3 ;

Particle radii, r_p : $10 \mu\text{m}$, $25 \mu\text{m}$, $50 \mu\text{m}$;

Specific heat, c_s : 766 J/kg/K .

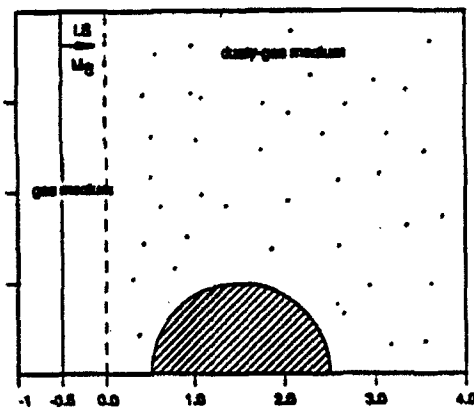


Figure 1. An illustration of the considered flow field.

The lower boundary and cylinder surface are solid walls and assumed adiabatic and impermeable. A reflecting boundary condition is assumed for both the gas and particle phase. Particles are assumed to experience a perfect elastic collision with the wall and reflect from the wall. The right and upper boundaries are open boundaries where a nonreflection boundary condition is used for the gas phase and a zero normal gradient condition is used for particle phase.

Numerical Method of Solutions

The system of partial differential equations described in the previous paragraph is integrated numerically. Equation (1) is repeated here:

$$\frac{\partial U}{\partial t} + \frac{\partial F}{\partial x} + \frac{\partial G}{\partial y} = S. \quad (1)$$

In order to solve this equation numerically, an operator time-splitting technique is used. Assuming that all flow variables are known at a given time, we can calculate its advancement in time by splitting the integration into two stages.

In the first stage, the conservative part of equation (1) is solved:

$$\frac{\partial U}{\partial t} + \frac{\partial F}{\partial x} + \frac{\partial G}{\partial y} = 0. \quad (10)$$

The Second Order Godunov method is used for the integration of the subsystem of equations describing the flow of the gaseous phase (first four components of equation (1)). The method is well documented in literature.^{5,6,7} The subsystem of equations describing the particle phase flow is integrated using a simple finite difference upwind scheme. This is done because there is no shock in the particle phase and the upwind scheme leads to a robust and accurate integration scheme.

In the second stage, the source term is added and the following equation is solved:

$$\frac{\partial U}{\partial t} = S. \quad (11)$$

To integrate this equation in time, we need to obtain S as a function of U . We calculate S through equations (5) to (8).

In order to produce a solution of the high spatial accuracy at minimal computational cost, an unstructured triangular grid with adaptive procedure is used. The adaptive procedure will automatically enrich the mesh by adding points in the high gradient (or high flow activity) region of the flow field and by removing points (coarsening mesh) where they are not needed. The dynamic nature of mesh enrichment is shown in Fig. 3 for two different time frames. One can see that a very fine mesh is generated around shock fronts and other steep density gradient regions.

2. RESULTS

Model Validation for One-Dimensional Shock Wave Propagation in A Dusty Gas

To test the momentum and heat exchange mechanism for the current two-phase model, we first simulate a one-dimensional problem of a normal shock wave propagating into a dusty gas. We numerically simulate the experiments conducted by Sommerfeld². In the experiments, small glass sphere particles of material density $\rho_s = 2500 \text{ kg/m}^3$, specific heat capacity $c_s = 766 \text{ J/kg/K}$, and average diameter of $27 \text{ }\mu\text{m}$ were used as suspension particle phase. The incoming shock, and particle loading ratio $\eta = \rho/\rho_p$, are two varying parameters. The experimental results and our numerical simulation results of shock Mach number as a function of distance for two test cases are shown in Fig. 2a ($\eta = 0.63$) and Fig. 2b ($\eta = 1.4$) for comparison purpose. As one can see, the agreement between the prediction of our present model and the experimental results is very good.

Two-Dimensional Simulation Results of Pure Gas Flow

To test the accuracy of the two-dimensional computation, we compute the pure gas flow case of a shock wave reflection and diffraction over a semicircular cylinder. We then compare the simulation with experimental results. Shock wave reflection on a wedge has been extensively studied by many researchers (see e.g., review paper of Hornung⁸). Shock wave reflection by circular cylinders was numerically simulated by Yang *et al.*⁹ and experiments were performed by Kaca¹⁰. Fig. 3a and 3b show density contours with adapted grids at two moments in time. In Figs. 4a and 4b, the interferogram from the experiment⁹ and density contours from the present simulation are compared for the same flow condition and same time. Note that the density levels are normalized by the ambient gas density in Fig. 4. As one can see from Fig. 4a and Fig. 4b, the results show an excellent quantitative as well as qualitative agreement between the numerical simulation and experimental results.

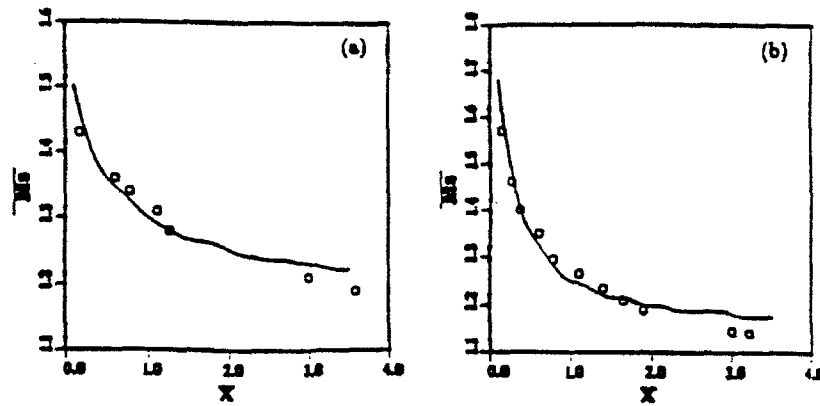


Figure 2. Comparison between computational prediction and experimental measurement of shock wave attenuation for (a) $M_0 = 1.40$, $\eta = \frac{p_2}{p_0} = 0.63$ and (b) $M_0 = 1.7$, $\eta = \frac{p_2}{p_0} = 1.4$ (o experiment, - calculation).

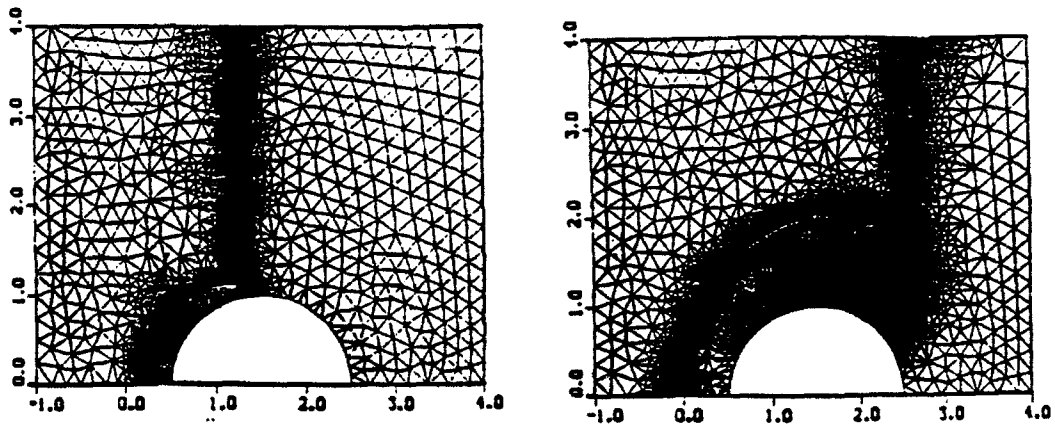


Figure 3. Computed density contours with adapted grid at two different times.

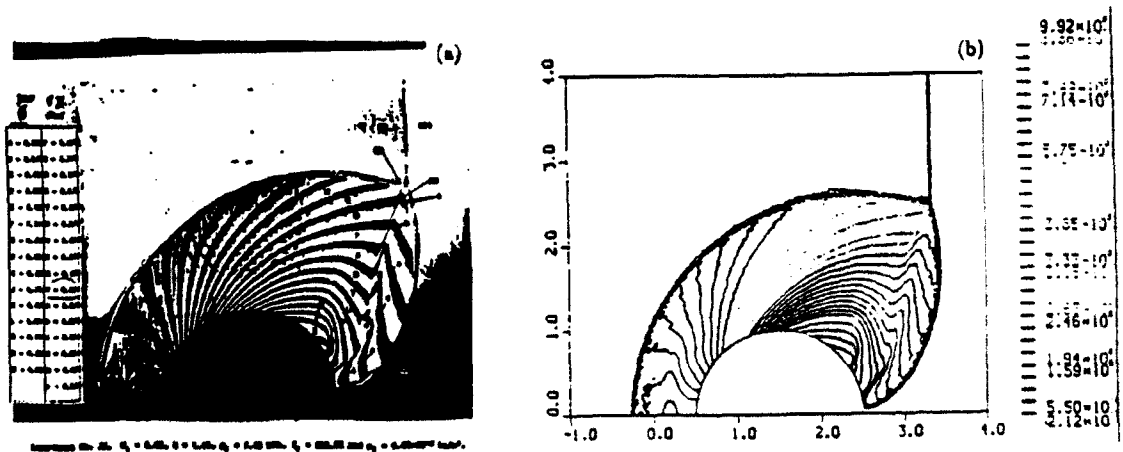


Figure 4. Comparison for $M_0 = 2.80$ gas-only flow, (a) interferogram from experiment conducted by Kaca (1988), (b) density contours from present calculation.

Two-Dimensional Simulation Results of Two-Phase Flow

The basic setup for the two-phase simulation is shown in Fig. 1. Here the planar shock with $M_s=2.3$ impinges on an area of a dusty gas. The interface between clear air and dusty air is located at $x=0.0$ of the computational domain. The area of the dusty air contains a semicylinder with a radius of 1m. The size of the computational domain, initial parameters of the gas, parameters of the incoming shock, size of the semicylinder and its location in the computational domain, are the same as in the reflection and diffraction simulation presented in the previous section.

The main objective of this set of simulations is to study the effects of particle size and particle loading on the parameters of the reflected and diffracted shock waves. It is also of interest to study the dynamics of reflection and diffraction in particle media. This is especially valuable since it is extremely difficult to observe these interactions experimentally in an optically thick dusty gas.

The first set of simulation results is shown for the case with dust parameters $r_p = 10\mu\text{m}$ and $\rho_p = 0.25\text{ kg/m}^3$. The gas parameters and the parameters of the incoming shock wave are the same as in the pure gas case presented above. In Figs. 5a and 5b, particle density contours and gas density contours are shown at the stage when the incident shock wave has reached the top of the semicylinder. At this stage, particles have very little effect on the dynamics and parameters of the shock in the gas phase. The presence of the particles causes a small widening of the shock that is more noticeable for the incident shock. Also, one can observe an additional contour line at the dusty gas/pure gas interface. The particle density contours depict significant piling up of the dust particles at the leading edge stagnation point of the cylinder.

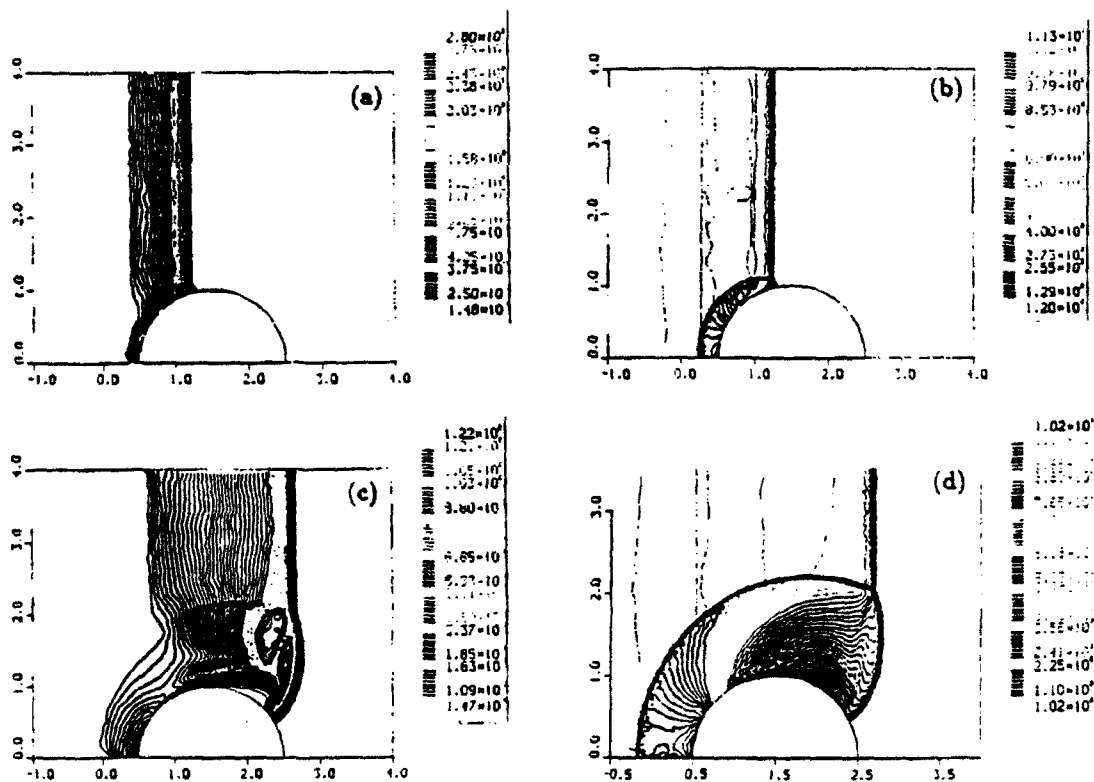


Figure 5. Density contours for the case; $M_s = 2.8$, $\rho_p = 0.25\text{ kg/m}^3$, $r_p = 10\mu\text{m}$ at two different times. (a) particle density at t_1 , (b) gas density at t_1 , (c) particle density at t_2 , and (d) gas density at t_2 .

In Figs. 5c and 5d, the particle density and gas density contours are shown at the stage where significant diffraction has taken place and the shock front is approaching the trailing edge of the cylinder. The small particle loading and small particle size leads to very small modification of the gas shock structure and parameters. One can observe further widening of the shock and some smearing of the slip

line that originates at the triple point. The particle density contours reveal that the particles piled up at the stagnation point were swept by the gas flow to the area of triple point and slip line for the gas flow, leaving a small amount of particles at the leading edge. We should note that this behavior is specific for our problem, where at $t=0$, the dusty gas area was located at $x=0$ and there is no influx of the dust from the left boundary. Also in Fig. 5c, we note that the particles reach a distinct local maxima at the distance about 25 cm behind the main shock front. At this maxima the particle density is 0.86 kg/m^3 , which is more than three times the initial particle density. The particle density reaches a maximum value at the location of the gas slip line. We observe a significant accumulation of the particles that have been moved along the slip line by the shear flow. The larger concentration of particles in the vicinity of triple point is, in fact, the remainder of the particles that have concentrated first at the leading edge and then were swept up with the flow. It is also interesting to observe that an essentially particle-free zone is formed due to the effects of particles slipping over the top of the cylinder and the rarefaction wave behind the cylinder.

3. CONCLUSIONS

In this paper, a computer program for two-phase compressible flow computation on adaptive grids using upwind schemes is described. The following validation study and conclusion can be made.

(1) The validation study for a one-dimensional shock wave propagating in a dusty gas shows a good agreement between the prediction of our model and the results of the experiment.

(2) For a two-dimensional gas-only flow, numerical results agree well with existing experimental data qualitatively and quantitatively, indicating that the gas phase is accurately simulated by adaptive grid technique.

(3) Particles in the gas can have a profound effect on the shock wave reflection and diffraction pattern, which is a function of particle size and loading. The smaller the particle and the lesser the particle loading, the less the inference of particle on the flow field.

(4) There is a particle accumulation behind the "back shoulder" of the semicircular cylinder due to the effect of particles inertia and gas rarefaction wave.

4. ACKNOWLEDGMENTS

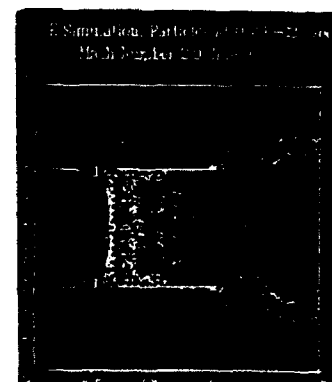
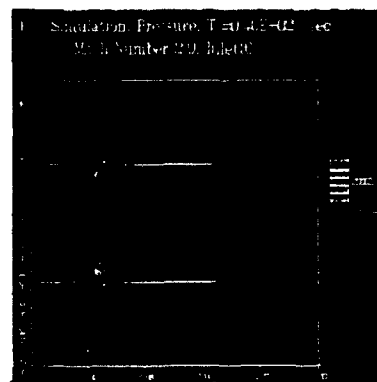
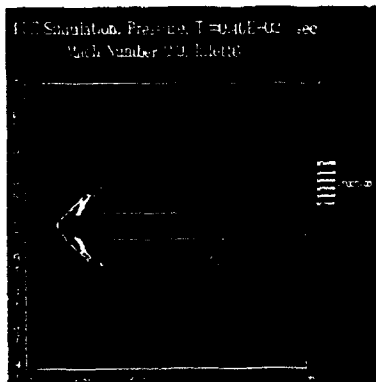
The work reported here was partially supported by DARPA and AFOSR under contract no. F49620-89-C-0087. The authors would like to thank Col. J. Crowley and Dr. A. Nachman for their interest in this project. The authors also acknowledge support from DNA under contract no. DNA001-91-C-0047, and A. Frederickson for his encouragement.

5. REFERENCES

1. S. L. Soo, Particulates and Continuum, Hemisphere Publishing Corporation, 1989.
2. M. Sommerfeld, "The Unsteadiness of Shock Waves Propagating through Gas-Particle Mixtures," Experiments in Fluids, Vol. 3, p. 197, 1985.
3. R. Clift, J. R. Grace, and M. E. Weber, Bubbles, Drops and Particles, Academic Press, New York, 1978.
4. R. M. Drake, Jr., "Discussions on G.C. Vliet and G. Leppert: Forced Convection Heat Transfer from an Isothermal Sphere to Water," Journal of Heat Transfer, Vol. 83, p. 170, 1961.
5. S. Eidelman, P. Collela, and R. P. Shreeve, "Application of the Godunov Method and Its Second Order Extension to Cascade Flow Modelling," AIAA Journal, Vol. 22, p. 10, 1984.
6. P. Collela, "A Direct Eulerian MUSCL Scheme for Gas Dynamics," SIAM J. Stat. Comput., Vol. 6, p. 104, 1985.
7. P. Collela, and H. M. Glaz, "Efficient Solution Algorithms for the Riemann Problem for Real Gases," J. Comput. Physics, Vol. 59, p. 264, 1985.
8. H. Hornung, "Regular and Mach Reflection of Shock Waves," Ann. Rev. Fluid Mech., Vol. 18, p. 33, 1986.
9. J. Y. Yang, Y. Liu, and H. Lomax, "Computation of Shock Wave Reflection by Circular Cylinder," AIAA Journal, Vol. 25, p. 683, 1987.
10. J. Kaca, "An Interferometric Investigation of Diffraction of a Planar Shock Wave over a Semicircular Cylinder," UTIAS Technical Note 269, 1988.



AIAA 92-3168
PULSED DETONATION ENGINE
EXPERIMENTAL AND THEORETICAL REVIEW
S. Eidelman and W. Grossmann
Science Applications International Corporation
McLean, VA



AIAA/SAE/ASME/ASEE
28th Joint Propulsion
Conference and Exhibit
July 6-8, 1992 / Nashville, TN

PULSED DETONATION ENGINE EXPERIMENTAL AND THEORETICAL REVIEW

Shmuel Eidelman and William Grossmann

Applied Physics Operation
Science Applications International Corporation
1710 Goodridge Drive, McLean, VA 22102

Abstract

A Review of past and current research on pulsed detonation engine devices connects early experimental work originating with the V1 pulsejet to recent interest in such propulsion devices. The recent interest has been, in part, stimulated by Aviation Week where sightings of aircraft contrails lead to question if some sort of PDE device has already been developed. This review summarizes what is known about PDEs, makes predictions for applications to realistic flight vehicles including missiles and full scale aircraft, and outlines what is yet required for successful PDE development.

1. Introduction

This paper reviews past and recent theoretical and experimental work related to the Pulsed Detonation Engine (PDE) concept. Such a review is timely since much interest in the PDE concept has been generated from several recent Aviation Week (AW) articles.^{1,2} The AW articles, in addition to describing SAIC PDE studies, describe observations of aircraft flight and engine sound generation that are similar to what would be expected from PDE operation. These observations are intriguing since, to our knowledge, there has been no previously reported use of PDE devices in any past or recent flight vehicles. The reported observations include loud pulsing sounds at Beale AFB and photographs of high altitude contrails with "cotton ball" like beads strung on the contrails in a repetitive pattern. It is tempting to try to connect the AW reports with what we understand about PDE operation. It has come to our attention that a ground observer has identified the frequency of the pulsing sounds emanating from the vehicle that made the contrails appearing in the AW article to be of the order of 50-60 Hertz. To obtain the source (aircraft engine) frequency we must correct the observed frequency for the Doppler effect, taking into account the temperature variation between ground and flight altitude. Assuming an altitude of 45-50,000 ft. (cirrus clouds are observed behind the trails in the published photographs), a flight Mach number

between one and two gives a source frequency, f , between 100 and 200 Hertz. As we show later in this paper, a PDE generating 25,000-50,000 lbs. thrust should, theoretically, operate in this frequency range. Of course we have no information concerning the subject aircraft characteristics and consequently we cannot conclude that PDEs are powering present day aircraft. On the other hand the observations appear to be consistent with expected PDE operation.

2. Early Pulsed Combustion Propulsion Devices

It is instructive to point out the differences between the PDE concept and the more commonly understood pulsejet devices. The first full scale application of pulsed propulsion devices was for the V1 flying "buzz bomb." The engine used for this vehicle was the Schmidt-Argus³ engine and has since been generally referred to as a "pulsejet." The pulsejet for the V1 engine was based on repetitive combustion ignitions accomplished through the use of mechanical reed valves that allowed fresh air charge to be drawn into the combustion chamber. The timing of the reed valve opening was pegged to the acoustical frequency (organ pipe modes) of the combustion chamber, which consisted of a central ignition region joined to an exhaust duct. Thus, the operating parameters of the engine were fixed with engine size; only narrow ranges of thrust level variation are possible in such an engine. An increase or decrease of thrust can only be made through changes in engine internal geometry. Ever since the first occurrence of pulsejets, these engines have been considered for other applications including full scale aircraft propulsion. One of the major obstacles in the early development of the pulsejet for wider applications was the complete absence of a theoretical approach to understanding the thermodynamic process in the combustion chamber. It was assumed that the pulsejet combustion process was similar to the steady-state Lenoir constant-volume cycle and that the frequency of the combustion pulsations could be predicted by means of steady-state acoustical wave motion. However, the efficiency of the pulsejet, as determined experimentally, was much lower than a

constant-volume process would predict. We know now that the early pulsejet devices operated on an approximately constant-pressure cycle, which is known to have a lower thermodynamic efficiency than the constant-volume cycle. We have previously argued that the lack of a firm theoretical understanding of the physics and thermodynamics was primarily responsible for the failure to develop the pulsejet further for a wider range of practical applications. This argument will be discussed again later in this review.

In the meantime, the term pulsejet has become generally understood to refer to a pseudo-generic series of engines. The term "propulsive duct" is a more comprehensive descriptor encompassing a wider range of pulsed combustion engine concepts. An early series of papers by Tharjatt⁴⁻⁶ described the status of work on such devices up to 1965, and provides a guide to the early attempts to understand the physics and aerodynamics of the internal gas flows in them. Even though these early investigations were seriously handicapped by a lack of knowledge of unsteady aerodynamics and the physics of repetitive combustion, it is remarkable that the conclusions offered in Tharjatt's papers are close to what we have concluded over the past several years for the PDE concept. Specifically, it was concluded that the propulsive duct engine concept should theoretically be capable of any desired level of thrust per unit area, with a corresponding reduction in specific fuel consumption. Valveless operation was also investigated and shown to offer a route to eliminating the dependency on fixed acoustical frequencies tied to a given chamber geometry. Figure 1 is representative of the valveless propulsive duct conceptualized by Tharjatt. Further, it was shown that the use of feedback techniques via multiple tube arrangements, which may not be practical from an engineering standpoint, leads to the possibility of very high frequency operation beyond the audible range. This would result in near silent operation. Finally, it was concluded that the propulsive duct should be capable of supersonic operation, and a Mach 3 engine was conceptualized; a schematic of this supersonic concept is shown in Figure 2.

Somewhat later, in a 1982 report by Kentfield,⁷ the pulsejet was analyzed for predicted flight performances based on well established experimental test-stand data and available theoretical studies.⁸ The results were compared against other engine alternatives suitable for small, high subsonic speed flight vehicles. The predicted performance for

valveless engine configurations was shown to be highly competitive with turbojets at high subsonic Mach numbers. Actual flight tests with a drone type aircraft at Mach 0.85 showed increased performance over predicted performance values due possibly to a combination of increased air-breathing, increased intake density, and a ram effect superimposed on the pulsejet cycle. Conclusions from these studies include suggestions that valveless pulsejet performance could be comparable and, in some cases, exceed that of turbojet engines. A strong point was made concerning the low cost, simplicity and relatively high thrust-to-weight ratio of pulsejets when compared with turbojets.

The main reason for including the preceding review of pulsejets and propulsive ducts is to draw attention to the similarity between the early conclusions concerning the future performance expectations of pulsejets and the conclusions drawn to date concerning expected PDE performance. As mentioned above, we believe a primary reason that such devices have not been pursued in the past is that adequate analysis and evaluation tools did not exist at the time to help understand the complexities of pulsed operation. Modern CFD techniques now allow a comprehensive analysis of the internal and external flows associated with pulsed propulsion devices. It may well be more than just an interesting exercise to re-examine the pulsejet engines using present day CFD tools, and to compare the results with those from similar PDE studies.

3. Constant Volume Combustion and Early Pulsed Detonation Studies

Constant Volume Combustion

A constant volume combustion process is known to have a higher thermodynamic efficiency than a constant pressure combustion process. Constant volume combustion was adopted very early for use in gas turbine engine development, and the first gas turbine engines in commercial use were based on the constant volume cycle. Jet propulsion engines were one of the applications of the constant volume cycle (or explosion cycle), which was explored in the late 1940s.⁸ Although the explosion cycle operates at a larger pressure variation in the combustion chamber than in a pulsejet, the cycle actually realized in these engines was not a fully constant volume one since the combustion chamber was open ended.⁹ In Reference 8 the maximum pressure ratio measured in an explosion cycle engine was 3:1, whereas the pressure ratio for the same mixture under the

assumption of a constant volume cycle would be 8:1. Also, this early engine was limited by the available cycle frequency, which in turn is limited by the reaction rate. A simple calculation⁸ showed that if the combustion time could be reduced in this engine from 0.006 sec to 0.003 sec, the thrust per pound of fuel-air mixture would increase 100%. Thus, a propulsion device based on an explosion-cycle has two main disadvantages:

- Constrained volume combustion (as distinguished from constant volume combustion) does not take full advantage of the pressure rise characteristic of the constant volume combustion process.
- The frequency of the explosion cycle is limited by the reaction rate, which is only slightly higher than the deflagration combustion rate.

The main advantage of the constant pressure cycle is that it leads to engine configurations with steady state processes of fuel and oxidizer injection, combustion, and expansion of the combustion products. These stages can be easily identified and the engine designer can optimize them on the basis of relatively simple steady state considerations.

Pulsed Detonation Studies

There have been numerous attempts in the past to take advantage of detonative combustion for engine applications. The following is a brief description of some of the most relevant past experimental and analytical studies of pulsed detonation.

The Work of N. Hoffmann. The first reported work on intermittent detonation is attributed to Hoffmann¹⁰ in 1940. Hoffmann's experiments on intermittent detonation were carried out in a long, narrow tube mounted on a test stand using acetylene-oxygen and benzene-oxygen fuel mixtures. Water vapor was added to prevent the highly sensitive acetylene-oxygen mixture from premature detonation. Hoffmann pointed out the importance of the detonation initiation (spark plug) location in reference to tube length and diffuser length. It was found that a continuous injection of the combustible mixture leads to only a narrow range of ignition frequencies that will produce an intermittent detonation cycle. These frequencies are governed by the time required for the mixture to reach the igniter, time of transition from deflagration to detonation, and time of expansion of the detonation products. Hoffmann attempted to find the optimum cycle frequency experimentally. It was

discovered that detonation-tube firing occurred at lower frequencies than the spark-plug energizing frequencies, indicating that the injection flow rate and ignition were out of phase. War-time events prevented further work by Hoffmann and his co-workers.

The Work of Nicholls and Co-Workers. A substantial effort in intermittent detonation research was made by a group headed by J. A. Nicholls¹¹⁻¹² of the University of Michigan beginning in the early 50's. The most relevant work concerns a set of experiments carried out in a six foot long detonation tube.¹¹ The detonation tube was constructed from a one inch internal diameter stainless steel tube. The fuel and oxidizer were injected under pressure from the (closed) left end of the tube and ignited at some distance down stream. The tube was mounted on a pendulum platform, suspended by support wires. Thrust for single detonations was measured by detecting tube (platform) movement relative to a stationary pointer. For multi-cycle detonations, thrust measurement was achieved by mounting the thrust end of the tube to the free end of a cantilever beam. In addition to direct thrust measurements, the temperature on the inner wall of the detonation tube was measured. Fuel mixtures of hydrogen/oxygen, hydrogen/air, acetylene/oxygen and acetylene/air mixtures were used. The gaseous oxidizer and fuel were continuously injected at the closed end of the detonation tube and three fixed flow rates were investigated. Under these conditions, the only parameters that could be varied were the fuel/oxidizer ratio and frequency of ignition. A maximum gross thrust of ~ 3.2lb was measured in the hydrogen/air mixture at the frequency of ~ 30 detonations per second. The most promising results were demonstrated for the H₂/air mixture, where a fuel specific impulse of $I_{sp} = 2100$ sec was reached. The maximum frequency of detonations obtained in all experiments was 35 Hz. The temperature measurements on the inner wall showed that for the highest frequency of detonations the temperature did not exceed 800° F. This temperature is approximately the mean between the temperature of the injected gasses and the detonation wave temperature averaged over the cycle frequency.

In their later work,¹³⁻¹⁵ the University of Michigan group concentrated on development of the Rotating Detonation Wave Rocket Motor. No further work on the pulsed detonation cycle was pursued.

The Work of L. J. Krzycki. In a setup very similar to Nicholls', L. J. Krzycki¹⁶ performed an

experimental investigation of intermittent detonations with frequencies up to 60 cps. An attempt was also made to analyze the basic phenomena using unsteady gas dynamic theory. Krzycki's attempt to analyze the basic phenomena relied on wave diagrams to trace characteristics, assumptions of isentropic flow for detonation and expansion, and incompressible flow for mixture injection processes. The most convincing data from the experiments are the measurement of thrust for a range of initiation frequencies and fuel mixture flow rates. Unfortunately no direct pressure measurement in the device is reported, so there is only indirect evidence of the nature of the process observed.

The basic test stand used by Krzycki is very similar to that used by Nicholls and his co-workers. The length of the detonation tube and the internal diameter were exactly the same as those in Nicholls' experiments. Figure 3 presents a schematic of the experimental apparatus containing common, generic elements of the Hoffmann-Nicholls-Krzycki experiments. A propane/air mixture was continuously injected through a reversed-flow diffuser for better mixing, and was ignited at the same distance as in the Nicholls' experiments from the injection point by an automobile spark plug. The spark frequency was varied from 1 to 60 cps. The spark plug power output was varied inversely with the initiation frequency, and at the frequency of 60 cps was only 0.65 Joule. This value is too low for direct initiation of a detonation wave by the spark, and consequently all of the experiments must have been based on transition from deflagration to detonation. According to experimental data and theory,¹⁷ direct initiation of a mixture of propane/air at the detonability limits requires an energy release on the order of 10^6 Joules. Thus, we conclude that the required deflagration-detonation transition region length in Krzycki's experiments would have been prohibitively large for the propane/air mixture. It follows that in all of the experiments a substantial part of the process was deflagrative. This resulted in low efficiency and negligible thrust. Krzycki repeated Nicholls' experiments using basically the same rates of injection of the detonable mixtures. Krzycki's experimental results are very well documented, allowing us to deduce a clear picture the physical processes occurring in the tube. The author arrived at the conclusion that thrust was possible from such a device but practical applications did not appear promising. It is unfortunate that, possibly based on Krzycki's extensive but misleading results, all

experimental work related to the pulsed detonation engine concept stopped at this time.

Russian Work on Pulse Detonation Devices. A review of the Russian literature has not uncovered work concerning applications of pulsed detonation devices to propulsion. However, there are numerous reports of applications of such devices for other purposes such as for producing nitrogen oxide¹⁸ (an old Zeldovich idea to bind nitrogen directly from air to produce fertilizers) and as rock crushing devices.¹⁹

Korovin et al.¹⁸ provide a most interesting account of the operation of a commercial detonation reactor. The main objective of this study was to examine the efficiency of thermal oxidation of nitrogen in an intermittent detonative process as well as an assessment of such technological issues as the fatigue of the reactor parts exposed to the intermittent detonation waves over a prolonged time. The reactor consisted of a tube with an inner diameter of 16 mm and length 1.3 m joined by a conical diffuser to a second tube with an inner diameter of 70 mm and length 3 m. The entire detonation reactor was submerged in running water. The detonation mixture was introduced at the end wall of the small tube. CH₄, O₂ and N₂ comprised the mixture composition and the mixture ratios were varied during the continuous operation of the reactor. The detonation wave velocity was measured directly by piezoelectric sensors placed in the small and large tubes. The detonation initiation frequency in the reactor was 2-16 Hz. It is reported that the apparatus operated without significant maintenance for 2000 hours.

Smirnov and Boichenko¹⁹ studied intermittent detonations of gasoline-air mixtures in a 3 m long and 22 mm inner diameter tube operating in the 6-8 Hz ignition frequency range. The main motivation for this work was to improve the efficiency of a commercial rock crushing apparatus based on intermittent detonations of the gasoline/air mixtures.²⁰ The authors investigated the dependence of the transitional region length from deflagration to detonation on the initial temperature of the mixture.

As a result of the information contained in the Russian reports, we conclude that reliable commercial devices based on intermittent detonations have been constructed and operated.

Pulsed Solid Explosion Studies at JPL. Work at the Jet Propulsion Laboratory (JPL) by Back, Varsi and others²¹⁻²⁴ concerned an experimental and

theoretical study of the feasibility of a rocket thruster based on intermittent detonations of solid explosive for propulsion in dense or high-pressure atmospheres of certain solar system planets. The JPL work was directed at very specific applications; however, these studies also addressed more general key issues concerning intermittent propulsion devices such as propulsion efficiency. In this work, a Deta sheet type C explosive was detonated inside a small detonation chamber attached to nozzles of various length and geometry. The nozzles, complete with firing plug, were mounted in a containment vessel that could be pressurized with mixtures of various inert gases from vacuum to 70 atm. The apparatus directly measured the thrust generated by single detonations of a small amount of solid explosive charge expanding into conical or straight nozzles. Thrust and specific impulse were measured by a pendulum balance system.

The results obtained from the JPL experimental study of an explosively driven rocket led to the following conclusions. First, rockets with long nozzles show increasing specific impulse with increasing ambient pressure in CO₂ and N₂. Short nozzles, on the other hand, show that specific impulse is independent of ambient pressure. Most importantly, most of the experiments obtained a relatively high specific impulse of 250 seconds and larger. This result is all the more striking since the detonation of a solid explosive yields a relatively low energy release of approximately 1000 cal/gm compared with 3000 cal/gm obtained in hydrogen oxygen combustion. Thus, it can be concluded that the total losses in a thruster based on unsteady expansion are not prohibitive and hence, in principle, very efficient intermittent detonation propulsion systems are possible.

