SOLUTION METHODS FOR RE-ENTRANT CORNERS IN ELLIPTIC PARTIAL DIFFERENTIAL EQUATIONS

By

DAVID PAUL WILLIAMS

Bachelor of Science

Southeastern Oklahoma State University

Durant, Oklahoma

1978

Submitted to the Faculty of the Graduate College
of the Oklahoma State University
in partial fulfillment of the requirements
for the Degree of
MASTER OF SCIENCE
May, 1983

SOLUTION METHODS FOR RE-ENTRANT CORNERS IN ELLIPTIC PARTIAL DIFFERENTIAL EQUATIONS

Thesis Approved:

J P Chandler
Thesis Adviser

D. E. Hedrick

K B Fisher

Norman N. Durham
Dean of the Graduate College

ACKNOWLEDGMENTS

I would like to thank the members of my committee, Dr. J. P. Chandler, Dr. D. D. Fisher, and Dr. G. E. Hedrick, for their assistance. I am indebted to Dr. Chandler for many interesting and informative discussions on a broad range of topics.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

OVERVIEW

Re-entrant corners in systems governed by elliptic partial differential equations strongly affect finite difference methods for determining solutions. A re-entrant corner is a boundary feature formed by intersecting line segments having an internal angle larger than π radians. The accuracy and, in some instances, the convergence characteristics of numerical solution methods are degraded. In this thesis, several finite difference techniques are examined. A technique for restoring their accuracy is the subject of analysis and experiment, here.

In Chapter II, necessary aspects of the theory of partial differential equations are presented: classification and boundary conditions. The focus there is on elliptic partial differential equations. Laplace's equation, which is elliptic, is introduced; it serves in the development and testing of the finite difference methods used here.

Chapter III deals with the discrete problem: the formation of discrete analogues of the solution domain and elliptic differential operator. The discrete problem consists of a network of points and a system of difference equations. The effects of curved boundaries and boundary conditions on the process are discussed. In addition, the basic iterative methods for solving the system of difference equations, the Jacobi and Gauss-Seidel methods, are introduced.

Techniques for accelerating the basic iterative methods are discussed in Chapter IV. Analysis of the accuracy and convergence characteristics of the basic and accelerated methods is given. The accelerated methods studied include the successive overrelaxation method (SOR), the symmetric successive overrelaxation method (SSOR), and the multigrid method. Vector extrapolation is also examined as an acceleration technique.

In Chapter V, re-entrant corners in problems governed by Laplace's equation are studied. Analysis shows that finite difference formulations are invalid in the vicinity of re-entrant corners. Errors propagated from the region of the re-entrant corner by the finite difference method affect the entire solution. Motz's method for singular points in partial differential equations is developed for use in combination with finite difference methods when re-entrant corners are present.

The results of several numerical experiments are given in Chapter VI. The various iterative techniques are used to solve a test problem in order that comparisons of their accuracy and convergence characteristics can be made. Efficient methods are chosen and used to solve a problem having a single re-entrant corner: the L-shaped region problem. Numerical solutions are formed with and without the use of Motz's method. Analysis of the solutions shows to what extent the re-entrant corner affects solutions by finite difference methods and how effective Motz's method is in restoring accuracy. Conclusions from the findings of the experiments and suggestions for further investigations are also presented.

CHAPTER II

ELLIPTIC PARTIAL DIFFERENTIAL EQUATIONS

Partial Differential Equations in Physical Systems

Second order partial differential equations (PDEs) describe the behavior of many important physical systems. Among the systems are fluids, mechanical stress in solid structures, gravitational and electrostatic potentials, and wave phenomena. A solution for the PDE of a particular system depends on the configuration of the system. Thus, for a particular system a wide variety of solutions is possible. Much effort is expended in determining solutions of PDEs, both analytical and numerical solutions. With a solution, the underlying mechanisms of a system can be examined; its future behavior or the effects of a change of configuration can be predicted. In the forecasting of weather, a fluid flow problem, solutions are projected into the future. The design of airfoils requires an understanding of the effects of small changes in shape on the behavior of the airfoils. Much computational effort goes into the solution of these two problems.

Classification of Second Order Partial Differential Equations

A technique for the classification of PDEs is useful. An appropriate solution method can be selected and general features of the

behavior of the system can be determined. Consider the general second order equation in two dimensions.

$$au_{xx} + bu_{xy} + cu_{yy} = f \tag{2.1}$$

with solution u(x,y) (4, 11, 19). The subscripts x and y in (2.1) indicate partial derivatives with respect to x or y. Thus, u_{xx} indicates the second partial derivative with respect to x. The coefficients a, b, c, and f in (2.1) are all functions of x, y, u, u_x, and u_y. An equation cast in the form of (2.1) can be classified on the basis of the behavior of the coefficients a, b, and c. As the coefficients may vary spatially, an equation may change type over the domain on which it is defined.

In a region where $b^2 - 4ac > 0$, equation (2.1) is hyperbolic (19). Wave phenomena and supersonic flow problems are governed by hyperbolic equations. The wave equation,

$$u_{xx} - u_{tt} = 0 \tag{2.2}$$

is hyperbolic. In the case of an ideal string vibrating without frictional losses, the terms u_{xx} and u_{tt} are proportional to the tension and acceleration in string elements, respectively.

Where $b^2 - 4ac = 0$, equation (2.1) is parabolic (19). Heat conduction and diffusion in isotropic media are described by parabolic equations. The diffusion equation,

$$u_{xx} - u_t = 0 \tag{2.3}$$

is parabolic. In (2.3), u_t is a velocity term. It may represent a frictional, retarding force. Characteristic of parabolic systems is the propagation of a solution in time or space.

In a region where $b^2 - 4ac < 0$, an equation of the form (2.1) is elliptic (19). Systems in equilibrium are described by elliptic equations: electrostatic and gravitational potentials, thermal equilibrium, and mechanical stress. Laplace's equation for potentials is elliptic,

$$u_{xx} + u_{yy} = 0 \ . \tag{2.4}$$

The Laplacian differential operator occurs frequently,

$$\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \ . \tag{2.5}$$

For the remaining discussion, equations will be considered to be of one type only on the domain of the problem.

Boundary Conditions

A problem to be solved is specified by the differential equation, the domain of the solution, and the conditions imposed on the solution on the boundary of the solution domain. The boundary conditions may specify the value of the solution there, the normal derivative of the solution, $\frac{\partial u}{\partial n}$, or combinations of the solution value and normal derivative. On a segment of the boundary where the solution only is specified, Dirichlet conditions exist; Neumann conditions occur where the normal derivative alone is specified. If both the value and normal derivative are specified on a segment, then Cauchy conditions are in effect there (19).

Elliptic problems require a closed, stationary boundary with static boundary conditions. In contrast, hyperbolic and parabolic problems may have open boundaries or moving boundaries, and boundary conditions that change with time. Figure 1 depicts an elliptic problem; Figure 2 depicts a parabolic problem.

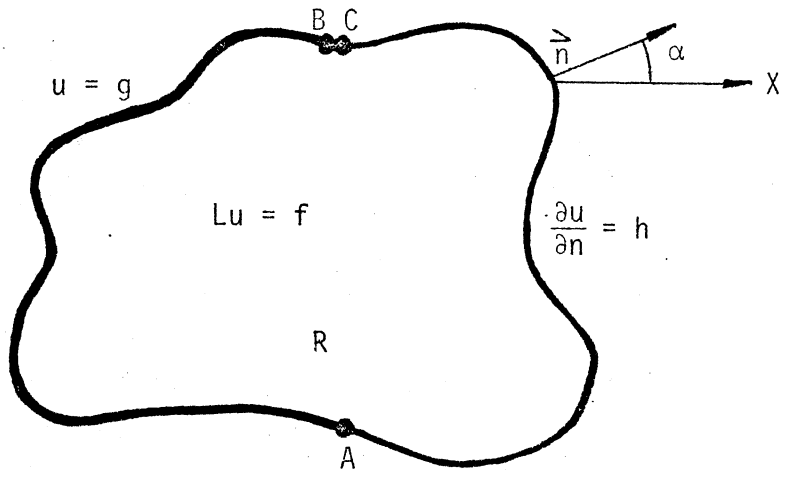


Figure 1. An Elliptic Problem With Mixed Neumann and Dirichlet Boundary Conditions

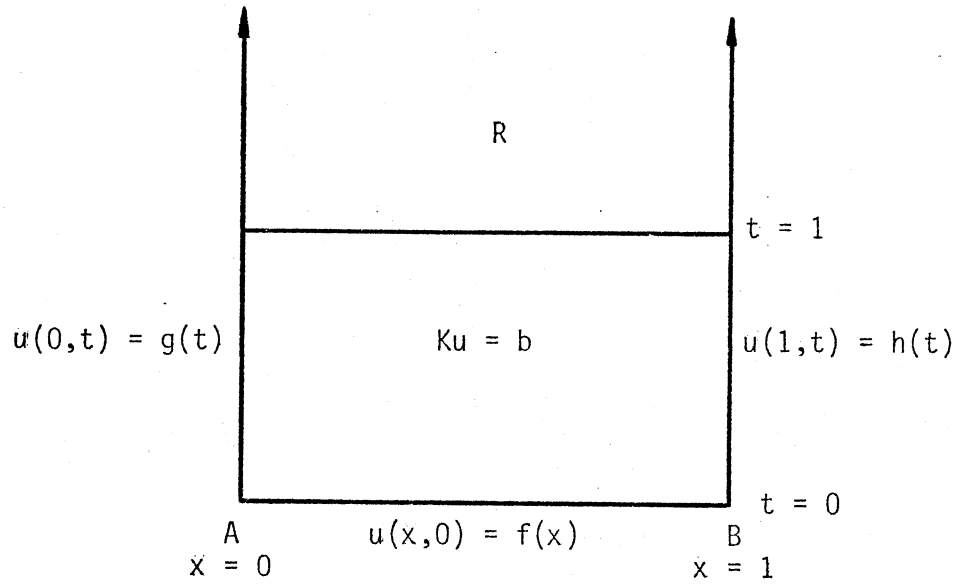


Figure 2. A Parabolic Problem--Solution Propagating in Time

The elliptic problem has Dirichlet conditions on the boundary segment AB and Neumann conditions on the segment AC. On AB, the solution is given by the function g(x,y). On AC, the normal derivative is given by h(x,y),

$$\frac{\partial u}{\partial n} = \frac{\partial u}{\partial x} \cos \alpha + \frac{\partial u}{\partial y} \sin \alpha = h(x,y) \ . \tag{2.6}$$

The angle α is formed by the x-axis of the problem and the normal vector n on AC. It is convenient to summarize the conditions of the boundary of R, as

$$\Lambda u(x,y) = \phi(x,y) \ . \tag{2.7}$$

The system of Figure 1 attains some equilibrium configuration on the domain R according to the elliptic equation Lu = f.

The initial value of the solution of the parabolic problem of Figure 2 (t = 0) is given by f(x). The solution propagates in time according to the equation Ku = b and the boundary conditions given by g(t) on the line x = 0 and h(t) on x = 1.

The solution of an elliptic problem is uniquely determined by Dirichlet or Neumann conditions, alone, or by a combination of Dirichlet and Neumann conditions as in Figure 1. Cauchy conditions are overly restrictive for elliptic problems and may rule out solutions altogether (19).

Laplace's Equation

Electrostatic and gravitational potentials and temperature distributions in bodies in thermal equilibrium are governed by Laplace's equation,

$$\nabla^2 u = 0 \ . \tag{2.8}$$

Sources of the potentials or temperature distributions are kept outside the solution domain. The related Poisson equation,

$$\nabla^2 u = f \ . \tag{2.9}$$

takes into consideration sources within the solution domain.