4. Description of the PDE Concept

Basic Principles

A detonation process, due to the very high chemical reaction rate in the detonation wave, leads to a propulsion concept in which the constant volume process can be fully realized. In detonative combustion, a strong shock wave, which is part of the detonation wave, acts like a valve between the detonation products and fresh charge; the detonation wave functions at the same time as a valveless compressor between the fresh fuel/air mixture and the detonation products. The speed of the detonation

wave is about two orders of magnitude higher than the speed of a typical deflagration wave. Because of this, very high power densities can be created in the detonation chamber. Each detonation can be initiated independently and, depending on the chamber geometry and external flow characteristics pertaining to a particular device, a wide range of frequencies is possible. There is no theoretical restriction on the range of operating frequencies; they are uncoupled from any acoustical chamber resonance. The independence of detonation cycle frequency is the feature that most differentiates the PDE concept from the pulsejet. It is also the feature that leads theoretically to scalability of PDE configurations for a wide range of flight applications. A key physical restriction on the range of allowable detonation frequencies arises from the rate at which the fresh fuel/air mixture can be introduced into the detonation chamber. Obviously the detonation products must be discharged from the chamber before fresh charge is injected.

First PDE Experiments

To our knowledge, the first experiments that successfully demonstrated repetitive or pulsed detonation was attainable in a propulsion-like device were carried out by Helman, Shreeve and Eidelman²⁵ at the Naval Postgraduate School in 1985-86. During these studies, several fundamentally new ideas were developed for pulsed detonation applications to propulsion. First, to overcome the energy requirements for detonation initiation, a pre-detonation was initiated in a small detonation tube where an oxygen rich fuel mixture could be detonated at substantially lower energies than those required for full fuel/air mixtures. Next, the experimental PDE was operated in a self-aspirating mode; the detonation exhaust gases were discharged through gasdynamic expansion and fresh air was drawn into the detonation chamber due to chamber overexpansion following detonation product exhaust. Figure 4 is a schematic of one of the variations of the PDE experimental configurations. The pre-detonation initiation tube is shown attached to a spark plug. The most important results were obtained when the fuel injection (injection was accomplished with a toroidal ring containing holes near the exhaust plane of the device) rate was timed appropriately (the lag time between the fuel/air travel to the pre-detonation port and the arrival of the pre-detonation pulse) with detonation initiation. The principle of repetitive detonation initiation and control was definitively established in these experiments. Pressure transducer traces unambiguously showed that a detonation wave was

formed in the chamber and propagated with the Mach number appropriate for the fuel-air mixture. The fuel used in the NPS experiments was ethylene and the maximum detonation frequency obtained was 25 Hz, limited only by the mechanical nature of the solenoid valve used for fuel injection control. Figures 5 and 6 are two frames from a videotape of the early NPS experiments. Figure 5 shows the experimental apparatus and Figure 6 shows the apparatus during repetitive detonation. The figures also show the fuel injector ring between the two concentric detonation chamber cylinders. It was determined that the duration of a single cycle was less than 7 msec. This means that the NPS device could have potentially operated at frequencies up to 150 Hz in the static or no flow ($M = 0$) case. At the time of the NPS experiments, performance extrapolations included thrust levels up to 40 lbs at 100 Hz. As described later, SAIC simulations of static operation show higher thrust levels at these frequencies due to new ideas and improvements in the PDE concept. These new ideas are incorporated in the generic PDE concept.

The Generic PDE Device

In this section, we refer to the generic PDE device, which is represented as a small engine in Figure 7. The figure shows a schematic of the basic detonation chamber attached to the aft end of a generic aerodynamic vehicle. A combustible gas mixture is injected at the closed end of the detonation chamber and a detonation wave is shown propagating through the mixture. Also shown are air injection inlets and an important part of the device that we have termed the thrust wall. The schematic suggests a small-payload aerodynamic vehicle; however, as we describe later, the concept can be extended to larger payloads simply by scaling up the size of the detonation chamber and possibly combining a number of chambers into one larger engine.

The geometry of the main detonation chamber, which determines the propulsion efficiency and the duration of the cycle (frequency of detonations), is a key issue for the PDE concept. Since the fresh charge for the generic engine is supplied from the external flow field, the efficiency of the engine depends on the interaction of the surrounding flow with the internal flow dynamics. Following is a partial list of the broad range of physical processes requiring simulation in order to model the complex flow phenomena associated with the detonation engine performance:

1. Initiation and propagation of the detonation wave inside the chamber;
2. Expansion of the detonation products from the chamber into the air stream around the chamber at flight Mach numbers;
3. Fresh air intake from the surrounding air into the chamber;
4. The flow pattern inside the chamber during post-exhaust pressure buildup, which determines the strategy for mixing the next detonation charge;
5. Strong mutual interaction between the flow inside the chamber and the external flow surrounding the engine.

All of these processes are interdependent, and interaction and timing are crucial to engine efficiency. Thus, unlike simulations of steady state engines, the phenomena described above cannot be evaluated independently. It is a challenging computational problem to resolve the flow regime inside the chamber to account for nozzles, air inlets, etc., and at the same time resolve the flow outside and surrounding the engine, where the flow regime varies from high subsonic, locally transonic and supersonic.

The single most important issue is to determine the timing of the air intake for the fresh charge that leads to repetitive detonations. It is sufficient to assume inviscid flow for the purpose of simulating the expansion of the detonation products and fresh air intake. The assumption of inviscid flow makes the task of numerically simulating the PDE flow phenomena somewhat easier than if a fully viscous flow model were employed. The effects of viscous boundary layers are negligible for the size of the generic device studied in this work, with the exception of possible boundary layer effects on the valve and inlet geometries discussed subsequently.

SAIC has performed an extensive study of the generic PDE over a wide range of operating conditions for a wide range of device configurations.²⁶⁻³⁰ Numerical simulations of the unsteady flow and detonation processes, in addition to theoretical analysis, have resulted in an understanding and an approach to analyzing and evaluating PDE propulsion performance. Although the basic concept remains the same, there are subtle differences in the PDE manifestation for particular applications. These will be described subsequently. Details of the

numerical simulations (including assumptions used for detonation wave physics and chemistry, use of adaptive unstructured grids and Godunov methods for the Euler gasdynamic equations) are given elsewhere.²⁶⁻³¹ The following section is a summary of the results from numerical and theoretical studies of various applications and operating regimes for the generic PDE.

5. Operating Regimes

In this section we summarize the results of several applications and operating regimes identified in the course of our studies of the PDE concept.

M = 0 Static Operation

Under static conditions, $M = 0$, the PDE is completely self-aspirating. Such was the case for the early NPS PDE studies. Without an external airstream, the PDE must obtain fresh air charge as a result of the detonation chamber overexpansion immediately following exhaust of air-fuel detonation products. To the lowest approximation, the available time for chamber refill due to this overexpansion process is, for a given chamber geometry and fuel-air combination, directly proportional to its length. For $M = 0$ operation, we assume that the PDE configuration does not contain any air inlets other than the aft end of the device or, if inlets are present, they are closed. Simulations²⁶ of $M = 0$ PDE operation show that the time required for fresh air refill for a device with dimensions equivalent to the NPS experimental apparatus is on the order of 6-7 msec. This agrees with the NPS results and means that a maximum frequency of 150 Hz should be possible. Simulated thrust levels were higher than those estimated from scaling the NPS results. This is due to a new operating scenario that was uncovered by the simulations: detonation initiation from the aft end results in the kinetic energy of the shock wave being transferred to the thrust wall. The amount of extra thrust obtained from this mode of operation is considerably larger than that expected from gasdynamic expansion following detonation initiation at the thrust wall. The physical reason for this is found in the shock wave energetics.

The importance of $M = 0$ PDE performance is associated with applications of the concept for full scale aircraft propulsion, including rollout and takeoff. Simple scaling laws derived from the numerical simulation results and described later, show that $M = 0$ thrust levels can be large (tens of

thousands of lbs.) depending on the engine cross sectional area, length and detonation frequency.

Subsonic-Transonic Operation

PDE operation in the subsonic-transonic regime differs from the static case in that the self aspiration effect decreases with increasing Mach number. This is due to the formation of a rear stagnation point behind the exhaust plane above certain Mach numbers for given geometries. The stagnation region prevents complete detonation product exhaust and subsequent fresh charge injection. For example, over the Mach number range, $0^+ < M < 0.5$, full to partial self aspiration occurs; the effect decreases rapidly for Mach numbers above 0.5, resulting in the need for some type of air inlet or air intake valve configuration. Simulations of various detonation chamber and air inlet geometries^{26,28} have shown that, depending on the free-stream Mach number, appropriate shaping of the air inlet geometry and total inlet area leads to propulsion engines that are attractive for certain applications. We present here a summary of studies²⁸ carried out in an attempt to find a satisfactory PDE configuration for a small missile engine (the final configuration was not optimum, by any means, since all variables were not parametrically varied).

A PENAID-type missile with associated mission requirements such as range, speed, system weight, total thrust, and specific fuel consumption was used for the study. The detonation chamber dimensions were 6 cm diameter and 9 cm length with a cylindrical cross-section. A schematic of PDE integration into such a missile configuration is shown in Figure 8. The simulations showed that, for practically all cases involving simple inlets (circumferential slits around the cylindrical cross-section), the thrust data were independent of whether the inlets open intermittently (valved) or remain open during operation. This is due partially to the very short time that detonation products have to escape from the inlets thereby adding to negative thrust; this negative thrust, determined in the simulations, is negligible compared to the total integrated thrust. The thrust data do indicate a strong dependence on external flow conditions, e.g., Mach number. The Mach number plays a role in the wave drag; the details of valve and inlet configuration geometry figure prominently in the total wave drag. These studies answered an important question: can an air inlet be configured such that the inlet remains open over the full flight regime and operating conditions? The answer is "yes." Thus, at least for this regime, the PDE offers the possibility of a no-

moving-parts propulsion device. For the PENAID missile under discussion here, a configuration was found that operates between $0.2 < M < 0.9$ with open air inlets.

The following performance data were obtained for the PENAID missile configuration. For $M=0.8$ at sea level altitude and a detonation frequency, $f=100$ Hz, the PDE characteristics are:

Thrust.....	79 lb.
Fuel flow rate.....	0.025 lb./sec.
Fuel weight for 12 min.....	18 lb.
Oxygen weight.....	1.8 lb.
Fuel for detonation tube.....	0.6 lb.
Total oxygen and fuel weight.....	20.4 lb.
Total engine weight.....	30.2 lb.
Specific fuel consumption.....	1.14 lb./(lb.*hr.)

Assuming the PDE device geometry is kept fixed, a higher detonation frequency will result in a linear increase in thrust and fuel flow rate at the same specific fuel consumption. For example, if the detonation frequency is increased to 200 Hz, the performance data are:

Thrust.....	157 lb.
Fuel flow rate.....	0.05 lb/sec.
Fuel weight for 12 min.....	36 lb.
Oxygen weight.....	3.6 lb.
Fuel for detonation tube.....	1.2 lb.
Total oxygen and fuel weight.....	40.8 lb.
Total engine weight.....	54.4 lb.
Specific fuel consumption.....	1.14 lb./(lb.*hr.)

At lower Mach numbers, $M=0.5$, the maximum operating frequencies for constant thrust will be lower since the external dynamic pressure responsible for supplying fresh air to the chamber is also lower. For the device under consideration here, the maximum frequency is 250 Hz. For a frequency of 100 Hz:

Thrust.....	100 lb.
Fuel flow rate.....	0.025 lb/sec.
Fuel weight for 12 min.....	18 lb.
Oxygen weight.....	1.8 lb.
Fuel for detonation tube.....	0.6 lb.
Total oxygen and fuel weight.....	20.4 lb.
Total engine weight.....	30.2 lb.
Specific fuel consumption.....	0.9 lb./(lb.*hr.)

Again, if the frequency is increased the thrust will increase linearly; operation at 200 Hz yields:

Thrust.....	200 lb.
Fuel flow rate.....	0.05 lb/sec.
Fuel weight for 12 min.....	36 lb.
Oxygen weight.....	3.6 lb.
Fuel for detonation tube.....	1.2 lb.
Total oxygen and fuel weight.....	40.8 lb.
Total engine weight.....	54.2 lb.
Specific fuel consumption.....	10.9 lb./(lb.*hr.)

The examples of the PDE device performance given above are based on point design conditions arising from the simulations reported earlier.²⁶ They cannot be extrapolated with any degree of reliability to other conditions or configurations. We conclude, however, that the performance computed for the indicated device is encouraging from the point of view of thrust, thrust control, simplicity of the device (no moving parts), and specific fuel consumption (SFC). The specific fuel consumption computed above is competitive with present day small turbojet engines. The SFC for a PDE could be significantly lower than for small turbojets (SFC's for small turbojets are in the range of 1.8-2.0 lb./(lb.*hr)). Thus, for a given mission and vehicle, a PDE propulsion unit may be more fuel efficient, resulting in increased range. Moreover, if the expected thrust control in PDE's is realizable, it may be possible to produce propulsion units that can slow down, loiter and maneuver, and finally accelerate to full thrust again rapidly. Depending on the detonation frequency, which determines the thrust for all other conditions fixed, the thrust-to-weight ratio for the PDE can be as high as 20:1. This value is certainly competitive with other propulsion concepts.

The results of the scaling studies at subsonic-transonic speeds lead to scaling laws that can be used to predict the performance of PDE's over some range of parameters, assuming that other parameters are held fixed. For example, holding the external Mach number and basic chamber and inlet geometry fixed suggests that the thrust at constant specific fuel consumption produced by the PDE scales as:

$$\text{Thrust} = T_1 * \left(\frac{v}{v_1}\right) * \left(\frac{f}{f_1}\right),$$

where T_1 , (v/v_1) and (f/f_1) are the thrust computed for a chamber of volume v_1 operating at frequency f_1 , the ratio of a new volume to v_1 and the ratio of the new frequency to f_1 , respectively. Thus, thrust should scale linearly with the parameter $(v/v_1) * (f/f_1)$ over some range of this parameter. Departure from this linear variation may occur due to the following argument: First, since volume is proportional to the

product of cross-sectional area and length, $v \sim r^2 l$, (r - detonation chamber radius, l - chamber length) physical limits will be placed on r and l ; if r is too small (less than 1 cm), a detonation will not be sustainable and if l is too small (less than 10 cm), it may be difficult to mix fuel and air effectively. Using the thrust relation established above, we make the following observations. For a PDE device producing 100 pounds thrust at 100 Hz, doubling the frequency and increasing the volume by a factor of 5 yields a thrust level of 1000 pounds. Assuming that the aspect ratio of the chamber (chamber length to radius) is fixed, this would require an engine only 25.5 cm in diameter and 25.5 cm in length. Of course, the relation between thrust and $(v/v_1) * (f/f_1)$ cannot be believed over too wide a range of parameters; but, it does serve to point out the flexibility permitted by the PDE concept.

The subsonic-transonic simulations showed that the timing of the fresh air refilling required to recharge the chamber for subsequent detonations is a strong function of the details of the valve and inlet geometry, the expansion of the combustion products, the resulting over-expansion of the chamber flow, and the external flow regime and interaction of the external flow with the internal flow. For subsonic flight, Mach 0.2-0.9, the fresh air entering the chamber comes from two separate principal flow processes; one comes from the flow through any valve or inlet and the other comes from the self-aspiration or reverse flow from the aft end of the chamber due to strong over-expansion. All these processes are interdependent and, in order to search for a given performance in a given device, require variation of many parameters. The simulation results obtained to date provide an understanding of the effects caused by variation of the above-mentioned parameters. With the information available, we conclude that a PDE propulsion unit can be optimized (although no optimization studies were carried out) for a given flight regime. The decrease in thrust with increasing Mach number has been described earlier to result from increased wave drag produced by the inlet geometry. Optimization of the inlet geometry could help to eliminate a large part of the wave drag. The simulation data can be used to determine the detonation frequency at a given Mach number yielding constant thrust. For example, for a constant thrust level of 90 pounds, the required detonation frequency varies from 84 Hz at $M=0.0$ to 140 Hz to $M=0.8$. In a similar fashion, we can obtain parametric variations of other important aspects of PDE performance, such as minimum time for refill at given Mach number as

a function of air inlet opening. To find an optimum configuration that satisfies given performance over a wide flight regime requires a more extensive simulation study. It was mentioned earlier that the simulations presented here were carried out under the assumption of inviscid flow: boundary layer effects were not included. Boundary layers are only significant for the air inlets and valves.

There is an important feature of PDE operation for missiles such as the one considered here: if the expected thrust control is attainable, then the detonation frequency can be varied to produce constant thrust over a given flight envelope, or the frequency can be varied to make the missile slow down, loiter and maneuver, and finally ramp back to full thrust more or less instantaneously. Since each detonation is controlled separately, this capability should depend only on on-board electronics and power.

Supersonic-Hypersonic Operation

Numerical simulations have been carried out for PDE operation in the supersonic and hypersonic flight regimes.²⁹ The results of these simulations show that there are differences when compared with the lower speed regimes. The main difference, with respect to operating characteristics, is the air intake inlet must be more carefully considered. For supersonic and hypersonic flow air scoops may be required, adding to wave drag. For PDEs enclosed in a duct connected to upstream air inlets, pressure recovery from free-stream to duct inlet and finally to PDE inlet must be accounted for. To date, several detailed studies have been carried out for the higher speed regimes; a supersonic, $M = 2$ PENAID missile engine simulation and a sizing analysis for a large engine operating in the supersonic to hypersonic flight regime.

Supersonic $M = 2$ PDE The $M = 2$ PENAID missile study has been reported earlier²⁹ and, representative simulation results are shown on the cover of this review paper. It was found that a fixed air inlet geometry could be conceptualized to operate over the Mach number range, $0.5 < M < 2$. By this is meant the timing for fresh air charge allowed a detonation frequency of 200 Hz at $M = 2$ and this, in turn, means that any lower frequency is allowable at any other Mach number below $M = 2$. Detonation frequency control may result in enhanced control over missile flight trajectory since a constant thrust, a cruise-dash-loiter-cruise or any other tailored thrust profile can be realized. We conclude that supersonic PDE operation appears possible for missile

applications, and there may also be advantages for longer range air-to-air missiles due to enhanced propulsion energy management capability.

Sizing Analysis for Large PDEs A zeroth order sizing analysis has been carried out to define and size a PDE configuration satisfying high thrust level requirements from sea level to 30,000 ft altitude and for a flight trajectory including the Mach number range, $0 < M < 4$. The nominal target thrust level was 50,000 pounds and we assume that the aircraft/engine integration requires an air inlet duct to deliver fresh air to the PDE. We sketch here an outline of the analysis and give the main results.

We use the simple scaling argument given and use the thrust data obtained from simulations of the smaller missile configurations. We also assume a nominal detonation frequency, $f = 100$ Hz. We then establish the following baseline PDE performance operating point. At 3×10^4 ft. altitude for $M = 2$ the thrust in pounds per cubic meter detonation chamber volume is 2.5×10^4 lbs/m³. Therefore, an engine producing 5×10^4 pounds thrust requires a 2 m³ chamber volume. The sizing study answers the following questions: what is the size and shape of the detonation chamber, required detonation chamber air inlet areas, frequency variation range, and effect of air inlet duct losses on a PDE developing the nominal target thrust?

We denote free-stream conditions by ()₀, PDE air inlet conditions by ()₂, and PDE detonation chamber conditions by ()₃. To account for air inlet duct losses we define the ratio of PDE inlet total pressure to free-stream total pressure by C or:

$$\frac{P_{t2}}{P_{t0}} = C. \quad (1)$$

The simplest condition to assume for the PDE air inlet is choked flow. Although this is not valid over much of the required regime, certainly not for subsonic external flow, it will result in a pessimistic bound on the sizing parameters. Using well known gasdynamic analysis²² the static and total pressures and density at the PDE inlet can be found as:

$$P_{t1} = CP_0 \left(1 + \frac{M_0^2}{5} \right)^{\frac{7}{2}} \quad (2)$$

$$P_2 = P_0 C \left(\frac{5 + M_0^2}{6} \right)^{\frac{7}{2}} \quad (3)$$

$$\rho_2 = 1.2 C \rho_0 \left(\frac{5 + M_0^2}{6} \right)^{\frac{7}{2}} \left(\frac{5 + M_0^2}{5} \right)^{-1} \quad (4)$$

The mass flow rate through the engine inlet is:

$$\dot{m} = \rho_2 U_2 A_2, \quad (5)$$

and, using equations 2-4, gives:

$$\dot{m}_2 = A_2 \left(1.2 \gamma C^2 \left(\frac{P_0^2}{RT_0} \right) \left(\frac{5 + M_0^2}{6} \right)^7 \left(\frac{5 + M_0^2}{5} \right)^{-1} \right)^{\frac{1}{2}} \quad (6)$$

An equation for the area ratio A_2/A_3 can be found as:

$$\frac{A_2}{A_3} = \frac{216}{125} M_3 \left(1 + \frac{M_3^2}{5} \right)^3, \quad (7)$$

where M_2 has been set equal to unity. Our analysis does not include the thermodynamics of the PDE cycle; the sizing analysis is based totally on a determination of the allowable detonation frequencies in the PDE chamber. We obtain a bound on allowable flow speeds in the detonation chamber by requiring the detonation chamber to refill in the time between detonations. We further require the fuel to mix and flow with the mean speed U_3 from inlet to chamber exit, a distance equal to L , the chamber length. Thus, we obtain the relation $U_3 = fL$, where f is the detonation frequency. A calculation of M_3 gives:

$$M_3 = \frac{U_3}{U_3^*} = fL \sqrt{\frac{\rho_3}{\gamma P_3}} \quad (8)$$

Since the total pressure in the chamber equals the total pressure at the PDE inlet, the static pressure in the chamber as a function of chamber Mach number, given in Eq. (8), can be related to the free-stream static pressure as follows:

$$P_3 = CP_0 \left(1 + \frac{M_0^2}{5} \right)^{\frac{7}{2}} \left(1 + \left(\frac{A_2}{A_3} \right) C_1 \frac{Lf}{\gamma 5P_3} \right)^{\frac{7}{2}} \quad (9)$$

where C_1 is:

$$C_1 = \left(1.2 \gamma C^2 \left(\frac{P_0^2}{RT_0} \right) \left(\frac{5 + M_0^2}{6} \right)^7 \left(\frac{5 + M_0^2}{5} \right)^{-1} \right)^{\frac{1}{2}}$$

Another relation between P_3 and P_0 as a function of M_3 can be given as:

$$P_3 = CP_0 \left(1 + \frac{M_0^2}{5} \right)^{\frac{7}{2}} \left(1 + \frac{M_3^2}{5} \right)^{\frac{7}{2}} \quad (10)$$

Equations (7), (9) and (10) form a closed set for the variables P_3 , A_2/A_3 and M_3 with parameters C , P_0 , M_0 , L , f , T_0 , g , and R , the universal gas constant. The volume, V , of the detonation chamber is given by the product, $V = LA_3$. Thus, for a given volume, Equations (7), (9), and (10) can be solved for A_2/A_3 versus L or A_3 . Figure 9 gives a schematic of the PDE showing the air inlet gap width "l" resulting in an inlet area of A_2 , the detonation chamber length L , and the chamber cross-sectional area A_3 . We choose first a square chamber cross-section; the total inlet area is therefore given by the expression $A_2 = 4l(A_3)^{1/2}$. Results obtained from solving Eqs. (7), (9) and (10) are presented in Figure 10 for the baseline conditions. There, the area ratio, A_2/A_3 , is given versus A_3 . If A_3 is chosen to be 1.2 m^2 then the length of the PDE is 1.67 m and the engine inlet opening is 15 cm. Also shown in Figure 10 is the effect of C , the pressure recovery factor. The range of values chosen for C was: $0.7 < C < 1$. The effect of C is negligible for the range studied here. More realistic estimates for duct losses resulting in much lower values of C at high Mach numbers may well have a more pronounced effect. If the cross-sectional area is held fixed, Eqs. (7), (9) and (10) yield the results shown in Figure 11. The curve cannot be extended below $M = 1$ since the assumption of choked flow at A_2 is not valid; indeed, the assumption is not valid somewhere before $M = 1$ due to duct loss effects. The results from Figure 11 can be translated into inlet gap widths as shown in Figure 12. Figure 12 shows a range of inlet openings that, when compared with the total engine length, is equivalent to 8-12% of the total engine length. Below $M = 1$, a combination of self-aspiration and recharge from air inlets must be considered depending on Mach number. For self-aspiration at $M = 0$, the ratio of A_2/A_3 is unity; the inlets are not needed. For Mach numbers between zero and say, 0.5, partial air inlet opening is required and for Mach numbers greater than 0.5, the inlets will be fully open. For a fixed PDE configuration, varying the detonation frequency changes the thrust according to the scaling law given

earlier. Figure 13 shows the effect of frequency variation on A_2/A_3 . Recall the design point was at $f = 100 \text{ Hz}$. Figures 10-13 contain the answers to the questions asked during this sizing analysis: reasonable physical sizes for PDEs developing high thrust levels are predicted. A more rigorous analysis is required to validate these predictions.

To conclude this section, we show the variation of thrust as a function of chamber volume derived from the baseline conditions used above. Figure 14 gives this variation and, if a circular cross-section engine is considered, varying the baseline thrust yields engine sizes shown in Figure 15. For example, a 45,000 pound thrust engine 1.67 meters long has an engine diameter of 1.2 meters. This number is not unreasonable and compares well with sizes of current turbojet engines. As mentioned, a more detailed analysis of PDE performance is needed, including an effective "steady state" thermodynamic cycle model, to validate the PDE as a credible alternative for high thrust propulsion engines.

6. Summary and Conclusions

Past and recent studies have shown that pulsed propulsion devices theoretically offer significant advantages over steady state engines. The advantages range from the possibility of a no-moving-parts configuration to high thermodynamic efficiency constant volume cycles. Numerical simulations, theoretical analysis and scaling studies of PDE performance have shown applicability to many different flight vehicles including small missiles and full scale aircraft. Configurational flexibility offered by the PDE include non-circular cross-sectional detonation chambers allowing consideration of unique aircraft/engine integration possibilities. Thus, the numerical simulation and theoretical studies of PDE performance to date have shown interesting and important propulsion applications.

In order to realize the PDE potential, experimental data is required to validate the theoretical predictions and, most importantly, provide a proof of principle demonstration of the PDE mode of operation described in this paper, namely, detonation initiation from the exhaust end of the engine. The principle of sustained repetitive detonation has already been demonstrated in the NPS experiments, but, this took place at the inner thrust wall. The next step in the development of practical PDE devices requires a comprehensive experimental program where such key

issues as detonation initiation, air inlet design including boundary layers, fuel/air injection and mixing can be studied and understood. In addition, thrust measurements, both static and in an external flow are required to validate the numerical and theoretical predictions. Plans for such an experimental program are presently under consideration.

Acknowledgments

The work reviewed in this paper received partial support from the Defense Advanced Research Projects Agency under the following contracts: N66001-88-D-0088 (NOSC) and F49620-89-C-0087 (AFOSR). The authors would like to acknowledge Robert Pegg and Larry Hunt of NASA LaRC, John Leingang of the Wright Aeronautical Laboratory, and Major David Neyland of DARPA for their interest and for stimulating discussions concerning PDE applications. Over the course of these studies, SAIC colleagues Ed Rogan, Adam Drobot, Isaac Lottati and Ahron Freidman provided support and advice on many of the technical areas required to carry out this work.

Note Added in Proof. The authors would like to thank Mr Robert Pegg of NASA's Langley Research Center for bringing their attention to early documents and reports describing pulsed detonation engine research.

References

1. Scott, William B., "Renewed Interest in Pulsed Engines May be Linked to Black Aircraft," *Aviation Week*, 28 October 1991 (68-69).
2. Scott, William B., "New Evidence Bolsters Reports of Secret, High-speed Aircraft," *Aviation Week*, May 11, 1992 (62-63).
3. Wolfe, M.O.W., Luck, G.A., "Pressure Measurements on the F.Z.G. 76 Flying Bomb Motor," Technical Note No. EA237/1, Royal Aircraft Establishment, Farnborough, 1944.
4. Tharratt, C.E., "The Propulsive Duct," *Aircraft Engineering*, November 1965, (327-337).
5. Tharratt, C.E., *ibid*, December 1965, (359-371).
6. Tharratt, C.E., *ibid*, February 1966, (23-25).
7. Kentfield, J.A.C., "Valveless Pulsejets and Allied Devices for Low Thrust, Subsonic, Propulsion Applications," AGARD Conf - Proc. No. 307, Ramjets and Rockets for Military Applications, March (1982).
8. Zipkin, M.A. and Lewis G.W., "Analytical and Experimental Performance of an Explosion-Cycle Combustion Chamber of a Jet Propulsion Engine," NACA TN-1702, September 1948.
9. Shultz-Grunow, F., "Gas-Dynamic Investigation of the Pulse-Jet Tube," NACA TM-1131, February 1947.
10. Hoffmann, N., "Reaction Propulsion by Intermittent Detonative Combustion," Ministry of Supply, Volkenrode Translation, 1940.
11. Nicholls, J.A., Wilkinson, H.R. and Morrison, R.B., "Intermittent Detonation as a Thrust-Producing Mechanism," *Jet Propulsion*, 27, 534-541, 1957.
12. Dunlap, R., Brehm, R.L. and Nicholls, J.A., "A Preliminary Study of the Application of Steady State Detonative Combustion of a Reaction Engine," *ARS J.*, 28, 451-456, 1958.
13. Nicholls, J.A., Gullen, R.E. and Ragland K.W., "Feasibility Studies of a Rotating Detonation Wave Rocket Motor," *Journal of Spacecrafts and Rockets*, 3, 893-898, 1966.
14. Adamson, T.C. and Olsson, G.R., "Performance Analysis of a Rotating Detonation Wave Rocket Engine," *Astronautica Acta*, 13, 405-415, 1967.
15. Shen, P.I. and Adamson, T.C., "Theoretical Analysis of a Rotating Two-Phase Detonation in Liquid Rocket Motors," *Astronautica Acta*, 17, 715-728, 1972.
16. Krzycki, L.J., *Performance Characteristics of an Intermittent Detonation Device*, Navweps Report 7655, U.S. Naval Ordnance Test Station, China Lake, California 1962.
17. Matsui, H. and Lee, J.H., "On the Measure of the Relative Detonation Hazards of Gaseous Fuel-Oxygen and Air Mixtures," *Seventeenth Symposium (International) on Combustion*, 1269-1280, 1978.
18. Korovin, L.N., Losev, A., S.G. Ruban and Smekhov, G.D. "Combustion of Natural Gas in a Commercial Detonation Reactor," *Fiz. Gor. Vzryva*, Vol. 17, No. 3, p. 86, 1981.
19. Smirnov, N.N. and Boichenko, A.P., "Transition from Deflagration to Detonation in Gasoline-Air Mixtures," *Fiz. Gor. Vzryva*, 22, No. 2, 65-67, 1986.
20. Lobanov, D.P., Fonbershtein, E.G. and Ekomasov, S.P., "Detonation of Gasoline-Air Mixtures in Small Diameter Tubes," *Fiz. Gor. Vzryva*, 12, No. 3, 446, 1976.

21. Back, L.H., "Application of Blast Wave Theory to Explosive Propulsion." *Acta Astronautica*, 2, No. 5/6, 391-407, 1975.
22. Varsi, G., Back, L.H. and Kim, K., "Blast Wave in a Nozzle for Propulsion Applications." *Acta Astronautica*, 3, 141-156, 1976.
23. Kim, K., Varsi, G. and Back, L.H., "Blast Wave Analysis for Detonation Propulsion." *AIAA Journal*, Vol. 10, October 1977.
24. Back, L.H., Dowler, W.L. and Varsi, G., "Detonation Propulsion Experiments and Theory," *AIAA Journal* Vol. 21, October 1983.
25. Helman, D., Shreeve, R.P. and Eidelman, S., "Detonation Pulse Engine," AIAA-86-1683, 24th Joint Propulsion Conference, Huntsville, 1986.
26. Eidelman, S., Grossmann, W. and Lottati, I., "Propulsion Applications of the Pulsed Detonation Engine Concept," SAIC Report Number 89/1684, December 31, 1989.
27. Eidelman, S., Grossmann, W. and Lottati, I., "A Review of Propulsion Applications of the Pulsed Detonation Engine Concept," *J. Propulsion and Power*, Vol. 7, No. 6, November-December 1991 (857-865).
28. Eidelman, S. and Grossmann, W., "Computational Analysis of Pulsed Detonation Engines and Applications," AIAA-90-0460, January 8-11, 1990/Reno, Nevada.
29. Eidelman, S., Grossmann, W. and Lottati, I., "Air-Breathing Pulsed Detonation Engine Concept: A Numerical Study," AIAA-90-2420, July 16-18, 1990/Orlando, Florida.
30. Eidelman, S., Lottati, I. and Grossmann, W., "A Parametric Study of the Air-Breathing Pulsed Detonation Engine," AIAA-92-0392, January 6-9, 1992/Reno, Nevada.
31. Lottati, I., Eidelman, S. and Drobot, A., "A Fast Unstructured Grid Second Order Godunov Solver (FUGGS)," AIAA-90-0649, January 8-11, 1990/Reno, Nevada.
32. Anderson, J. D., *Modern Compressible Flow*, McGraw-Hill, 2nd Edition, New York, 1982.

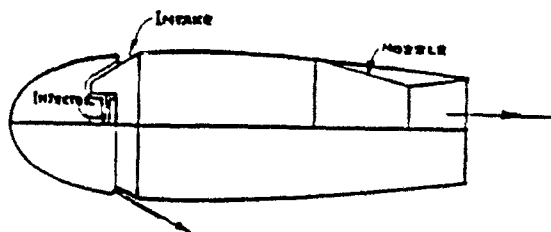


Figure 1. Valveless propulsive duct concept due to Tharjatt.

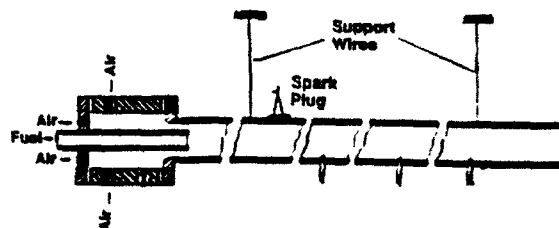


Figure 3. Schematic of the Hoffmann - Nicholls - Krzycki detonation tube experimental apparatus.

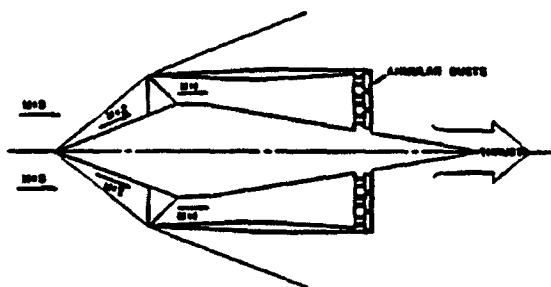


Figure 2. Supersonic, $M = 3$ conceptualization of the propulsive duct.

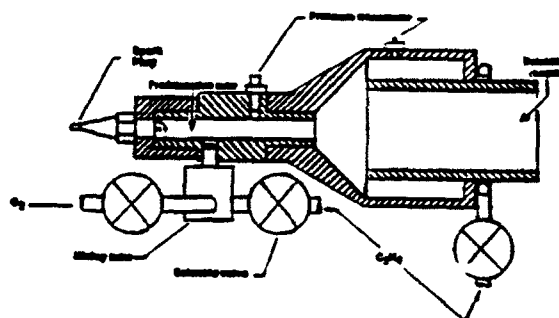


Figure 4. Schematic of the Helman, Shreeve, Eidelman PDE experimental configuration from the NPS studies.

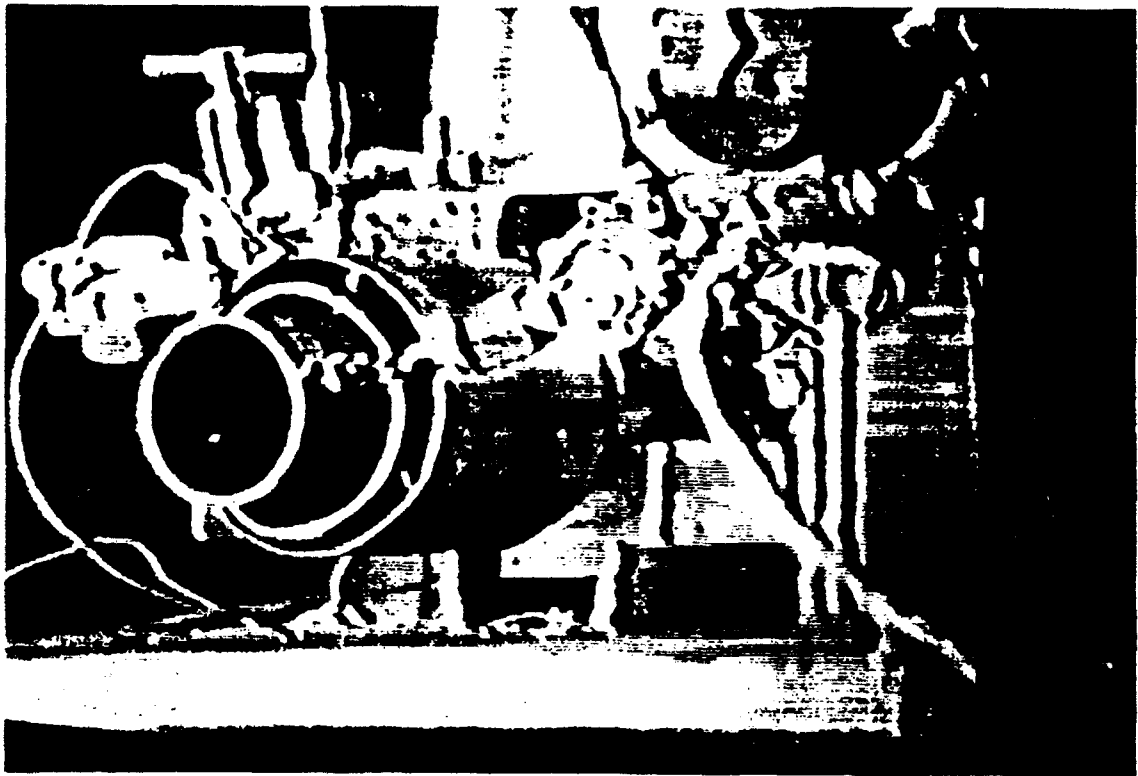


Figure 5. The PDE experimental apparatus used in the NPS studies.

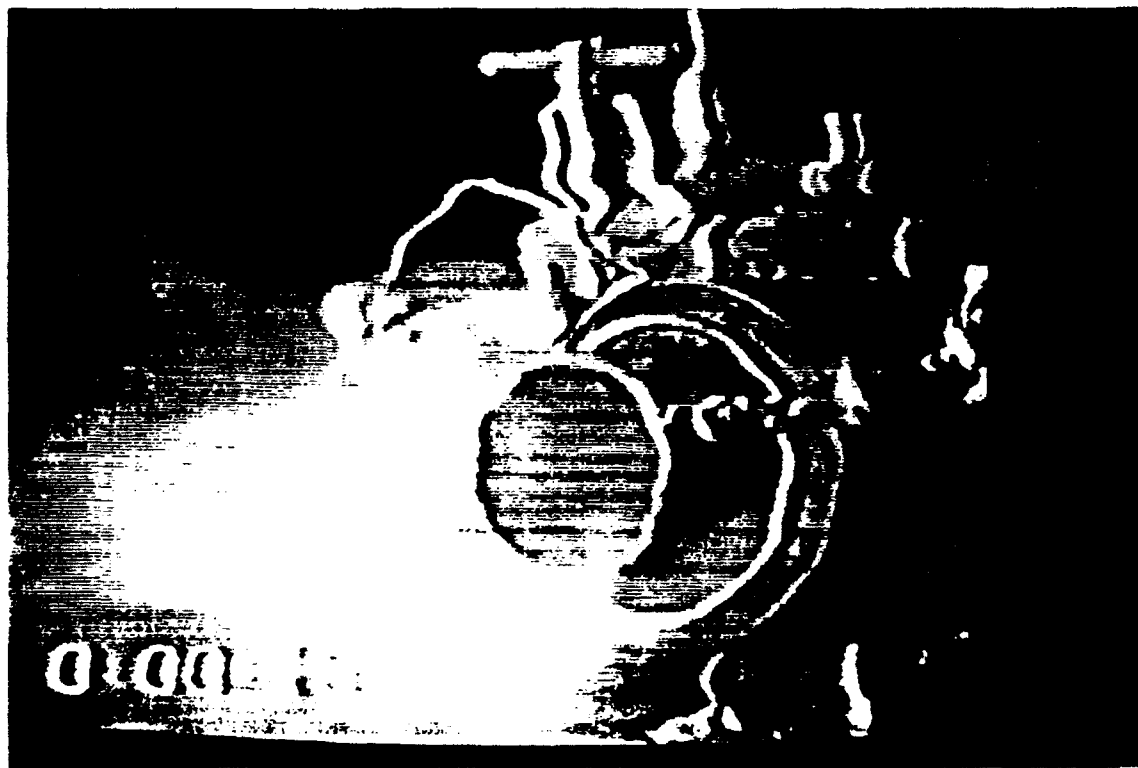


Figure 6. The PDE experiment during repetitive detonation.

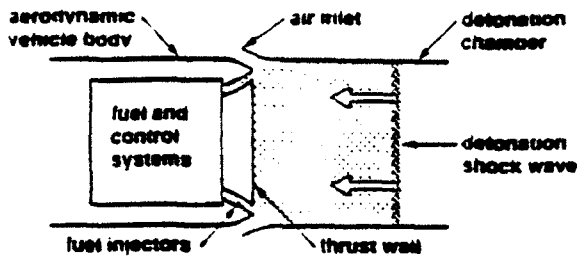


Figure 7. Schematic of the generic PDE.

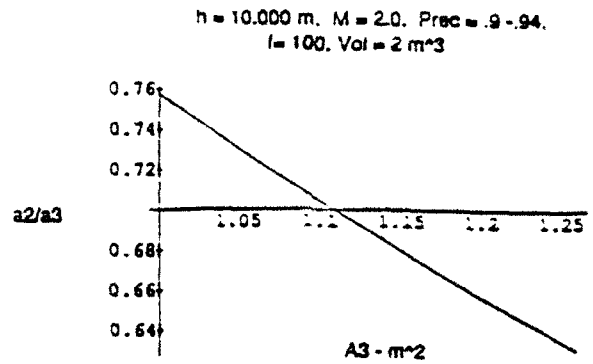


Figure 10. Results for A_2/A_3 as a function of A_3 . The results are, for the chosen conditions, independent of pressure recovery.

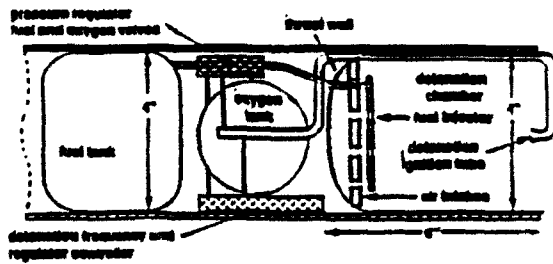


Figure 8. Schematic of PDE/PENAID missile integration.

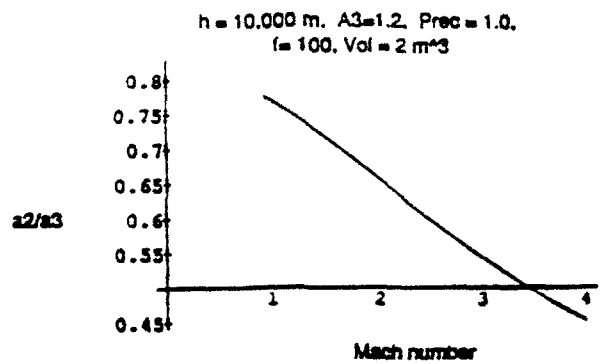


Figure 11. Results for A_2/A_3 as a function of Mach number.

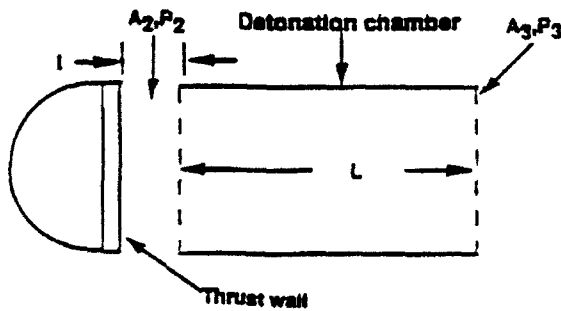


Figure 9. Schematic of PDE describing key sizing variables.

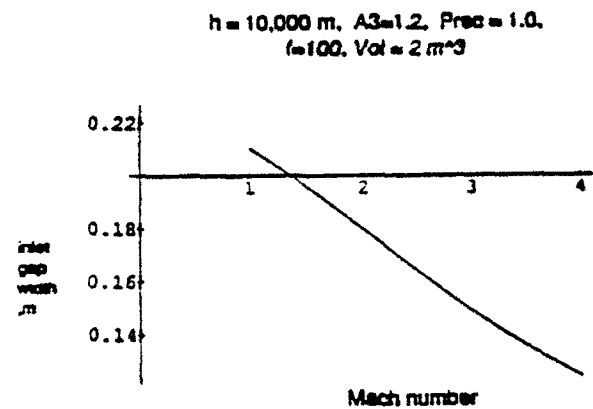


Figure 12. Results for inlet gap width, l , as a function of Mach number.

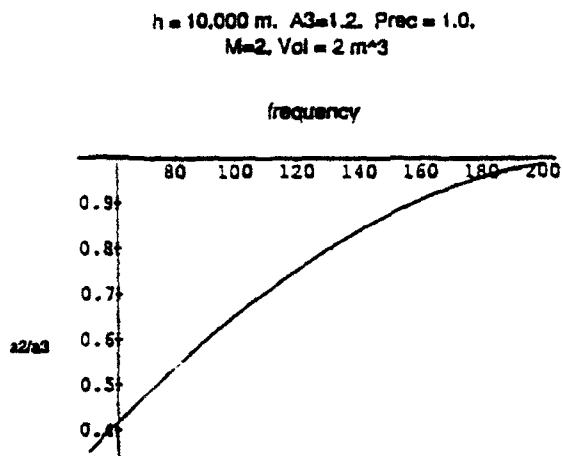


Figure 13. Results for A_2/A_3 as a function of detonation frequency.

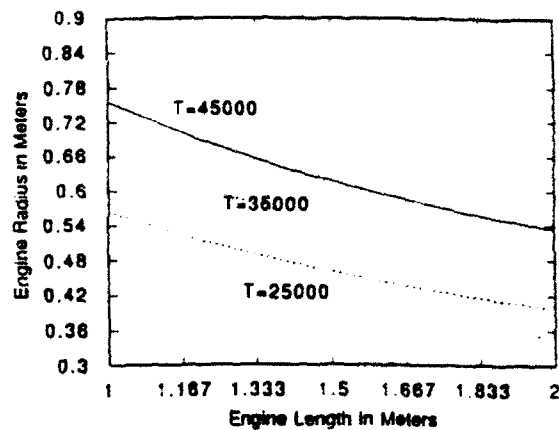


Figure 15. PDE engine radius (cylindrical cross-section) versus engine length.

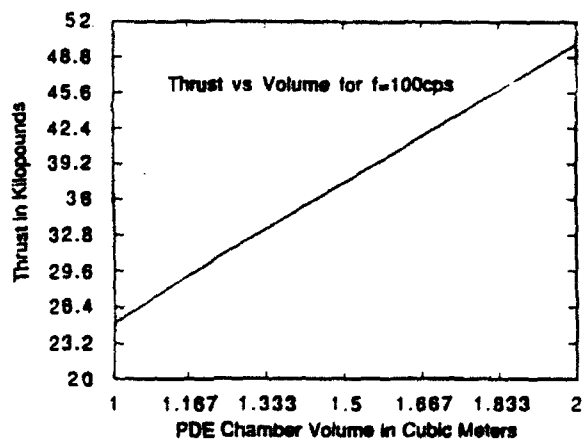


Figure 14. PDE thrust versus detonation chamber volume at a given frequency, $f = 100 \text{ Hz}$.

**Synthesis of Nanoscale Materials Using
Detonation of Solid Explosives**

*Shmuel Eidelman and Anatoly Altshuler
Science Applications International Corporation
1710 Goodridge Drive
Mail Stop 2-3-1
McLean, Virginia 22102*

**First International Conference on
Nanostructured Materials**
Cancun, Mexico
September 21-26, 1992

Synthesis of Nanoscale Materials Using Detonation of Solid Explosives

*Shmuel Eidelman and Anatoly Altshuler
Science Applications International Corporation
McLean, Virginia 22102*

Abstract

Direct synthesis of nanophase materials in detonations is considered. Article discusses a number of methods that can lead to formation of super saturated states of media that in turn will precipitate as nanoscale particles when the detonation products are quenched in the expansion process. Several examples are given of reactions that will lead to production of nanophase particles of metals, oxides, diamond and other unique materials. It is shown that conditions of nucleation and growth of nanoscale material can be analysed using advanced methods of computer simulation of detonation and blast wave phenomena. A sample of this kind of simulations is given. It is concluded that detonative synthesis of nanophase material can lead to low cost technology that will produce a range of unique materials.

1. Introduction

Recent enhanced interest in nanoscale materials is merited by the discovery of a set of unconventional material properties in the form of particles that are less than 10 nm in size. Anomalous chemical activity, lower critical temperatures of oxidation and sintering, sintering of composite materials with manifold increase in tensile strength, and sintering unique semiconducting and ferromagnetic materials, have all been demonstrated for nanoscale materials. This wide range of applications makes nanosize materials an extremely interesting and important material state that is the subject of intense study by many researchers.

The synthesis of nanoscale materials is accomplished through methods such as ion-sputtering and ion-deposition, laser ablation, evaporation and condensation in a vacuum, solgel, electroprecipitation, and plasma-jet techniques. Each of these techniques has produced 2-10 nm particles of various materials; however, the yield of such processes is extremely low and the cost of materials obtained is very high.

In this article we will consider detonative synthesis, a method of nanosize materials synthesis that offers an alternative to other more costly methods of production. Detonative synthesis is extremely advantageous because it allows very high pressure and temperature conditions to be created using low cost explosive materials and simple processing equipment. The synthesis occurs directly in the plasma created by the detonation wave. Conditions for detonative synthesis can be modified by changing the physical and chemical conditions of the detonation wave and expanding detonative products. For example, for nanoscale diamond powder synthesis, rapid expansion and cooling of the detonation products are required to prevent diamond graphitization. Thus, the explosive charge and atmosphere surrounding it should be designed to create these conditions.

In the following, we will review a range of conditions necessary for nanosize material synthesis that is provided by detonative synthesis methods, and examine their applicability for specific materials.

2. Detonation Waves as Generators of High Energy Density Plasmas

Detonations are reactive wave phenomena in which a reaction is initiated by the shock waves propagating at supersonic speeds through an explosive mixture. This wave consists of a shock wave discontinuity followed by a narrow zone of homogeneous chemical reaction. The shock wave compresses the explosive from its initial state with pressure P_0 and density ρ_0 to the shocked state P_s, ρ_s , with subsequent reaction of explosive in the reaction zone that extends up to the Chapman-Jouguet (CJ) state.

Condensed Explosive Detonations

Table I gives some typical parameters for detonation waves in solid explosives. We can see from this data that temperatures of about 3000°C at pressures of 30 GPa are typical for solid explosive detonations. These parameters create extremely oversaturated conditions for some detonation products. Subsequent ultra-fast quenching can lead to synthesis of nanophase material. Behind the detonation wave reaction zone, temperatures and pressures are high and detonation products will usually contain various active chemical components. It is challenging in this environment to preserve nanosize material from further reaction.

TABLE I
Some Typical Conditions for Detonation of Solid Explosives (1)

	Pressure GPa	Temperature $^\circ\text{K}$	D velocity m/sec
TNT, $\rho=1.6 \text{ g/cm}^3$	20.6	2940	6950
RDX, $\rho=1.8 \text{ g/cm}^3$	34.7	2590	8750
HMX, $\rho=1.9 \text{ g/cm}^3$	39.5	2364	9160
PbN_3 , $\rho=4.0 \text{ g/cm}^3$	23.1	2660	5000

Multi-Phase Detonations

Multi-phase detonations can cover a range of conditions between gaseous and condensed material detonations. Multi-phase detonable mixtures can be composed of solid or liquid fuel particles dispersed in gaseous oxidizer, solid particles of explosive material dispersed in gas, gaseous explosive mixture mixed with the inert or reactive liquid phase (2), or explosive slurries. All these possible methods of generating detonation waves greatly extend the range of conditions available for material synthesis. It should be noted that there is a difference in the character of condensed explosive detonation and gaseous detonations. With condensed explosives, high rate decomposition reactions usually take place. For gaseous detonations, reactions can be characterized as detonative combustion. Multi-phase detonations can be based on detonative combustion, high rate decomposition, and combinations of these processes.

Nonstandard Regimes for Detonative Reaction

A classical self-sustained detonation wave has a fixed wave structure that moves through the explosive with constant velocity. In a self-sustained detonation, a balance is achieved between the compression work of the shock wave and energy released in the reaction zone. If a self-sustained detonation is possible in a given explosive mixture at given initial conditions, it will propagate with a constant speed.

However, for many important reactive mixtures it is either very difficult or impossible to obtain a self-sustained detonation wave.

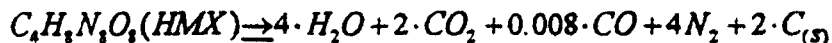
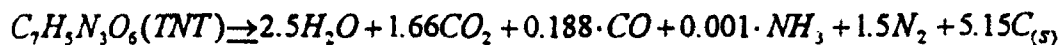
Over the last forty years, many nonstandard detonation regimes have been discovered that significantly reduce the restrictive limitations of the classical self-sustained detonation. The following is an incomplete list of the detonation regimes that significantly deviate from the classical self-sustained detonation wave:

- a. Transient detonation (forms when a deflagration wave undergoes transition to detonation);
- b. Overdriven detonation (compression work of the leading shock is partially sustained by an external source of energy);
- c. Spinning detonation (formed by small number of detonative combustion fronts that propagate through the mixture by spinning);
- d. Multi-layer detonation (propagates in layers of explosives where the detonation wave in one layer can lead to lateral initiation of an overdriven detonation wave in the adjacent layer);
- e. SWACER (Shock Wave Amplification by Coherent Energy Release) detonation;
- f. Light supported detonation (detonation front is supported by a laser beam heating the area behind the shock front).

All these possible regimes for initiating and sustaining detonation waves allow substantial flexibility in adapting a detonative process for the purpose of material synthesis.

3. Detonative Synthesis Chemistry for Nanophase Materials

The elementary composition of known explosives is quite limited. The most common class, CHNO explosives, produces only one condensed phase under normal thermodynamic conditions – ultra fine carbon (1):



These reactions have the following yield limits for solid phase carbon: 9% for RDX or HMX, and 29% for TNT.

More "exotic" BCHNO explosives can decompose, which produces solid BN or B₂O₃. For example, the powerful explosive B₁₀H₁₀₀C_{5.75}N₁₅O₃₀, decomposes with the 26% yield of BN by weight, while less hydrogenized B₁₀H₁₈C_{5.75}N₁₅O₃₀ produces primarily B₂O₃ with 34% yield.

From the point of view of chemical productivity, the most promising class of explosives is presented by acetylides and azides. For example, explosive decomposition of silver acetylide ($Ag_2C_2 \rightarrow \%Ag + 2C + 87kcal/mol$) generates a 90% silver yield. A more powerful explosive decomposition of $Ag_2C_2 \cdot AgNO_3 \rightarrow 3Ag(vapor) + CO_2 + CO + 0.5N_2 + 185kcal/mol$, gives 80% silver yield but much finer dispersity is expected. The decomposition of silver acetylides is interesting to compare with a silver azide explosion, $2Ag(N_3) \rightarrow 2Ag(v) + 3N_2$, with a respective yield of silver on

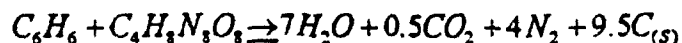
the order of 72% by weight. Over two dozen metals form explosive azides, while explosive acetylides are less common. Among the most interesting azides for nanosize powder production are explosive azides of cobalt, gold, strontium, and platinum. The main challenge in producing nanophase metals by explosive decomposition of azides or acetylides will be to assure rapid quenching of nanoscale phase components of the explosive products.

Loaded Explosive Synthesis

Explosive compositions are unknown for some chemical elements, as in the case of aluminum. The most obvious solution is to mix the explosive carrier with the powder or liquid form of the desired chemical. There is already a substantial history of adding aluminum powder to explosives in order to increase their performance. It has been established that at a grain size of several microns, aluminum does not have time to sublime in the detonation wave reaction zone; thus, it will not affect the reaction rates. On the other hand, detonation energies and temperatures are high enough to evaporate a substantial amount of additive. In order to overcome the diffusion barrier, we are considering mixing a melted explosive carrier with a liquid aluminum compound like $AlBr_3$, which has a melting point of $97^\circ C$ and comparatively low evaporation energy and temperature. Aluminum azide is also a possibility.

The same approach can be implemented in the loaded explosive synthesis of the nanoscale Hf. In this case, we can use detonation mixture of $Hf(BH_4)_4$ and an explosive carrier. Similarly, Ir can be produced using an IrF_6 load; Pu using a PuF_6 load; Re using a ReF_6 load; U using an UF_6 load; W using a WCl_6 load; V using a VF_5 load; Ti by means of a $TiCl_4$ load, etc. The reduction of metals in all these cases is taking place both physically, as the result of shock-temperature dissociation of molecules, and chemically, by ionized hydrogen and, in some cases, lithium vapors.

For carbon synthesis, loading the explosives cited above with benzol (C_6H_6), 1-hexadecan ($C_{16}H_{32}$), hexacozan ($C_{26}H_{54}$), dibenzyl ($C_{14}H_{14}$) etc., can greatly increase the yield of carbon without substantially diminishing the energetic characteristics of detonation. For example, a mixture of benzol with HMX on mol to mol basis will decompose in the detonative reaction as follows:



This reaction yields 30% by weight of solid carbon that has a potential to be preserved in nanoscale form.

All these examples illustrate that the loading of explosives for nanophase material synthesis expands the range of opportunities beyond the synthesis that results from the detonative decomposition of explosives.

Phase Composition of Synthesis Products

The crystalline structure of nanoscale powders obtained from detonation generally corresponds to high-pressure modifications of the solids. This is the result of high temperature and high pressure conditions in the detonation wave reaction zone and subsequent ultra-fast quenching and cooling of detonation products. In the case of carbon, diamond is formed. The phase diagram for carbon shown below easily illustrates this point. Area marked with number 1 on the phase diagram reflects parameters typical for detonation of HMX, while the area marked with 2 corresponds to detonation of TNT. It is quite obvious from Figure 1 that the detonation of TNT cannot produce diamond, while the detonation of HMX brings all condensed carbon into diamond form.

The same situation occurs with the synthesis of BN, when explosive decomposition of boron azide $B(N_3)_3$ produces hexagonal modification of BN, while powerful BCHNO explosives can produce BN with cubic sflerite structure. Other compounds that can be obtained include interesting compositions such as ZrO_2 , HfC, and WC, sometimes in their metastable modifications. Much more diverse are crystalline modifications of nanoscale metals. In cases like Gadolinium (Gd) and Samarium (Sm), five different structure modifications could be obtained as a result of different experimental conditions.

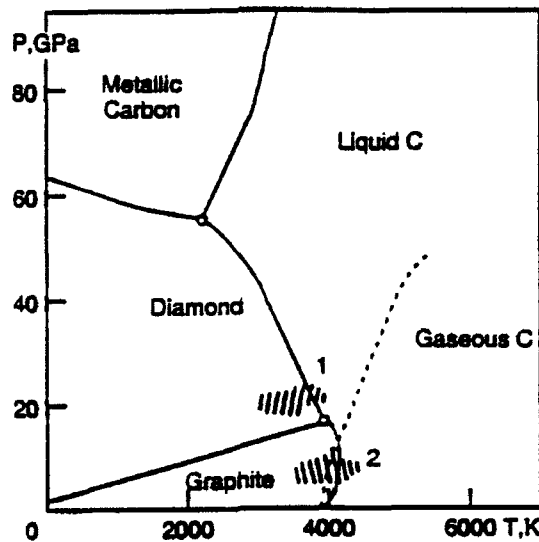


Figure 1. Carbon phase diagram schematics.

Nucleation and Growth of Nanophase Material Behind the Detonation Waves

We have discussed above the detonation wave structure in solids. We made important assumptions in our previous analysis regarding chemical equilibrium and physical stationarity of the processes on detonation front. The characteristic time of typical explosive decomposition reactions under detonation conditions in solid explosives lies in the range of $10^{-11} + 10^{-12}$ sec. As we noted above, the characteristic time length of the reaction zone for detonations in solids is $10^{-7} + 10^{-8}$ sec. This difference in time scales allows us to consider reactions behind the detonation front as equilibrium decompositions. Phenomenologic criteria of nucleation stationarity according to V. Shreidman (2) can be presented as follows:

$$v \leq \left(\frac{W}{T} \right)^{-1} \rho \sigma^2 / \eta^2 \quad (1)$$

where v - characteristic frequency of external forces; W - activation energy of nucleation; T - temperature in energetic units; ρ and η - density and viscosity of gases; σ - surface tension coefficient for nuclei.

Following are Fo'ner theory (3) we present activation energy through thermodynamic parameters:

$$W = \frac{16\pi}{3} \frac{\sigma^3 V^2}{(T \ln P / P_e)^2} \quad (2)$$

Here, P - partial pressure in gaseous precipitous phase; P_e - equilibrium pressure of saturation for condensate at given T ; V - atomic volume in condensed phase.

For the conditions typical for diamond condensation in the process of detonative synthesis, the barrier of nucleation at 100 kbar pressure and 3000°k temperature behind the detonation front, is $W = 13 \cdot 10^{-12}$ erg. Criteria (1) in this case gives: $\nu \leq 10^{13} \div 10^{14}$ Hz. Considering the time span of the detonation wave reaction zone ($10^{-7} \div 10^{-8}$ sec), we can assume stationarity of diamond nucleation. Calculations made for metals and some inorganic compounds lead to the same conclusion.

In accordance with the stationary approximation, the nucleation rate can be presented as follows (3):

$$I = \frac{2\alpha P^2 V \sigma^{1/2}}{(2\pi m T)^{1/2} T^{3/2}} \exp\left(-\frac{W}{T}\right) \quad (3)$$

α - condensation coefficient, m - atomic mass of condensate. For our reference case of diamond nucleation,

calculation using equation (3) gives: $I \approx 10^{21} \frac{\text{nuclei}}{\text{sec} \cdot \text{cm}^3}$

The diameter of critical nuclei can be estimated from activation barrier: $D = \sqrt{\frac{3W}{\pi\sigma}}$

For diamond it gives $D \sim 5\text{\AA}$.

4. Solid Explosive Charge Detonation in a Confined Volume

Experimentally developed conditions for diamond powder synthesis rely on the multi-layered detonation of several explosives and inert material. This system undergoes a complex detonation under conditions that are overdriven for the explosive producing diamond powder, and are standard for the driver detonation with some complex multi-dimensional expansion into the surrounding media. The details of the detonation process in this system have never been studied computationally, but experimental methods indicate that very specific conditions are required. It is known that the end result of this process is extremely sensitive to conditions of the multi-layered detonation. Currently, it is not clear what variables control particle sizes, or the maximum amount of free carbon released during the detonative combustion process that can be synthesized into diamond. Experimental work in this field is sketchy; numerical analysis of this complex process will enable us to understand the sensitivity to the basic parameter variations controlling diamond synthesis. Below are the results of numerical simulation of detonation and detonation products expansion for a composite TNT/RDX charge detonated in a 1 M^3 chamber. This simulation will give the conditions of the detonative products at various stages of expansion that determines the environment prevalent in the detonative synthesis.

In Figure 2 schematics of the blast sphere cross section are shown with the solid explosive charge located at the sphere's center. The inner volume of the sphere is 1 M^3 . Solid explosive is a composite charge formed from a TNT main charge with the layer of RDX around it. Detonation of a high energy RDX layer leads to the formation of an overdriven detonation wave in the main charge. Because the problem is symmetric, it is sufficient to simulate one quarter of the sphere volume to describe the full range of blast interaction that will occur for this condition. To increase the simulation's accuracy, we have divided the numerical modeling in the near field and global blast simulations. For the near field, a square grid with $DR = DX = 1 \text{ mm}$ was used to describe a region $10 \text{ cm} \times 10 \text{ cm}$ containing the solid explosive charge. The simulation results from the near field region are mapped on the larger computational domain, which includes the inner wall of the blast sphere. For higher resolution and computational efficiency, we

have used structured/unstructured grids to describe the sphere's inner volume. The mathematical formulation and numerical method for the solution used in the near field are described in detail in Reference 2. These computational techniques are implemented in the MPHASE code. The model and numerical methods used for simulations in the computational domain shown in Figure 2 are described in Reference 5. These computational techniques are implemented in AUGUST code. Both MPHASE and AUGUST have been validated for the range of detonation and strong shock wave reflection and diffraction problems.(6)

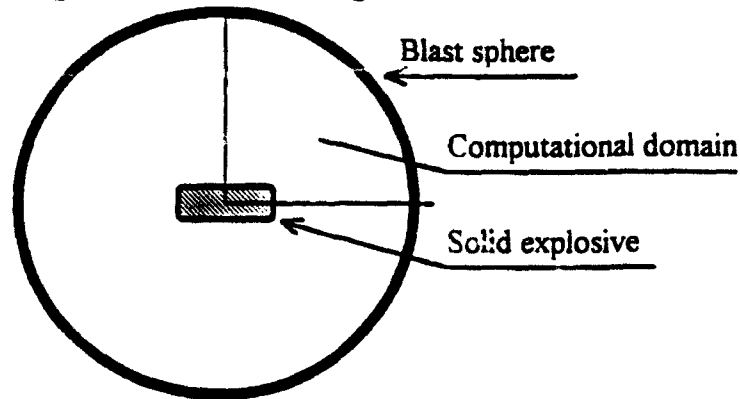


Figure 2. Schematics of the blast sphere cross section with the solid explosive charge. The computational domain is covering the upper right quadrant.

In Figure 3, simulation results for the near field region are shown as pressure density and temperature contour plots for an instant of time when the detonation wave is at 2 mm distance from the right edge of the charge. $t = 0$ is the time of detonation wave initiation in a solid explosive. Pressure and temperature contour plots are shown using a linear scale. In Figure 3 we observe propagation of the complex detonation front through the composite charge and the initial stages of detonation product expansion. The outer layer of the RDX leads to the formation of an overdriven detonation wave in the TNT charge that has shorter reaction zone, higher wave speed, higher temperatures, and higher pressures as compared with a homogeneous TNT charge detonation. The maximum temperature is reached in the air strata located in the immediate vicinity of the charge. This temperature maximum is created by a strong shock wave produced by expanding detonation products in air. The following conditions are reached at the detonation wave front in the TNT charge: $P = 62.6$ GPa; $T = 6000^\circ\text{C}$; $\rho = 2900$ kg/m³. It should be noted that because of high resolution of the numerical scheme we are simulating the Von Neumann spike of the detonation wave front, where the pressure is considerably higher than at the Chapman-Jouguet point.

When the shock wave reaches the edges of the computational domain for the near field simulation, the simulation results are mapped to the grid of the global domain shown in Figure 2 and are continued on larger grid. In Figure 4 pressure and temperature contour plots are shown for three consecutive instances of time for the global domain simulation. In Figure 4a results are shown at $t = 0.05$ μsec , shortly before the detonation products reached the walls of the sphere. Here we can observe significantly lower pressures as compared with Figure 3 values due to strong expansion; however, the propagating shock is leading to considerable heating of the surrounding air. In Figure 4b pressure and temperature contour plots are shown at some stage of the wave front reflection from the inner wall of the blast sphere. The average pressures and temperatures are significantly lower; however, several focus points are created during the reflection that have significantly higher pressure and temperature values. In Figure 4c, the shock wave complex is converging towards the blast sphere center, with significant amplification of the shock strength and temperature at the front. It is obvious that this system of shock waves will undergo a number of reflections, focusing, and expansions until quiescent conditions are reached in the blast sphere.

The simulations illustrated above will provide the global conditions in the blast chamber as a function of time. This information can be used for the nucleation simulations of the material behind the shock front, and estimates of possible phase transformation or reaction of the newly formed material. As a result of this multi-step approach, we can consider all the stages of the detonative synthesis process that are important for nanoscale material formation. This approach will allow us to minimize the number of experiments, understand the physics of detonative synthesis, and control the quality and yield of nanoscale materials produced experimentally.

5. Conclusions

Detonative synthesis of nanoscale material is a new technology and the nature of this process is widely unexplored. More studies should be done in addressing chemical and phase transformations under extreme and fast changing conditions in waves of detonation, shock and rarefaction. An unlimited array of elements and compounds, as well as their structural modifications (some highly metastable), is attainable through such processing.

Detonation synthesis combines the best features of traditional nanophase material technology -- the most effective generation of hot plasma and vapors, and fast quenching of a condensing product. The most unique feature of the process is the extreme density of the generated plasmas, which makes them highly supersaturated in regard to pressure and temperature.

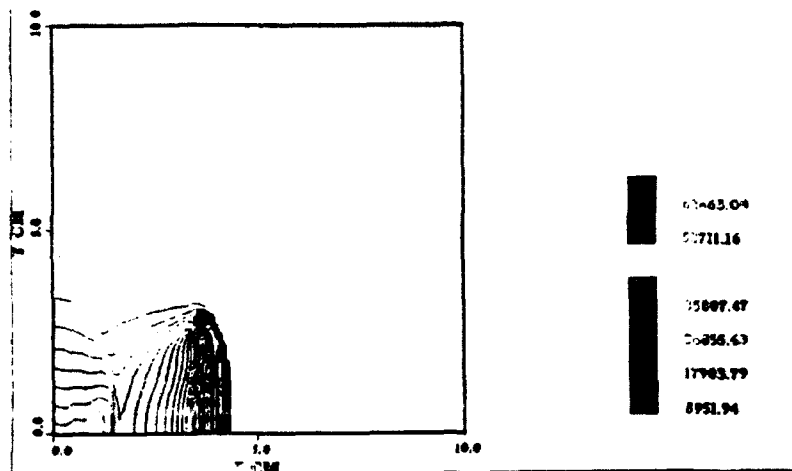
Detonative technology has promising industrial prospects, due to very low production cost and the unique materials it yields. As a reference we can use ultra-fine carbon. In this case common nanotechnology produces carbon black; detonative synthesis diamond. These factors are completely changing the traditional view of nanomaterials applications. (7)

6. Acknowledgment

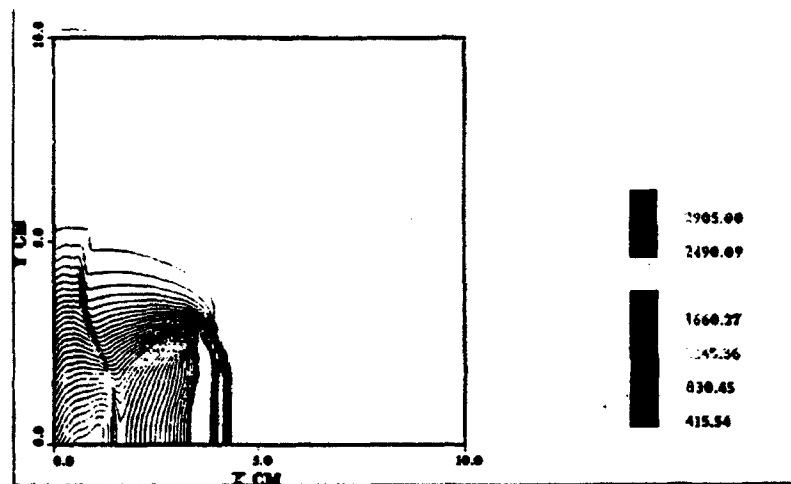
The work reported here was partially supported by ONR under Contract #N00014-92-C-0168. The authors would like to thank Dr. L.T. Kabacoff for his interest in this project.