Solutions for Laplace's and Poisson's equations contain information about vector fields that is often the object of a solution attempt. If $u(x,y)$ is an electrostatic potential, the electric field is given by

$$\tilde{E} = - \tilde{\nabla} u \tag{2.10}$$

where

$$\tilde{\nabla} = \hat{i} \frac{\partial}{\partial x} + \hat{j} \frac{\partial}{\partial y} \ .$$

The vectors $\hat{i}$ and $\hat{j}$ are unit vectors in the x- and y- directions, respectively (15).

Solutions for Laplace's equation have two important properties to consider here. The maximum and minimum values of the solution occur on the boundary of the solution domain. Also, the solution is harmonic. That is, complex transformations of the solution and solution domain produce functions that are, in turn, solutions of Laplace's equation (19). Transformations can be used to simplify problems having complicated geometries.

CHAPTER III

FINITE DIFFERENCE METHODS FOR SOLVING ELLIPTIC PARTIAL DIFFERENTIAL EQUATIONS

Introduction

Finite difference methods are important tools for the solution of PDEs. Problems that will not yield to analytical techniques can be solved with finite difference methods. The tools of analysis are supplanted by the ability of a computer to perform elementary operations rapidly. The finite difference approach seeks an economical solution for a system of difference equations that approximates the behavior of a PDE on some solution domain. Solution of the system is easily adapted to the use of high speed digital computers.

Finite difference methods can be classified according to the type of problem to be solved. There are methods specific to elliptic, parabolic, and hyperbolic problems.

Among the first to discuss the use of finite difference methods and to identify techniques specific to the different types of problems was Richardson (4) in 1910. He identified methods for elliptic and parabolic problems and presented a solution for a problem of stress in an earthen dam based on those methods. In a 1925 paper, Richardson (25) characterized propagation and equilibrium problems as "marching" and "jury" problems, respectively. The solution is "marched" out from the

boundary in a propagation problem; the boundary parts of an equilibrium problem sit in "judgment" of solution values within the solution domain.

Another important paper in the development of finite difference methods was released in 1928 by Courrant, Friedrichs, and Lewy (9). Their work covered techniques for elliptic and hyperbolic problems. The 1938 paper by Shortley and Weller (27) dealt with solution methods for Laplace's equation. In it, extrapolation methods were proposed to speed the solution of the system of difference equations.

The Discrete Solution Domain

The finite difference solution consists of discrete points that are representative of the continuous solution. These are usually arranged on the solution domain in some regular grid; uniform triangles, rectangles, or hexagons are the cells of the network. For convenience, a grid is usually formed of rectangular or square cells. Consider the domain R of Figure 3 with a grid of rectangular cells superimposed. A grid point P in R with coordinates (x,y) can be located in a number of ways; if numbered, the row and column of the grid point can serve. Thus, a value in the discrete solution at point (x,y) can be represented as

$$u(x,y) = u(ih,jk) = u_{ij} \tag{3.1}$$

where h and k are the dimensions of the grid cells. Grids with square cells are used most commonly (h = k). Here, they are called uniform grids.

A set of points u_{ij} making up the discrete solution will be called the solution vector and will be designated $\vec{u}$. As will be seen in coming discussions, a method for storing the solution vector in

computer memory is needed that will allow ready access to a point u_{ij} and its immediate neighbors. The two-dimensional array of FORTRAN, whether simulated or used as it exists, is a storage scheme which will allow such access. For a solution domain that has an irregular shape, such as the one of Figure 3, a system of pointers will be necessary to establish the proper shape in the two-dimensional array.

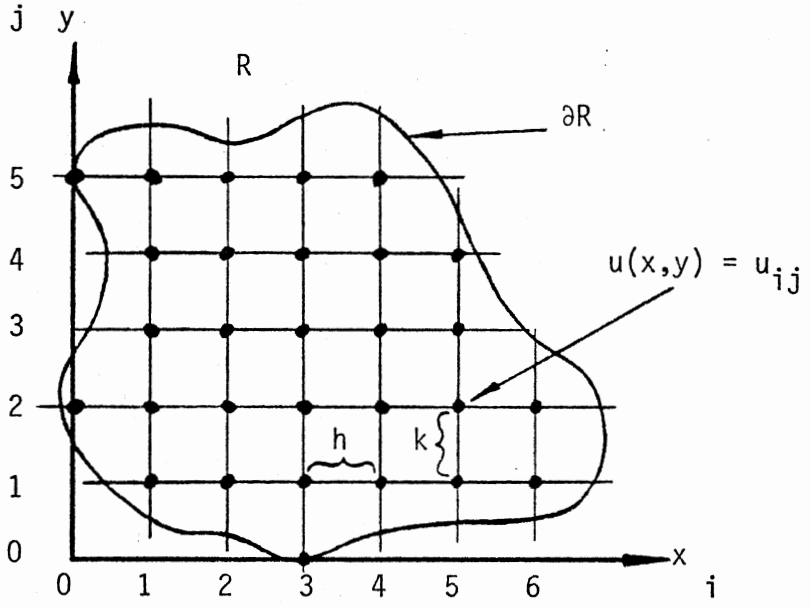


Figure 3. A Rectangular Grid Superimposed on the Solution Domain R

The mesh of Figure 3 has few grid points that coincide with the boundary of R, ∂R. Then, R is being misrepresented and information supplied by the boundary is lost. The boundary should be represented as closely as is possible to insure that the final solution has the most accuracy attainable (4) (11).

Figure 4 shows a boundary segment, AB, that is represented more closely by the solution grid. Auxiliary points are added to the solution grid where grid lines intersect the boundary. The penalty for adding the auxiliary points is a more complicated storage scheme in the computer implementation; the distance from each auxiliary point to the adjacent grid point must be stored.

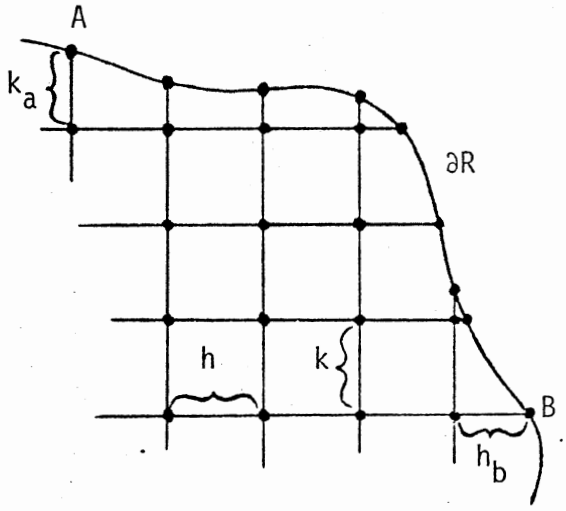


Figure 4. An Improved Boundary Treatment--Grid Points Added at Intersections With ∂R

The Difference Equation

Now that the solution domain, R, has been made discrete, the partial differential equation whose solution is sought there must be made discrete. Assume that the PDE with differential operator L,

$$L\,u(x,y) = f(x,y) \tag{3.2}$$

is to be solved on R with the conditions

$$L\ u(x,y) = \phi(x,y) \tag{3.3}$$

on the boundary, ∂R. Some discrete analogue of the differential operator, L, is needed which, when applied to the points of the discrete solution, closely approximates the behavior of L applied to the continuous solution.

For the remainder of this discussion L will be the Laplacian operator

$$\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \ . \tag{3.4}$$

The discrete operator will approximate the behavior of the Laplacian. Consider the grid of Figure 5. It is a portion of the grid of Figure 3, having spacing h and k. Taylor series expansions of the solution u(x,y) at the grid points adjacent to the point (x,y) can be used to develop approximations for the derivatives of the Laplacian operator, (3.4) (11). The expansion for u(x+h,y) is

$$u(x+h,y) = u + h\frac{\partial u}{\partial x} + \frac{h^2}{2!}\frac{\partial^2 u}{\partial x^2} + \frac{h^3}{3!}\frac{\partial^3 u}{\partial x^3} + O(h^4) \tag{3.5}$$

with all terms of the expansion evaluated at (x,y). The expansion for u(x-h,y) is similar

$$u(x-h,y) = u - h\frac{\partial u}{\partial x} + \frac{h^2}{2!}\frac{\partial^2 u}{\partial x^2} - \frac{h^3}{3!}\frac{\partial^3 u}{\partial x^3} + O(h^4) \ . \tag{3.6}$$

Adding the two together gives a difference equation for the second derivative in x that is centered about the point (x,y).

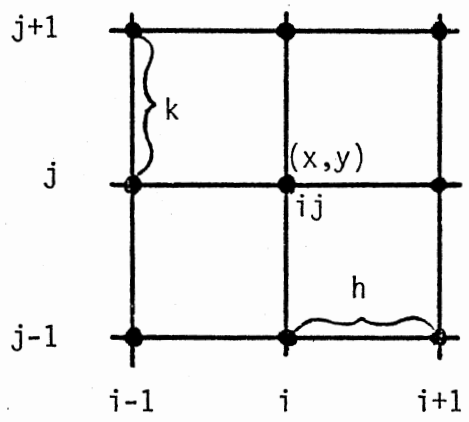


Figure 5. Cells From a Rectangular Grid

$$\frac{\partial^2 u}{\partial x^2}\Big|_{x,y} = \frac{1}{h^2}\left(u(x-h,y) - 2u(x,y) + u(x+h,y)\right) + O(h^2) \; . \qquad (3.7)$$

Terms with odd powers of h cancel.

Written in the notation of grid points, the expression becomes

$$\frac{\partial^2 u}{\partial x^2}\Big|_{i,j} = \frac{1}{h^2}\left(u_{i-1j} - 2u_{ij} + u_{i+1j}\right) + O(h^2) \; . \qquad (3.8)$$

The second derivative in y follows the same development

$$\frac{\partial^2 u}{\partial y^2}\Big|_{i,j} = \frac{1}{k^2}\left(u_{ij-1} - 2u_{ij} + u_{ij+1}\right) + O(k^2) \; . \qquad (3.9)$$

Dropping the terms $O(h^2)$ and $O(k^2)$, equations (3.8) and (3.9) become the finite difference approximations for the second derivatives of Laplace's equation. The deleted terms are the truncation error of the two approximations.

The most common choice for the cells of a grid is a square (h=k). Then, equations (3.8) and (3.9) combine conveniently to form the finite difference representation of the Laplacian

$$\nabla^2 u|_{ij} = \frac{1}{h^2}\left(u_{i-1j} + u_{i+1j} + u_{ij-1} + u_{ij+1} - 4u_{ij}\right) + O(h^2) \quad . \quad (3.10)$$

This is the important five-point Laplacian difference operator. Near a curved boundary, as in Figure 4, equations (3.8) through (3.10) will not suffice. The centered difference equations must be replaced with less accurate formulas that take into consideration the unequal spacing of the auxiliary points (4) (11). No problem will be considered here with curved boundaries.

Higher order approximation formulas (larger powers of h in the truncation term) can be obtained by including more grid points in the difference equations (11). A rotation of (3.10) to make use of the points of $\pi/4$, $3\pi/4$, ..., $7\pi/4$ in Figure (3.3) gives

$$\nabla_x^2 u|_{ij} = \frac{1}{2h^2}\left(u_{i+1\ j+1} + u_{i-1\ j+1} + u_{i+1\ j-1} + u_{i-1\ j-1} - 4u_{ij}\right) + O(h^2) \quad . \quad (3.11)$$

No apparent improvement in the order of the truncation is made.

A judicious combination of (3.10) and (3.11), however, can produce a difference equation with truncation $O(h^6)$ for Laplace's equation,

$$\nabla^2 u(x,y) = f(x,y) \quad (3.12)$$

when the solution is harmonic, and $O(h^2)$ in other circumstances. Weighting (3.10) and (3.11) properly causes truncation terms through

$O(h^5)$ to vanish (11). The result is the nine-point difference operator given by

$$\nabla_{(9)} = \frac{1}{3} \nabla_x^2 + \frac{2}{3} \nabla^2 \tag{3.13}$$

or

$$\nabla_{(9)} = \frac{1}{6h^2} (u_{i+1\ j+1} + u_{i-1\ j+1} + u_{i+1\ j-1} + u_{i-1\ j-1} + 4(u_{i-1j} + u_{i+1j} + u_{ij-1} + u_{ij+1}) - 20u_{ij}) . \tag{3.14}$$

Schematic representations of (3.10) and (3.14), computational molecules, are given in Figure 6.

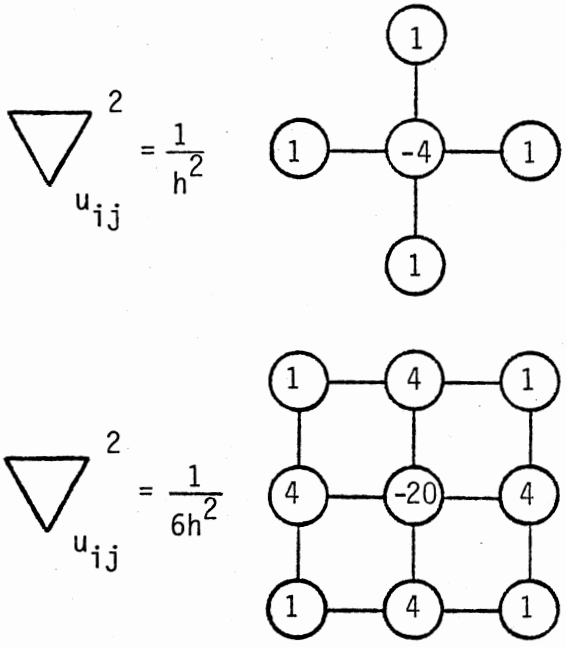


Figure 6. Computational Molecules for the Five- and Nine-Point Laplacians

The finite difference representations of other differential operators can be built up using the methods demonstrated here. Sources for other finite difference operators are (1) and (11).

Dirichlet and Neumann conditions must be considered in the formation of difference equations on or near the boundary of the solution domain. Figure 7 shows a boundary segment, AC, with Dirichlet conditions on BC and Neumann conditions on AB. Dirichlet conditions are easily dealt with; a grid point falling on BC is assigned a permanent value based on the function that specifies the Dirichlet conditions. The value may enter into other difference equations, but no different equation applies at its location.

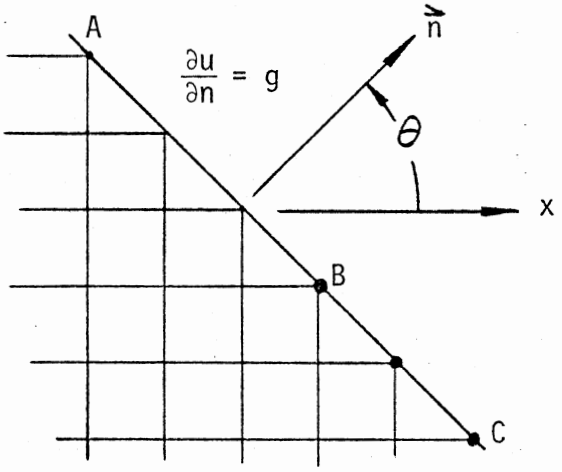


Figure 7. Neumann and Dirichlet Conditions

Treating Neumann conditions can be quite involved (4) (11). A difference equation is required where Neumann conditions hold, but points may be missing that are required for the centered differences of equations (3.10) and (3.14). In Figure 7 the normal derivative is given by

$$\frac{\partial u}{\partial n} = \frac{\partial u}{\partial x} \cos \theta + \frac{\partial u}{\partial y} \sin \theta = g \tag{3.15}$$

where θ is the angle between the normal vector and the x-axis. For a particular point the terms in θ and g can be computed and saved. Taylor series expansions involving adjacent points can be used to give the derivatives in x and y. Enough information can be derived from equation (3.15) to make up for missing points.

The problems considered here have two elements that simplify the treatment of Neumann points: g is everywhere zero and θ is either 0, $\pi/2$, π, or $3\pi/2$. Then, for Neumann points the requirements become

$$\frac{\partial u}{\partial x} = 0, \quad \theta = 0, \pi$$

or (3.16)

$$\frac{\partial u}{\partial y} = 0, \quad \theta = \pi/2, 3\pi/2$$

The Taylor series of equations (3.5) and (3.6) can be combined to form a centered difference approximation for the x-derivative of (3.16),

$$\frac{\partial u}{\partial x}\Big|_{ij} = \frac{1}{2h} (u_{i+1j} - u_{i-1j}) + O(h^2) \ . \tag{3.17}$$

When (3.17) is used at a Neumann point on the boundary one of the values on the right will correspond to a point that is not in the solution

domain. That does not prevent the determination of a value that will satisfy (3.17). Since the derivative is zero (3.17) becomes

$$u_{i+1j} = u_{i-1j} + O(h^3) \ . \tag{3.18}$$

For the y-derivative, a similar procedure gives

$$u_{ij+1} = u_{ij-1} + O(h^3) \ . \tag{3.19}$$

The boundary, then, serves to mirror the solution; the missing value is the same as the value on the opposite side of the boundary. With this information, difference equations can be constructed at Neumann points.

Suppose that $\theta = 0$ for some Neumann point at location (i,j). The point (i+1,j) will not be in the solution domain. Equation (3.10) will take the form

$$\nabla^2 u|_{ij} = \frac{1}{h^2} (2u_{i-1j} + u_{ij+1} + u_{ij-1} - 4u_{ij}) + O(h) \ . \tag{3.20}$$

The modification of the truncation term is discussed in (11).

Figure 8 gives the computational molecules of the difference equations that would be used for a problem with the Neumann conditions of equation (3.16) on all of the boundary.

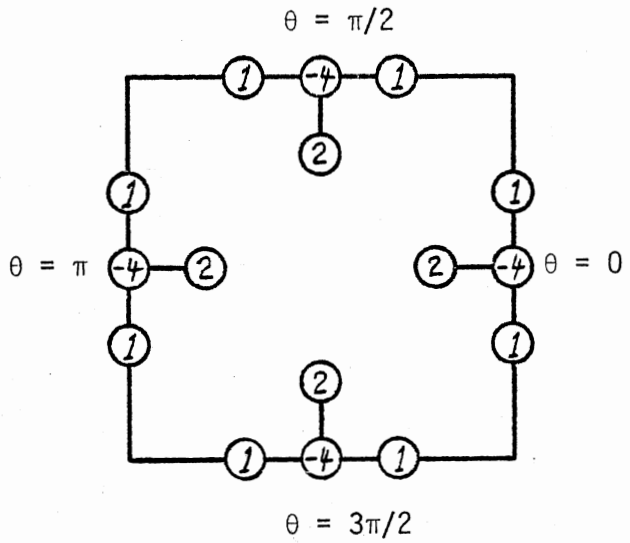


Figure 8. Computational Molecules for Neumann Conditions

The System of Difference Equations

The difference equations of the previous section provide a framework in which a discrete solution can be determined. To determine an approximate solution for Laplace's equation on domain R,

$$\nabla^2 u(x,y) = 0, \tag{3.21}$$

we can require that the five-point (or nine-point) Laplacian difference equation, (3.10), satisfy

$$\nabla^2 u_{ij} = 0$$

for each point of a uniform grid on R. The truncation term is dropped and the discrete solution satisfies

$$\frac{1}{h^2}\left(u_{i-1j} + u_{i+1j} + u_{ij+1} + u_{ij-1} - 4u_{ij}\right) = 0 \qquad (3.22)$$

for each grid point that is not a boundary point expressing Dirichlet conditions.

Consider the square domain of Figure (3.7), S, where Laplace's equation is to be solved. There are Dirichlet conditions on the boundary, and a uniform grid with spacing h = 1/3 is placed on S.

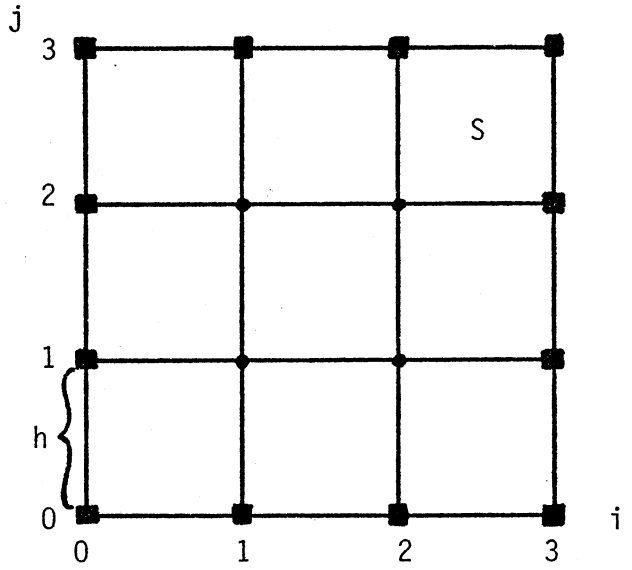


Figure 9. A Uniform Grid on Domain S

The solution is unknown for the four central points only; these will form the solution vector, $\vec{u}$. Writing equation (3.22) for each of the four points gives a system of four equations in four unknowns of the form

$$A\vec{u} = \vec{b}. \qquad (3.23)$$

The system, with the components of $\vec{u}$ ordered by rows, is

$$\begin{bmatrix} -4 & 1 & 1 & 0 \\ 1 & -4 & 0 & 1 \\ 1 & 0 & -4 & 1 \\ 0 & 1 & 1 & -4 \end{bmatrix} \begin{bmatrix} u_{11} \\ u_{21} \\ u_{12} \\ u_{22} \end{bmatrix} = \begin{bmatrix} -u_{01} - u_{10} \\ -u_{20} - u_{31} \\ -u_{02} - u_{13} \\ -u_{23} - u_{32} \end{bmatrix} \qquad (3.24)$$

The right-hand side of (3.24), $\vec{b}$, reflects the Dirichlet conditions of the problem. The coefficient matrix, A, reflects in its rows the relationship between a component of the solution vector and its immediate neighbors in the solution grid. Note that A is symmetric as a result of the ordering of the components of the solution vector. Other orderings also exist which will cause A to be symmetric.

Writing the system of difference equations for other problems is little different. Neumann conditions add components to the solution vector that require modified difference equations. The size of the coefficient matrix is very sensitive to the number or components in the solution vector.

If h is reduced to 1/32, the grid of Figure 9 would have 961 interior points. There would be 923,521 entries in A. Of these, 4,681 would be non-zero; 841 equations would have non-zero elements, 116 would have four non-zero elements, and four would have three non-zero elements.

Coefficient matrices tend to be highly sparse, that is, zeros predominate as coefficients. In computer implementations, the coefficient matrix is not stored at all, or the non-zero coefficients only are stored.

Solution Methods for the System of Difference Equations

The determination of the approximate solution of the PDE now depends on the solution of a linear system. A number of methods can be used and many computer implementations exist. There are two categories of methods: direct and iterative.

A direct method involves the algebraic manipulation of a linear system to determine its solution. Gaussian elimination is commonly used. Solving sparse systems directly requires complicated routines for storage management in computer implementations. Symmetry in the linear system or banding, the clustering of elements along diagonals, can be exploited to simplify the solution process. Systems resulting from the finite difference process with regular grids are often symmetric and are generally banded (11). The program package ITPACK2B contains routines written in FORTRAN for the direct solution of systems resulting from the discretization of elliptic problems (18).