7. References

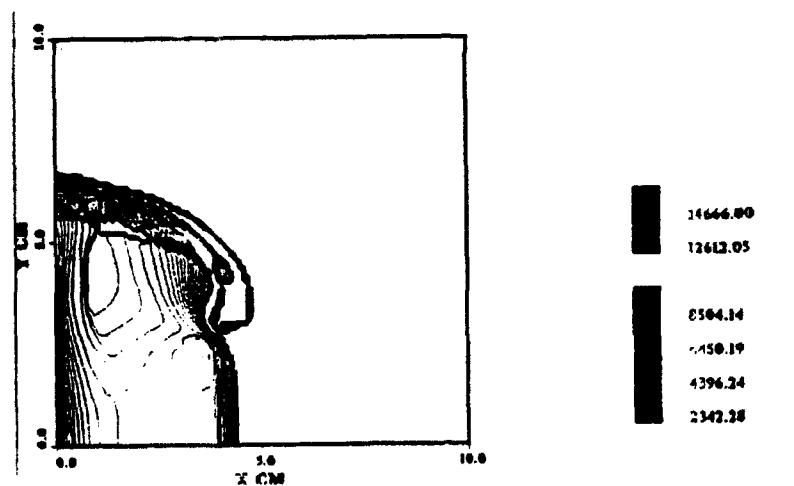
1. C.L. Mader, "Numerical Modeling of Detonations," University of California Press Ltd., London, England. (1979).
2. V. Shreidman, Zhurn Eksperim, Teor. Fiz., v 91, 8, p 520, (1986).
3. D. Fedoseev, Uspekhi Khimii, V 53, p 7 53, (1983).
4. S. Eidelman and X. Yang, "Detonation Wave Propagation in Variable Density Multi-Phase Layers," AIAA 92-0346, 30th Aerospace Sciences Meeting, Reno, NV, Jan. 6-10, 1992.
5. I. Lotatti, S. Eidelman, A. Drobot, "A Fast Unstructured Grid Second Order Godunov Solver (FUGGS)," AIAA 90-0699, 28th Aerospace Sciences Meeting, Reno, NV, Jan. 8-11, 1990a.
6. I. Lottati and S. Eidelman, "Second Order Godunov Solver on Adaptive Unstructured Grids." Proceedings of the 4th International Symposium on Computational Fluid Dynamics, Davis, CA, September 1991.
7. A. Altshuler and J.L. Sprague, "The Synthesis, Properties, and Applications of Diamond Ceramic Materials." 41st Electronic Components Technology Conference, Atlanta, Georgia, May 1991.



a. Pressure



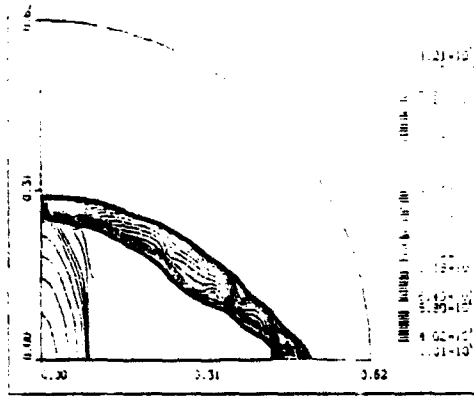
b. Density



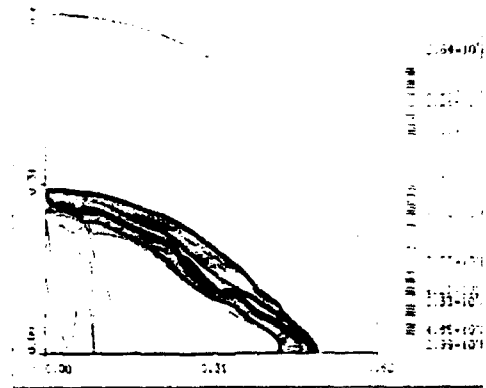
c. Temperature.

Figure 3. Pressure, density and temperature contour plots for a composite charge detonation $t = 0.01$ msec.

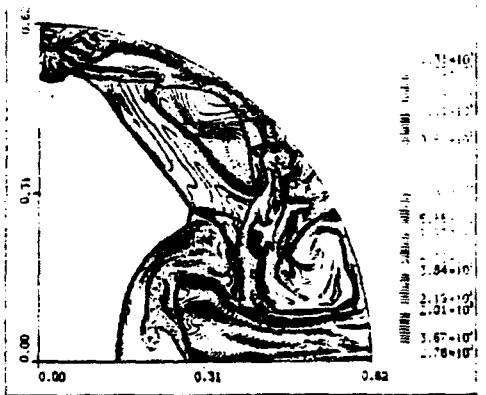
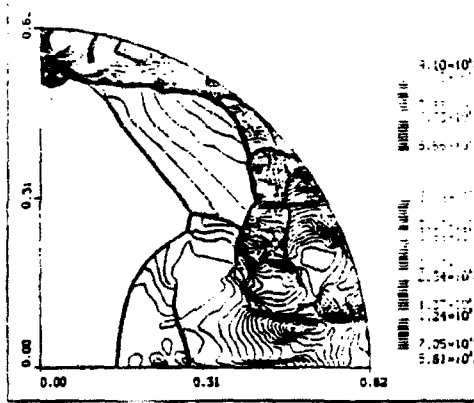
Pressure



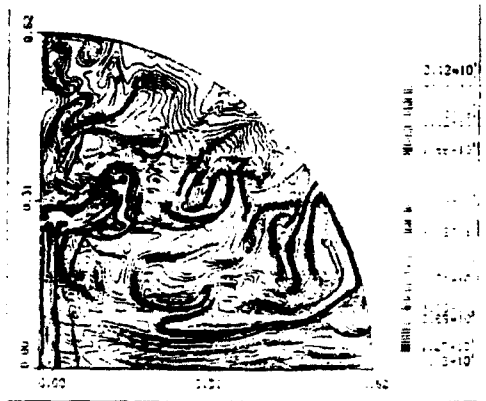
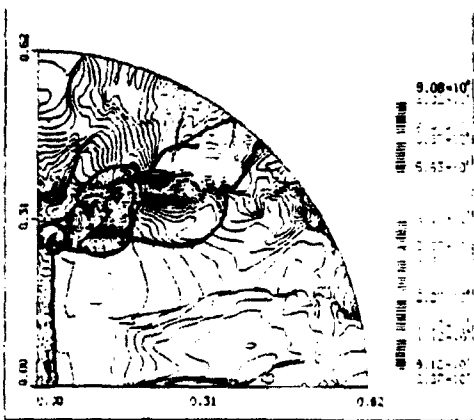
Temperature



a. $t = 0.05$ msec



b. $t = 0.2$ msec



c. $t = 0.3$ msec

Figure 4. Simulation of blast wave reflection from the inner wave of a blast sphere.

Detonation Wave Propagation in Combustible Mixtures with Variable Particle Density Distributions

Shmuel Eidelman* and Xiaolong Yang*

Science Applications International Corporation, McLean, Virginia 22102

Abstract

A mathematical model is presented describing a physical system of detonation waves propagating in a solid particle/air mixture with a wide range of solid-phase concentrations. The mathematical model was solved numerically using the Second Order Godunov method, and numerical solutions were validated for detonation waves propagating in mixtures with concentrations of solid phase from 0.75 kg/m^3 to 1000 kg/m^3 . Numerical solution was obtained for detonation waves propagating in a system consisting of clouds with a small concentration of particles and a ground layer in which solid particle densities are three orders of magnitude larger than in the cloud. Three different particle concentration distributions in the ground layer were simulated and compared in terms of detonation wave structure and parameters.

Introduction

When combustible particles are intentionally or unintentionally dispersed into the air, the resulting mixture can be detonable. Formation of this potentially explosive dust environment and the properties of its detonation are of significant practical interest in view of its destructive or creative effects.

Copyright © 1993 by the American Institute of Aeronautics and Astronautics, Inc. All rights reserved.

*Research Scientist, Applied Physics Operation.

The experimental and theoretical study of these phenomena until now has addressed only homogenous particle/oxidizer mixtures. However, intentional or accidental processes of the explosive dust dispersion will always lead to inhomogeneous particle density distribution. Some industrial methods of explosive-forming rely on detonation of explosive powder. This powder can be deposited as a thin layer over the surface area of the forming metal, with some remaining concentration in the vicinity of the layer. The phenomenology of detonation wave initiation and propagation in this environment is the main subject of this paper.

When the detonation wave is generated in a homogeneous mixture by a "direct initiation," it starts with a strong blast wave from the initiating charge. As the blast wave decays, combustion of the reactive mixture behind its shock front starts to have a larger role in support of the shock wave motion. When the initial explosion energy exceeds some critical value, transition to steady-state detonation occurs.¹⁻⁴ In explosive dust mixtures with a nonuniform distribution of particle density, the initiation dynamics are significantly more complicated. The critical initiation energy sufficient for one of the explosive particle density strata regions is not necessarily adequate for other regions. Also, when there is a significant variation in density between the different layers (regions) of the mixture, steady detonation in one layer can result in an overdriven detonation in an adjacent layer. Our paper demonstrates that the phenomenology of these interactions is distinctly different from the classical studies of multilayer detonations in gases. This is primarily because the energy content of adjacent layers in a typical multigas layer experiment⁵ varies by a factor of two or four, whereas the energy content in explosive dust/air mixtures can vary by several orders of magnitude.

In this paper we use detailed numerical simulation to study the initiation dynamics and propagation phenomenology for a general case of explosive dust dispersion. We will consider particle density variation from 1000 kg/m^3 in the ground layer to 0.5 kg/m^3 or 0 for the upper edges of the cloud. The effects of variation of the cloud density on detonation wave parameters

will be examined for different cases of cloud particle density distribution. When possible, the results of computer simulations are validated in comparison with experimental and theoretical studies.

The outline of this paper is as follows. Section 2 gives a description of a mathematical model that includes governing conservation equations for two phases and the constitutive laws. We describe the model for a particle-gas interaction, combustion, and equation-of-state for gas phase. The numerical integration technique for solving the mathematical model will also be outlined. In Section 3, we present our numerical simulation results. We first validate our model by comparing one-dimensional detonation wave simulation with available experimental results. We then give the two-dimensional simulation for detonation wave propagation in combustible particles/air mixtures with variable particles density distribution. Concluding remarks are given in Section 4.

Mathematical Model and the Numerical Solution

The mathematical model consists of conservation governing equations and constitutive laws that provide closure relations for the model. The basic formulation adopted here follows the two-phase fluid dynamics model presented in the text by Kuo.⁶ The approach assumes that there are two distinct continua, one for gas and one for solid particles, each moving at its own velocity through its own control volume. The sum of these two volumes represents an average mixture volume. With these assumptions, distinct equations for continuity, momentum, and energy are written for each phase. The interaction effects between the two phases are accounted for by the source terms on the right-hand side of the governing equation. The following is a short description of the two-phase flow model used in our study, with conservation equations written in Eulerian form for two-dimensional flow in Cartesian coordinates:

Continuity of gaseous phase

$$\frac{\partial \rho_1}{\partial t} + \frac{\partial(\rho_1 u_g)}{\partial x} + \frac{\partial(\rho_1 v_g)}{\partial y} = \Gamma \quad (1)$$

Continuity of solid-particle phase

$$\frac{\partial \rho_2}{\partial t} + \frac{\partial(\rho_2 u_p)}{\partial x} + \frac{\partial(\rho_2 v_p)}{\partial y} = -\Gamma \quad (2)$$

Conservation of momentum of gaseous phase in x direction

$$\frac{\partial(\rho_1 u_g)}{\partial t} + \frac{\partial(\rho_1 u_g^2 + \phi p_g)}{\partial x} + \frac{\partial(\rho_1 u_g v_g)}{\partial y} = -F_x + \Gamma u_p \quad (3)$$

Conservation of momentum of gaseous phase in y direction

$$\frac{\partial(\rho_1 v_g)}{\partial t} + \frac{\partial(\rho_1 u_g v_g)}{\partial x} + \frac{\partial(\rho_1 v_g^2 + \phi p_g)}{\partial y} = -F_y + \Gamma v_p \quad (4)$$

Conservation of momentum of solid-particle phase in x direction

$$\frac{\partial(\rho_2 u_p)}{\partial t} + \frac{\partial(\rho_2 u_p^2)}{\partial x} + \frac{\partial(\rho_2 v_p u_p)}{\partial y} = F_x - \Gamma u_p \quad (5)$$

Conservation of momentum of solid-particle phase in y direction

$$\frac{\partial(\rho_2 v_p)}{\partial t} + \frac{\partial(\rho_2 u_p v_p)}{\partial x} + \frac{\partial(\rho_2 v_p^2)}{\partial y} = F_y - \Gamma v_p \quad (6)$$

Conservation of energy of gas phase

$$\frac{\partial(\rho_1 E_g T)}{\partial t} + \frac{\partial(\rho_1 u_g E_g T + u_g \phi p_g)}{\partial x} + \frac{\partial(\rho_1 v_g E_g T + v_g \phi p_g)}{\partial y} =$$

$$\Gamma \left(\frac{u_p^2 + v_p^2}{2} + E_{chem} + C_s T_p \right) - (F_x u_p + F_y v_p) - \dot{Q} \quad (7)$$

Conservation of energy of solid-particle phase:

$$\frac{\partial(\rho_2 E_{pT})}{\partial t} + \frac{\partial(\rho_2 E_{pT} u_p)}{\partial x} + \frac{\partial}{\partial y}(\rho_2 E_p v_p) = \dot{Q} + (F_x v_p + F_y u_p) - \Gamma \left(\frac{u_p^2 + v_p^2}{2} + E_{chem} + C_p T_p \right) \quad (8)$$

Conservation of number density of solid-particle

$$\frac{\partial N_p}{\partial t} + \frac{\partial(N_p u_p)}{\partial x} + \frac{\partial(N_p v_p)}{\partial y} = 0 \quad (9)$$

In the above equations, we have the following definitions and constitutive laws:

Phase densities

$$\rho_1 = \phi \rho_g, \quad \rho_2 = (1 - \phi) \rho_p \quad (10a)$$

and fractional porosity

$$\phi = 1 - \frac{N_p M_p}{\rho_p} = \frac{\text{Volume of void}}{\text{total volume}} \quad (10b)$$

where M_p is the mass of each particle and ρ_p is the solid-particle density.

Total internal energy of gaseous phase

$$E_{gT} = E_g + \frac{1}{2}(u_g^2 + v_g^2) \quad \text{and} \quad E_g = E_g(p_g, \rho_g) \quad (11)$$

where $E_g(p_g, \rho_g)$ is the equation-of-state for gas phase, which will be discussed later.

Total internal energy of solid-particle phase

$$E_{pT} = E_p + \frac{1}{2}(u_p^2 + v_p^2) \quad \text{and} \quad E_p = E_{chem} + C_p T_p \quad (12)$$

In order to close the above system of conservation equations, it is necessary to define certain criteria and interaction laws between the two phases, which include mass generation rate, Γ , drag force between particles and gas, F_x, F_y , and the interphase heat transfer rate \dot{Q} . The model for particle and gas interaction and particle combustion that results in the constitutive relation for the conservation equations is explained in detail in the next subsection.

Model for a Particle Gas Interaction and Combustion

Presently, the physics of the energy release mechanisms in solid-particles/air mixtures is not clearly understood. This can be attributed to the obvious difficulties of making a direct nonobtrusive measurement in the optically thick environment typical for this system. In the experimental and theoretical work done for the grain dust detonation conditions,⁷ it was demonstrated that the volatile components released by the particle heated behind the shock front play a major role in determining the detonability limits of the mixture. Eidelman and Burcat⁸ successfully applied a combination of fast evaporation and aerodynamic shattering mechanisms to simulate a two-phase detonation process.

The chemical processes of a single particle combustion, which mainly occur in the gaseous phase, are significantly faster than the physical processes of particle gasification or disintegration. Thus, in the multiphase mixtures, the rate of energy release will be mostly determined by physics of particle disintegration. It is very difficult to describe the details of particle disintegration in the complex environment prevalent behind the shock or detonation wave. For example, Reinecke and Waldman⁹ defined five different disintegration regimes for a relatively simple environment of water droplets passing through a weak shock. Fortunately, in most cases of multiphase detonation, only the main features of the particle disintegration dynamics need to be captured to describe the phenomena. For example, Eidelman and Burcat¹⁰ used simple models for particle evaporation and shattering to obtain simulation results that compared very favorably with experimental data. Because of

our inability to resolve the particle disintegration problem in all its complexity, the validation of the model against known experimental data is essential.

In this paper, we consider solid particles consisting of explosive material. Explosive material contains fuel and oxidizer in a passive state at low temperature; however, when the temperature rises the fuel and oxidizer react, leading to detonation or combustion. The initiation of reaction for explosives occurs at relatively low temperature. For example, TNT will detonate when heated to the temperature¹¹ of 570°C. Only particles larger than a critical detonation size can detonate directly when initiated by a shock wave. Here, consider particles smaller than 4 mm in diameter that will not detonate when heated, but will burn when the temperature on the particle surface reaches a critical value. Since the heat conduction inside the explosive material is relatively slow, the process of particle heating needs to be resolved in detail. Our simulations numerically solve the temperature field in the particles at every step of numerical integration of the global conservation equations. The explosive particle combustion model examined in this paper assumes that the fraction of the particle that reaches the critical temperature will burn instantaneously.

Energy transfer by convection and conduction is simulated by solving the unsteady heat conduction equation in each computational cell at each time step. Assuming a particle's temperature to be a function of time and radial position only, the unsteady heat conduction equation may be transformed to:

$$\frac{d^2 w}{dr^2} = \frac{1}{\alpha} \frac{dw}{dt} \quad (13)$$

subject to the boundary conditions:

$$w = 0 \quad \text{at} \quad r = 0, \quad t > 0$$

$$k \frac{dw}{dr} + \left(h - \frac{1}{R} \right) w = hRT_g \quad \text{at} \quad r = R, t > 0 \quad (14)$$

where

$$w(r, t) = rT(r, t)$$

r = radial position

$T(r, t)$ = temperature

R = particle radius

T_g = temperature of surrounding gas

k = thermal conductivity of particle

h = convective heat transfer coefficient

The Nusselt number, used to find h , is given by an empirical relation given by Drake.¹² The gas viscosity is derived from Sutherland's Law. The gas thermal conductivity is calculated by assuming a constant Prandtl number. Finally, the boiling temperature at a given pressure is derived from the Clapeyron-Clausius equation under the following assumptions: 1) phrasing-constant latent enthalpy of phase-change; 2) the vapor obeys the ideal equation-of-state; and 3) the specific volume of the solid/liquid is negligible compared to that of the vapor. A critical temperature is also employed to serve as an upper limit to the boiling point, regardless of pressure.

Equation 13 with boundary condition 14 can be numerically integrated using either implicit or explicit schemes.

Since the particle radius R becomes very small due to evaporation, the implicit Crank-Nicolson algorithm is used because of its stability properties and its second order temporal and spatial accuracy. Using the Crank-Nicolson scheme to predict the particle temperature profiles at times t_1 and t_2 permits easy calculation of the total energy exchange Q between t_1 and t_2 , due to convection and conduction.

Knowledge of the particle temperature profile also allows the precise determination of the quantity of the mass to transfer from the particle to the gas Γ . Once any point at a radial location $0 \leq r \leq R$ has a temperature exceeding the boiling temperature, the entire mass between r and R is transferred to the gas phase in one time step. In so doing, an energy equal to the product of the mass lost and the particle intrinsic energy is transferred by the particle to the gas.

The interphase drag force F_x, F_y is determined from the experimental drag for a sphere, as presented by Schlichting.¹³

$$F_x = \left(\frac{\pi}{8}\right) N_p \rho_g C_D |V_g - V_p| (u_g - u_p) R^2 \quad (15)$$

where

$$C_D = \begin{cases} \frac{24}{Re} \left(1 + \frac{Re^{2/3}}{6}\right) & \text{for } Re < 1000; \\ 0.44 & \text{for } Re > 1000 \end{cases} \quad (16)$$

and $Re = \frac{2R|V-V_p|}{\mu_g}$, R is radius of particle, and μ_g is gas viscosity at temperature of $T_{film} = \frac{1}{2}(T_g + T_p)$. Similarly, the formulae for F_y is

$$F_y = \frac{\pi}{8} N_p \rho_g C_D |V_g - V_p| (v_g - v_p) R^2 \quad (17)$$

Equation-of-State for Detonation Products

To close the system of governing equations, one needs a constitutive relation between pressure, temperature, and energy for gas phase, which is an equation-of-state. This study uses the Becker-Kistiakowsky-Wilson (BKW) equation-of-state,^{14,15} that is,

$$p_g V_g / \bar{R} T_g = 1 + x e^{bx} \quad (18)$$

where

- V_g = volume of gas phase
- p_g = pressure of gas phase
- T_g = temperature of gas phase
- \bar{R} = universal gas constant
- $x = k/V_g(T + \Theta)^a$
- $k = K \sum_j X_j k_i$

with empirical constants a , b , K , Θ , and k_i . The constants k_i , one for each molecular species, are covolumes. The covolumes are multiplied by their mole fraction of species X_i and are added to find an effective volume for a mixture. For a particular explosive, if we know the composition of detonation products, a , b , Θ , K , and all k_i s can be found in Ref. 15.

The internal energy is determined by thermodynamics relation

$$\left(\frac{\partial E_g}{\partial V_g}\right)_T = T_g \left(\frac{\partial p_g}{\partial T_g}\right)_V - p_g \quad (19)$$

Integration of this equation for a fixed composition of the detonation products will allow us to calculate the energy of the detonation products as a function of temperature and volume. For each component, its thermodynamic properties as functions of temperature were calculated from the NASA tables compiled by Gordon and McBride.¹⁶

The BKW equation-of-state is the most commonly used and well-calibrated of those equations-of-state used to calculate the properties of detonation products. The detailed discussion and review of the BKW equation-of-state can be found in Ref. 15.

Numerical Method of Solutions

The system of partial differential equations described in the previous paragraph is integrated numerically. The Second Order Godunov method is used for the integration of the subsystem of equations describing flow of gaseous phase material and is described in Ref. 17. In the following, we will elaborate only on some specifics of its application to simulations of detonation products. The subsystem of equations describing the flow of particles is integrated using a simple upwind integration. This is done because our mathematical model neglects the pressure of interparticle interaction, and that prevents formulation of a Second Order Godunov scheme for particles.

The physical system under study will have concentrations of solid explosive powder ranging from 1000 kg/m³ near the

ground to 0.75 kg/m^3 or less in the cloud. Detonation of this mixture will create detonation products with effective γ ranging from 3 to 1.1. To describe the flow of detonation products, we use the BKW equation-of-state described above. Since the Second Order Godunov method uses primitive variables to calculate Riemann problems at the edges of the cells, its implementation for non-ideal EOS is difficult. In our simulations, we have resolved this problem by using direct and inverse equations-of-state. After integrating a system of gas conservation laws, we use the direct BKW equation-of-state to calculate pressure, gamma, and temperature as functions of thermal energy, density, and mixture composition. After this step, we have a complete set of parameters allowing calculation of the fluxes in the Second Order Godunov method as well as interaction of the multiphase processes. The "inverse" EOS calculates internal energy as a function of density, pressure, and mixture composition. In our code, we use the "inverse" EOS to calculate the fluxes of conserved variables after calculation of the flux of primitive variables.

For the multiphase system under study, $dx=dy=1\text{mm}$ was used to allow explicit integration of the gasdynamic and physical processes of evaporation and heat release. When a mismatch occurred between the physical and gasdynamical characteristic times, the time step was adjusted by some fraction to assure stability. However, this did not result in a significantly smaller time step than the one calculated using CFL criteria. For larger cell sizes, this approach will be impractical. Recently, we implemented a scheme in which multiphase processes are calculated implicitly; however, this will be reported elsewhere.

The numerical method is implemented in a code named MPHASE, which is fully vectorized and supported by number of graphics and diagnostics codes.

Results

Model Validation for One-Dimensional Detonation Wave Problem

The main advantage of our particle combustion model is its description of the phenomenology of detonation for a wide

Table 1 One-dimensional validation result

D_{det}(m/s) - Detonation wave velocity,
 P_{CJ}(Pa) - Pressure at Chapman-Jouguet Point
 P_p(Pa) - Peak pressure; ρ_p(kg/m³) - Peak density

RDX density (kg/m ³)	Parameters	Present calculation	Expt'l Ref. 1	Tiger calculation - Ref. 3	BKW calculation - Ref. 1	Soviet experiments - Ref. 2
1000 kg/m ³	D	6156	5361		6128	
	P _{CJ}	1.220 × 10 ¹⁰			1.00 × 10 ¹⁰	1.00 × 10 ¹⁰
	P _p	2.57 × 10 ¹⁰				
	ρ _p	1936				
500 kg/m ³	D	6081		6000		
	P _{CJ}	0.905 × 10 ¹⁰		0.80 × 10 ¹⁰		0.82 × 10 ¹⁰
	P _p	1.00 × 10 ¹⁰				
	ρ _p	1722				
400 kg/m ³	D	4000		4000		
	P _{CJ}	0.370 × 10 ¹⁰		0.20 × 10 ¹⁰	0.3 × 10 ¹⁰	
	P _p	0.625 × 10 ¹⁰				
	ρ _p	926				
250 kg/m ³	D	4000		3000		
	P _{CJ}	0.2478 × 10 ¹⁰		0.13 × 10 ¹⁰		
	P _p	0.4836 × 10 ¹⁰				
	ρ _p	552				
100 kg/m ³	D	2400				
	P _{CJ}	0.6093 × 10 ⁹				
	P _p	0.7686 × 10 ⁹				
	ρ _p	720				
0.75 kg/m ³	D	1632	1410 ^a	1670 ^b		
	P _{CJ}	0.26 × 10 ⁷	0.200 × 10 ^{7c}	0.26 × 10 ^{7c}		
	P _p	0.404 × 10 ⁷				
	ρ _p	0				

Ref. 1 - Meder, C., *NUMERICAL MODELING OF DETONATIONS*, (University of California Press, Ltd., 1978) p. 47. Ref. 2 - Windermann, A., "An Evaluation of Homotopy Layer Loading Effects," *ITRI Report*, Feb. 1968. Ref. 3 - Stamborovich, K. P., "Physics of Explosions" (in Russian), Nauka, 1975.

range of explosive particle sizes and densities. We will demonstrate this capability on a set of one-dimensional test problems. For these test problems, we simulated the initiation and propagation of the detonation waves in a shock tube-like setting, where the explosive particles are distributed uniformly through the shock tube volume.

Results of these simulations are summarized in Table 1, which shows detonation wave velocity, peak pressure, and peak density given as a function of the average density of the solid explosive. Here, the explosive two-phase mixture is composed from RDX particle and air, where RDX particle concentration varies from 0.75 kg/m³ to 1000 kg/m³. This concentration variation covers a whole range of solid explosive concentrations of interest to our problem. The simulations performed with the MPHASE code were compared with the experimental results^{15,18} and calculations done with the TIGER code that are presented in Ref. 19.

From Table 1, it is clear that our simulation results compare favorably with other simulation results and experimental data. The maximum deviation between our results and referenced results is no greater than 15% for the entire range of explosives densities. Considering that our results were obtained with a single model for particle combustion applied to the extreme range of densities, our model gives an excellent prediction of the detonation wave parameters.

Two-Dimensional Simulation Results

Figure 1 shows a setup for a typical simulation with a computational domain of 25 cm \times 25 cm. The explosive powder density is distributed according to the 4th power law of vertical distance, starting from the ground where the density is 1000 kg/m³, to 1.2 cm, where the density is 0.75 kg/m³. From this point to 25 cm height, the density is constant and equal to 0.75 kg/m³. The density distribution in the direction of the "x" axis is uniform. The boundary conditions for the computational domain shown in Fig. 1 are specified as follows: solid wall along the "x" axis, symmetry conditions along the "y" axis, supersonic outflow for upper boundary, and at the

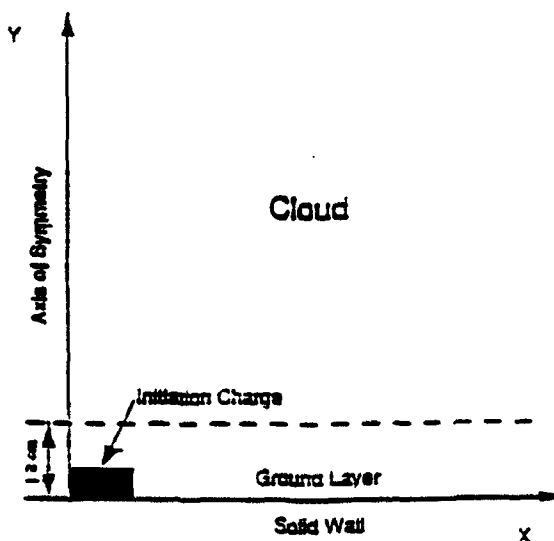


Fig. 1 Computational domain and boundary conditions.

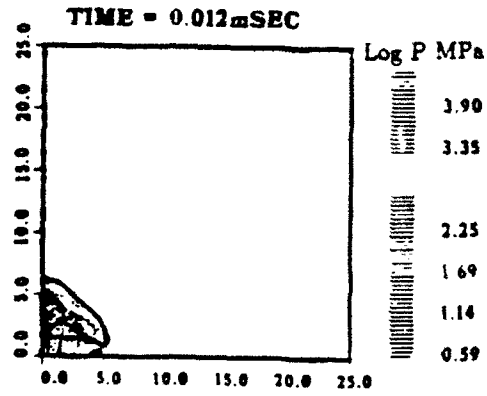
right of the computational domain. The mixture consists of RDX powder and air at ambient conditions, and it is assumed to be quiescent at the time of initiation.

The simulation starts at $t=0$ when the mixture is initiated at the lower left corner of the computational domain, as shown in Fig. 1. The energy released by the initiating explosion leads to formation of the detonation wave propagating through the multiphase media. Figure 2a shows pressure contours for the propagating detonation wave at the time of $t=0.012$ msec after initiation. The pressure contour levels are shown on the logarithmic scale in MPa. The maximum pressure value of 7940 MPa is observed in the layer of condensed explosive located near the ground. The pressure in the layer is two to three orders of magnitude higher than pressure behind the detonation wave in the 0.75 kg/m^3 RDX cloud and air located above the distance of 1.2 cm from the ground. Figure 2a demonstrates that the detonation wave in the cloud is overdriven, since the pressure behind the shock continuously rises and reaches its maximum in the layer. From this figure, we also observe that the overdriven wave propagates faster in the cloud than in the layer. This is explained by the fact that it is easier to compress air that is very lightly loaded with particles and located above the ground layer than it is to compress air heavily loaded with a particle mixture near the ground. It is interesting to note a discontinuous pressure change between the yellow contours and the light blue and green contours behind the detonation front. This discontinuity is overemphasized by our presentation of contour lines on the logarithmic scale; however, further examination of our simulation results indicates this feature is real and is similar in nature to barrel shocks observed for strong jets.

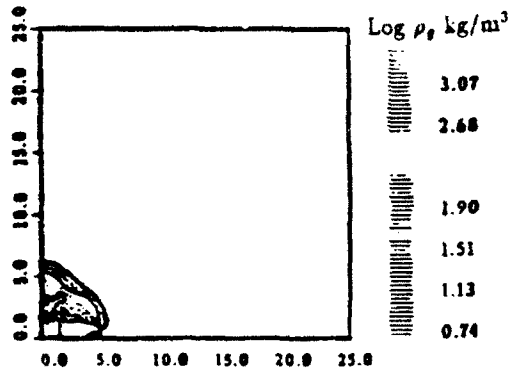
In Fig. 2b, gas-phase density contours are shown for the time $t=0.012$ msec. Here the contour lines are distributed on the logarithmic scale. The main features of the shock wave structure are very similar to those observed in the pressure contours figure. We see that a jet of high-density gases reflects from the center of symmetry axis, which will create a contact discontinuity that we will observe at later times. The barrel

shock is clearly visible in this figure. In Fig. 2c, the particle density contour plots are shown for $t=0.012$ msec. The contour levels in Fig. 2c are given on the logarithmic scale and the initial deposition of the explosive material in the ground layer of the computational domain can be clearly observed. The white contour line delineates the beginning and the end of the reaction zone in the cloud. To the left of these contours lies an area with combustion products and to the right are unburned particles in the cloud. The reaction zone length is of the order of 1 cm.

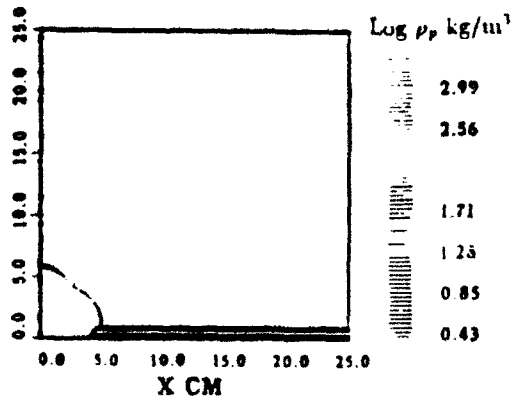
Figure 2d shows pressure contours for the same simulation for the time $t=0.055$ msec, just before the detonation wave leaves the computational domain. In this figure, we see that the global structure of the wave did change slightly from Fig. 2a. We observe that the barrel shock wave is fully developed and has a half-ellipse shape. The detonation wave in the cloud is still overdriven; however, part of the shock wave front that propagates vertically weakened because it gets further away from the detonation front in the layer. Another noticeable feature is the increase in distance between the detonation front in the layer and in the cloud area close to the layer. This is a result of the fact that the lightly loaded two-phase media above the layer can be compressed much more easily than the particle-heavy ground layer. In Fig. 2e, temperature contours are shown for $t=0.055$ msec. Comparing this figure with an early stage of the wave propagation, we observe a significant cooling of the front area propagating upwards, which indicates transition from the overdriven detonation regime to a self-sustained detonation. We also observe in Fig. 2a clear development of two detonation fronts, one moving vertically in the cloud and another moving horizontally in the layer. Because the energy density of the explosive powder in the layer is about three orders of magnitude larger than in the cloud, the vertical parts of the front represent an overdriven detonation wave in the cloud. Even though the vertical front has slowed down compared with the horizontal front, its speed and parameters far exceed those typical for detonation waves in a cloud. In fact, the self-sustained detonation regime in the cloud will



a) Pressure.

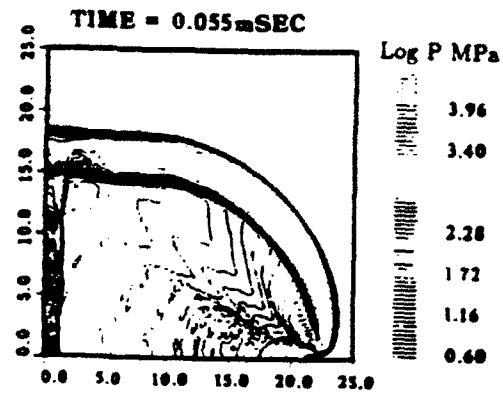


b) Gas Density.

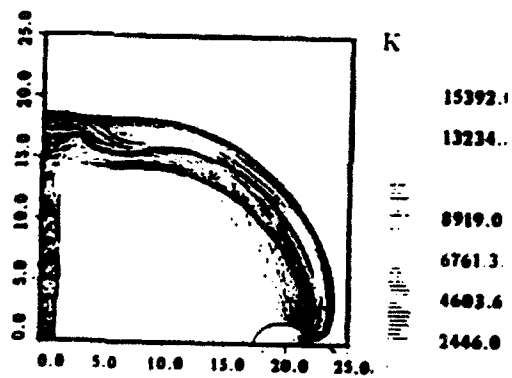


c) Particle Density.

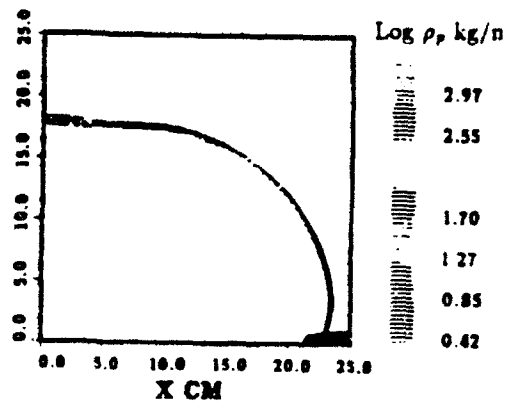
Fig. 2 Fourth power layer distribution; maximum density in the layer 800 kg/m³; density in the cloud 0.75 kg/m³; time 0.012 m/s and 0.055 m/s after initiation.



d) Pressure.



e) Temperature.



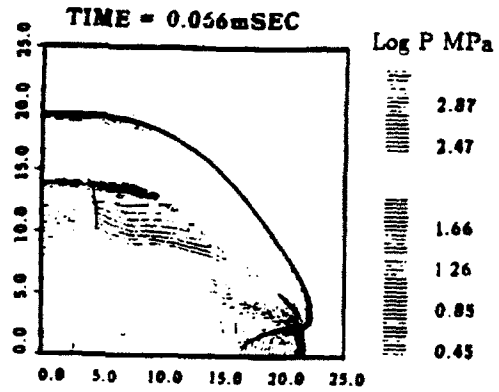
f) Particle Density.

Fig. 2 (continued) Fourth power layer distribution: maximum density in the layer 800 kg/m^3 ; density in the cloud 0.75 kg/m^3 ; time 0.012 m/s and 0.055 m/s after initiation.