An iterative method produces a vector sequence that converges to the solution of a problem as the number of iterations approaches infinity. An iterate is generally produced by multiplying the previous iterate by a matrix that is some function of the coefficient matrix of the linear system. An iterate deemed a sufficiently accurate solution estimate is selected after some number of iterations. Iterative methods for solving the sparse systems generated here require only enough storage for the solution vector and boundary values. The matrix mulitplications require numbers of additions and multiplications proportional to N rather than N^2 for a solution vector with N components. Sophisticated acceleration schemes can be applied to the iterative processes to

enhance their performance. Iterative methods will be considered exclusively, here.

Basic Iterative Solution Methods: The Jacobi and Gauss-Seidel Methods

In solving the linear system of order N,

$$A\vec{u} = \vec{b} \tag{3.23}$$

an iterative method produces the sequence

$$\vec{u}^0, \vec{u}^1, \ldots, \vec{u}^n, \ldots \; . \tag{3.25}$$

The vector $\vec{u}^0$ is some initial solution estimate. The limit of the sequence is $\vec{u}$. An iterate $\vec{u}^n$ is produced by a function of n, A, $\vec{b}$, and some number of the previous iterates; a method of order k involves the previous k iterates (4) (32):

$$\vec{u}^n = F(A, \vec{b}, n, \vec{u}^{n-1}, \ldots, \vec{u}^{n-k}). \tag{3.26}$$

The most common methods have no dependence on n (they are stationary), they are of order k=1, and they are linear in $\vec{u}^{n-1}$. For these methods, equation (3.26) becomes

$$\vec{u}^n = F(A, \vec{b}, \vec{u}^{(n-1)}). \tag{3.27}$$

A general linear, stationary iterative method can be written as

$$\vec{u}^n = G\,\vec{u}^{n-1} + \vec{r}^n \tag{3.28}$$

where G is a matrix that depends on A and $\vec{b}$. The vector $\vec{r}^n$ also depends

on A and $\vec{b}$. Using the relation $\vec{u} = A^{-1}\vec{b}$, equation (3.28) can be rewritten to give the error at each iteration,

$$\vec{e}^{n} = \vec{u} - \vec{u}^{n} = G\vec{e}^{n-1} \tag{3.29}$$

or

$$\vec{e}^{n} = G^{n}\vec{e}^{0}. \tag{3.30}$$

For the iterative scheme of (3.28) to converge to the solution, $\vec{u}$, we must have

$$\vec{e}^{n} \to 0 \quad \text{as } n \to \infty. \tag{3.31}$$

This implies

$$G^{n} \to 0 \quad \text{as } n \to \infty. \tag{3.32}$$

The condition (3.32) is crucial, and it must be predictable from the structure of G. It will be examined more closely in the following sections.

To describe the basic iterative methods it is convenient to partition A. The three matrices L, D, and U formed from the elements of A below the diagonal, on the diagonal, and above the diagonal, respectively, with zeros elsewhere give the partitioned representation L + D + U. It is also convenient to adopt a single subscript in referring to the components of the solution vector, numbering them from 1 to N.

For the Jacobi (J) method, each component of $\vec{u}^{n-1}$ is allowed a displacement in order to satisfy one of the equations of the linear system. The displaced components make up $\vec{u}^{n}$; they are given by

$$u_j^n = a_{jj}^{-1}\left(\sum_{k\neq j} a_{jk}\, u_k^{n-1} + b_j\right) \tag{3.33}$$

where a_{ij} is an element of A. The desired effect of the displacements is an iterate that better approximates the solution, $\vec{u}$. Equation (3.33) can be rewritten in matrix form as (32):

$$\vec{u}^n = D^{-1}(D - A)\vec{u}^{n-1} + D^{-1}\vec{b} \tag{3.34}$$

The requirement that D^{-1} exist is satisfied since the diagonal elements of A are all non-zero (from equation (3.22)). Although the J method is not used as a solution method, its behavior is the basis for the selection of parameters for more sophisticated methods.

The Gauss-Seidel (GS) method uses a strategy similar to that of the J method. As soon as a component of $\vec{u}^n$ is determined, though, it is used to help determine the remaining components of $\vec{u}^n$. Component j is given by

$$u_j^n = a_{jj}^{-1}\left(\sum_{k=1}^{j-1} a_{jk}\, u_k^n + \sum_{k=j+1}^{N} a_{jk}\, u_k^{n-1} + b_j\right) \tag{3.35}$$

In matrix form this becomes

$$\vec{u}^n = KU\vec{u}^{n-1} + KD^{-1}\vec{b} \tag{3.36}$$

where

$$K = (I - L)^{-1}$$

The matrix I is the identity matrix. The matrix (I - L) is non-singular since all the diagonal entries are non-zero and the matrix is lower triangular (32).

There is one important difference between the J and GS methods to note here. The order of the difference equations in the system $A\vec{u} = \vec{b}$ is immaterial for the J method; the sequence of values for a particular grid location will be the same whether $A\vec{u} = \vec{b}$ or a shuffled version of the system is used. This same statement can not be made for the GS method; the sequence of values at a particular location will change as the order of the difference equations changes.

The successive overrelaxation (SOR) method involves a slight modification of the GS method. Each displacement of a component of $\vec{u}^{n-1}$ is weighted before it is used to produce $\vec{u}^{n}$. The weighting factor or overrelaxation parameter, ω, is in the range

$$1 \leq \omega < 2 . \tag{3.37}$$

Defining u_j^n to be the result of (3.35), the basic operation of the GS method, the components of $\vec{u}^{n}$ are given by

$$u_j^n = u_j^{n-1} + \omega(\bar{u}_j^n - u_j^{n-1}) . \tag{3.38}$$

For $\omega = 1$, the SOR method reduces to the GS method. The matrix representation of SOR is

$$\vec{u}^{n} = M(I - \omega(U - I))\vec{u}^{n-1} + MD^{-1}\vec{b} \tag{3.39}$$

where

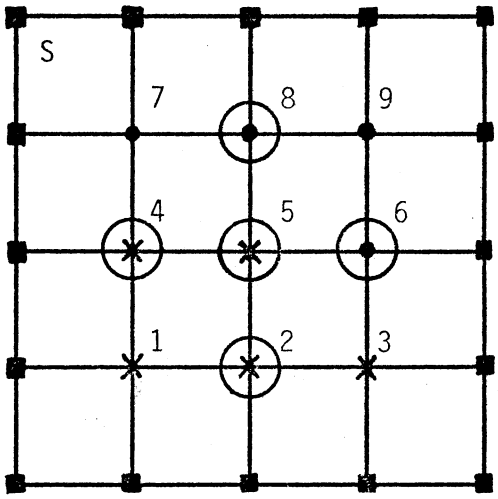
$$M = (I - \omega L)^{-1}.$$

SOR shares the sensitivity of the GS method to the order of the components of the solution vector (32). For an optimal choice of ω, SOR is far superior to the J and GS methods for solving the system of

(3.23). The choice of the overrelaxation parameter is discussed in Chapter IV.

Implementation of the Basic Iterative Methods

Figure 10 shows a uniform grid on domain S; there are Dirichlet conditions on the boundary. The solution methods of the previous section can be easily implemented for such a problem. As noted earlier, the two-dimensional array of FORTRAN can be used to store the values of solution components and boundary values and maintain the organization imposed by the grid.



• - old component
x - new component
- boundary value

Figure 10. A Gauss-Seidel or SOR Iteration

Let the array contain the components of $\underset{\sim}{u}^{n-1}$. For component j, designate the values at the four nearest grid points u_A, u_B, u_C, and u_D; as many as two of the values can be boundary values.

The equations that govern the J and GS method, (3.33) and (3.35), can be written explicitly from the difference equation (3.21). For the J method a component of $\underset{\sim}{u}^{n}$ is given by

$$u_j^n = \frac{1}{4}(u_A + u_B + u_C + u_D). \tag{3.40}$$

There is no place to put the new component in the single array so a duplicate array is needed. An iteration of the J method involves performing (3.40) for all values of j. The uses of the two arrays interchange for the next iteration.

For the GS method, components of $\underset{\sim}{u}^{n}$ replace components of $\underset{\sim}{u}^{n-1}$ as they become available; old values of u_A, ..., u_D are replaced with new ones. It is natural, then, to use a single array and put the new values of components in place of the old values. During the course of a GS iteration parts of the old and new iterate will be contained in storage. The GS method requires the choice of some order for the components. The results of an iteration are not independent. Figure 10 shows a GS iteration where the order is by rows. Modifying a GS iteration according to (3.38) gives an SOR iteration.

A varient of SOR is the symmetric successive overrelaxation method (SSOR). An iteration consists of an SOR sweep of the solution components in order followed by an SOR sweep of the components in reverse order. The primary advantage of SSOR is that its iteration matrix is symmetric (32).

Convergence of the Basic Iterative Methods

As noted earlier, the behavior of G^n for increasing n determines the convergence of the error sequence, $\vec{e}^n$. Some idea of the convergence characteristics of the J, GS, and SOR methods is needed before proceeding.

The three iteration matrices, G_j, G_g, and G_s for the J, GS, and SOR methods, respectively, each have performances determined by their eigenvalue spectra (32). The eigenvalue spectrum of a square matrix B of order N consists of the set of all Λ_i such that

$$B\vec{V}_i = \Lambda_i \vec{V}_i \tag{3.41}$$

is satisfied; the $\vec{V}_i$ are eigenvectors of B. There are N eigenvalues of B; some may be complex and a single value may be repeated.

The spectral radius of B, S(B), is defined by

$$S(B) = \max_i |\Lambda_i| \ . \tag{3.42}$$

The eigenvalue corresponding to S(B) is the dominant eigenvalue of B. For large N, it is difficult to determine the eigenvalue spectrum of B; S(B) can be determined indirectly, though.

A necessary and sufficient condition for $G^n \to 0$ (also $\vec{e}^n \to 0$) as $n \to \infty$ is $S(G) < 1$ (32). Thus, the spectral radii of G_j, G_g, and G_s must be less than one for the J, GS, and SOR methods to converge for a particular problem.

A norm is a useful measure of the convergence of a vector sequence. For some norm a, $\| \cdot \|_a$, a vector sequence $\vec{u}^n$ is convergent if and only if

$$\| \vec{u} - \vec{u}^n \|_a \to 0 \text{ as } n \to \infty. \tag{3.43}$$

A norm that is easily computed is

$$\|\vec{u}\| = \left(\sum_{i=1}^{N} |u_i|\right)/N . \tag{3.44}$$

Then, $\|\vec{e}^n\|_a \to 0$ as n increases (32). The improvement in the solution estimate made by a single iteration can be gauged by the ratio

$$\|\vec{e}^n\| / \|\vec{e}^{n-1}\| . \tag{3.45}$$

This ratio approaches the spectral radius of the iteration matrix, S(G) as n increases. It often serves as an estimate of S(G) (4) (32). Young (32) uses a measure he calls the assymptotic rate of convergence,

$$R(G) = - \log S(G), \tag{3.46}$$

which can be estimated using (3.45).

Only in artificial situations is $\vec{e}^n$ available during the computation of $\vec{u}$, hence, the ratio (3.45) must be determined indirectly. The displacement vector, $\vec{d}^n$, given by

$$\vec{d}^n = \vec{u}^n - \vec{u}^{n-1} \tag{3.47}$$

should approach zero as $\vec{u}^n$ approaches $\vec{u}$. An operation common to the J, GS, and SOR methods is that of equation (3.40). A residual vector, $\vec{r}^n$, whose components are given by

$$r_j^n = u_A + u_B + u_C + u_D - 4u_j^{n-1} \tag{3.48}$$

is another displacement measure. It, also, should approach zero as $\vec{u}^n$ converges. It can be shown (4) that the displacement and residual vectors have behavior similar to that of the error vector. The sequence of residual vectors is given by

$$\vec{r}^{\,n} = (AGA^{-1})^n \vec{r}^{\,n-1} \quad . \tag{3.49}$$

Either of the two ratios, $\|\vec{d}^{\,n}\| / \|\vec{d}^{\,n-1}\|$ or $\|\vec{r}^{\,n}\| / \|\vec{r}^{\,n-1}\|$, can be used in place of (3.45) to determine S(G).

The requirement for the convergence of a general iterative method has been described, $S(G) < 1$. No general criteria have been offered to determine which systems of difference equations will produce iteration matrices that fulfill this requirement. The coefficient matrix, A, that results when the five-point Laplacian difference equation is written for a uniform grid has properties that guarantee that S(G) will be less than one for the J, GS, and SOR methods. First, A has weak diagonal dominance, that is,

$$|a_{ii}| \geq \sum_{j \neq 1} |a_{ij}| \quad , \; i = 1, 2, \ldots, N, \tag{3.50}$$

and

$$|a_{ii}| > \Sigma \, |a_{ij}|$$

for at least one i. This is assured by the form of the Laplacian equation (3.22). The diagonal entry of each row is -4, and the sum of the remaining elements is never more than 4. Second, A is irreducible. This follows from the grid structure; the components of the solution vector are all interrelated. No subsystem of $A\vec{u} = \vec{b}$ can be solved independently. With these two properties for A it follows that S(G) is less than one for the three methods (4) (11) (32).

Young (32) has described an important property found among many of the coefficient matrices that results when the discrete problem is formed. It has been mentioned that the GS and SOR methods produce sequences of iterates that are different for the different orderings of the difference

equations that make up the linear system. Young's observations limit the consideration that must be given to the order of the difference equations.

The Property (A) that Young describes assures that for each "consistent" ordering of the coefficient matrix the eigen values of the iteration matrix, G_g or G_{sor}, will be the same. This means that every consistently ordered system for a particular problem will have the same convergence characteristics when the GS method is used. The same will be true when SOR is used.

The matrix A has Property (A) if and only if there exists a permutation matrix P such that $P^{-1}AP$ has the form

$$P^{-1}AP = \begin{bmatrix} D_1 & H \\ K & D_2 \end{bmatrix} \tag{3.51}$$

where D_1 and D_2 are square diagonal matrices and H and K are rectangular matrices. Consistent orderings can be formed for any matrix having Property (A) (32). If A is symmetric and has Property (A) then the eigenvalues of G_j are real and occur in pairs (positive and negative values) (4). Also, the eigenvalues of the SSOR iteration matrix are real.

With the assurance that the spectra of G_g and G_s are invariant, relations can be formed between the eigenvalues of G_j, G_g, and G_s. For an eigenvalue μ of G_j there is a corresponding eigenvalue Λ of G_s. The two are related by

$$(\Lambda + \omega - 1)^2 = \omega\mu^2 \Lambda \tag{3.52}$$

where ω is the overrelaxation parameter (4) (11) (32). For $\omega = 1$, the GS method, the eigenvalues are related by

$$\Lambda = \mu^2 . \tag{3.53}$$

Equations (3.52) and (3.53) can serve to relate spectral radii. The implication of (3.53) is that the GS method converges at twice the rate of the J method (from (3.46)).

Assuming that S(G) is known for the J or GS method for a particular problem, then the optimum value of ω, ω_b, can be determined. It is chosen to minimize $S(G_s)$. In terms of $S(G_j)$, ω_b is given by (4) (11)

$$\omega_b = 2(1 + \sqrt{1 - S(G_j)^2}\ )^{-1} . \tag{3.54}$$

If the value of $S(G_j)$ is given for a problem, the relative effectiveness of the J, GS, and SOR methods can be demonstrated. If $S(G_j)$ is .980, $S(G_g)$ will be .960, and $S(G_s)$ is .668. Comparing the assymptotic convergence rates of convergence, the GS method is 2.02 times as fast as the J method, and SOR is 20.0 times as fast.

Discussion of the determination of ω_b for SOR will be given in the next chapter.

CHAPTER IV

ACCELERATED ITERATIVE SOLUTION METHODS

Techniques for accelerating the convergence of the basic iterative solutions have been developed over the last 30 years. These techniques include methods to reduce the value of S(G), and the use of extrapolation in order to skip unnecessary iterations. The methods discussed here include SOR, symmetric SOR, vector Aitken acceleration, and the multigrid method.

Behavior of the Eigenvalues of the SOR Iteration Matrix

Because SOR with the optimum value of ω is the basic iteration for a number of accelerated methods, much study has gone into the determination of that value. Although there are relations between the eigenvalues of the SOR, GS, and J iteration matrices and ω_b, those eigenvalues are, in practice, difficult to obtain. Any error in determining the dominant eigenvalue for the J or GS matrices, and consequently in ω_b, can seriously degrade the performance of SOR.

Solving the equation relating eigenvalues,

$$(\Lambda + \omega - 1)^2 = \omega^2 \mu^2 \Lambda \ , \tag{3.52}$$

for Λ, the dominant eigenvalue of G_s, in terms of ω, the dominant eigenvalue of G_j, gives

$$\Lambda = \frac{1}{4}\,(\omega\mu + (\omega^2\mu^2 - 4(\omega - 1)^{\frac{1}{2}})^2 \qquad (4.1)$$

for $1 \leq \omega \leq \omega_b$ (4). For $\omega_b \leq \omega < 2$, Λ is given by

$$\Lambda = \omega - 1. \qquad (4.2)$$

The eigenvalues of G_s lie along the positive real axis for $\omega = 1$ (the GS method). As ω increases, eigenvalues migrate onto a circle with radius $\omega - 1$ centered about the origin in the complex domain. When $\omega = \omega_b$, all eigenvalues lie on the circle. Equation (4.1) is no longer appropriate since the circle continues to expand. For $\omega > \omega_b$ all the eigenvalues are complex and lie on the circle (11) (14).

A graph of $S(G_2)$ versus ω using (4.1) and (4.2) illustrates the sensitivity of SOR to the choice of ω. Figure 11 shows the relation for $S(G_j) = \mu = .981$. The line AB, a reflection of BC, is useful for evaluating the effects of errors in an estimate of ω_b ; ω_a and ω_c will correspond to the points A and C. Clearly, an estimate that falls in the interval $[\omega_a, \omega_b]$ will cause a larger value of $S(G_s)$ than an estimate falling in $[\omega_b, \omega_c]$. Gross errors should result in underestimates due to the steeper slope of BC as opposed to the curve on $[1, \omega_a]$. Graphs of $S(G_s)$ in (8) indicate that the interval $[\omega_a, \omega_c]$ becomes shorter as $S(G_j)$ approaches 1 making the choice of ω_b more critical.

Numerical Determination of the Optimum Overrelaxation Parameter

Determining the dominant eigenvalue of either G_j or G_g accurately is of the first importance in determining ω_b. There are two environments in which ω_b must be determined: a problem is solved repeatedly

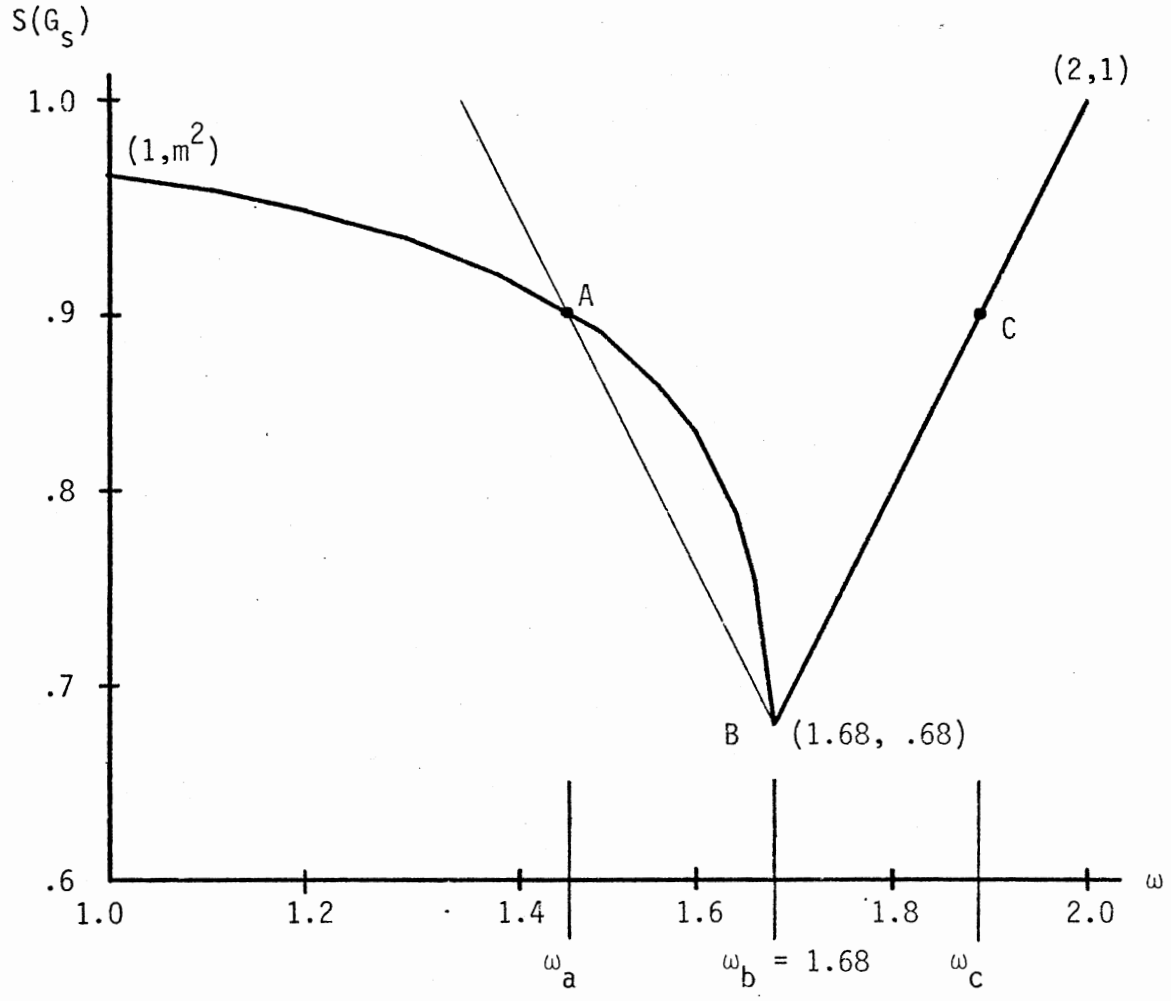


Figure 11. The Dependence of the Spectral Radius for SOR on ω ($S(G_j) = .981$)

with no change in the iteration matrix; a range of unrelated problems is encountered.

In the first case an exhaustive effort to determine ω_b might be justified. In the second, a solution produced with a poor choice of ω_b might be less costly than a modest effort to determine ω_b.

All methods employ one of the vector sequences produced by the iteration scheme. Here, the residual vector defined in Chapter III is used. The most important characteristic of the sequence is

$$P_n = \|\vec{r}^n\| / \|\vec{r}^{n-1}\| \to S(G), \text{ as } n \to \infty \tag{4.3}$$

for iteration matrix G.

Most of the techniques discussed here are variations on the power method (17). Assume that matrix B (order N) has eigenvalues Λ_i and eigenvectors $\vec{x}_i$; the eigenvalues are ordered so that Λ_1 is dominant and Λ_N is smallest.

Multiplying an arbitrary vector $\vec{v}^0$ by B k times gives

$$\vec{v}^k = B^k\vec{v}^0 = \Lambda_1^k c_1\vec{x}_1 + \Lambda_2^k c_2\vec{x}_2 + \ldots + \Lambda_N^k c_N\vec{x}_N \tag{4.4}$$

where the c_i are constants.