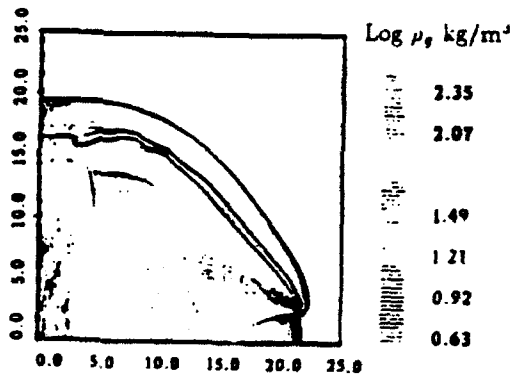
develop at the distance of about 3 m from the layer. The area of the front close to the detonation wave in the layer will remain hot and overdriven, since it is located very close to the detonation front in the layer. In Fig. 2f, particle density contours are shown on a logarithmic scale. We can clearly observe the reaction zone delineated by black contour lines. In this case, the reaction zone length in the cloud is about 1 cm. Consistent with the gradual transition from overdriven to self-sustained detonation, the reaction zone length is larger for the vertical part of the detonation front. The detonation wave velocity observed in our simulation is approximately 4048 msec, which is significantly lower than the detonation wave velocity observed in RDX with a density of 860 kg/m^3 (see Table 1), the highest density in the ground layer. This can be explained by high gradient of particle density distribution in the layer, where the density drops rapidly from 860 kg/m^3 at the bottom of the layer to 1 kg/m^3 at the top strata of the layer at 12 mm above the ground.

To further explore properties and phenomenology of the detonation waves propagating in the layer/cloud systems, we simulated additional cases in which explosive powder density distribution was different from the case reported above, although total weight of fuel per unit area remained the same.

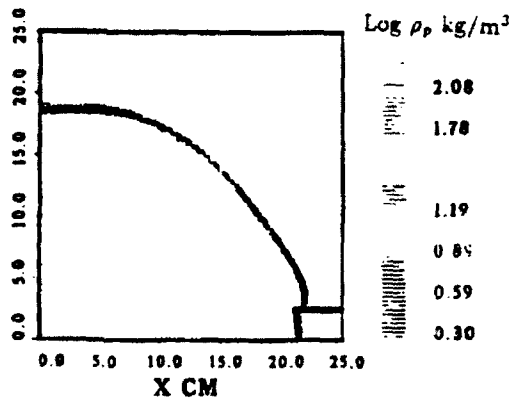
In Fig. 3, results are shown for the case of a uniform 2.5 cm-thick layer of RDX with a density of 100 kg/m^3 and a 0.75 kg/m^3 cloud initiated under the same conditions as in the previous example. Figures 3a, 3b, and 3c show pressure, gas density, and particle density contour plots at $t=0.066 \text{ msec}$. We observe that because the layer has considerably smaller density compared to the case reported above, the precursor effect of the detonation wave in the cloud preceding the wave in the layer is less pronounced. Also, one can observe a significant difference in the shape of the strong contact discontinuity in the region of the shock front close to the layer. In Fig. 3b, we can clearly distinguish two contact surfaces. One is between condensed explosive detonation products in the layer and in the cloud, and another is between the detonation products from



a) Pressure

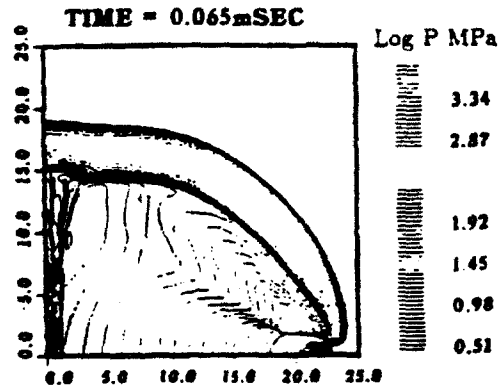


b) Gas Density

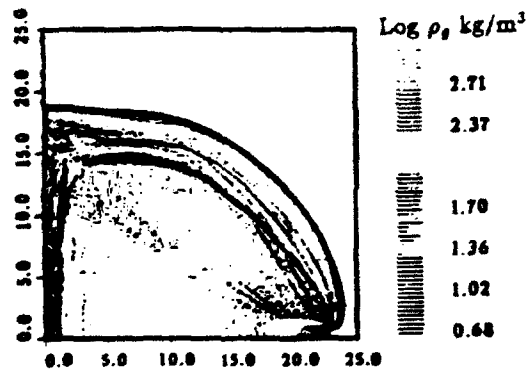


c) Particle Density

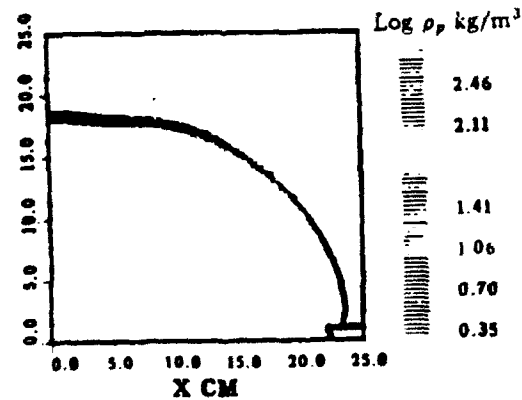
Fig. 3 Constant density 2.5-cm-thick layer; maximum density in the layer 100 kg/m³; density in the cloud 0.75 kg/m³; time 0.055 m/s after initiation.



a) Pressure.



b) Gas Density.



c) Particle Density.

Fig. 4 Constant density 1.2-cm-thick layer; maximum density in the layer 250 kg/m³; density in the cloud 0.75 kg/m³.

layer explosive detonation and from cloud particle detonation. We should note that these contact surfaces are overemphasized by the logarithmic display of the contour plot levels. The maximum pressure observed in this simulation is 955 MPa, which is about one order of magnitude smaller than in previous simulation. This is consistent with the one order of magnitude difference in the maximum density of the ground layer in the two cases. The detonation wave speed in the case presented in Fig. 3 is 3407 msec. That is only slightly lower than the speed predicted by one-dimensional simulations presented in Table 1, which reflects the influence of the two-dimensional expansion on the detonation wave propagation.

Figure 4 presents results for the case of a uniform density of 250 kg/m^3 in 1.2 cm ground layer. All other parameters are the same as in the previous two cases. In Figs. 4a, 4b, and 4c, pressure, gas density, and particle density contour plots are shown at the time $t=0.066$ msec after initiation of the detonation wave. Here, the detonation wave propagates faster than in the previous cases $U=3660$ msec. This is about 400 msec slower than in the case of parabolic density distribution. Maximum pressure on the ground is 2150 MPa, which is consistent with the increase of powder density in the layer. The basic structure of the detonation front and the contact surfaces is similar to the case of parabolic density distribution.

Conclusions

We have presented a mathematical model and numerical solution for the simulation of initiation and propagation of the detonation waves in multiphase mixtures consisting of solid combustible particles and gas. Using this model, we studied detonations in mixtures of solid RDX particles and air, with the objective of examining the effects of wide variation in particle density distribution on the dynamics and structure of detonation waves. We considered a physical system of solid particle clouds in air, in which a significant amount of particles settle on the ground and the condensed-phase concentrations in the particle/air mixture range from 0 to 1000 kg/m^3 . This range of solid-phase densities necessitated development of the

model and its numerical implementation for a wide range of particle concentrations. Our validation study has shown good agreement between the simulations and referenced results for the whole range of particle concentrations.

Two-dimensional simulations were done for the system of low particle density concentration clouds and ground layers formed by high concentrations of the RDX powder. We examined three cases of ground layer density distribution: a fourth power distribution within 12 mm above ground with a maximum density on the ground of 860 kg/m^3 ; a uniform 25 mm-thick layer with a density of 100 kg/m^3 ; and a 12 mm-thick uniform layer with a density of 250 kg/m^3 . In all these cases, the weight of condensed phase per unit area was the same, which allowed examination of the effects of the particle density distribution on detonation wave parameters.

In all examined two-dimensional cases, the detonation wave in the cloud in the computational domain was significantly overdriven and did not play an important role. We estimated that the self-sustained regime of the detonation wave in the cloud for the examined cloud concentrations can occur only at the distances of 2-3 m above ground. At the same time, the particle density distribution in the layer determines the dynamics of the detonation wave as well as pressure on the ground.

In all three two-dimensional simulations, we observed a very distinct shape of the detonation wave front in the vicinity of the layer. In this area, the overdriven detonation in the cloud is preceding the detonation wave in the ground layer. This feature of the detonation front can be explained by the fact that the energy released in the detonation wave in the ground layer produces a faster shock wave in the dilute cloud than in those heavily loaded with solid particle stratas from the ground layer. However, these structures were not observed experimentally, and more studies are needed to examine their parameters.

The maximum pressure affecting the ground was directly related to the maximum particle density in the lower strata of the layer. However, the detonation front velocity for the fourth

power distribution case was considerably lower than calculated for a one-dimensional case with 860 kg/m^3 particle density, reflecting the significant effect of two-dimensional expansion. Two other cases with 250 kg/m^3 and 100 kg/m^3 maximum densities had the detonation wave velocity only slightly lower than the one-dimensional simulations of the same RDX/air concentrations. It is interesting to compare the simulation of the fourth power density distribution case and 250 kg/m^3 case. In both cases, the same amount of explosive was distributed in the same physical space; however, the parameters of developed detonations were vastly different. Existence of the high-density strata at the bottom of the ground layer in the fourth power case significantly increased the maximum pressure at the ground and produced higher detonation wave velocity.

Using a variable density layer, one can reach a combination of pressure and velocity conditions outside of Chapman-Jouget limitations. The range of conditions that can be obtained in the variable density system and the parametrics for this range need a more systematic study. In this article, we introduced only the mathematical formulation and numerical simulation method validated for the range of conditions of interest. In addition, we have given some examples of its application for two-dimensional simulations. However, this methodology should be linked to an experimental study for a more in-depth analysis of the phenomenology discussed here.

References

¹Eidelman, S., Timnat, Y. M., and Burcat, A., "The Problem of a Strong Point Explosion in a Combustible Medium," 6th Symposium on Detonation, Office of Naval Research, Coronado, CA, 1976, p. 590.

²Burcat, A., Eidelman, S., and Manheimer-Timnat, Y., "The Evolution of a Shock Wave Generated by a Point Explosion in a Combustible Medium," Symposium of High Dynamic Pressures (H.D.P.), Paris, 1978, p. 347.

³Oved, Y., Eidelman, S., and Burcat, A., "The Propagation of Blasts from Solid Explosives to Two-Phase Medium," *Propellants and Explosives*, Vol. 3, No. 105, 1978.

⁴Eidelman, S., and Burcat, A., "The Evolution of a Detonation Wave in a Cloud of Fuel Droplets; Part I, Influence of the Igniting Explosion," *AIAA Journal*, Vol. 18, 1980, p. 1103.

- ⁵Liu, J. C., Kauffman, C. W., and Sichel, M., "The Lateral Interaction of Detonating and Detonable Mixtures," Private communication, 1990.
- ⁶Kuo, K., *Principles of Combustion*, John Wiley and Sons, Inc., New York, NY, 1990, pp. 513-626.
- ⁷Kauffman, C. W., et al., "Shock Wave Initiated Combustion of Grain Dust," *Symposium on Grain Dust*, Manhattan, KS, 1979.
- ⁸Eidelman, S., and Burcat, A., "Numerical Solution of a Non-Steady Blast Wave Propagation in Two-Phase ('Separated Flow') Reactive Medium," *Journal of Computational Physics*, Vol. 39, 1981, p. 456.
- ⁹Reinecke, W. G., and Waldman, G. D., "Shock Layer Shattering of Cloud Drops in Reentry Flight," AIAA Paper 75-152, 1975.
- ¹⁰Eidelman, S., and Burcat, A., "The Mechanism of Detonation Wave Enhancement in a Two-Phase Combustible Medium," 18th Symposium on Combustion, The Combustion Institute, Waterloo, Ontario, Canada, 1980, pp. 1661-1670.
- ¹¹*Engineering Design Handbook, Explosives Series, Properties of Explosives of Military Interest*, AMC Pamphlet, AMCP 706-7177, 1971.
- ¹²Drake, R. M., Jr., "Discussions on G. C. Vliet and G. Leppert: Forced Convection Heat Transfer from an Isothermal Sphere to Water," *Journal of Heat Transfer*, Vol. 83, 1961, p. 170.
- ¹³Schlichting, H., *Boundary Layer Theory* 7th ed., McGraw-Hill, New York, 1983.
- ¹⁴Cowan, R. D., and Fickett, W., "Calculation of the Detonation Products of Solid Explosives with the Kistiakowsky-Wilson Equation of State," *Journal of Chemical Physics*, Vol. 24, 1956, p. 932.
- ¹⁵Mader, C. L., *Numerical Modeling of Detonation*, University of California Press, Ltd. London, England, 1979.
- ¹⁶Gordon, S., and McBride, B. J., "Computer Program for Calculations of Complex Chemical Equilibrium Compositions, Rocket Performance, Incident and Reflected Shocks and C-J Detonations," NASA SP-273, 1976 (revision).
- ¹⁷Eidelman, S., Collela, P., and Shreeve, R. P., "Application of the Godunov Method and Its Second Order Extension to Cascade Flow Modelling," *AIAA Journal*, Vol. 22, 1984, p. 10.
- ¹⁸Stanukovitch, K. P., *Physics of Explosion* (in Russian), Nauka, 1975.
- ¹⁹Wiedermann, A., "An Evaluation of Bimodal Layer Loading Effects," *IITRI Report*, February 1990.

Detonation Wave Propagation in Combustible Particle/Air Mixture with Variable Particle Density Distributions

SHMUEL EIDELMAN and XIAOLONG YANG *Science Applications International Corporation McLean, VA 22102*

(Received June 11, 1991; in final form June 18 1992)

Abstract—A mathematical model is presented describing a physical system of detonation waves propagating in a solid particle/air mixture with a wide range of solid phase concentrations. The mathematical model was solved numerically using the Second Order Godunov method, and numerical solutions were validated for detonation waves propagating in mixtures with concentrations of solid phase from 0.75 kg/m^3 to 1000 kg/m^3 . Numerical solution was obtained for detonation waves propagating in a system consisting of clouds with a small concentration of particles and a ground layer in which solid particle densities are three orders of magnitude larger than in the cloud. Three different particle concentration distributions in the ground layer were simulated and compared in terms of detonation wave structure and parameters.

Key words: detonation wave, two-phase flow, numerical simulation

1. INTRODUCTION

When combustible particles are intentionally or unintentionally dispersed into the air, the resulting mixture can be detonable. Formation of this potentially explosive dust environment and the properties of its detonation are of significant practical interest in view of its destructive or creative effects. The experimental and theoretical study of these phenomena until now has addressed only homogenous particle/oxidizer mixtures. However, intentional or accidental processes of the explosive dust dispersion will always lead to inhomogeneous particle density distribution. Some industrial methods of explosive forming rely on detonation of explosive powder. This powder can be deposited as a thin layer over the surface area of the forming metal, with some remaining concentration in the vicinity of the layer. The structure of the detonation waves and the phenomenology of their initiation and propagation in these environments are the main subjects of this paper.

When the detonation wave is generated in a homogeneous mixture by a "direct initiation," it starts with a strong blast wave from the initiating charge. As the blast wave decays, combustion of the reactive mixture behind its shock front starts to have a larger role in support of the shock wave motion. When the initial explosion energy exceeds some critical value, transition to steady state detonation occurs (cf. Eidelman *et al.*, 1976; Burcat *et al.*, 1978; Oved *et al.*, 1978; Eidelman and Burcat, 1980). In explosive dust mixtures with a nonuniform distribution of particle density, the initiation dynamics is significantly more complicated. The critical initiation energy sufficient for one of the explosive particle density strata regions is not necessarily adequate for other regions. Also, when there is a significant variation in density between the different layers (regions) of the mixture, steady detonation in one layer can result in an overdriven detonation in an adjacent layer. Our paper demonstrates that the phenomenology of these interactions is distinctly different from the classical studies of multi-layer detonations in gases. This is primarily because the energy content of adjacent layers in a typical multi-gas layer experiment varies by a factor of two or four (Liu *et al.*, 1990), whereas the energy content in explosive dust/air mixtures can vary by several orders of magnitude.

In this paper we use detailed numerical simulation to study the initiation dynamics and propagation phenomenology for a general case of explosive dust dispersion. We will consider particle density variation from 1000 kg/m^3 in the ground layer to 0.75 kg/m^3 for the upper edges of the cloud. The effects of the cloud density variation on detonation wave parameters will be examined for different cases of cloud particle density distribution. When possible, the results of computer simulations are validated in comparison with experimental and theoretical studies.

The outline of this paper is as follows. Section 2 gives a description of mathematical model that includes governing conservation equations for two phases and the constitutive laws. We describe the model for a particle gas interaction, combustion and equation-of-state for gas phase. The numerical integration technique for solving the mathematical model will also be outlined. In Section 3, we present our numerical simulation results. We first validate our model by comparing one dimensional detonation wave simulation with available experimental results. We then give the two dimensional simulation for detonation wave propagation in combustible particles/air mixtures with variable particle density distribution. Concluding remarks are given in Section 4.

2. THE MATHEMATICAL MODEL AND THE NUMERICAL SOLUTION

The mathematical model consists of conservation governing equations and constitutive laws that provide closure relations for the model. The basic formulation adopted here follows the two-phase fluid dynamics model presented in the text by Kuo (1990). The approach assumes that there are two distinct continua, one for gas and one for solid particles, each moving at its own velocity through its own control volume. The sum of these two volumes represents an average mixture volume. Furthermore, particles in their own control volume are assumed monodisperse and they are moving with the same velocity. With these assumptions, distinct equations for continuity, momentum and energy are written for each phase. The interaction effects between the two phases are accounted as the source terms on the right hand side of the governing equation. The following is a short description of the two phase flow model used in our study, with conservation equations written in Eulerian form for two dimensional flow in Cartesian coordinates.

Conservation Equations

Continuity of gaseous phase:

$$\frac{\partial \rho_1}{\partial t} + \frac{\partial(\rho_1 u_g)}{\partial x} + \frac{\partial(\rho_1 v_g)}{\partial y} = \Gamma; \quad (2.1)$$

Continuity of solid particle phase:

$$\frac{\partial \rho_2}{\partial t} + \frac{\partial(\rho_2 u_p)}{\partial x} + \frac{\partial(\rho_2 v_p)}{\partial y} = -\Gamma; \quad (2.2)$$

Conservation of momentum of gaseous phase in x -direction:

$$\frac{\partial(\rho_1 u_g)}{\partial t} + \frac{\partial(\rho_1 u_g^2 + \phi p_g)}{\partial x} + \frac{\partial(\rho_1 u_g v_g)}{\partial y} = -F_x + \Gamma u_p; \quad (2.3)$$

Conservation of momentum of solid particle phase in y -direction:

$$\frac{\partial(\rho_1 v_g)}{\partial t} + \frac{\partial(\rho_1 u_g v_g)}{\partial x} + \frac{\partial(\rho_1 v_g^2 + \phi p_g)}{\partial y} = -F_y + \Gamma v_p; \quad (2.4)$$

Conservation of momentum of solid particle phase in x -direction:

$$\frac{\partial(\rho_2 u_p)}{\partial t} + \frac{\partial(\rho_2 u_p^2)}{\partial x} + \frac{\partial(\rho_2 v_p u_p)}{\partial y} = F_x - \Gamma u_p; \quad (2.5)$$

Conservation of momentum of solid particle phase in y -direction:

$$\frac{\partial(\rho_2 v_p)}{\partial t} + \frac{\partial(\rho_2 u_p v_p)}{\partial x} + \frac{\partial(\rho_2 v_p^2)}{\partial y} = F_y - \Gamma v_p; \quad (2.6)$$

Conservation of energy of gas phase:

$$\begin{aligned} \frac{\partial(\rho_1 E_g T)}{\partial t} + \frac{\partial(\rho_1 u_g E_g T + u_g \phi p_g)}{\partial x} + \frac{\partial(\rho_1 v_g E_g T + v_g \phi p_g)}{\partial y} = \\ \Gamma \left(\frac{u_p^2 + v_p^2}{2} + E_{chem} + C_s \bar{T}_p \right) - (F_x u_p + F_y v_p) = \dot{Q}; \end{aligned} \quad (2.7)$$

Conservation of energy of solid particle phase:

$$\begin{aligned} \frac{\partial(\rho_2 E_p T)}{\partial t} + \frac{\partial(\rho_2 E_p T u_p)}{\partial x} + \frac{\partial(\rho_2 E_p T v_p)}{\partial y} = \dot{Q} + (F_x v_p + F_y v_p) \\ - \Gamma \left(\frac{u_p^2 + v_p^2}{2} + E_{chem} + C_s \bar{T}_p \right); \end{aligned} \quad (2.8)$$

Conservation of number density of solid particle:

$$\frac{\partial N_p}{\partial t} + \frac{\partial(N_p u_p)}{\partial x} + \frac{\partial(N_p v_p)}{\partial y} = 0. \quad (2.9)$$

In the above equations, $\phi = 1 - \frac{N_p M_p}{\rho_p}$, $\rho_1 = \phi \rho_g$, $\rho_2 = (1 - \phi) \rho_p$, where N_p and M_p are the number density of particles and mass of each particle, respectively, and ρ_g and ρ_p are the material density of gas and particle densities, respectively. u_g , v_g , p_g are gas phase x -velocity, y -velocity and pressure, respectively; u_p , v_p , \bar{T}_p are x -velocity, y -velocity and average particle temperature, respectively. C_s is the solid particle specific heat, and $E_{chem} = E_{comb} - E_{evap}$, where E_{comb} is heat of combustion and E_{evap} is heat of evaporation. Γ is the rate of phase change from solid to gas and Q is heat transfer between the two phases; F_x , F_y are drag force between the two phases in x and y directions, respectively.

Equations (2.2) and (2.9) are linked through the relation $\rho_2 = N_p M_p$. In the case of a reactive solid phase, M_p decreases due to combustion. The mass of a single particle at any point can be obtained from $M_p = \rho_2(x, y) / N_p(x, y)$, and the diameter of a particle at any spatial location is $D(x, y) = [6M_p(x, y) / \pi \rho_p]^{1/3}$. The total internal energy of gaseous phase

$$E_g T = E_g + \frac{1}{2}(u_g^2 + v_g^2) \quad \text{and} \quad E_g = E_g(p_g, \rho_g) \quad (2.10)$$

where $E_g(p_g, \rho_g)$ is the equation-of-state for gas phase, which will be discussed later.

The total internal energy of solid particle phase is

$$E_{pT} = E_p + \frac{1}{2}(u_p^2 + v_p^2) \quad \text{and} \quad E_p = E_{comb} + C_s \bar{T}_p. \quad (2.1)$$

In order to close the above system of conservation equations, it is necessary to define certain criteria and interaction laws between the two phases, which include mass generation rate, Γ , drag force between particles and gas, F_x, F_y and the interphase heat transfer rate \dot{Q} . The model for particle and gas interaction and particle combustion that results in the constitutive relation for the conservation equations, is explained in detail in the next subsection.

Model for a Particle Gas Interaction and Combustion

Presently the physics of the energy release mechanisms in solid particles/air mixtures is not clearly understood. This can be attributed to the obvious difficulties of making a direct non-obtrusive measurement in the optically thick environment typical for this system. In the experimental and theoretical work done for the grain dust detonation conditions (Kauffman *et al.*, (1979)), it was demonstrated that the volatile components released by the particle heated behind the shock front play a major role in determining the detonability limits of the mixture. Eidelman and Burcat (1981) successfully applied a combination of fast evaporation and aerodynamic shattering mechanisms to simulate a two-phase detonation process.

The chemical processes of a single particle combustion, which mainly occur in the gaseous phase, are significantly faster than the physical processes of particle gasification or disintegration. Thus, in the multi-phase mixtures, the rate of energy release will be mostly determined by physics of particle disintegration. It is very difficult to describe the details of particle disintegration in the complex environment prevalent behind the shock or detonation wave. For example, Reinecke and Waldman (1975) defined five different disintegration regimes for a relatively simple environment of water droplets passing through a weak shock. Fortunately, in most cases of multi-phase detonation, only the main features of the particle disintegration dynamics need to be captured to describe the phenomena. For example, Eidelman and Burcat (1980) used simple models for particle evaporation and shattering to obtain simulation results that compared very favorably with experimental data. Because of our inability to resolve the particle disintegration problem in all its complexity, the validation of the model against known experimental data is essential.

In this paper we consider solid particles consisting of explosive material. Explosive material contains fuel and oxidizer in a passive state at low temperature; however, when the temperature rises the fuel and oxidizer react, leading to detonation or combustion. The initiation for explosives will occur at a relatively low temperature. For example, TNT will detonate when heated to the temperature of 570°C. Only particles larger than a critical detonation size can detonate directly when initiated by a shock wave. We consider here particles smaller than 4mm in diameter that will not detonate when heated, but will burn when the temperature on the particle surface reaches a critical value. Since the heat conduction inside the explosive material is relatively slow, the process of particle heating needs to be resolved in detail. Our simulations numerically solve the temperature field in the particles at every time step of numerical integration of the global conservation equations. The explosive particle combustion model examined in this paper assumes that the fraction of the particle that reaches the critical temperature will burn instantaneously. Energy transfer by convection and conduction is simulated by solving the unsteady heat

conduction equation in each computational cell at each time step. Assuming a particle's temperature T_p to be a function of time and radial position only, the unsteady heat conduction equation may be transformed to:

$$\frac{d^2w}{dr^2} = \frac{1}{\alpha} \frac{dw}{dt}, \quad (2.12)$$

subject to the boundary conditions:

$$w = 0 \text{ at } r = 0, t > 0$$

$$k \frac{dw}{dr} = (h - \frac{1}{R})w = hRT_g \text{ at } r = R, t > 0 \quad (2.13)$$

where:

- $w(r, t)$ = $r T_p(r, t)$
- r = radial position
- $T(r, t)$ = temperature
- R = partial radius
- T_g = temperature of surrounding gas
- k = thermal conductivity of particle
- h = convective heat transfer coefficient.

The Nusselt number, used to find h , is given by an empirical relation given by Drake (1961). The gas viscosity is found from Sutherland's Law. The gas thermal conductivity is calculated by assuming a constant Prandtl number. Lastly, the boiling temperature at a given pressure is found from the Clapeyron-Clausius equation under the assumptions of: 1) constant latent enthalpy of phase change, 2) the vapor obeys the ideal equation-of-state, and 3) the specific volume of the solid/liquid is negligible compared to that of the vapor. A critical temperature is also employed to serve as an upper limit to the boiling point, regardless of pressure.

Equation (2.12) with boundary condition (2.13) can be numerically integrated using either implicit or explicit schemes, which will be explained later.

Knowledge of the particle temperature profile also allows us to determine, Γ , the rate of phase change from solid particle to gas. Once any point at a radial location $0 \leq r \leq R$ has a temperature exceeding the boiling temperature, the entire mass between r and R is transferred to the gas phase in one time step. In so doing, an energy equal to the product of the mass lost and the particle combustion of heat minus heat of evaporation energy is transferred from the particle to the gas.

The interphase drag forces (F_x, F_y) are determined from the experimental drag for a sphere, as presented by Schlichting (1983).

$$F_x = \left(\frac{\pi}{8} \right) N_p \rho_g C_D |V_g - V_p| (u_g - u_p) R^2 \quad (2.14)$$

where

$$C_D = \begin{cases} \frac{24}{Re} \left(1 + \frac{Re^{2/3}}{6} \right) & \text{for } Re < 1000; \\ 0.44 & \text{for } Re > 1000. \end{cases} \quad (2.15)$$

and $Re = \frac{2R|V - V_p|}{\mu_g}$, R is the radius of the particle and μ_g is gas viscosity at a temperature of $T_{film} = \frac{1}{2}(T_g + \bar{T}_p)$. Similarly, the formula for F_y is

$$F_y = \frac{\pi}{8} N_p \rho_g C_D |v_g - v_p| (v_g - v_p) R^2. \quad (2.16)$$

Equation of-State for Detonation Products

To close the system of governing equations, one needs a constitutive relation between density, pressure, temperature, and energy for gas phase, which is an equation-of-state. This study uses the Becker-Kistiakowsky-Wilson (BKW) equation-of-state (cf. Cowan and Fickett, 1956; Mader, 1979), which is.

$$p_g V_g / \bar{R} T_g = 1 + x e^{bx}. \quad (2.17)$$

where V_g = volume of gas phase
 p_g = pressure of gas phase
 T_g = temperature of gas phase
 \bar{R} = universal gas constant
 x = $k/F_g(T + \Theta)^a$
 k = $K \sum_i X_i k_i$

with empirical constants a, b, K, Θ and k_i . The constants k_i , one for each molecular species, are co-volumes. The co-volumes are multiplied by their mole fraction of species, X_i , and are added to find an effective volume for a mixture. For a particular explosive, if we know the composition of detonation products a, b, Θ, K , and all k_i 's can be found in the book by Mader (1979).

The internal energy is determined by thermodynamics relation

$$\left(\frac{\partial E_g}{\partial V_g} \right)_T = T_g \left(\frac{\partial p_g}{\partial T_g} \right)_V - p_g. \quad (2.18)$$

Integration of this equation for a fixed composition of the detonation products will allow us to calculate the energy of the detonation products as a function of temperature and volume. The thermodynamic properties as functions of temperature were calculated for each component from the NASA tables compiled by Gordon and McBride (1976).

The BKW equation-of-state is the most used and well calibrated of those equations-of-state used to calculate the properties of detonation products. The detailed discussion and review of the BKW equation-of-state can be found in the literature (cf. Cowan and Fickett, 1956; Mader, 1979).

Numerical Method of Solutions

The system of partial differential equations described in the previous paragraph is integrated numerically. Equations (2.1)–(2.9) can be written in the following vector form.

$$\frac{\partial \Phi}{\partial t} + \frac{\partial F}{\partial x} + \frac{\partial G}{\partial y} = \Omega. \quad (2.19)$$

In order to numerically solve this equation, an operator time-splitting technique is used. Assuming that all flow variables are known at a given time, we can calculate its advancement in time by splitting the integration into two stages.

In the first stage, the conservative part of Eq. (2.19) is solved:

$$\frac{\partial \Phi}{\partial t} + \frac{\partial F}{\partial x} + \frac{\partial G}{\partial y} = 0. \quad (2.20)$$

The Second Order Godunov method is used for the integration of the subsystem of equations describing the gaseous phase flow. The method is well documented in the literature (cf. Eidelman *et al.*, 1984; Colella, 1985; Colella and Glaz, 1985). In the following we will elaborate only some specifics of application of the method with BKW equation-of-state to simulate detonation product.

The physical system under study will have concentrations of solid explosive particle ranging from 1000 kg/m³ near the ground to 0.75 kg/m³ in the cloud. Detonation of this mixture will create detonation products with effective γ ranging from 3 to 1.1. To describe the flow of detonation products, we use the BKW equation-of-state described previously. Since the Second Order Godunov method uses primitive variables to calculate Riemann problems at the edges of the cells, its implementation for non-ideal EOS is difficult. In our simulations, we have resolved this problem by involving a local parameterization of EOS and by using direct and inverse equations-of-state. After integrating a system of gas conservation laws, we use the direct BKW equation-of-state to calculate pressure, gamma, and temperature as functions of thermal energy, density, and mixture composition. After this step, we have a complete set of parameters allowing calculation of the fluxes obtained from solving the Riemann problem (Colella and Glaz, 1985). The "inverse." EOS calculates internal energy as a function of density and pressure. In our code we use the "inverse" EOS to calculate the fluxes of conserved variables after calculation of the flux from Riemann problem of primitive variables.

The subsystem of equations describing the particle phase flow is integrated using a simple finite difference upwind scheme. This is done because there is no shock in the particle phase and the upwind scheme leads to a robust and accurate integration scheme.

In the second stage, the source term is added and the following equation is solved:

$$\frac{\partial \Phi}{\partial t} = \Omega. \quad (2.21)$$

To integrate this equation in time, we need to obtain Ω as a function of Φ . To do this, we first solve the particle heat conduction and heat transfer equation (2.12) with a boundary condition (2.13) that gives the temperature distribution as a function of particle radius and time using a local particle grid. Since the particle radius, R , will become very small due to evaporation, the implicit Crank-Nicolson algorithm is used because of its stability properties and its second order temporal and spatial accuracy. Using the Crank-Nicolson scheme to predict the particle temperature profiles at times t_1 and t_2 permits easy calculation of the total energy exchange, Q , between t_1 and t_2 , due to convection and conduction. Knowing the temperature distribution inside the particle, we can calculate gas generation rate Γ , drag force F_r , F_v , and heat exchange Q , between two phases and hence, Ω of Eq. (2.21). After obtaining the source term, we can integrate Eq. (2.21) by an explicit scheme.

For the multiphase system under study, $\Delta_r = \Delta_v = 1mm$ was used to allow explicit integration of the gasdynamic and physical processes of evaporation and heat release. When a mismatch occurred between the physical and gasdynamical characteristic times, the time step was adjusted by some fraction to assure stability. However, the resulting time step was not significantly smaller than that calculated by CFL criteria. For larger cell sizes, this approach will be impractical.

The numerical method is implemented in a code named MPHASE, which is fully vectorized and supported by number of graphics and diagnostics codes.

Table 1.
One Dimensional Validation Result

D[m/sec]—Detonation wave velocity,

P_{CJ} [Pa]—Pressure at Chapman-Jouguet Point

P_p [Pa]—Peak pressure; ρ_p [kg/m³]—Peak density

RDX Density (kg/m ³)	Parameters	Present Calculation	Expt'l Ref. 1	Tiger Calculation Ref. 2	BKW Calculation Ref. 1	Soviet Experiments Ref. 3
1000 kg/m ³	D	6155	5981		6128	
	P_{CJ}	1.220×10^{10}			1.08×10^{10}	1.00×10^{10}
	P_p	2.57×10^{10}				
	ρ_p	1936				
860 kg/m ³	D	6031		5900		
	P_{CJ}	0.986×10^{10}		0.88×10^{10}		0.82×10^{10}
	P_p	1.95×10^{10}				
	ρ_p	1722				
466 kg/m ³	D	4800		4500		
	P_{CJ}	0.379×10^{10}		0.30×10^{10}	0.3×10^{10}	
	P_p	0.625×10^{10}				
	ρ_p	924				
250 kg/m ³	D	4049		3600		
	P_{CJ}	0.2478×10^{10}		0.13×10^{10}		
	P_p	0.4538×10^{10}				
	ρ_p	552				
100 kg/m ³	D	3495				
	P_{CJ}	0.5013×10^9				
	P_p	0.7658×10^9				
	ρ_p	220				
0.75 kg/m ³	D	1622	1410*	1870*		
	P_{CJ}	0.25×10^7	0.284×10^7 *	0.26×10^7 *		
	P_p	0.484×10^7				
	ρ_p	8				

Ref. 1—Mader, C., "Numerical Modeling of Detonation," (University of California Press, Ltd., 1979), p. 47.

Ref. 2—Wiedermann, A., "An Evaluation of Bimodal Layer Loading Effects," IITRI Report, Feb., 1990.

Ref. 3—Stanukovitch, K.P., "Physics of Explosion" (in Russian), Nauka, 1975.

3. RESULTS

Model Validation for a One Dimensional Detonation Wave Problem

The main advantage of our particle combustion model is its description of the detonation phenomenology for a wide range of explosive particle sizes and densities. We will demonstrate this capability on a set of one dimensional test problems. For these test problems we have simulated the initiation and propagation of the detonation waves in a shock tube-like setting, where the explosive particles are distributed uniformly through the shock tube volume.