If $|\Lambda_1|$ is greater than $|\Lambda_2|$ and k is sufficiently large, then (4.4) can be reduced to

$$v^k = B^k v^0 \cong \Lambda_1^k c_1\vec{x}_1. \tag{4.5}$$

Some vector norm can be used to extract Λ_1,

$$\|\vec{v}^k\| / \|\vec{v}^{k-1}\| = \Lambda_1. \tag{4.6}$$

Therefore, the use of (4.3) to determine S(G) is equivalent to the power method with the iteration matrix replacing B.

A hindrance in the use of (4.3) is the fact the two eigenvalues with the largest magnitude for G_j and G_g (Λ_1 and Λ_2 in (4.4)) have magnitudes that differ only slightly (8). Then, many iterations are required before (4.4) applies. The sequence P_n in (4.2) is slowly convergent, but a process like Aitken extrapolation can be used to skip many iterations and resolve the dominant eigenvalue (8). Using the ratios of (4.2), an Aitken extrapolation is given by (2)

$$P = P_n - \frac{(P_n - P_{n-1})^2}{(P_n - 2P_{n-1} + P_{n-2})} \quad . \tag{4.6}$$

The extrapolated value, P, can be used as an estimate of S(G). Acceleration through the use of Chebyshev polynomials is also possible for the power method (13).

A simple method for determining ω_b requires 15 to 20 SOR iterations with ω_b set to one (a GS iteration); equation (4.6) is used with ratios from the last three iterations to determine $S(G_g)$. Then, with results from the previous chapter, ω_b is given by

$$\omega_b = 2/(1 + \sqrt{1 - S(G_g)}) \, . \tag{4.7}$$

Carré (8) presents an iterative method for determining ω_b that operates while SOR iterations are conducted. A value ω_0 is used as an initial estimate of ω_b. Ten to 15 iterations are allowed, and (4.6) is used to determine Λ, the dominant eigenvalue of G_s corresponding to ω_0. Then, a combination of (3.52) and (3.53) is used,

$$\omega_i = 2/(1 + \sqrt{1 - (\Lambda + \omega_{i-1} - 1)^2/(\Lambda\omega_{i-1}^2)} \quad) . \tag{4.8}$$

The new estimate of ω_b, ω_i, is used for several iterations, and (4.6) and (4.8) are used again. As the method proceeds ω_i will approach ω_b

and may exceed it. Carré includes a mechanism that assures that ω_b will not be exceeded.

The method involving equations (4.6) and (4.7) was used here to determine ω_b. This was done once for each of the iteration matrices used. One hundred iterations were usually sufficient to determine ω_b to four digits.

Acceleration of Convergent Vector Sequences

A number of methods exist for speeding the convergence of the vector sequences produced by iterative methods such as SOR. These use combinations of several iterates to produce an improved solution, skipping many iterations in the process. Some of the most recent methods include Chebyshev semi-iterative techniques (12) (14) (32) and conjugate gradient techniques (14) (17) (22). These are typically used with SOR or SSOR iterations (33).

The method used here is vector Aitken acceleration adapted to vector sequences. In a 1937 paper, Aitken presents his δ^2-process for accelerating the determination eigenvalues and eigenvectors (2). The extrapolation of equation (4.6) is applied to the eigenvalue sequence and to corresponding components of the eigenvector sequence. An assumption is made that each component of the eigenvector sequence is convergent. This may not be true for the initial iterations and following extrapolations. Then, applying (4.6) may produce absurd values for some components of the new iterate. A modified vector Aitken method is due to Jennings (16) (17). An extrapolation factor, s, is produced that is a composite of the tendencies of the individual components. For a vector sequence $\vec{P}^n$, the improved iterate is given by

$$\tilde{P} = \tilde{P}^n + s(\tilde{P}^n - \tilde{P}^{n-1}) \tag{4.9}$$

where

$$s = \frac{(\tilde{P}^{n-2} + \tilde{P}^{n-1})^T (\tilde{P}^{n-1} - \tilde{P}^n)}{(\tilde{P}^{n-2} + \tilde{P}^{n-1})^T (\tilde{P}^{n-2} - 2\tilde{P}^{n-1} + \tilde{P}^n)} \tag{4.10}$$

Two additional constraints can limit the effects that fluctuations in a few component produce: upper and lower limits can be placed on s to prevent absurd extrapolations, and extrapolation can be postponed until the last two iterates are closely aligned. The angle θ between the last two iterates can be determined from

$$(\tilde{P}^n)^T \tilde{P}^{n-1} = q_n \, q_{n-1} \cos \theta \tag{4.11}$$

where q_n and q_{n-1} are the lengths of $\tilde{P}^n$ and $\tilde{P}^{n-1}$, respectively. If cos θ is close to one, then an extrapolation can be allowed. The routine VAITK written by Dr. J. P. Chandler of the Computing and Information Sciences Department of Oklahoma State University uses Jenning's method and employs the constraints mentioned above. It is included in Appendix A as part of the program CORNER.

The modified Aitken acceleration can be used for sequences generated by matrices with real eigenvalues. Then, the J, GS, and SSOR methods are candidates for Aitken acceleration.

The Multigrid Method

The multigrid method is a recent development in finite difference methods due to Brandt (5) (6) (7). The performance of multigrid methods exceeds that of the most recent accelerated iterative methods. The method used here is the Cycle-C algorithm that appears in (6). It is intended for elliptic problems.

A practice that speeds the convergence of iterative methods is the use of a collection of n grids with spacing h, 2h, 4h, ..., $2^{n-1}h$. A solution is obtained on each grid to prescribed accuracy, and interpolation is used to fill in values missing in the next grid. The solution is eventually governed by the rate of convergence on the fine grid, but most of the initial work required to build up a solution is bypassed. An iteration on a coarse grid requires roughly one fourth the effort required on the next finest grid.

This process is inverted in the Cycle-C algorithm: iterations begin on the solution grid and then proceed to coarser grids. The multigrid philosophy encompasses the range of errors that occurs in a problem: there are error components that have short wavelengths (on the order of h) and there are components that span the solution domain. The short wavelength components represent local roughness of the solution; the longer components result from the slow propagation of boundary values into the solution domain.

On the finest grid, error components with wavelength 2h or less can be damped effectively by GS iterations. The magnitudes of the components can be reduced by an order of magnitude with just a few iterations. Due to the limited range of the five-point Laplacian difference equation, longer components are essentially unaffected. If coarser grids are used, though, GS iterations can damp the longer components.

Rather than computing the solution on coarse grids, corrections are calculated. If the system to be solved is

$$A\vec{u} = \vec{b}, \tag{4.10}$$

the GS method will produce the sequence $\vec{u}^n$. Defining the residual vector $\vec{r}^n$,

$$\vec{r}^n = \vec{u}^n - \vec{u}^{n-1} \tag{4.11}$$

the vector $\vec{u}^n$ satisfies

$$A\vec{u}^n = \vec{b} - \vec{r}^n \ . \tag{4.12}$$

Combining equations (4.10) gives

$$A(\vec{u} - \vec{u}^n) = \vec{r}^n \tag{4.13}$$

Defining the correction $\vec{v}^n$,

$$\vec{v}^n = \vec{u} - \vec{u}^n \tag{4.14}$$

(4.13) becomes

$$A\vec{v}^n = \vec{r}^n. \tag{4.15}$$

Assume that there is a system of grids numbered 1 though k with spacing h, 2h, ..., 2^{k-1}h. If the short wavelength components are smoothed on grid 1, the residual will be free of that error and the system (4.15) can be solved economically on grid 2. The same steps can be taken in solving (4.15) on grid 2: short wavelength components are damped and an equation similar to (4.15) is formed and solved on grid 3. The process can be extended through grid k. Whenever a solution on a coarse grid meets some convergence criterion it is interpolated and added to the next finest grid as a correction.

It becomes necessary to define solution vectors and residual vectors for each grid. For grid k the residual and solution vectors are $\vec{r}_k^n$ and $\vec{v}_k^n$.

Figure 12 shows the behavior of the residual vector for several GS iterations on grid k. The initial steep slope reflects the reduction of short wavelength error components. The initial rapid progress towards a solution abates as long wavelength error components begin to dominate. A parameter η can be used to govern the jump to a coarser grid. If two successive residuals obey

$$\| \vec{r}_k^{\,n} \| / \| \vec{r}_k^{\,n-1} \| < \eta \tag{4.16}$$

then the system of (4.15) should be formed and solved on grid k+1. If the residual norm on grid k becomes less than some fraction δ of the last residual norm on grid k-1, the correction of grid k can be interpolated and added to the solution of grid k-1. Typical values for η and δ are .7 and .2, respectively.

Figure 13 shows the interactions of a three level grid system. The right-hand side of the system (4.15) must be stored; it accounts for the duplicate grids at each level. The difference equation (3.22) becomes

$$\frac{1}{h^2} (u_{i\ -1j} + u_{i+1j} + u_{ij+1} + u_{ij-1} - 4u_{ij}) = r_{ij} \tag{4.17}$$

The extra grid on level 1 can be employed for solutions of Poisson's equation, (2.9). The term r_{ij} in (4.17) is replaced by $h^2 f_{ij}$.

A Cycle-C multigrid routine, VBOUND, is included in Appendix B. It can be used to solve Laplace's equation or Poisson's equation on a region with irregular boundaries; a uniform square mesh approximates the region.