Results of these simulations are summarized in Table I, which shows detonation wave velocity, peak pressure, and peak density given as a function of the average density of the

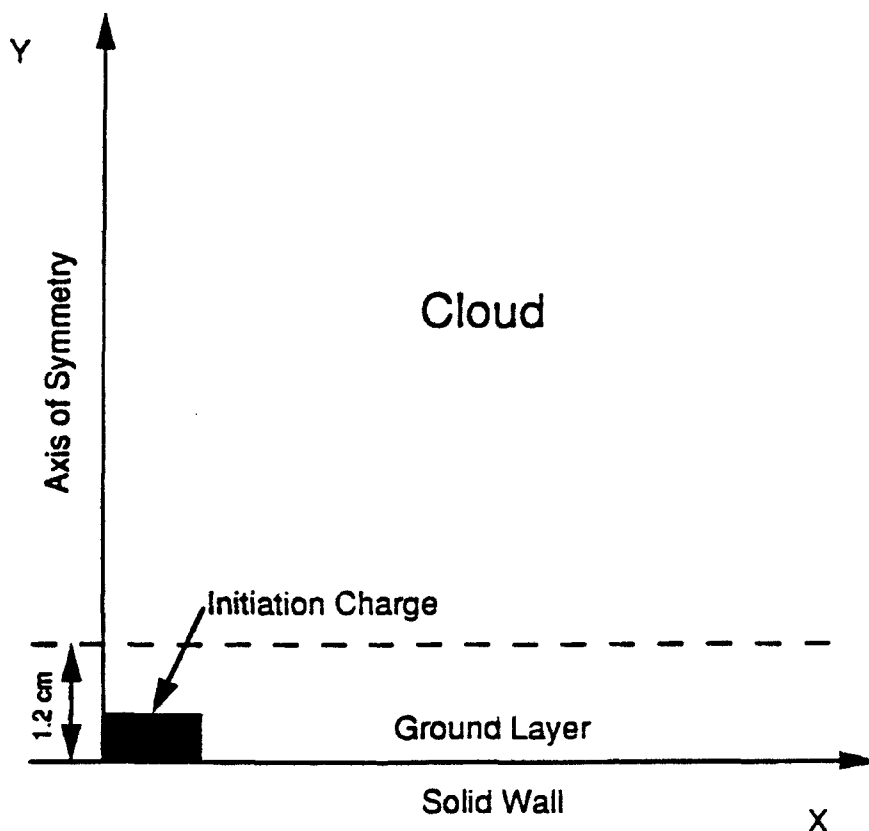


FIGURE 1 Computational domain and boundary conditions.

solid explosive. Here the explosive two-phase mixture is composed from RDX particle and air, where RDX particle concentration varies from 0.75 kg/m^3 to 1000 kg/m^3 . This concentration variation covers the whole range of solid explosive concentrations of interest to our problem. The simulations performed with the MPHASE code were compared with the experimental results (Mader, 1979; Stanukovitch, 1975), and calculations were done with the TIGER code presented by Wiedremann (1990).

From Table I, it is clear that our simulation results compare favorably with other simulation results and experimental data. The maximum deviation between our results and referenced results is no greater than 15% for the entire range of explosives densities. Considering that our results were obtained with a single model for particle combustion applied to the extreme range of densities, our model gives an excellent prediction of the detonation wave parameters.

Two Dimensional Simulation Results

Figure 1 shows a setup for a typical two dimensional simulation. Here the computational domain is $25\text{cm} \times 25\text{cm}$. The explosive powder density is distributed according to the 4th power law of vertical distance, starting from the ground where the density is 800 kg/m^3 , and rising to 1.2cm, where the density is 0.75 kg/m^3 . From this point to 25cm height, the density is constant and equal to 0.75 kg/m^3 . The density distribution in the

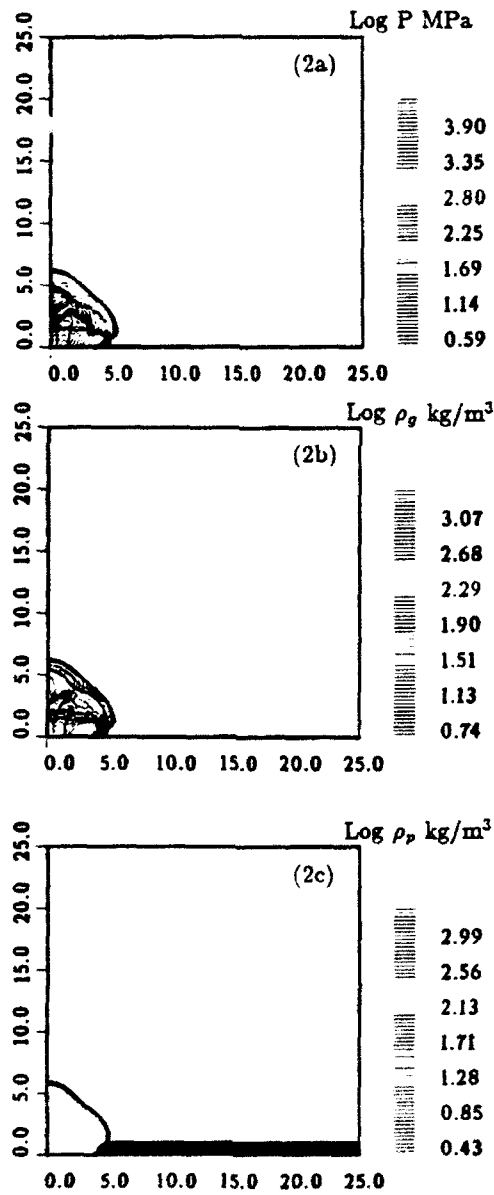


FIGURE 2 Fourth power distribution of particle density in the layer. The maximum density in the layer is 800 kg/m^3 . (2a), (2b), and (2c) are gas pressure, gas density, and particle density at $12 \mu\text{sec}$, respectively. See COLOR PLATE IV.

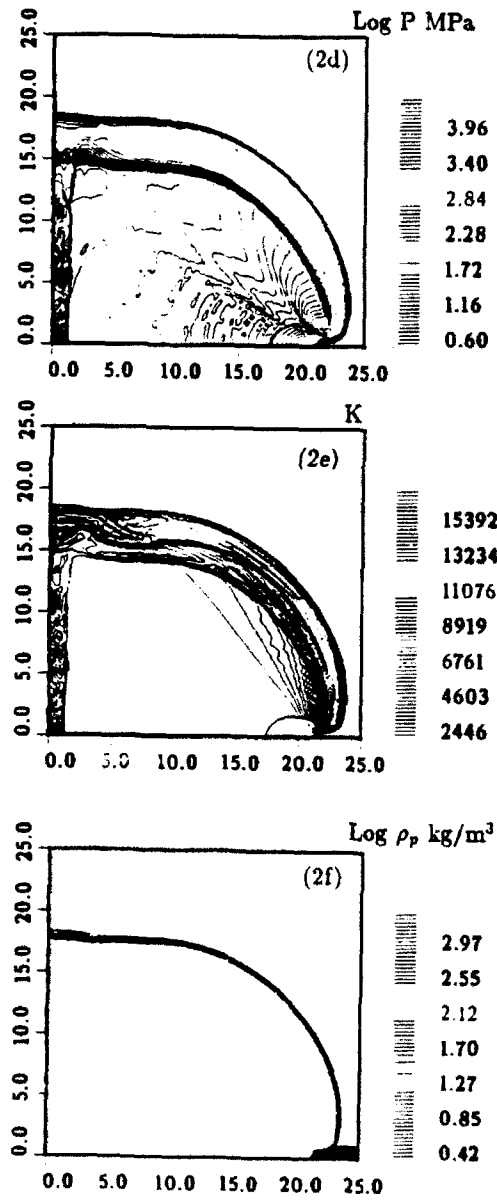


FIGURE 2 (Continued) (2d), (2e), and (2f) are gas pressure, temperature, and particle density at 55 μ sec, respectively. See also COLOR PLATE IV.

direction of the "x" axis is uniform. The boundary conditions for the computational domain shown in Fig. 1 are specified as follows: solid wall along the "x" axis; symmetry conditions along the "y" axis; supersonic outflow for upper boundary and at the right of the computational domain. The mixture consists of RDX powder and air at ambient conditions and it is assumed to be quiescent at the time of initiation.

The simulation starts at $t = 0$ when the mixture is initiated at the lower left corner of the computational domain by an initiating charge, as shown in Fig. 1. The initiating charge is $6 \text{ mm} \times 10 \text{ mm}$, with pressure of 4 GPa and density of 450 kg/m^3 . The energy released by the initiating explosion leads to formation of the detonation wave propagating through the multiphase media. Figure 2a shows pressure contours for the propagating detonation wave at the time of $t = 12 \text{ } \mu\text{sec}$ after initiation. Here the pressure contour levels are shown on logarithmic scale in MPa. The maximum pressure value of 7940 MPa is observed in the layer of condensed explosive located near the ground. The pressure in the layer is two to three orders of magnitude higher than pressure behind the detonation wave in the 0.75 kg/m^3 RDX cloud and air located above the distance of 1.2cm from the ground. Figure 2a demonstrates that the detonation wave in the cloud is overdriven, since the pressure behind the shock continuously rises and reaches its maximum in the layer. From this figure, we also observe that the overdriven wave propagates faster in the cloud than in the layer. This is explained by the fact that it is easier to compress air that is very lightly loaded with particles and located above the ground layer, than it is to compress air heavily loaded with a particle mixture near the ground. It is interesting to note a discontinuous pressure change between the yellow contours and the light blue and green contours behind the detonation front. This discontinuity is over-emphasized by our presentation of contour lines on the logarithmic scale; however, further examination of our simulation results indicates this feature is real and is similar in nature to barrel shocks observed for strong jets.

In Fig. 2b, gas phase density contours are shown for the time $t = 12 \text{ } \mu\text{sec}$. Here the contour lines are distributed on logarithmic scale. The main features of the shock wave structure are very similar to those observed in the pressure contours figure. Here we see that a jet of high density gases reflects from the center of symmetry axis, creating a contact discontinuity that we will observe at later times. The barrel shock is clearly visible in this figure. In Fig. 2c, the particle density contour plots are shown for $t = 12 \text{ } \mu\text{sec}$. The contour levels in Fig. 2c are given on the logarithmic scale and the initial deposition of the explosive material in the ground layer of the computational domain can be clearly observed. The black contour lines delineate the beginning and the end of the reaction zone in the cloud. To the left of these contours lies an area with combustion products and to the right unburned particles in the cloud. Here we can see that the reaction zone length is of the order of 1cm.

Figure 2d shows pressure contours for the same simulation for the time $t = 55 \text{ } \mu\text{sec}$ just before the detonation wave leaves the computational domain. In this figure we see that the global structure of the wave did change slightly from Fig. 2a. We observe that the barrel shock wave is fully developed and has a half ellipse shape. The detonation wave in the cloud is still overdriven; however, part of the shock wave front that propagates vertically becomes weaker as it gets further away from the detonation front in the layer. In Fig. 2e, gas temperature contours are shown at $t = 55 \text{ } \mu\text{sec}$. In this case, it is interesting to note that the highest temperatures are observed behind the front of the overdriven cloud detonation wave in immediate vicinity of the layer's upper strata. Very high temperatures in this region can be explained by the high pressure generated from the detonation of the explosive material in the layer and by relatively low density of cloud strata in the layer's immediate vicinity. Here, as in the pressure contours graph, the area of barrel shock can be clearly identified.

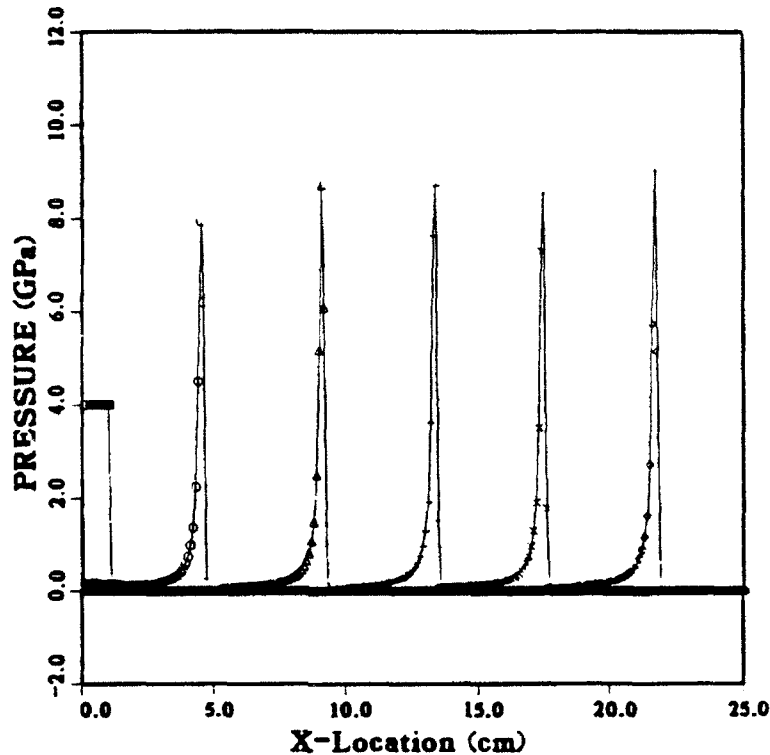


FIGURE 3 History of pressure distribution on the ground from initiation to steady detonation: \square - 0 μ sec, \circ - 12 μ sec, \triangle - 24 μ sec, $+$ - 34 μ sec, \times - 44 μ sec and \diamond - 55 μ sec.

We also observe in Fig. 2 a clear development of two detonation fronts, one moving vertically in the cloud and another moving horizontally in the layer. Because the energy density of the explosive particle in the layer is about three orders of magnitude larger than it is in the cloud, the vertical parts of the front represent an overdriven detonation wave in the cloud. Even though the vertical front has slowed down compared with the horizontal front, its speed and parameters far exceed those typical for detonation waves in a cloud. In fact, the self-sustained detonation regime in the cloud will develop at the distance of about three meters from the layer. The area of the front close to the detonation wave in the layer will remain hot and overdriven, since it is located very close to the detonation front in the layer. In Fig. 2f, particle density contours are shown on a logarithmic scale. We can clearly observe the reaction zone delineated by black contour lines. In this case, the reaction zone length in the cloud is about 1cm. Consistent with the gradual transition from overdriven to self-sustained detonation, the reaction zone length is larger for the vertical part of the detonation front. The detonation wave velocity observed in our simulation is approximately 4048 m/sec, which is significantly lower than the detonation wave velocity observed in RDX with a density of 860 kg/m^3 (see Table I), which is the highest density in the ground layer. This can be explained by a high gradient of particle density distribution in the layer, where the density drops rapidly from 800 kg/m^3 at the bottom of the layer to 0.75 kg/m^3 at the top strata of the layer at 12 mm above the ground.

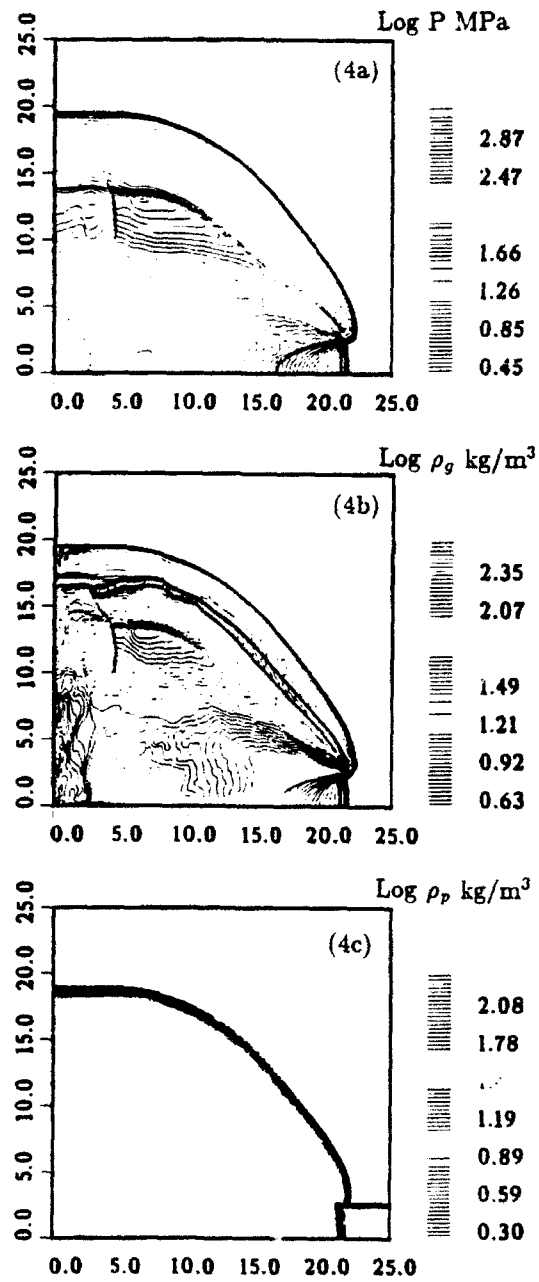


FIGURE 4 2.5 cm thick layer at constant density of 100 kg/m^3 . Density in the cloud is 0.75 kg/m^3 . (4a), (4b), and (4c) are gas pressure, gas density, and particle density at $66 \mu\text{sec}$, respectively. See COLOR PLATE V.

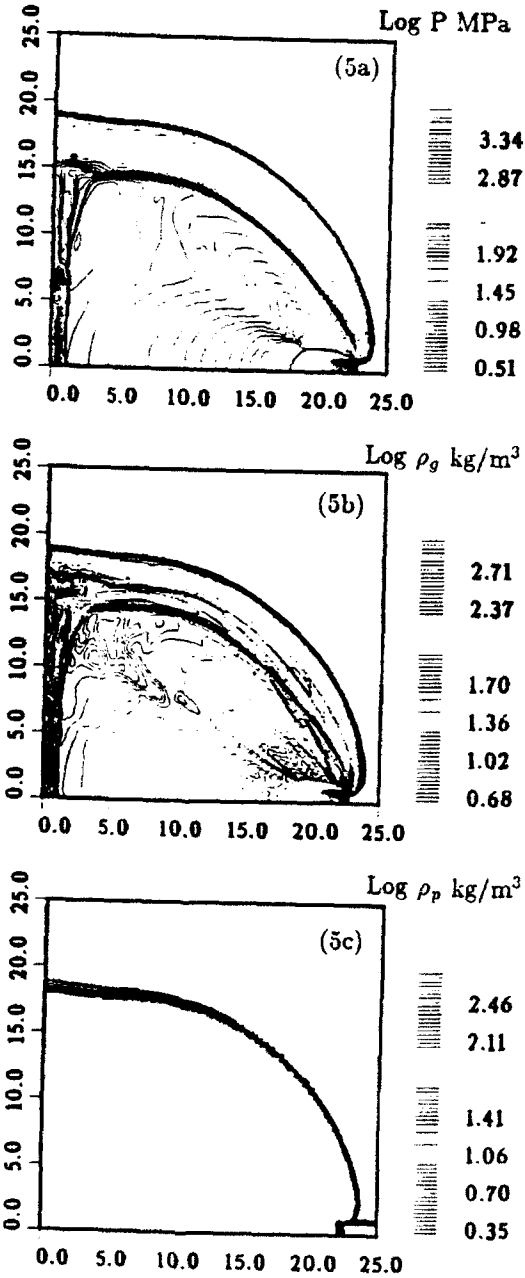


FIGURE 5 1.2 cm thick particle layer at constant density of 250 kg/m^3 . Particle density in the cloud is 0.75 kg/m^3 . (5a), (5b), (5c) are gas pressure, gas density, and particle density at $65 \mu\text{sec}$, respectively. See COLOR PLATE VI.

To show the transient process from initiation to steady-state detonation, we plot the pressure-distance profiles at six separate times after ignition (Fig. 3). Here the pressure is taken on the ground. Examining the profiles, we observe that the steady detonation is reached after 10cm. For each profile, we see that the pressure distribution is characterized by a strong detonation front followed by a fast expansion wave because of lateral expansion.

To further explore properties and phenomenology of the detonation waves propagating in the layer/cloud systems, we simulated additional cases in which explosive powder density distribution was different from the case reported above, although total weight of particle per unit area remained the same.

In Fig. 4, results are shown for the case of a uniform 2.5 cm thick layer of RDX with density of 100 kg/m^3 , and a 0.75 kg/m^3 cloud initiated under the same conditions as in the previous example. Figures 4a, 4b, and 4c show pressure, gas density, and particle density contour plots at $t = 66 \mu\text{sec}$. Here we observe that because the layer has much less density than the case reported above, the precursor effect of the detonation wave in the cloud preceding the wave in the layer is less pronounced. We also observe a significant difference in the shape of the strong contact discontinuity in the region of the shock front close to the layer. In Fig. 4b, we can clearly distinguish two contact surfaces, one between condensed explosive detonation products in the layer and in the cloud, and another between the detonation products from layer explosive detonation and from cloud particle detonation. We should note that these contact surfaces are over-emphasized by the logarithmic display of the contour plot levels. The maximum pressure observed in this simulation is 955 MPa, which is about one order of magnitude smaller than in the previous simulation. This is consistent with one order of magnitude difference in the maximum density of the ground layer in the two cases. The detonation wave speed (3407 m/sec) for the case presented in Fig. 4, which is only slightly lower than the speed predicted by the one dimensional simulations presented in Table I, reflects the influence of the two dimensional expansion on the detonation wave propagation.

Figure 5 presents results for the case of a uniform density of 250 kg/m^3 in a 1.2 cm ground layer. All other parameters are the same as in the previous two cases. In Figs. 5a, 5b, and 5c, pressure, gas density, and particle density contour plots are shown at the time $t = 65 \mu\text{sec}$ after detonation wave initiation. Here, the detonation wave propagates faster than in the previous cases $U = 3660 \text{ m/sec}$. This is about 400 m/sec slower than in the case of fourth power density distribution. Maximum pressure on the ground is 2150 MPa, which is consistent with the increase of powder density in the layer. The basic structure of the detonation front and the contact surfaces is similar to the case of fourth power density distribution.

4. CONCLUSIONS

We presented a mathematical model and numerical solution for the simulation of detonation wave initiation and propagation in multiphase mixtures consisting of solid combustible particles and gas. Using this model, we studied detonations in mixtures of solid RDX particles and air, with the objective of examining the effects of wide variation in particle density distribution on the dynamics and structure of detonation waves. We considered a physical system of solid particle clouds in air where a significant amount of particle can settle on the ground and the particle phase concentrations in the particle/air mixture can range from 0 to 1000 kg/m^3 . This range of solid phase densities necessitated development of the model and its numerical implementation for a wide range of particle concentrations. Our validation study has shown good agreement between the simulations and referenced results for the whole range of particle concentrations.

Two dimensional simulations were done for the system of low particle density concentration clouds and ground layers formed by high concentrations of the RDX powder. We examined three cases of ground layer density distribution: a fourth power distribution within 12 mm above ground with a maximum density on the ground of 800 kg/m^3 ; a uniform 25 mm thick layer with a density of 100 kg/m^3 ; a 12 mm thick uniform layer with a density of 250 kg/m^3 . In all these cases, the weight of condensed phase per unit area was the same, which allowed examination of the effects of the particle density distribution on detonation wave parameters.

In all examined two dimensional cases, the detonation wave in the cloud in the computational domain was significantly overdriven and did not play an important role. We estimated that the self-sustained regime of the detonation wave in the cloud for the examined cloud concentrations can occur only at the distances of 2–3 M above ground. At the same time, the particle density distribution in the layer determines the dynamics of the detonation wave as well as the pressure on the ground.

We observed in all three two dimensional simulations a very distinct shape of the detonation wave front in the vicinity of the layer. In this area, the overdriven detonation in the cloud is preceding the detonation wave in the ground layer. This feature of the detonation front can be explained by the fact that the energy released in the ground layer detonation wave produces a faster propagating shock wave in the dilute cloud than in the ground layer which is heavily loaded with solid particles. However, these structures were not observed experimentally, and more studies are needed to examine their parameters.

The maximum pressure affecting the ground was directly related to the maximum particle density in the lower strata of the layer. However, the detonation front velocity for the fourth power distribution case was considerably lower than calculated for a one dimensional case with 860 kg/m^3 particle density, reflecting the significant effect of two dimensional expansion. Two other cases with 250 kg/m^3 and 100 kg/m^3 maximum densities had detonation wave velocity only slightly lower than the one dimensional simulations of the same RDX/air concentrations. It is interesting to compare the simulation of the fourth power density distribution case and the 250 kg/m^3 case. In both, the same amount of explosive was distributed in the same physical space; however, the parameters of developed detonations were vastly different. Existence of the high density strata at the bottom of the ground layer in the fourth power case significantly increased the maximum pressure at the ground, and produced higher detonation wave velocity.

Using a variable density layer, we can reach a combination of pressure and velocity conditions outside of the Chapmen-Jougett limitations. The range of conditions that can be obtained in the variable density system and its parametrics needs a more systematic study. In this article, we introduced only the mathematical formulation and numerical simulation method validated for the range of conditions of interest. In addition, we have given some examples of the method's application for two dimensional simulations. However, this methodology should be linked to an experimental study for a more in-depth analysis of the phenomenology discussed here.

ACKNOWLEDGMENTS

The work reported here was partially supported by DARPA and AFOSR under contract no. F49620-89-C-0087. The authors would like to thank Col. J. Crowley and Dr. A. Nachman for their interest in this project.

REFERENCES

- Burcat, A., Eidelman, S., and Manheimer-Timnat, Y. (1978). "The Evolution of a Shock Wave Generated by a Point Explosion in a Combustible Medium," *Symp. of High Dynamic Pressures (H.D.P.)*, Paris, 347.
- Colella, P. (1985). "A Direct Eulerian MUSCL Scheme for Gas Dynamics," *SIAM J. Stat. Comput.* 6, 104.
- Colella, P. and Glaz, H.M. (1985). "Efficient Solution Algorithms for the Riemann Problem for Real Gases," *J. Comput. Physics*, 59, 264.

- Cowan, R.D., and Fickett, W. (1956). "Calculation of the Detonation Products of Solid Explosives with the Kistiakowsky-Wilson Equation of State." *Journal of Chemical Physics*, 24, 932.
- Drake, R.M., Jr. (1961). "Discussions on G.C. Vliet and G. Leppert: Forced Convection Heat Transfer from an Isothermal Sphere to Water." *Journal of Heat Transfer*, 83, 170.
- Eidelman, S., Timnat, Y.M., and Burcat, A. (1976). "The Problem of a Strong Point Explosion in a Combustible Medium." *6th Symp. on Detonation, Coronado, CA, Office of Naval Research*, 590.
- Eidelman, S., and Burcat, A. (1980). "The Evolution of a Detonation Wave in a Cloud of Fuel Droplets: Part I, Influence of the Igniting Explosion." *ALAA Journal*, 18, 1103.
- Eidelman, S., Coljela, P., and Shreeve, R.P. (1984). "Application of the Godunov Method and Its Second Order Extension to Cascade Flow Modelling." *ALAA Journal*, 22, 10.
- Eidelman, S., and Burcat, A. (1980). "The Mechanism of Detonation Wave Enhancement in a Two-Phase Combustible Medium." *18th Symposium on Combustion, The Combustion Institute, Waterloo, Ontario, Canada*.
- Eidelman, S., and Burcat, A. (1981). "Numerical Solution of a Non-Steady Blast Wave Propagation in Two-Phase (Separated Flow) Reactive Medium." *J. Comput. Physics*, 39, 456.
- Gordon, S., and McBride, B.J. (1976). "Computer Program for Calculations of Complex Chemical Equilibrium Compositions, Rocket Performance, Incident and Reflected Shocks and C-J Detonations." *NASA SP-273*, 1976 Revision.
- Kauffman, C.W., Wolanski, P., Vral, E., Nicholls, J.A. and Van Dyke, R. (1979). "Shock Wave Initiated Combustion of Grain Dust," *Proc. of the Intl. Symp. on Grain Dust*, p. 164. Manhattan, KS.
- Kuo, K. (1990). "Principles of Combustion." John Wiley and Sons, Inc.
- Liu, J.C., Kauffman, C.W., and Sichel, M. (1990). "The Lateral Interaction of Detonating and Detonable Mixtures." (Private communication).
- Mader C.L. (1979). "Numerical Modeling of Detonation." University of California Press, Ltd. London, England.
- Oved, Y., Eidelman, S., and Burcat, A. (1978). "The Propagation of Blasts from Solid Explosives to Two-Phase Medium." *Propellants and Explosives*, 3, 105.
- Reinecke, W.G., and Waldman, G.D. (1975). "Shock Layer Shattering of Cloud Drops in Reentry Flight." *ALAA Paper*, 75-152.
- Schlichting, H. (1983). "Boundary Layer Theory," 7th ed. McGraw-Hill.
- Stanukovitch, K.P. (1975). "Physics of Explosion" (in Russian), Nauka.
- Wiedermann, A. (1990). "An Evaluation of Bimodal Layer Loading Effects." *IITRI Report*, February.



AIAA 93-2940

**Computation of Shock Wave Reflection and
Diffraction Over a Semicircular Cylinder in
a Dusty Gas**

X. Yang, S. Eidelman, and I. Lottati

Science Applications International
Corporation

McLean, VA 22102

**AIAA 24th
Fluid Dynamics Conference
July 6-9, 1993 / Orlando, FL**

COMPUTATION OF SHOCK WAVE REFLECTION AND DIFFRACTION OVER A SEMICIRCULAR CYLINDER IN A DUSTY GAS

Xiaolong Yang,* Shmuel Eidelman,† and Itzhak Lottati*
Science Applications International Corporation

Abstract

The unsteady shock wave reflection and diffraction generated by a shock wave propagating over a semicircular cylinder in a dusty gas are studied numerically. The mathematical model is a multi-phase system based on a multi-fluid Eulerian approach. A Second Order Godunov scheme is used to solve the gas phase Euler equations and an upwind scheme is used to solve the particle phase conservation equations on an unstructured adaptive mesh. For the validation of the model, the numerically predicted one dimensional shock wave attenuation is compared with experimental results. Shock wave reflection and diffraction over a semicircular cylinder in a pure gas flow is simulated first to show the excellent agreement between the present computation and the experimental results. For a shock wave reflection and diffraction in a dusty gas, the effects of particle size and particle loading on the flow field are investigated. Gas and particle density contour plots are presented. It has been shown that the shock wave configuration differs remarkably from pure gas flow depending on the particle parameters. The difference is explained as the result of momentum and heat exchange between the two phases.

Introduction

Shock wave propagation into a gas particle suspension medium has attracted great attention in recent years due to its many engineering applications. Some of these applications include blast wave propagating over a dusty surface, exhaust from a solid propellant rocket, and coal or grain dust detonation. Many studies dealing with two phase environment can be found in literature. A general description and theoretical analysis of such flow can be found in review papers by Marble¹ and by Rudinger,² and in a book by Soo.³ Numerical models for dilute gas-particle flows were reviewed by Crown.⁴ Numerical studies of gas-particle flow in a solid rocket nozzle can be

*Drs. Yang and Lottati are Research Scientists with Science Applications International Corp. (SAIC), 1710 Goodridge Dr., MS 2-3-1, McLean, VA 22102. † Shmuel Eidelman, Research Scientist, SAIC, Associate Fellow AIAA

found in Refs. 5 and 6. Miura and Glass⁷ theoretically and numerically studied the oblique shock waves in a dusty-gas flow over a wedge. The one-dimensional unsteady structure of shock waves propagating through a gas-particle mixture was investigated both experimentally and numerically by Sommerfeld.⁸ Recently, Kim and Chang⁹ illustrated a numerical simulation of shock wave propagation into a dusty gas and the reflection of the wave from a wedge. Shock wave ignition of different reactive dust is experimentally investigated by Sichel *et al.*¹⁰ and comprehensive model for the structure of dust detonations is also described by Fan and Sichel.¹¹

In this paper, we study shock wave reflection and diffraction over a semicircular cylinder in a dusty gas. We numerically simulate the problem of a shock wave initiated in a pure gas section moving into a dusty region and impinging on a semicircular cylinder. We first formulate the compressible two-phase flow on the basis of a Eulerian multi-fluid formulation. We consider the two phases (i.e., gas and particle) to be interpenetrating continua. The dynamics of the flow are governed by conservation equations of each phase and the two phases are coupled by interactive drag force and heat transfer. We solve the system of conservation equations numerically on an unstructured adaptive grid. The objectives of the study are: (a) to solve the two-phase compressible flow field and compare the simulation with available experimental results; (b) to observe and investigate the reflection and diffraction wave patterns when a shock wave propagates over a semicircular cylinder in a dusty gas, with particle radius and loading as parameters.

The outline of this paper is as follows. Section 2 gives a description of the mathematical model and method of numerical solution, including governing conservation equations for two phases, the constitutive laws, the initial and boundary conditions, and particle parameter. A brief outline of numerical schemes and the adaptive unstructured grid is also given. In Section 3, we present our numerical simulation results. We validate our model by comparing a one-dimensional simulation of a shock wave propagating into a dusty gas with available experimental results. We also show the excellent agreement between our two-dimensional gas-only simulation with existing experimental results. Results for reflection and diffraction of shock wave over a semicircular cylinder are given for different particle parameters. Concluding remarks are given in Section 4.

Mathematical Model and the Numerical Solution

Conservation Equations

The mathematical model consists of conservation governing equations and constitutive laws that provide closure for the model. The basic formulation adopted here follows the gas and dilute particle flow dynamics model presented by Soo.³ The following assumptions are used during the derivation of governing equations:

- (1) The gas is air and is assumed to be ideal gas;
- (2) The particles do not undergo a phase change because for particles considered here (sand) phase transition temperature is much higher than the temperatures typical for the simulated cases;
- (3) The particles are solid spheres of uniform diameter and have a constant material density;
- (4) The volume occupied by the particles is negligible;
- (5) The interaction between particles can be ignored;
- (6) The only force acting on the particles is drag force and the only heat transfer between the two phases is convection. The weight of the solid particles and their buoyancy force are negligibly small compared to the drag force;
- (7) The particles have a constant specific heat and are assumed to have a uniform temperature distribution inside each particle.

Under the above assumptions, distinct equations of continuity, momentum, and energy are written for each phase. The interaction effects between the two phases are listed as the source terms on the righthand side of the governing equation. The two-dimensional unsteady conservation equations for the two phases can be written in the vector form in Cartesian coordinates:

$$\frac{\partial U}{\partial t} + \frac{\partial F}{\partial x} + \frac{\partial G}{\partial y} = S. \quad (1)$$

Here U is the vector of conservative variables, F and G are fluxes in x and y direction, respectively, and S is the source term for momentum and heat exchange. The definition of these vectors are:

$$U = \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ e \\ \rho_p \\ \rho_p u_p \\ \rho_p v_p \\ e_p \end{pmatrix}, \quad F = \begin{pmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ u(e + p) \\ \rho_p u_p \\ \rho_p u_p^2 \\ \rho_p u_p v_p \\ u e_p \end{pmatrix}, \quad G = \begin{pmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ v(e + p) \\ \rho_p v_p \\ \rho_p u_p v_p \\ \rho_p v_p^2 \\ v e_p \end{pmatrix},$$

$$S = \begin{pmatrix} 0 \\ -f_x \\ -f_y \\ -q - u_p f_x - v_p f_y \\ 0 \\ f_x \\ f_y \\ q + u_p f_x + v_p f_y \end{pmatrix}$$

where ρ , u , v , and e are gas density, velocities, and energy, respectively; ρ_p , u_p , v_p and e_p are particle density, velocities, and energy, respectively; (f_x, f_y) and q denotes drag force components acting on the particles and heat transfer to the particles, respectively. The gas pressure p is related to ρ , u , v and e for by

$$p = (\gamma - 1)[e - 0.5\rho(u^2 + v^2)] \quad (2)$$

where γ is the specific heat ratio. The gas temperature can be found through the equation-of-state for ideal gas

$$p = \rho RT \quad (3)$$

where R is the gas constant.

The particle temperature T_p is calculated through relation

$$e_p = \rho_p c_p T_p + 0.5\rho_p(u_p^2 + v_p^2). \quad (4)$$

The source terms on the righthand side of Eq. (1) are momentum and heat exchange between gas and particle phases. If we let τ_p and ρ_s be the particle radius and material density, respectively, then the drag forces are

$$\begin{pmatrix} f_x \\ f_y \end{pmatrix} = \frac{3}{8} \frac{\rho_p \rho}{\rho_s r_p} C_d [(u - u_p)^2 + (v - v_p)^2]^{1/2} \begin{pmatrix} (u - u_p) \\ (v - v_p) \end{pmatrix}. \quad (5)$$

The particle drag coefficient C_d depends on relative Reynolds number, Re and relative Mach number, M_r . In the present study, since the relative Mach number is small ($M_r < 0.5$), the effect of M_r on C_d is neglected. The Reynolds number, Re , is based on the relative velocity between the gas and particle phases. After testing the drag coefficients given by Sommerfeld⁸ and by Clift *et al.*,¹² the following two were adopted:

$$C_d = \frac{24}{Re} (1 + 0.15 Re^{0.687}) \text{ for } Re < 800.$$

and

$$C_d = \frac{24}{Re} (1 + 0.15 Re^{0.687}) + \frac{0.42}{1 + 42500 Re^{-1.16}} \text{ for } Re > 800. \quad (6)$$

Here the Reynolds number Re is defined as

$$Re = \frac{2\rho r_p [(u - u_p)^2 + (v - v_p)^2]^{1/2}}{\mu} \quad (7)$$

Viscosity, μ , is calculated at film temperature, namely, $T_f = 0.5(T_p + T)$, and the temperature dependency of the viscosity is evaluated according to Sutherland's law

$$\mu = \mu_r \left(\frac{T}{T_r} \right)^{3/2} \frac{T_r + \Phi}{T + \Phi} \quad (8)$$

where μ_r is the dynamic viscosity of the gaseous phase at the reference temperature and Φ is an effective temperature, called the Sutherland constant.