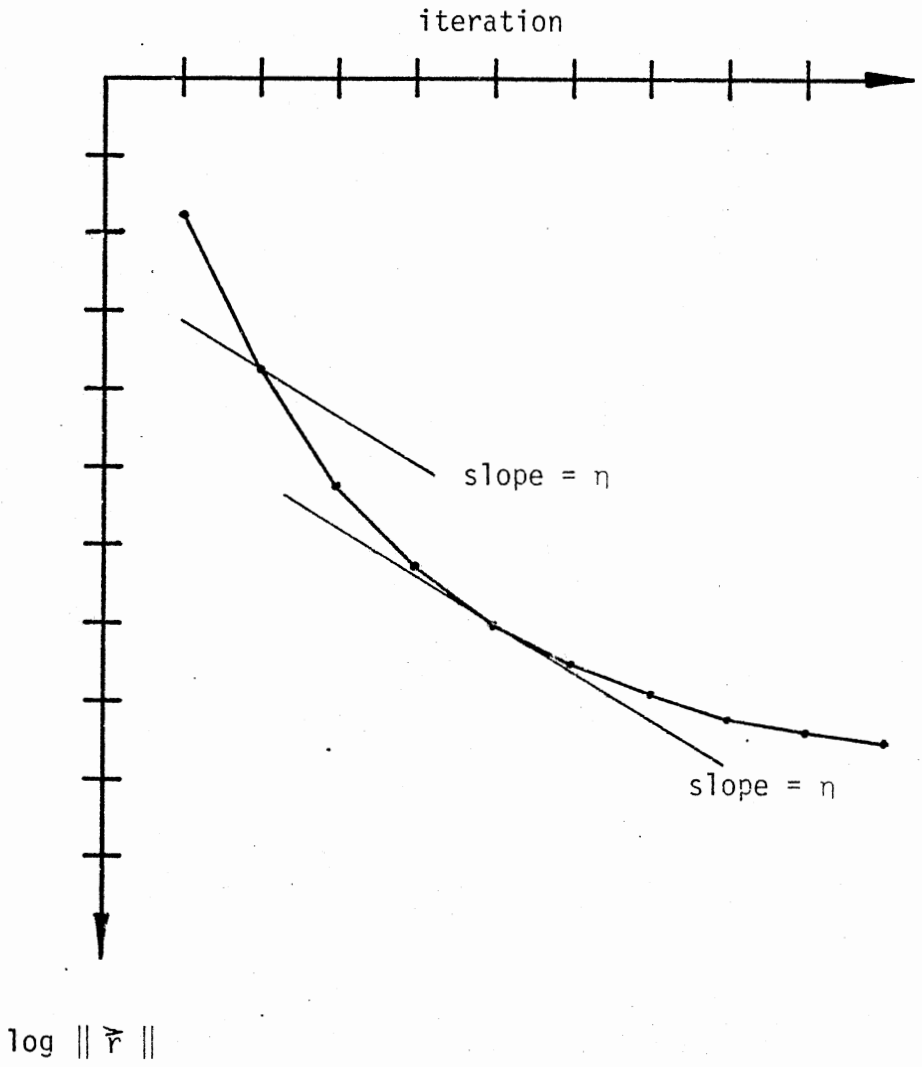
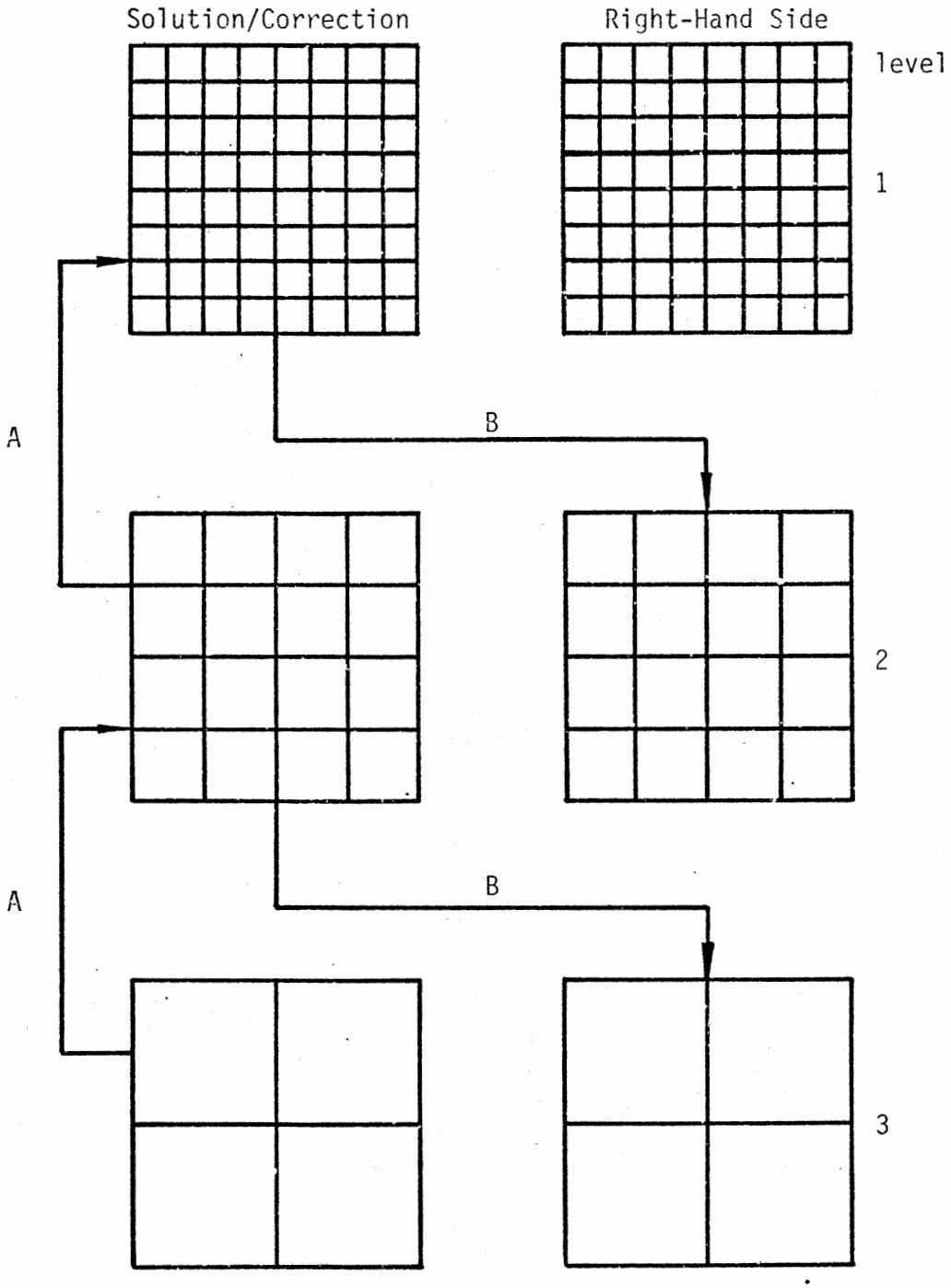


Figure 12. Behavior of the Residual Vector for Gauss-Seidel Iterations



A - Interpolation and Addition of Corrections
B - Calculation of residuals

Figure 13. The Interactions of a Three Grid System in a Multigrid Solution

CHAPTER V

RE-ENTRANT CORNERS IN ELLIPTIC PARTIAL DIFFERENTIAL EQUATIONS

Determining the Form of the Solution

For the general elliptic equation, (2.1), posed on some domain S,

$$au_{xx} + bu_{xy} + cu_{yy} = f \tag{2.1}$$

the solution has a singularity wherever the coefficients a, b, c, and f are singular or discontinuous, and where boundary features cause derivatives of the solution to be undefined (4). A common singularity arises in elliptic problems when the boundary makes a discontinuous change of direction that forms an internal angle of greater than π radians. This feature is called a re-entrant corner. In Figure 14, the boundary of domain S changes direction at O, the origin of a rectangular coordinate system. The internal angle formed by the segments AO and OB, α, is greater than π.

Re-entrant corners cause finite difference methods to produce results that contain error greater than that which can be attributed to discretization. It is useful in studying re-entrant corners to determine the functional dependences of a solution in its vicinity given some idealized problem. Laplace's equation affords a representative study of re-entrant corners in elliptic problems and their effect on finite difference methods.

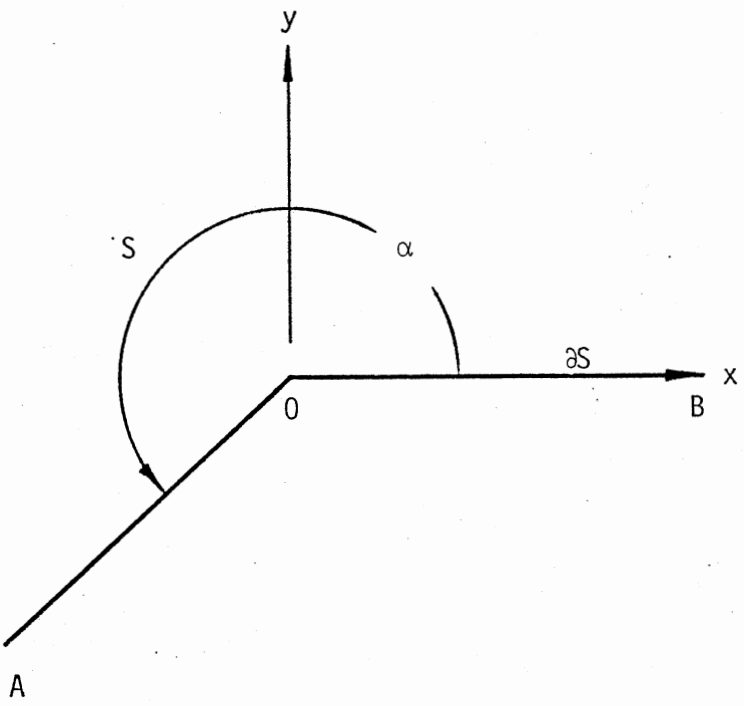


Figure 14. A Re-entrant Corner

Writing Laplace's equation in polar coordinates simplifies the analysis of the solution near the re-entrant corner of Figure 14 (15). It becomes

$$\left(\frac{1}{r} \frac{\partial}{\partial r} + \frac{\partial^2}{\partial r^2} + \frac{1}{r^2} \frac{\partial^2}{\partial \theta^2} \right) u(r_1 \theta) = 0 \ . \tag{5.1}$$

Assuming that u is made up of separate functions in r and θ,

$$u(r,\theta) = R(r)\ T(\theta), \tag{5.2}$$

equation (5.1) reduces to two related second order ordinary differential equations (15):

$$\frac{r}{R(r)} \left[\frac{dR(r)}{dr} + r \frac{d^2R(r)}{dr^2} \right] = c^2 \tag{5.3}$$

and

$$\frac{1}{T(\theta)} \frac{d^2T(\theta)}{d\theta^2} = -c^2 \tag{5.4}$$

General solutions are

$$R(r) = ar^c + br^{-c} \tag{5.5}$$

and

$$T(\theta) = A \cos (c\theta) + B \sin (c\theta) . \tag{5.6}$$

Since $u(r,\theta)$ must be defined at the corner ($r = 0$), we can choose to have $c > 0$ and $b = 0$. Then (5.5) becomes

$$R(r) = ar^c . \tag{5.7}$$

Particular solutions can be determined from the conditions on $u(r,\theta)$ on the boundary of S, ∂S. If u is zero on ∂S (Dirichlet conditions), then requiring that $T(\theta)$ vanish there satisfies that condition. Setting A to zero and letting c have the values

$$c = 0, \pi/\alpha, 2\pi/\alpha, 3\pi/\alpha, \ldots \tag{5.8}$$

causes T to vanish on ∂S. The solution becomes

$$u(r,\theta) = \sum_{n=1}^{\infty} b_n r^{n\pi/\alpha} \sin (n\pi\theta/\alpha) . \tag{5.9}$$

A constant can be added to (5.9) if u is to have some constant value on ∂S.

For Neumann conditions, the normal derivative on ∂S is $\partial u/\partial\theta$. If the normal derivative is to vanish on ∂S, then setting B to zero and

allowing c to have the values in (5.8) will satisfy that requirement. The solution becomes

$$u(r,\theta) = \sum_{n=1}^{\infty} a_n r^{n\pi/\alpha} \cos(n\pi\theta/\alpha) . \qquad (5.10)$$

For each of the solutions, the exponent of the first term in r is π/α. Taking the first derivative with respect to r as in (5.1) the exponent becomes $\pi/\alpha - 1$. If α is greater than π then both derivatives in r in (5.10) are undefined at the corner 0 ($r = 0$).

Implicit in the development of the finite difference methods of Chapter III is the requirement that derivatives of the solution through a specified order exist and that they be continuous over all of the solution domain. For the development of the five-point approximation of the Laplacian operator, equation (3.10), the solution must be three times continuously differentiable (11). Since the solutions given by (5.9) and (5.10) are not differentiable at the re-entrant corner, the five-point approximation is invalid there. Its use at the corner introduces error in excess of the expected $0(h^2)$ discretization error (4) (29).

The L-Shaped Region Problem

The effects of a re-entrant corner on the solution of Laplace's equation by finite difference methods can be studied conveniently for the region S of Figure 15. The coordinates of the vertices, A through F, are (-1,1), (1,1), (0,1), (0,0), (0,-1), and (-1,-1), respectively. All of the internal angles are either $\pi/2$ or $3\pi/2$. The problem requires a solution of Laplace's equation on S subject to mixed boundary conditions. The solution has the value -1 on EF and +1 on BC. On AB, AF, CD, and DE, the normal derivative must be zero.