The rate of heat transfer from gaseous phase to the particle phase is given by

$$Q = \frac{3}{2} \frac{\rho_p}{\rho_s} \frac{\mu C_p}{Pr} Nu (T - T_p) \quad (9)$$

where $Pr = \mu c_p / k_g$ is the Prandtl number, and c_p and k_g are the specific heat and thermal conductivity of gas, respectively. The Nusselt number Nu is a function of Reynolds number and the Prandtl number as given by Drake¹³

$$Nu = \frac{2r_p h}{R} = 2 + 0.459 Re^{0.55} Pr^{0.33} \quad (10)$$

Initial and Boundary Conditions

The geometry of the computational domain is shown in Fig. 1. The initial conditions for gas are $\rho_o = 1.2 \text{ kg/m}^3$ and $p_o = 101.3 \text{ kpa}$, with a coming shock at $x = -0.5$. There are no particles from $-1.0 \leq x \leq 0.0$. From $x \geq 0.0$, particles are initially in thermal and kinematic equilibrium with surrounding gas. The particles that are uniformly distributed in the dusty region have the following parameters for different test problems:

- Mass loading, ρ_p : 0.25 kg/m^3 , 0.76 kg/m^3 ;
- Mass material density, ρ_s : 2500 kg/m^3 ;
- Particle radii, r_p : $10 \mu\text{m}$, $25 \mu\text{m}$, $50 \mu\text{m}$;
- Specific heat, c_s : 766 J/kg/K .

The lower boundary and cylinder surface are solid walls and assumed adiabatic and impermeable. A reflecting boundary condition is assumed for both the gas and particle phase. Particles are assumed to experience a perfect elastic collision with the wall and reflect from the wall. The right and upper boundaries are open boundaries where a nonreflection boundary condition is used for the gas phase and a zero normal gradient condition is used for particle phase.

Numerical Method of Solutions

The system of partial differential equations described in the previous paragraph is integrated numerically. Equation (1) is repeated here:

$$\frac{\partial U}{\partial t} + \frac{\partial F}{\partial x} + \frac{\partial G}{\partial y} = S. \quad (1)$$

In order to solve this equation numerically, an operator time-splitting technique is used. Assuming that all flow variables are known at a given time, we can calculate its advancement in time by splitting the integration into two stages.

In the first stage, the conservative part of Eq. (1) is solved:

$$\frac{\partial U}{\partial t} + \frac{\partial F}{\partial x} + \frac{\partial G}{\partial y} = 0. \quad (11)$$

The Second Order Godunov method is used for the integration of the subsystem of equations describing the flow of the gaseous phase (first four components of Eq. (1)). The method is well documented in literature.^{14,15,16} The subsystem of equations describing the particle phase flow is integrated using a simple first order finite difference upwind scheme¹⁷. This is done because there is no shock in the particle phase and the upwind scheme leads to a robust and accurate integration scheme.

In the second stage, the source term is added and the following equation is solved:

$$\frac{\partial U}{\partial t} = S. \quad (12)$$

To integrate this equation in time, we need to obtain S as a function of U . We calculate S through Eqs. (5) to (10).

In order to produce a solution of the high spatial accuracy at minimal computational cost, an unstructured triangular grid with adaptive procedure is used. The adaptive procedure will automatically enrich the mesh by adding points in the high gradient (or high flow activity) region of the flow field and by removing points (coarsening mesh) where they are not needed. The dynamic nature of mesh enrichment is shown in Fig. 4 for three different time frames. One can see that a very fine mesh is generated around shock fronts and other steep density gradient regions.

Results

Model Validation for One-Dimensional Shock Wave Propagation in Dusty Gas

To test the momentum and heat exchange mechanism for the current two-phase model, we first simulate a one-dimensional problem of a normal shock wave propagating into a dusty gas. We numerically simulate the experiments conducted by Sommerfeld.⁸ In the experiments, small glass spherical particles of material density $\rho_s = 2500 \text{ kg/m}^3$, specific heat capacity $c_s = 766 \text{ J/kg/K}$, and average diameter of $27 \mu\text{m}$

were used as the suspension particle phase. The incoming shock Mach number M , and particle loading ratio $\eta = \rho_p/\rho$, are two varying parameters. The experimental results and our numerical simulation results of shock Mach number as a function of distance for two test cases are shown in Fig. 2a ($\eta = 0.63, M_s = 1.49$) and Fig. 2b ($\eta = 1.4, M_s = 1.7$) for comparison purposes. It is clear that the agreement between the prediction of shock wave attenuations from our present model and the experimental results is very good.

Two-Dimensional Simulation Results for Pure Gas Flow

To test the accuracy of the two-dimensional computation, we first compute the pure gas flow case of a shock wave reflection and diffraction over a semicircular cylinder. We then compare the simulation with experimental results. Shock wave reflection on a wedge has been extensively studied by many researchers (see e.g., review papers of Ben-Dor and Dewey¹⁸ and Hornung¹⁹). Shock wave reflection over circular cylinders was numerically simulated by Yang *et al.*²⁰. Recently, Glass *et al.*²¹ using high order Godunov scheme numerically simulated the shock wave reflection over a half diamond and semicircular cylinder and compared the simulation with experimental results obtained by Kaca.²² Figure 3 is a schematic sketch to show four stages of a shock wave reflection over a semicircular cylinder and terminologies which will be used to describe the flow fields. Figures 4a, 4b and 4c show the calculated density contours at three moments in time. When the planar shock wave propagates and encounters the cylinder, it first experiences a head-on collision with the front stagnation point of the semicylinder and then immediately reflects from the first quarter of the cylinder, forming a regular reflection (RR), which is shown in Fig. 4a. The regular reflection consists of two shocks, i.e., the incident shock and reflected shock, both originating from a common point on the cylinder wall. As the shock wave propagates up the cylinder, the angle between the incident shock and the tangent of the cylinder becomes larger and the regular reflection changes into a Mach Reflection (MR) as shown in Fig. 4b. The MR is characterized by three waves, incident shock (I), reflected shock (R), and Mach stem (M). All three shocks intersect at one common point called triple point (T). For Mach reflection, one can further observe both Simple Mach Reflection (SMR) and Complex Mach Reflection (CMR). Later, as the incident shock wave passes over the top of the semicircular cylinder, it experiences a rarefaction on the back side of the cylinder. The shock wave system grows upward and rightward with a curved Mach stem and forms a slipline(S) or a contact discontinuity (CD) as shown in Fig. 4c. In Figs. 5a and 5b, the interferogram from

the experiment²² and density contours from the present simulation are compared for the same flow condition and same time. Note that the density levels are normalized by the ambient gas density in Fig. 5. As one can see from Fig. 5, the results show an excellent quantitative as well as qualitative agreement between the numerical simulation and experimental results.

Two-Dimensional Simulation Results of Two-Phase Flow

The basic setup for the two-phase simulation is shown in Fig. 1. Here the planar shock with $Ms = 2.8$ propagates into an area of a dusty gas and impinges on a semicircular cylinder. The interface between pure air and dusty air is located at $x = 0.0$ of the computational domain. The area of the dusty air contains a semicylinder with a radius of 1m. The size of the computational domain, initial parameters of the gas, parameters of the incoming shock, size of the semicylinder and its location in the computational domain, are the same as in the reflection and diffraction simulation presented in the previous section.

The main objective of this set of simulations is to study the effects of particle size and particle loading on the parameters of the reflected and diffracted shock waves. It is also valuable to study the dynamics of particle media, since it is extremely difficult to observe these interactions experimentally in an optically thick dusty gas.

The first set of simulation results is shown for the case with dust parameters $r_p = 10\mu m$ and $\rho_p = 0.25 kg/m^3$. The gas parameters and the parameters of the incoming shock wave are the same as in the pure gas case presented above. In Figs. 6a and 6b, particle density contours and gas density contours are shown at the stage when the incident shock wave has reached the top of the semicylinder. At this stage, the largest difference of velocity and temperature between the two phases exists and the nonequilibrium between the two phases causes extensive heat and momentum exchange between particles and the gas. The presence of the particles causes a widening of the shock that is more noticeable for the incident shock. Also, an additional contour line is observed at the dusty gas/pure gas interface. Comparing gas density for pure gas flow field shown in Fig. 4b and the dusty gas density of Fig. 6b, we see that Mach stem and contact discontinuity resulting from Mach reflection are smeared in the dusty gas flow due to the presence of the particle. The particle density contours depict significant piling up of the dust particles at the leading edge stagnation point of the cylinder.

In Figs. 6c and 6d, the particle density and gas density contours are shown at the stage where significant diffraction has taken place and the shock front is

approaching the trailing edge of the cylinder. Further widening of the shock and some smearing of the slip line that originates at the triple point is evident. The particle density contours reveal that the particles were swept by the gas flow to the area of triple point and slip line for the gas flow, leaving a small amount of particles at the leading edge. We should note that this behavior is specific for our problem, where at $t = 0$, the dusty gas area was located at $x = 0$ and there is no influx of the dust from the left boundary. Also in Fig. 6c, we note that the particles reach a distinct local maxima at the distance about 25 cm behind the incident shock front. At this maxima the particle density is 0.86 kg/m^3 , which is more than three times the initial particle density. The particle density reaches a maximum value at the location of the gas slip line. We observe a significant accumulation of the particles that have been moved along the slip line by the shear flow. The larger concentration of particles in the vicinity of triple point is, in fact, the remainder of the particles that were swept up with the flow. It is also interesting to observe that an essentially particle-free zone is formed due to the effects of particles slipping over the top of the cylinder and the rarefaction wave behind the cylinder.

To study the influence of particle loading on the dynamics of reflection and diffraction, we have simulated the case with a dust density of $\rho_p = 0.76$, and with $r_p = 10 \mu\text{m}$. The results for this simulation are shown in Figs. 7a and 7b in the form of particle and gas density contour plots. In Fig. 7a, the particle density contours are shown at the diffraction phase. Here we can observe two local maxima for particles accumulated in the regions along the slip line characteristic for the shock diffraction process. It should be noted here that in our problem the conditions behind the incident shock wave and its structure are in constant flux. At higher loading, dust will have a profound effect on the gas dynamics of reflection and diffraction. Figure 7b shows gas density contours for the reflection stage corresponding to the particle density contours shown in Fig. 7a. We observe from Fig. 7b that the incident shock wave is significantly smeared and the triple point cannot be clearly identified. Because of the widening of the incident shock, the area where the reflected and incident shock join is spread over 50 cm distance. From Fig. 7a, we see that the high density particle region is spread wider than in the previous case, and the particle density reaches its maximum at about 25 cm behind the front. There is a visible maximum in gas density in the area where the reflected shock is interacting with the area of maximum particle density behind the incident shock. A part of the reflected shock front that is moving to the left side of the computational domain is not affected by the dust since it is propagating into an area with little dust concentra-

tion. The parameters and structure of this part of the front remain basically the same as in the case of pure gas flow.

To examine the effect of particle size on the reflection-diffraction process, we simulated a case where the particle loading and gas flow conditions are the same as in the previous case with particle density $\rho_p = 0.76$. However, the particle size is $r_p = 50 \mu\text{m}$. In Figs. 8a and 8b, results for this simulation are illustrated by particle density and gas density contours correspondingly. The particle contour plots depict a significantly wider particle relaxation zone than in the previous case. The longer relaxation zone is caused by the larger inertia of larger particles. The maximum particle density of 2.64 kg/m^3 is reached 50 cm behind the incident shock front. This value is significantly lower than 4.01 kg/m^3 reached behind the shock in calculation with $10 \mu\text{m}$ particles. Larger particles skip above the apex of the cylinder creating a void where particle density is very small. Also, because of larger particle size, the maxima of particle concentration that has been created by a slip surface of the reflected Mach stem is indistinct. The main reason for this is that the particles do not follow the gas flow as closely as they did in the previous case due to the inertia of large particles. The maximum particle density is reached here at the slip line behind the Mach stem.

Comparing gas density of Fig. 8b to the previous case shown in Fig. 7b, we observe that the slip line behind the curved Mach stem becomes less distinguishable in Fig. 7b. This result is expected, since at fixed particle loading, smaller particles have a larger surface/volume ratio and the larger surface/volume ratio increases momentum and heat exchange between the two phases.

One general comment regarding all three cases presented above: Due to the heat and momentum exchange between the two phases, the shock is decaying as it traverses the cylinder. Ultimately, it will reach a new equilibrium state as suggested by Fig. 2. It should be noted that the shock considered in the previous three cases is still in the process of transition in the gas-particle mixture.

Conclusion

In this paper, numerical study for a two-phase compressible flow is performed for the reflection and diffraction of a shock wave propagating over a semicircular cylinder in a dusty gas. The following conclusions can be made:

(1) The validation study for a one-dimensional shock wave propagating in a dusty gas shows a good agreement between the prediction of our model and the results of the experiment;

(2) For a two-dimensional gas-only flow, numerical results agree well with existing experimental data quali-

tatively and quantitatively, indicating that the gas phase is accurately simulated by the adaptive grid technique;

(3) Particles in the gas can have a profound effect on the shock wave reflection and diffraction pattern, which is a function of particle size and loading. The lesser the particle loading, the less the influence of particle on the flow field;

(4) In the three simulation cases, there is a particle accumulation behind the "back shoulder" of the semi-circular cylinder due to the effect of particle inertia and gas rarefaction wave;

(5) For different particle size at fixed particle loading, the larger particle will have a longer relaxation zone and less accumulation at "back shoulder" and behind incident shock. The gas density contours show a less distinguishable slip line in small particle case than in the large particle case.

Acknowledgments

The work reported here was partially supported by DARPA and AFOSR under contract no. F49620-89-C-0087. The authors would like to thank LtCol. J. Crowley and Dr. A. Nachman for their interest in this project.

References

¹Marble, F., "Dynamics of Dusty Gases," *Annual Review of Fluid Mechanics*, Vol. 2, 1970, pp. 369-377.

²Rudinger, G., "Some Properties of Shock Relaxation in Gas Flows Carrying Small Particles," *Physics of Fluids*, Vol. 7, 1964, pp. 658-663.

³Soo, S. L., *Particulates and Continuum*, Hemisphere Publishing Corporation, New York, 1989, pp.266-324.

⁴Crowe, C. T., "Review-Numerical Models for Dilute Gas Particle Flow," *Journal of Fluids Engineering*, Vol. 104, Sept. 1982, pp. 297-303.

⁵Hwang, C. J. and Chang, G. C., "Numerical Study of Gas-Particle Flow in a Solid Rocket Nozzle," *AIAA Journal*, Vol. 26, No. 6, 1988, pp. 682-689.

⁶Chang, I-Shih, "Three-Dimensional, Two-Phase, Transonic, Canted Nozzle Flows," *AIAA Journal*, Vol. 28, No. 5, 1989, pp. 790-797.

⁷Miura, H. and Glass I. I., "Oblique Shock Wave in a Dusty-Gas Flow Over a Wedge," *Proceedings of Royal Society of London A*, Vol 408, 1986, pp. 61-68.

⁸Sommerfeld, M., "The Unsteadiness of Shock Waves Propagating through Gas-Particle Mixtures," *Experiments in Fluids*, Vol. 3, No. 2, 1985, pp. 197-206.

⁹Kim, S-W. and Chang, K-S., "Reflection of Shock Wave from a Compression Corner in a Particle-laden Gas Region," *Shock Waves*, Vol. 1, No. 1, 1991, pp. 65-73.

¹⁰Sichel, M., Baek, S. W., Kaufman, C. W., Maker, B. and Nicholls, "The Shock wave Ignition of Dusts," *AIAA Journal*, Vol. 23, No. 9, 1985, pp. 1374-1380.

¹¹Fan, B. and Sichel, M., "A Comprehensive Model for the Structure and Dust Detonations," *Twenty-Second Symposium (International) on Combustion*, The Combustion Institute, PA, 1988, pp. 1741-1750.

¹²Clift, R., Grace, J. R. and Weber, M. E., *Bubbles, Drops and Particles*, Academic, New York, 1978.

¹³Drake, R.M., Jr., "Discussions on G.C. Vliet and G. Leppert: Forced Convection Heat Transfer from an Isothermal Sphere to Water," *Journal of Heat Transfer*, Vol. 83, No. 2, 1961, pp. 170-179.

¹⁴Eidelman, S., Colella, P. and Shreeve, R.P., "Application of the Godunov Method and Its Second Order Extension to Cascade Flow Modelling," *AIAA Journal*, Vol. 22, No. 11, 1984, pp. 1609-1615.

¹⁵Colella, P., "A Direct Eulerian MUSCL Scheme for Gas Dynamics," *SIAM Journal Scientific and Statistical Computation*, Vol. 6, 1985, pp. 104-117.

¹⁶Colella, P. and Glaz, H.M., "Efficient Solution Algorithms for the Riemann Problem for Real Gases," *Journal of Computational Physics*, Vol. 59, No. 3, 1985, p. 264-289.

¹⁷Peyret, R. and Taylor, T. D., *Computational Methods for Fluid Flow* Springer-Verlag, New-York, 1983, pp. 18-31.

¹⁸Ben-Dor, G. and Dewey J. M., "The Mach Reflection Phenomenon: A Suggestion for an International Nomenclature," *AIAA Journal*, Vol. 23, N. 10, 1985, pp. 1650-1652.

¹⁹Hornung, H., "Regular and Mach Reflection of Shock Waves," *Annual Review of Fluid Mechanics*, Vol. 18, 1986, pp. 33-58.

²⁰Yang, J. Y., Liu, Y. and Lomax H., "Computation of Shock Wave Reflection by Circular Cylinder," *AIAA Journal*, Vol. 25, 1987, No. 5, pp. 683-689.

²¹Glass, I. I., Kaca, J. Zhang, D. L. Glas, H. M., Bell, J. B. and Tangenstein, J., *Current Topics in Shock Waves, 17th Int'l Symp. on Shock Tubes and Waves*, edited by Y. W. Kim, AIP Conference Proceedings 208, American Institute of Physics, New York, 1990, pp. 246-251.

²²Kaca, J., "An Interferometric Investigation of Diffraction of a Planar Shock Wave over a Semicircular Cylinder," *UTIAS Technical Note 269*, Institute for Aerospace Studies, University of Toronto, 1988.

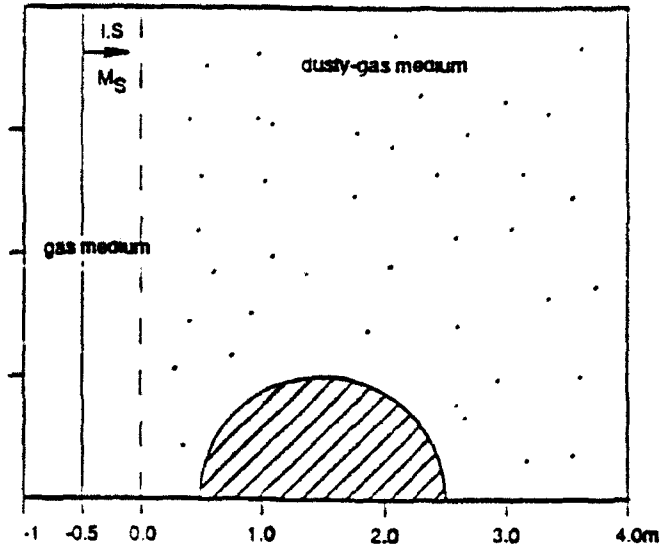
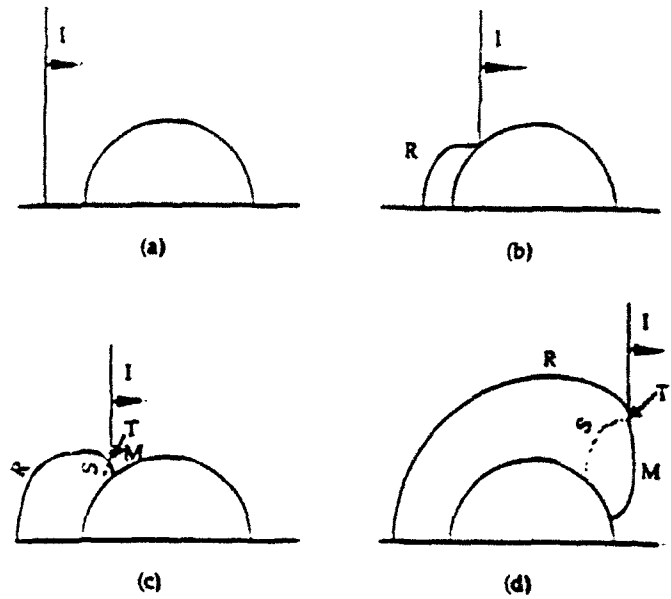


Figure 1. An illustration of the considered flow field.



I - Incident Shock R - Reflected Shock
M - Mach Stem S - Slipline
T - Triple Point

Figure 3. Stages of shock wave reflection over a semicircular cylinder, (a) before collision, (b) regular reflection, (c) Mach reflection, (d) well developed Mach reflection.

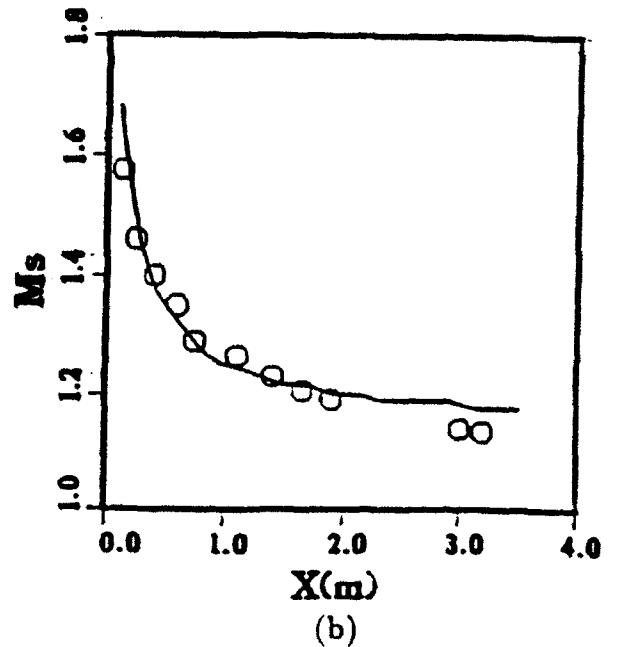
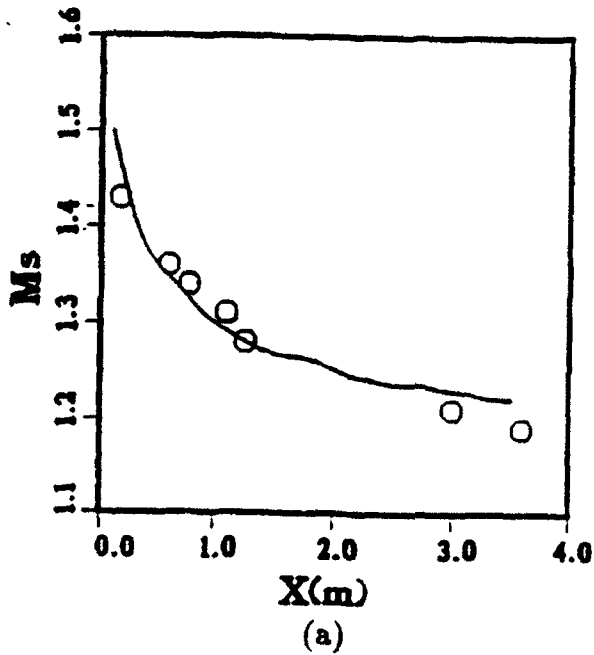
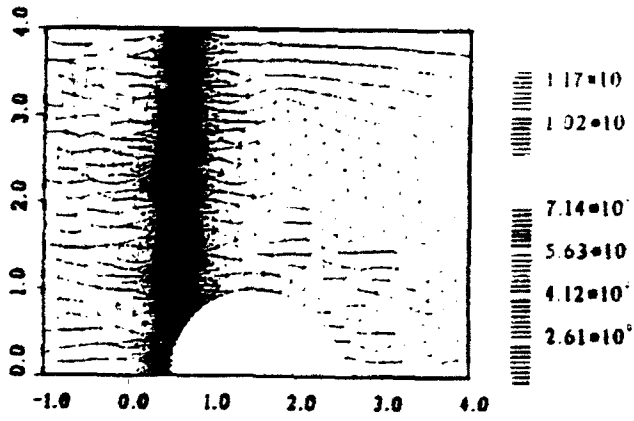
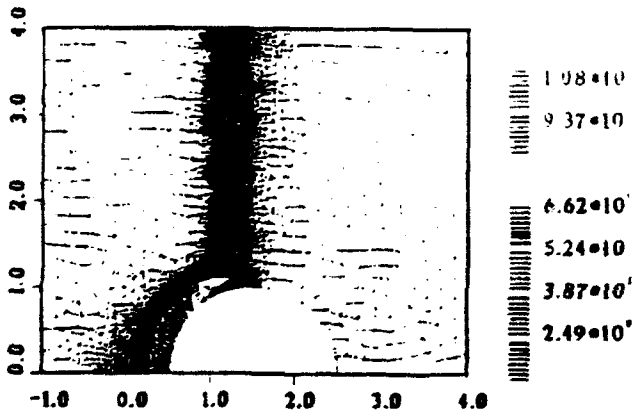


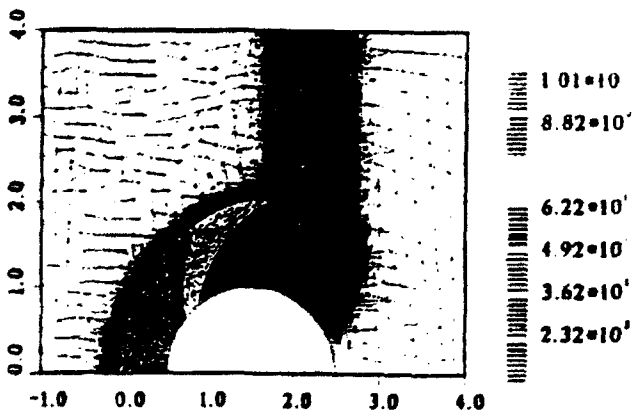
Figure 2. Comparison between computational prediction and experimental measurement of shock wave attenuation for (a) $M_1 = 1.49$, $\eta = \frac{\rho_2}{\rho_0} = 0.63$ and (b) $M_1 = 1.7$, $\eta = \frac{\rho_2}{\rho_0} = 1.4$ (o experiment, - calculation).



(a)

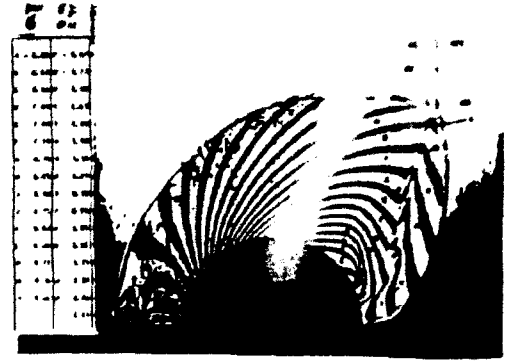


(b)

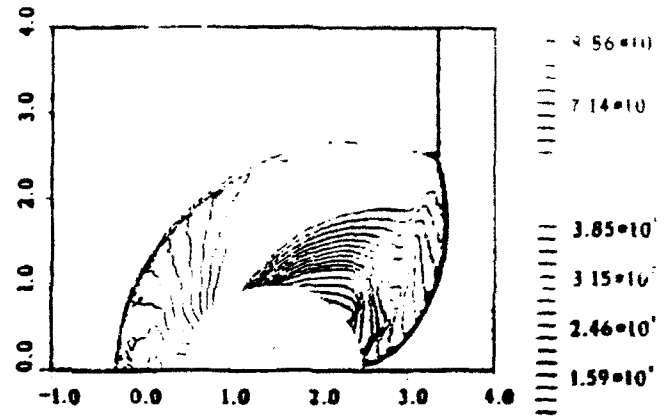


(c)

Figure 4. Computed density contours with adapted grid at three different times: (a) regular reflection (RR), (b) Mach reflection (MR) and (c) diffraction with slip line (S).

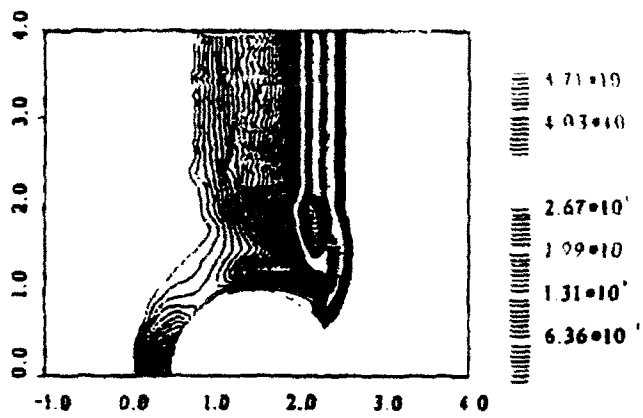


(a)

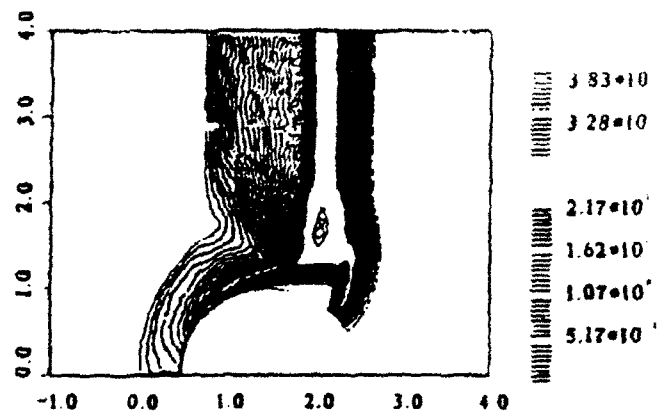


(b)

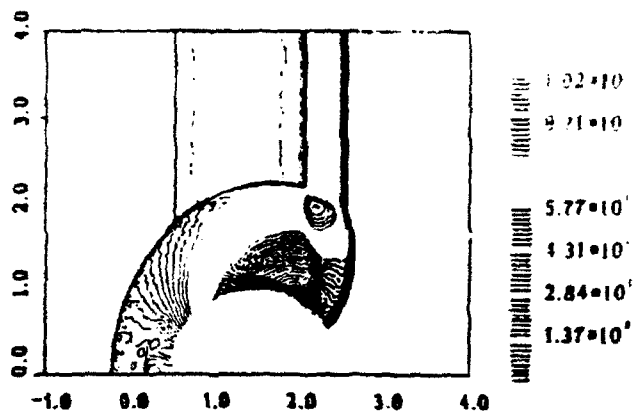
Figure 5. Comparison for $M_\infty = 2.80$ gas - only flow. (a) interferogram from experiment conducted by Kaca (1988), (b) density contours from present calculation.



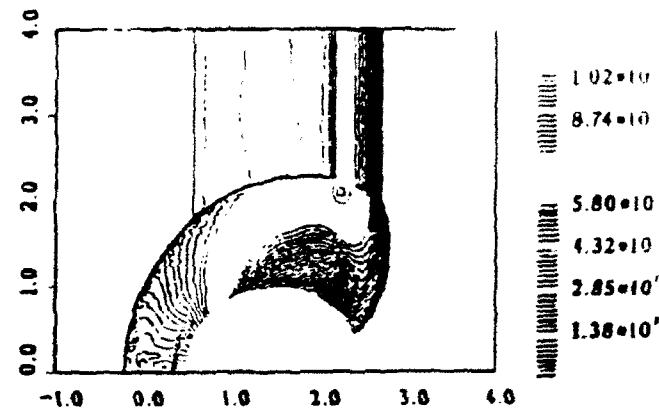
(a)



(a)



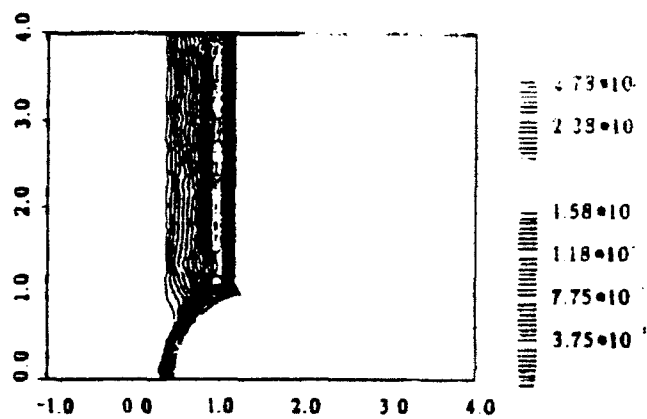
(b)



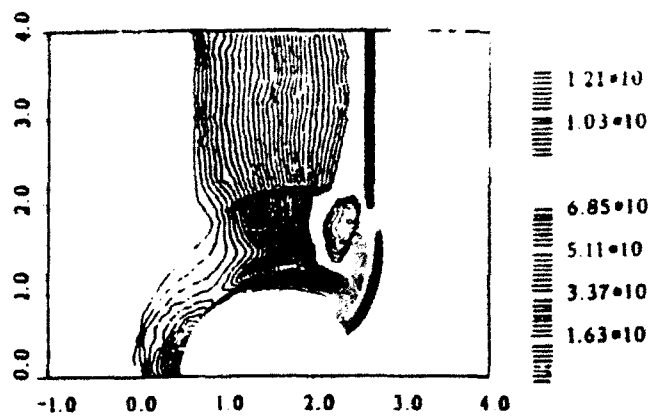
(b)

Figure 7. Density contours for the case: $M_s = 2.8$, $\rho_p = 0.76 \text{ kg/m}^3$ and $r_p = 10 \mu\text{m}$, (a) particle density and (b) gas density.

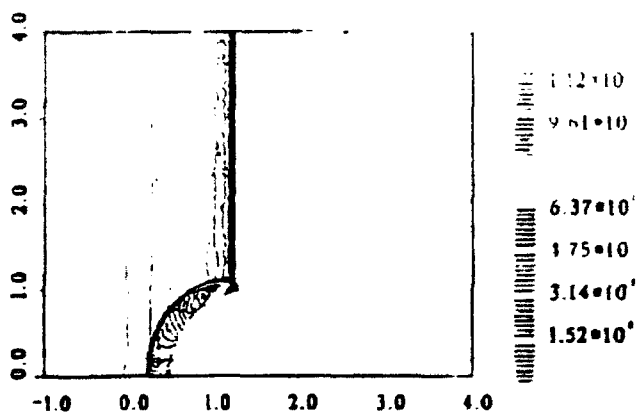
Figure 8. Density contours for the case: $M_s = 2.8$, $\rho_p = 0.76$ and $r_p = 50 \mu\text{m}$, (a) particle density and (b) gas density.



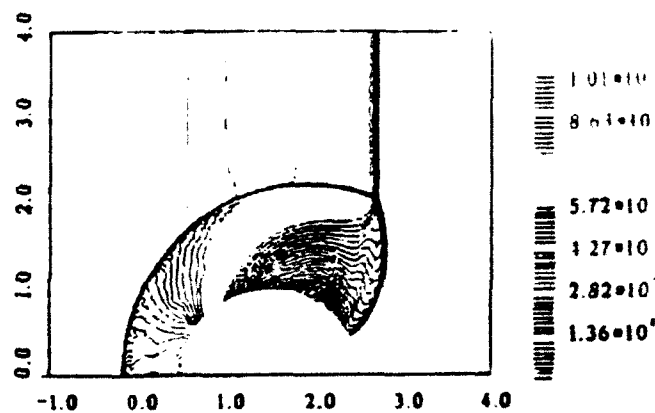
(a)



(c)



(b)



(d)

Figure 6. Density contours for the case: $M_\infty = 2.8$, $\rho_p = 0.25 \text{ kg/m}^3$, $r_p = 10 \mu\text{m}$ at two different times, (a) particle density at t_1 , (b) gas density at t_1 , (c) particle density at t_2 , and (d) gas density at t_2 .



AIAA 93-3089

**Acoustic Wave Focusing in an Ellipsoidal
Reflector for Extracorporeal Shock-wave
Lithotripsy**

Itzhak Lottati and Shmuel Eidelman

Science Applications International
Corporation

McLean, VA 22102

AIAA 24th

Fluid Dynamics Conference

July 6-9, 1993 / Orlando, FL

ACOUSTIC WAVE FOCUSING IN AN ELLIPSOIDAL REFLECTOR FOR EXTRACORPOREAL SHOCK-WAVE LITHOTRIPSY

by

Itzhak Lottati* and Shmuel Eidelman†
Science Applications International Corporation

Abstract

Simulations of acoustic wave focusing in an ellipsoidal reflector for extracorporeal shock-wave lithotripsy (ESWL) are presented. The simulations are done on a structured/unstructured grid with a modified Tait equation of state for water. The Euler equations are solved by applying a second-order Godunov method. The computed results compare very well with the experimental results.

Introduction

Research relating to focusing of shock and acoustic waves is of practical interest for extracorporeal shock-wave lithotripsy (ESWL). A considerable body of work is dedicated to this subject (see e.g., review in Ref. 1), and numerical simulations play a prominent role in research on these devices. It is conceivable that real-time numerical simulation can be used for better assessment of shock-wave impact on the targeted areas and more effective focusing. Requirements for these real-time simulations in terms of robustness, accuracy and efficiency are very stringent, and can be satisfied only with the most advanced numerical methods.

Structured rectangular grids allow the construction of numerical algorithms that integrate the fluid conservation equations efficiently and accurately. The efficiency of these schemes results from the extremely low storage overhead needed for domain decomposition and the efficient and compact indexing, which also defines domain connectivity. These two factors allow code construction based on a structured domain decomposition that can be highly vectorized and parallelized. Integration in physical space on orthogonal and uniform grids produces numerical algorithms with the highest possible accuracy. The disadvantage of structured rectangular grids is that they cannot be used to decompose computational domains with complex geometries. Thus it

is difficult to represent computationally a complex computational domain with the curved boundaries characteristic of typical reflectors used in ESWL devices.

The early developers of computational methods realized that, for many important applications of Computational Fluid Dynamics (CFD), it is unacceptable to describe curved computational domain boundaries using the stair-step approximation available with the rectangular domain decomposition technique. To overcome this difficulty, the techniques of boundary-fitted coordinates were developed. With these techniques, the computational domain is decomposed on quadrilaterals that can be fitted to the curved domain. The solution is then obtained in physical space using the geometrical information defining the quadrilaterals, or in the computational coordinate system that is obtained by transformation of the original domain into a rectangular domain. The advantage of this technique is that it employs the same indexing method as the rectangular structured domain decomposition methods that also serve to define domain connectivity. The boundary fitted coordinate approach leads to efficient codes, with approximately a 4:1 penalty in terms of memory requirement per cell as compared with rectangular domain decomposition. However, this approach is somewhat restricted in its domain decomposition capability, since distortion or large size variations of the quadrilaterals in one region of the domain lead to unwanted distortions or increased resolution in other parts of the domain. An example of this is the case of structured body-fitted coordinates used to simulate flows over a profile with sharp trailing edges. In this case, increasing the resolution in the vicinity of the trailing edge increases resolution in the whole row of elements connected to the trailing edge elements.

The most effective methods of domain decomposition developed to overcome this disadvantage are those using unstructured triangular grids. These methods were developed to cope with very complex computational domains. The unstructured grid method, while efficient and powerful in domain decomposition, results in codes that must store large quantities of information defining the grid geometry and connectivity, and have large computational and storage overheads. As a rule, a code with an unstructured grid requires greater storage by a factor of 10, and will run about 5 times slower on a per cell per iteration basis than a structured rectangular code.

Unstructured triangular meshes are designed to pro-

*Dr. Lottati is a Research Scientist with Science Applications International Corp. (SAIC), 1710 Goodridge Dr., MS 2-3-1, McLean, VA 22102, †Dr. Eidelman, Research Scientist, SAIC, Associate Fellow AIAA

vide a grid that is fitted to the boundary of complex geometry. The flexibility of the unstructured mesh that allows complex geometry to be gridded should be weighed against the huge memory requirement needed to define the interconnectivity of the triangles. To cut down on the memory overhead, unstructured grid methods are used to their best advantage when combined with grid adaptivity. This feature usually allows the dynamic reallocation of triangles according to the physics and geometry of the problem solved, which leads to a substantial reduction in the number of cells needed for the domain decomposition. However, this advantage is highly dependent on the problem solved. Adaptive unstructured grids have an advantage over nonadaptive unstructured domain decomposition if the area of high resolution needed is around one-tenth of the global area of the computational domain. As a result, while the adaptive unstructured method may be extremely effective for simulating flow with multiple shock waves in complex geometries, it becomes extremely inefficient when high resolution is needed in a substantial area of the computational domain.

Our approach to domain decomposition for ESWL applications combines the structured and unstructured methods to achieve better efficiency and accuracy. Under this method, structured rectangular grids are used to cover most of the computational domain, and unstructured triangular grids are used only to patch between the rectangular grids (Fig. 1) or to conform to the curved boundaries of the computational domain (Fig. 2). In these figures, an unstructured triangular grid is used to accurately define the curved internal or external boundaries and a structured rectangular grid is used to decompose the regions of the computational domain that have a simple geometry.

Mathematical Model

We consider a system of two-dimensional Euler equations written in conservation law form:

$$\frac{\partial U}{\partial t} + \frac{\partial F}{\partial x} + \frac{\partial G}{\partial y} = 0 \quad (1)$$

where

$$U = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ e \end{bmatrix}, \quad F = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ u(e + p) \end{bmatrix}, \quad G = \begin{bmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ v(e + p) \end{bmatrix}$$

Here u, v are the x, y velocity vector components, p is the pressure, ρ is the density, and e is total energy of the fluid.

The equation of state for water was adopted from Ref. 2. The actual pressure and density $\bar{p}, \bar{\rho}$ in water are modified and then applied in the Euler solver. The modified pressure and density are given as

$$p = \bar{p} + B, \quad (2)$$

$$\rho = \bar{\rho} / (1 + \bar{p}/B)^n, \quad (2a)$$

where $B = 2955$ bar and $n = 7.44$ to adjust the velocity of sound to that for water ($a_0 = 1483$ m/sec).

The initial pressure distribution $\bar{p}(r)$ in the left focal point is chosen as

$$\bar{p}(r) = 1.0 \text{ bar} + \Delta p \exp[-(r - r_0)/(a_0 \tau)], \quad (2b)$$

where Δp is the intensity of the blast, τ is a time scale and a_0 is sound speed in water ($\tau = 3 \mu\text{sec}$).

It is assumed that an initial distribution of the fluid parameters is given at $t = 0$, and the boundary conditions defining a unique solution are specified for the computational domain.

Integration Algorithm

The system of governing equations (1) can be written in the following form:

$$\frac{\partial U}{\partial t} + \nabla \cdot Q = 0, \quad (3)$$

where Q represents the convective flux vector. By integrating Eq. (3) over space and using Gauss' theorem, we obtain the following expression:

$$\frac{\partial}{\partial t} \int_{\Omega} U dA + \oint_{\partial\Omega} Q \cdot d\mathbf{l} = 0, \quad (4)$$

where $d\mathbf{l} = \mathbf{n} dl$, \mathbf{n} is the unit normal vector in the outward direction, and dl is the element of length on the boundary of the domain. The variable Ω is the domain of computation and $\partial\Omega$ is the domain boundary.

Equation (4) can be discretized for each element (cell-triangle) of the domain:

$$\frac{(U_i^{n+1} - U_i^n)}{\Delta t} A_i = \sum_{j=1}^3 Q_j^{n+\frac{1}{2}} \mathbf{n}_j \Delta l_j, \quad (5)$$

where A_i is the area of the cell; Δt is the marching time step; U_i^{n+1} and U_i^n are the primitive variables at the center of the cell at time n and at the updated $(n+1)$ st timestep; $Q_j^{n+\frac{1}{2}}$ are the value of the fluxes across the three boundaries edges on the circumference of the cell, where \mathbf{n}_j is the unit normal vector to edge j of the boundary, and Δl_j is the length of the boundary edge

j. Equation (5) is used to update the physical primitive variables U_i according to computed fluxes for each timestep Δt . The time step is subjected to the Courant-Fredrichs-Levy (CFL) constraint.

To ensure a second order spatial accuracy, the gradient of each primitive variable is computed in the baricenter of the cell. This gradient is used to define the projected values of primitive variables at the two sides of the cell edge, as shown in Fig. 3. The gradient is approximated by a path integral

$$\int_{\Omega} \nabla U_i^{\text{cell}} dA = \oint_{\partial\Omega} U_j^{\text{edge}} dl. \quad (6)$$

The notation is similar to the one used for Eq. (5), except that the domain Ω is a single cell and U_i^{cell} and U_j^{edge} are values at the baricenter and on the edge respectively. The gradient is estimated as

$$\nabla U_i^{\text{cell}} = \frac{1}{A} \sum_{j=1}^3 \bar{U}_j^{\text{edge}} n_j \Delta l_j, \quad (7)$$

where \bar{U}_j^{edge} is an average value representing the primitive variable value for edge j .

The gradients that are computed at each baricenter are used to project values for the two sides of each edge by piecewise linear interpolation. The interpolated values are subjected to monotonicity constraints.³ The monotonicity constraint assures that the interpolated values are not creating new extrema.

The monotonicity limiter algorithm can be written in the following form:

$$U_{\text{pro}}^{\text{edge}} = U_i^{\text{cell}} + \phi \nabla U_i \cdot \Delta r, \quad (8)$$

where Δr is the vector from the baricenter to the point of intersection of the edge with the line connecting the baricenters of the cells over the two sides of this edge. ϕ is the limiter coefficient that limits the gradient ∇U_i .

First, we compute the maximum and minimum values of the primitive variable in the i 's cell and its three neighboring cells that share common edges (see Fig. 3):

$$\left. \begin{aligned} U_{\text{cell}}^{\text{max}} &= \max(U_k^{\text{cell}}) \\ U_{\text{cell}}^{\text{min}} &= \min(U_k^{\text{cell}}) \end{aligned} \right\} k = i, 1, 2, 3. \quad (9)$$

The limiter can be defined as:

$$\phi = \min \{1, \phi_k^{lr}\}, \quad k = 1, 2, 3, \quad (10)$$

where the superscript lr stands for left and right of the three edges (6 combinations altogether). ϕ_k^{lr} is defined by:

$$\phi_k^{lr} = \frac{[1 + \text{Sgn}(\Delta U_k^{lr})] \Delta U_{\text{cell}}^{\text{max}} + [1 - \text{Sgn}(\Delta U_k^{lr})] \Delta U_{\text{cell}}^{\text{min}}}{2\Delta U_k^{lr}} \quad (11)$$

$$k = 1, 2, 3,$$

where $\Delta U_k^{lr} = \nabla U_i^{lr} \cdot \Delta r_k$ and

$$\left. \begin{aligned} \Delta U_{\text{cell}}^{\text{max}} &= U_{\text{cell}}^{\text{max}} - U_i^{\text{cell}} \\ \Delta U_{\text{cell}}^{\text{min}} &= U_{\text{cell}}^{\text{min}} - U_i^{\text{cell}} \end{aligned} \right\}. \quad (12)$$

To obtain second-order accuracy in space and time, we subject the projected values of the left and right side of the cell edge to characteristic constraints following Ref. 4. The one-dimensional characteristic predictor is applied to the projected values at the half timestep $t^n + \Delta t/2$. The characteristic predictor is formulated in the local system of coordinates for the one dimensional Euler equation. We illustrate the implementation of the characteristic predictor in the direction of the unit vector n_c . The Euler equations for this direction can be written

$$W_t + A(W)W_{n_c} = 0, \quad (13)$$

where

$$W = \begin{Bmatrix} \tau \\ u \\ p \end{Bmatrix}; \quad A(W) = \begin{pmatrix} u & -\tau & 0 \\ 0 & u & \tau \\ 0 & \rho c^2 & u \end{pmatrix}, \quad (14)$$

where $\tau = \rho^{-1}$, ρ denotes density, and u, p are the velocity and pressure. The matrix $A(W)$ has three eigenvectors ($l^{\#}, r^{\#}$) (l for left and r for right, where $\#$ denote $+, 0, -$) associated with the eigenvalues $\lambda^+ = u + c$, $\lambda^0 = u$, $\lambda^- = u - c$.

An approximation of the value projected to an edge, accurate to second order in space and time, can be written

$$\begin{aligned} W_{i+\Delta r}^{n+1/2} &\approx W_i^n + \frac{\Delta t}{2} \frac{\partial W}{\partial t} + \Delta r \frac{\partial W}{\partial r_{n_c}} \\ &\approx W_i^n + \left[\Delta r - \frac{\Delta t}{2} A(W_i) \right] \frac{\partial W}{\partial r_{n_c}} \end{aligned} \quad (15)$$

An approximation for $W_{i+\Delta r}^{n+1/2}$ can be written as

$$W_{i+\Delta r}^{n+1/2} = W_i + (\Delta r_i - \frac{\Delta t}{2} (M_x M_n) n_c) \cdot \nabla W_i, \quad (16)$$

where

$$(M_x M_n) = \begin{cases} \text{Max}(\lambda_i^+, 0) & \text{for cell left to the edge} \\ \text{Min}(\lambda_i^-, 0) & \text{for cell right to the edge} \end{cases} \quad (17)$$

The gradients calculated in the process of computing the projected values at $t^n + \Delta t/2$ are subjected to the monotonicity limiter.

Following the characteristic predictor described above, the full Riemann problem is solved at the edge. The solution of the Riemann problem defines the flux $Q_j^{n+1/2}$ through the edge. The fluxes through the edges of triangles are then integrated (Eq. 5), thus updates the variables at t^{n+1} . One of the advantages of this algorithm is that calculation of the fluxes is done over the largest loop in the system (the loop over edges) and can be carried out in the vectorized or parallelized loop. This makes the algorithm efficient.

The algorithm presented is a modification of the algorithm of Ref. 5, which was derived for a structured mesh. The present algorithm has been applied to simulate a wide range of flow problems and has been found to be very accurate in predicting the features of the physics. The performance of the algorithm is well documented in Refs. 6-9. The algorithm for the rectangular cells are identical except the cell has four edges (Eq. 5).

Sound Wave Focusing in an Ellipsoidal Reflector

For our simulations, we chose a deep reflector shaped like an ellipsoid, which was used for ESWL by Dornier and other companies. A schematic of the cross section of this reflector is shown in Fig. 4. Strong acoustic waves are generated in the left focal point of the ellipsoid by an instantaneous release of energy and are refocused at the right focal point. Ideally, a reflector should employ waves of acoustic intensity, since the nonlinear reflections of strong shock waves lead to significant distortions in wave propagation and impair simple geometrical focusing.

Figure 2 shows the computational domain and grid for the ellipsoidal reflector that we used in our study. In order to illustrate the concept of the composite structured/unstructured grid, we have shown only every sixteenth cell of the grid that was actually used for the simulation. In this example, we observe that the structured rectangular grid covers about 90% of the computational domain, and the unstructured triangular grid is restricted to the curved surface of the ellipsoid and covers about 10% of the domain. The major axis of the ellipsoid is 150 mm and the minor axis is 90 mm.

Two simulations were conducted with two different Δp values to study how the intensity of the blast affects focusing of waves in the reflector. The first simulation was done with $\Delta p = 725$ bar and $\tau = 3\mu s$ where $|r - r_0| < 10$ mm. The other simulation was done by using pressure three times larger than in first simulation.

In Figs. 5a-5d simulation results for the $\Delta p = 725$ bar conditions are shown in the form of pressure con-

tour plots. Figure 5a shows pressure distribution for the initial stage of wave propagation before the wave front has reached the surface of the reflector. The contour plots are shown at $t = 1.10 \times 10^{-5}$ sec. At this time the maximum pressure in the wave has dropped to 173 bar. In Fig. 5b pressure contours are shown at $t = 3.32 \times 10^{-5}$ sec. Here we observe that the wave reflected from the surface of the reflector has maximum pressure about five times than that of the incident wave. However, both wave fronts propagate through the water with a constant speed equal to the speed of sound, and the phase shift observed in Fig. 5b holds through the calculation. In Fig. 5c the simulation results are shown at the stage when the incident wave is crossing the center of symmetry of the reflector. Here $t = 8.88 \times 10^{-6}$ sec. It is interesting to note that the value of the overpressure at this location was used in Ref. 1 as a normalizing value for presentation of the experimental and computational results. In our case for the initialization with $\Delta p = 725$ bar the incident pressure at the center of the ellipsoid is $p = 11.1$ bar. In Fig. 5d simulation results are shown at $t = 19.2 \times 10^{-5}$ sec, when maximum focusing of the reflected wave take place. The pressure values in the focal point reaches 188 bar. This maximum is immediately followed by a negative phase with a minimal pressure of 163 bar. This strong pressure variations can cause disintegration of the stones by the ESWL apparatus.

In Figs. 6a-6b simulation results are shown for the second case of $\Delta p = 2175$ bar. As we can see in Fig. 6a, this value of the initial overpressure produces an incident wave with about 33 bar, which is a bit higher than the 29 bar value observed in Ref. 1. The wave structure at the time of focusing is shown in Fig. 6b. Here we can observe that for this case the maximum pressure reaches 494 bar, followed by a 371 bar minimum. Comparing this case with that reported above, we conclude that the amplification at the focal point is smaller in the second case.

The waves observed in the system are of acoustic intensity and are propagating at the speed of sound. The reflected wave will therefore not be able to catch up with the incident wave. Except for some compressibility effects in the initiation and focusing stages when pressures are high, the fluid will behave as incompressible. Figure 7 shows the density contour for the first case ($\Delta p = 725$ bar). As expected, the compressibility effect is negligible.

In Fig. 8 the simulation results are compared with the experimental results in a plot of normalized pressures as function of distance from the focal point. In this figure the simulation results for the case of initiation with $\Delta p = 725$ bar and $\Delta p = 2175$ bar are shown by the curves marked with triangles and rectangles respectively. The experimental results for the 29-bar incident pressure

(which most closely fits our second simulation) are shown by the curve marked by circles. In Fig. 8 we see that the maximum reflection factor is achieved for weaker waves, which is consistent with the results reported in Ref. 1. The simulation results are very close to the experimental ones in the case of $\Delta p = 2175$ bar initiation for focal point location and pressure amplification factor, which validates the simulation methodology.

In all the figures presented, the method of composite domain decomposition works extremely well, producing solutions with no seams at the interfaces. We should mention here that our test problem is particularly sensitive because the main acoustic waves are weak, and any inaccuracy introduced at the grid interfaces would produce a distortion in the phase or in the intensity of the traveling waves that would be a visible disturbance evident in the results.

Conclusions

A composite method of structured/unstructured domain decomposition is introduced as an efficient technique for dealing with the computational domains of complex geometry. We have simulated a demanding acoustic wave focusing problem and have shown that our approach leads to accurate wave propagation without any reflection or distortion at the structured/unstructured grid interfaces. Note that for the acoustic focusing problem as simulated and presented in this paper, both structured and unstructured methods of domain decomposition can be shown to be inadequate if used separately. The structured method has difficulty describing the curved boundaries of the computational domain, while the unstructured method is totally inefficient in describing phenomena with wide fronts that occupy a large portion of the computational domain. Our hybrid method combines the advantages of structured and unstructured methods of domain decomposition. This hybrid technique combines the efficiency of the unstructured grid, which accurately represents curved walls, with the computational and memory efficiency of the structured grid in the majority of the computational domain. We also attribute the quality of the numerical result to the Second Order Godunov method, which allows a consistent, accurate and robust formulation for handling both grids and boundary conditions.

Acknowledgment

The work reported here was partially supported by ARPA and ONR. The authors would like to thank Col James Crowley and Dr. Lawrence Kabecoff for their interest in this project.

References

1. H. Gronig, "Past, Present and Future of the Shock Focusing Research," Proceedings of the International Workshop on Shock Wave Focusing, Sendai, Japan, March 1989.
2. K. Isuzugawa and M. Horiuchi, "Experimental and Numerical Studies of Blast Wave Focusing in Water," Proceedings of the International Workshop on Shock Wave Focusing, Sendai, Japan, March 1989.
3. B. van Leer, "Towards the Ultimate Conservative Difference Scheme, V.A. Second Order Sequel to Godunov's Method," J. Comp. Phys. 32, 101-136 (1979).
4. P. Collela and P. Woodward, "The Piecewise Parabolic Method (PPM) for Gasdynamic Simulations," J. Comp. Phys. 54, 174-201 (1984).
5. S. Eidelman, P. Collela, and R.P. Shreeve, "Application of the Godunov Method and Its Second Order Extension to Cascade Flow Modeling," AIAA Journal 22, 10, 1984.
6. I. Lottati, S. Eidelman and A. Drobot, "A Fast Unstructured Grid Second Order Godunov Solver (FUGGS)," 28th Aerospace Sciences Meeting, AIAA-90-0699, Reno, NV (1990).
7. I. Lottati, S. Eidelman, and A. Drobot, "Solution of Euler's Equations on Adaptive Grids Using a Fast Unstructured Grid Second Order Godunov Solver," Proceeding of the Free Lagrange Conference, Jackson Lake, WY, June 1990.
8. I. Lottati and S. Eidelman, "Second Order Godunov Solver on Adaptive Unstructured Grids," Proceeding of the 4th International Symposium on Computational Fluid Dynamics, Davis, CA, September 1991.
9. I. Lottati and S. Eidelman, "A Second Order Godunov Scheme on Spatial Adapted Triangular Grid," To appear in a special issue of Applied Numerical Mathematics (Proceedings of U.S. Army Workshop on Adaptive Methods for Partial Differential Equations, R.P.I., March 1992).

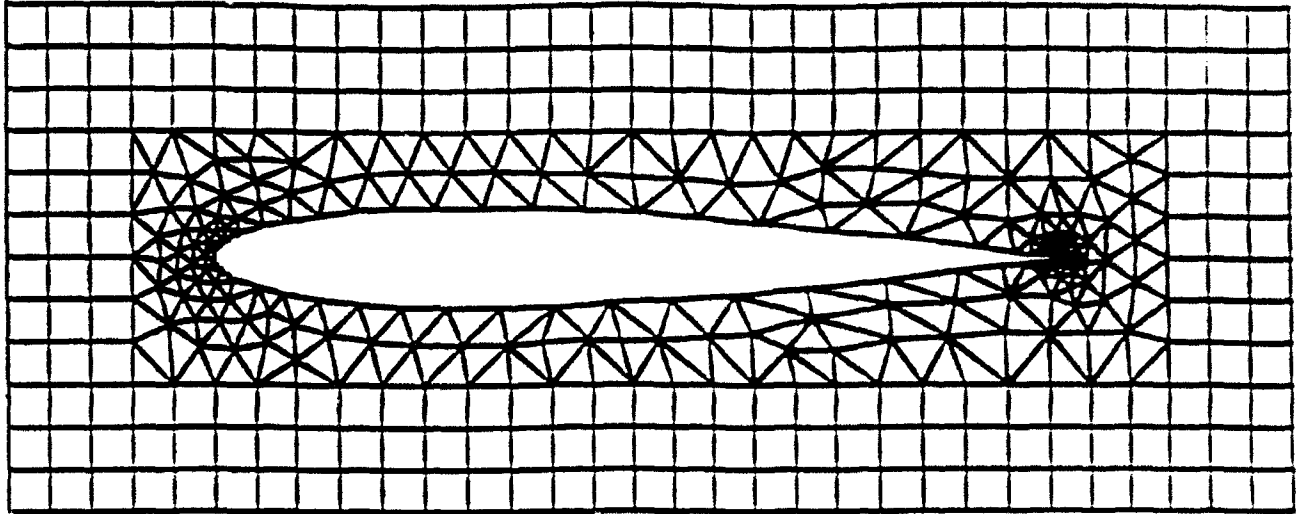


Figure 1. A possible candidate configuration for hybrid structured/unstructured domain decomposition.

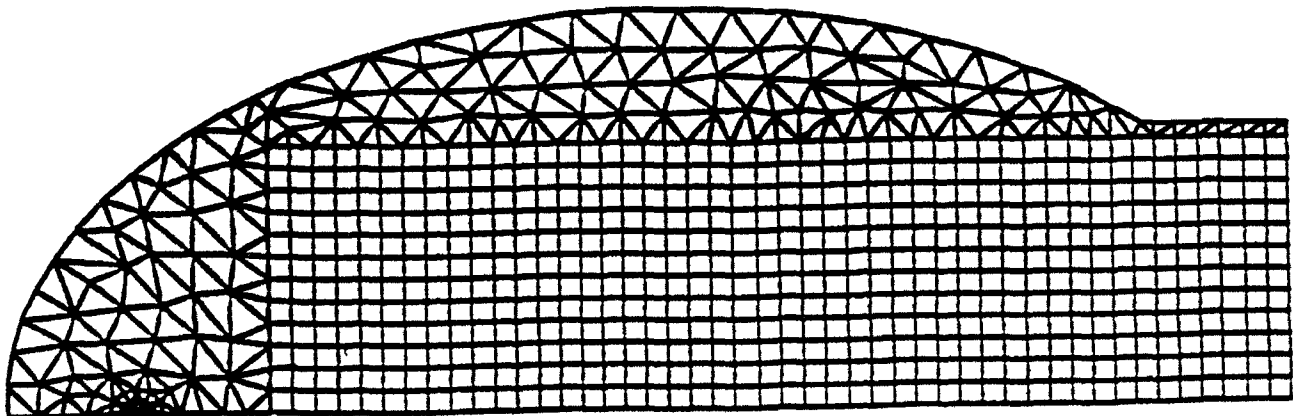
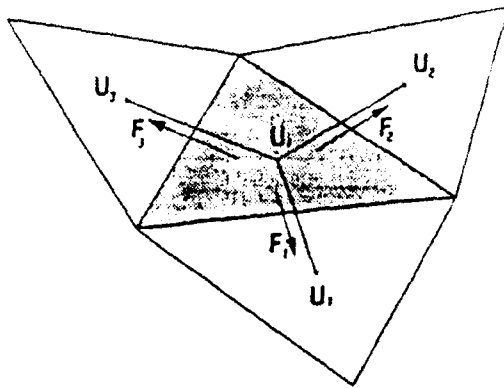


Figure 2. A possible candidate configuration for hybrid structured/unstructured domain decomposition, representing the ellipsoid reflector grid used for the numerical simulation.



Representative triangular cell in the mesh and its neighbors showing fluxes across the edges

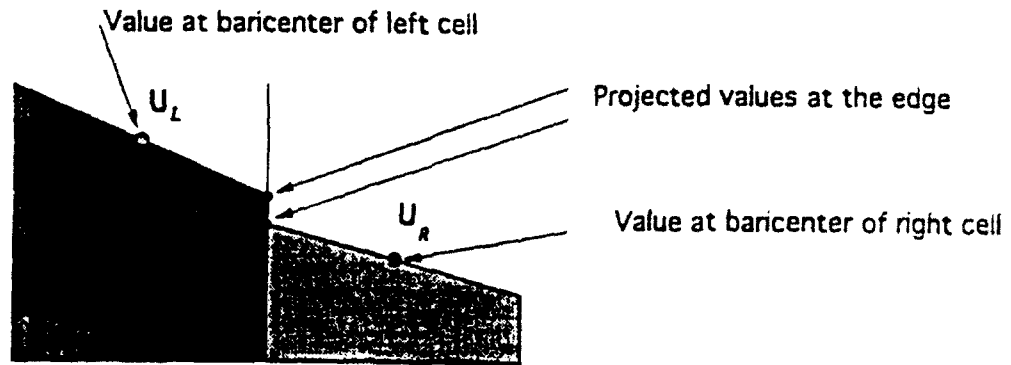


Figure 3. Second order triangular based flux calculation.

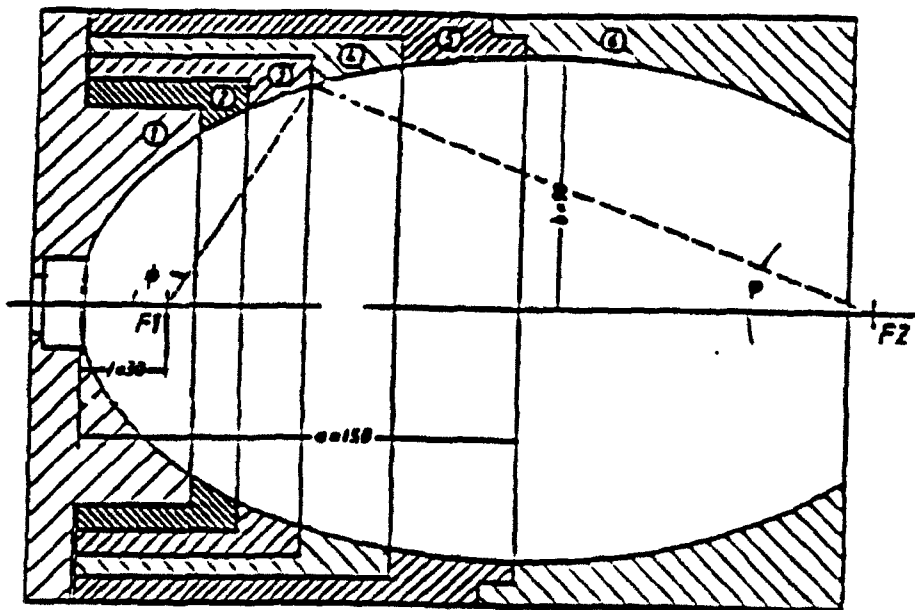
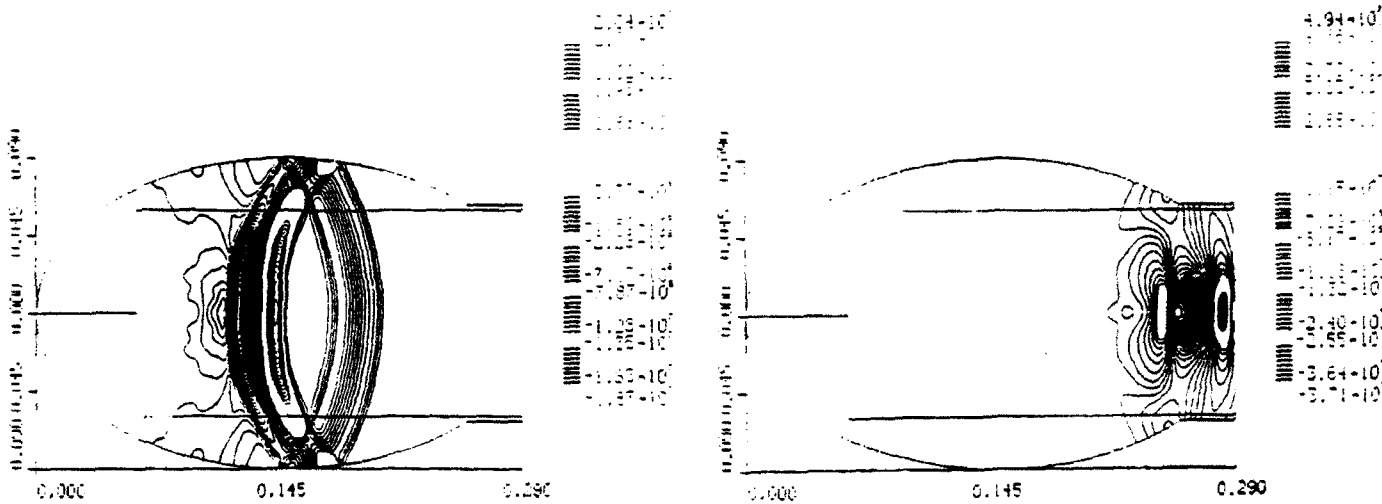


Figure 4. A schematic drawing of the center cross section of the ellipsoid reflector.



a. Time = 1.08×10^{-4} sec

b. Time = 1.96×10^{-4} sec

Figure 6. Pressure contours showing the incident wave and the reflected wave pattern for $\Delta p = 2175$ bar.

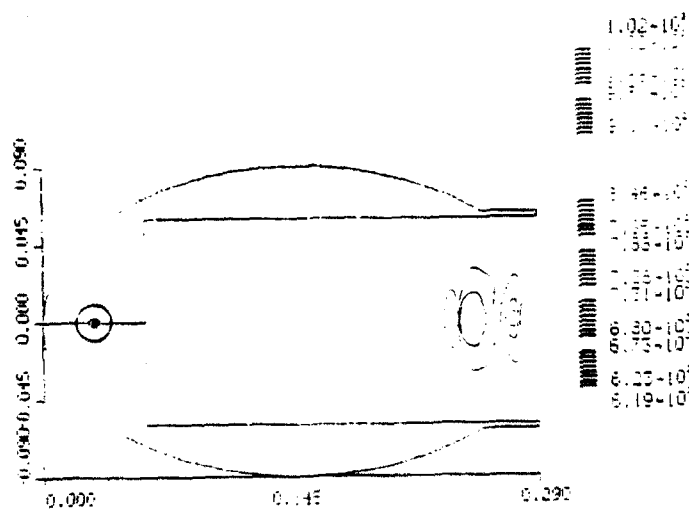


Figure 7. Density contours emphasizing the fact that the compressibility effect is negligible ($\Delta p = 725$ bar at $t = 1.92 \times 10^{-4}$ sec).

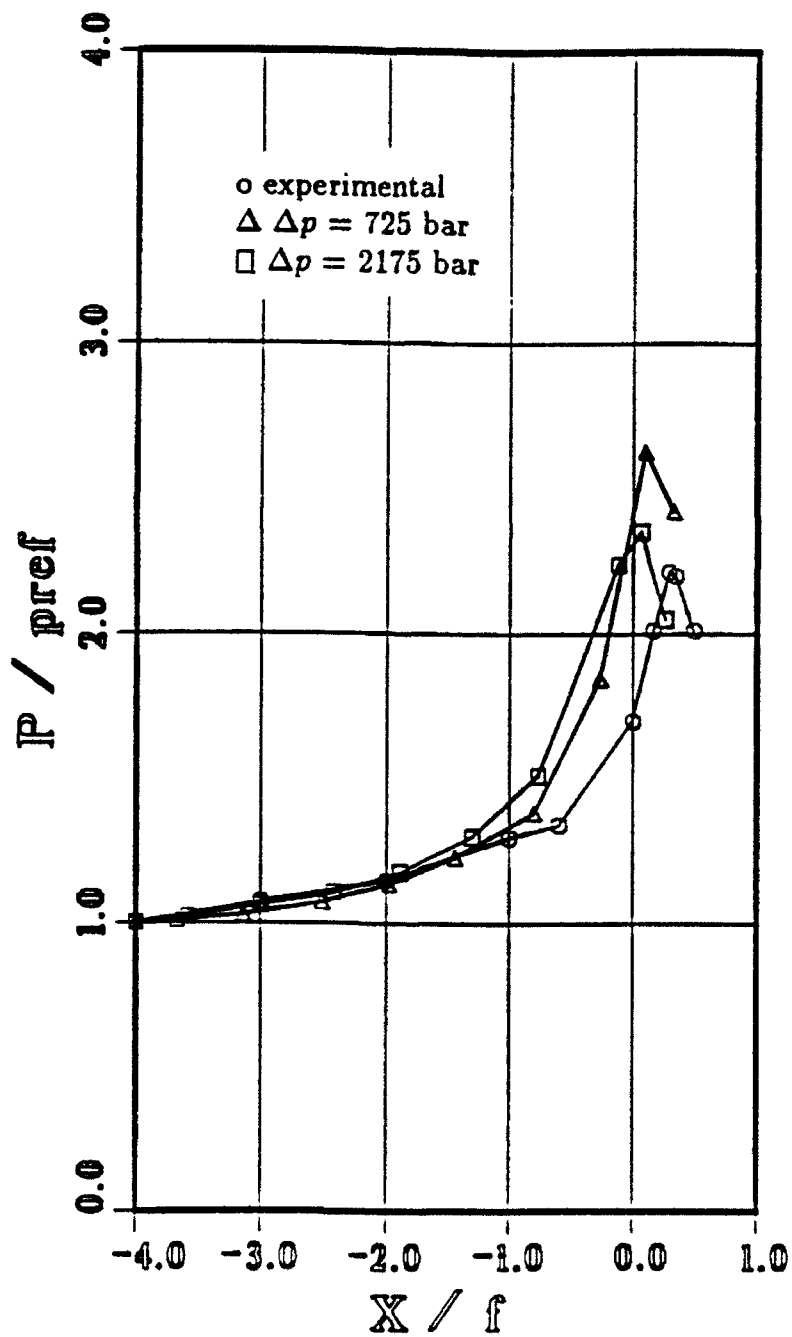


Figure 8. Normalized maximum pressure distribution on the axis of symmetry. A comparison between computed and experimental results.