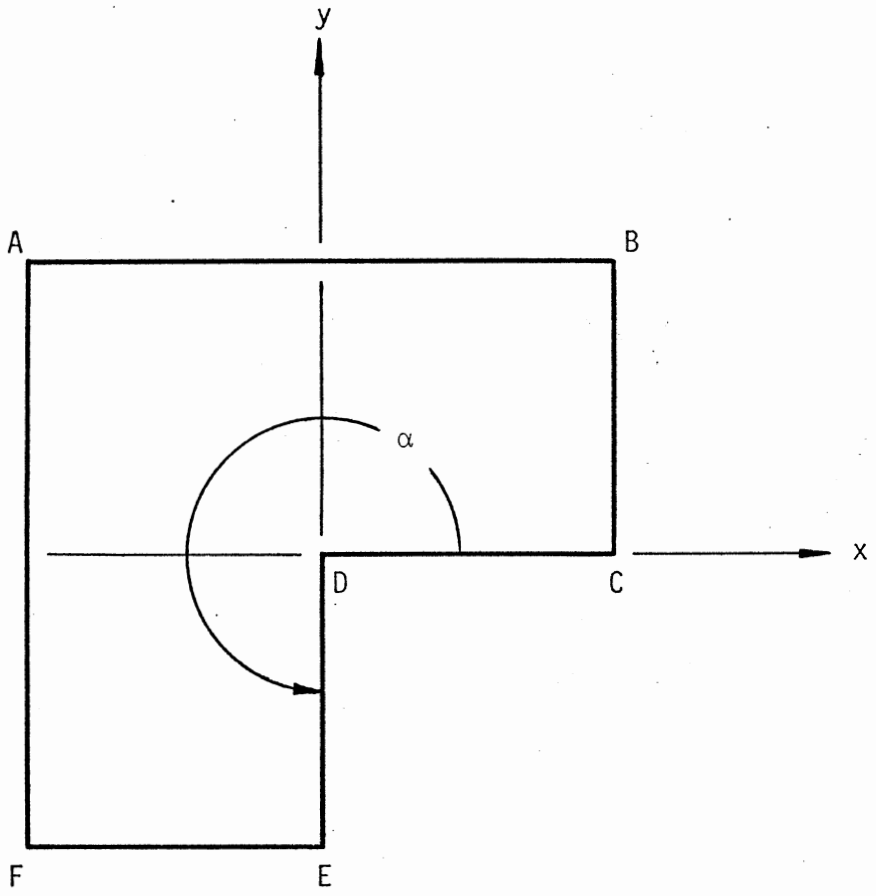


Figure 15. The L-Shaped Region

This is commonly called the L-shaped region problem; a related problem is the L-shaped membrane problem (11) (23). A physical system that corresponds to the problem described above is that of a conducting plate of the same shape with bar contacts attached at BC and EF. The bar contacts are maintained at +1 volt and -1 volt. The solution of the problem describes the electrostatic potential (voltage) on the surface of the plate. The electric field at the surface of the plate can be determined from the potential. Written in polar coordinates, the electric field has a dependence on the first derivative of the

solution with respect to r. Thus, at the re-entrant corner, the electric field becomes undefined (from (5.9) and (5.10)).

At the re-entrant corner, point D, α is $3\pi/2$. The expansion of equation (5.10) applies in the vicinity of the corner due to the Neumann conditions on CD and DE. It can be rewritten as

$$u(r,\theta) = \sum_{n=0}^{\infty} a_n r^{2n/3} \cos(2n\theta/3) \tag{5.11}$$

The solution in this form can be used to determine the truncation error in the five-point approximation of the Laplacian near the singularity.

Figure 16 shows a uniform grid with spacing h in the vicinity of the re-entrant corner. The expected truncation error in the approximation at point P is $O(h^2)$,

$$\nabla^2 u_p = \frac{1}{h^2}(u_D + u_G + u_H + u_J - 4u_p) + O(h^2) , \tag{5.12}$$

(from (3.10)). Substituting the series expansion of the solution (5.11), for each value appearing on the right of (5.12) demonstrates that the truncation error is not $O(h^2)$ (29)

$$\nabla^2 u_p = h^{-2}(u_D + u_G + u_H + u_J - 4u_p) + O(h^{-4/3}) + O(h^{-2/3}) + \ldots \tag{5.13}$$

The same truncation error occurs at point N in Figure 16. The symmetry in the placement of the values making up the five-point approximation at point D substantially reduces the truncation error,

$$\nabla^2 u_D = h^{-2}(u_M + u_N + u_P + u_Q - 4u_D) + O(h^{-2/3}) + O(h^{2/3}) + \ldots \tag{5.14}$$

Thus, the Laplacian is poorly represented by finite difference approximations near the re-entrant corner. From (5.13) and (5.14) an unusual finding is that the truncation error increases as the solution grid is refined (h grows small).

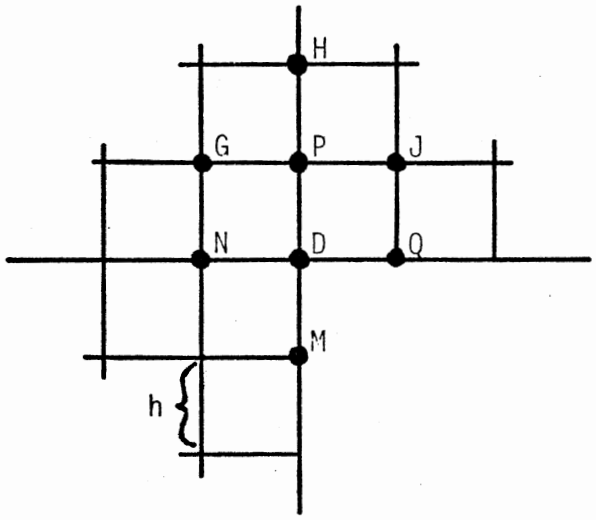


Figure 16. A Uniform Grid Near a Re-Entrant Corner

The error in the solution near the singularity is not as serious as that in the approximation of the Laplacian. For Laplace's equation, (5.13) can be rewritten for u_p in terms of the surrounding points,

$$u_P = \frac{1}{4}(u_D + u_G + u_H + u_J) + O(h^{2/3}) + O(h^{4/3}) + \ldots \qquad (5.15)$$

The truncation error in u_D has the leading term $O(h^{4/3})$. The truncation error in the solution is large compared to that at points some distance from the re-entrant corner. Solution iterations spread this error over

the entire solution domain; the re-entrant corner pollutes the whole solution.

There are methods for restoring the accuracy of the solution. These use some sort of special treatment in the region about the re-entrant corner, or they seek to restate the problem in a way that eliminates the re-entrant corner altogether. Near the corner, the analytical form of the solution can be used (when it is known), or a finer grid can be employed.

Motz's Method for Re-Entrant Corners

When the analytical form of the solution is known near a re-entrant corner or any other type of singularity, it can be incorporated into a finite difference solution method. Such modified methods are due to Motz (20) and Woods (31); the methods are applied by Reid and Walsh (23) and Whiteman (30). The method considered here is the one due to Motz.

The form of the solution of Laplace's equation near a re-entrant corner is given by one of (5.9) and (5.10). Thus, the solution of Laplace's equation can serve to test and develop Motz's method. Returning to the L-shaped region problem, the solution near the re-entrant corner is given by equation (5.10). The coefficients, a_n, are not given, though they can be determined from the continuous solution. Making the assumption that values of the discrete solution some distance away from the corner are exact, approximation for the first few coefficients can be determined. Equation (5.10) must be truncated after k terms and k remote values of the discrete solution must be selected in order to determine a set of coefficients. Equation (5.10) becomes

$$u_j = \sum_{n=0}^{k-1} a_n c_{jn} \tag{5.16}$$

for each remote value, u_j. The coefficients c_{jn} depend on the coordinates of u_j, (r_j,θ_j),

$$c_{jn} = r_h^{2n/3} \cos (2n\theta_j/3). \tag{5.17}$$

Defining the vector $\vec{u}_r$ made up of remote values and the vector $\vec{b}$ made up of the coefficients, b_n, the system

$$C\vec{b} = \vec{u}_r \tag{5.18}$$

results from writing (5.16) for each u_j. The system of (5.18) can be solved directly or iteratively for the vector of coefficients, $\vec{b}$.

Figure 17 shows the grid points near the re-entrant corner of Figure 15. The truncated series used to give the solution value at each grid point within a distance d of the corner (23),

$$u_m = \sum_{h=0}^{k-1} b_n d_{mn} \tag{5.19}$$

The coefficients d_{mn} depend on the coordinates of u_m,

$$d_{mn} = r_m^{2n/3} \cos (2n\theta_m/3). \tag{5.20}$$

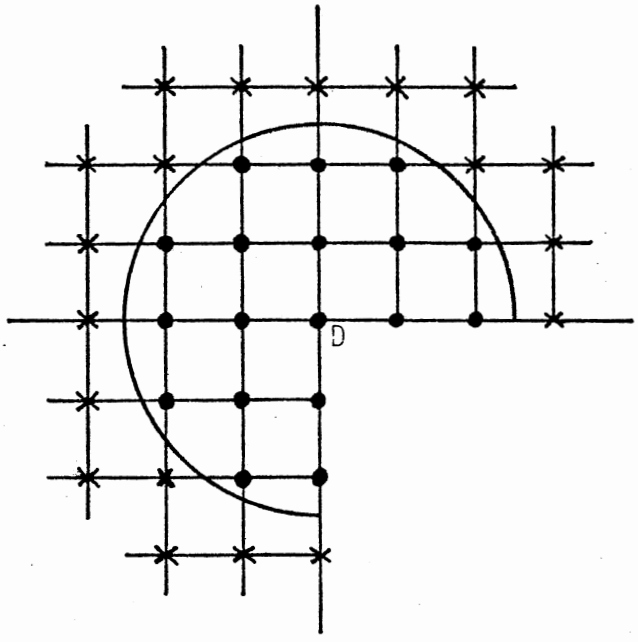
Defining the vector u_s of series replacement values, the system

$$D\vec{b} = \vec{u}_r \tag{5.21}$$

results when (5.19) is written for each u_m; D is not necessarily a square matrix. Values at grid points farther than d from the re-entrant corner, outside the arc in Figure 17, are candidates for components of $\vec{u}_r$.

Motz's method has been presented as if an accurate solution exists. In practice, the steps outlined here must be used at each iteration.

The vector of series values approaches the correct solution with each iteration causing the vector of finite difference values, $\vec{u}_r$, to approach the correct solution, in turn. Information flows in two directions across the boundary between the series process and the finite difference process. The vector $\vec{u}_s$ is built up from information propagated from distant boundaries by the finite difference process. Values of $\vec{u}_r$ that are linked to values of $\vec{u}_s$ by the five-point approximation, (5.12), are prevented from seeking the erroneous values of the uncorrected solution. They influence the remaining points of $\vec{u}_r$ through the iteration process.



- components of $\vec{u}_s$

- choices for components of $\vec{u}_r$

Figure 17. Remote Points and Series Replacement Points

It is important to note that the matrices C and D of (5.18) and (5.21) do not change from iteration to iteration. The matrix C can be put in a form that requires only $O(k^2)$ operations to determine $\vec{b}$ from a new $\vec{u}_r$ as opposed to $O(k^3)$ operations if C changes. The routines DECOMP and SOLVE from (10) provide this economy. An iteration of the hybrid method involves a sweep with SOR or some other method followed by the determination of $\vec{b}$ and $\vec{u}_s$.

Conformal Mapping for Re-entrant Corners

As mentioned in Chapter II, complex transformations, conformal mappings, can be used to simplify the geometry of problems with complicated features. Also, a solution of Laplace's equation for the transformed problem is a solution of the original problem when it is transformed to the original setting. Thus, conformal mappings can be used to eliminate re-entrant corners and dispense with the need for a special treatment near a corner (23) (28) (29) (30).

For the L-shaped region of Figure 15, a useful mapping is

$$w = z^{2/3} \tag{5.22}$$

Domains w and z are given by $z = (x + iy)$ and $w = (t + iv)$. The corner formed by segments CD and DE is eliminated. Figure 18 shows the transformed region; the vertices are labeled so that they correspond to the vertices of the L-shaped region. Only the segments CD and DE are lines in the transformed problem; all other segments become curves.

The potential problem described earlier for the L-shaped region can be solved on the domain of Figure 18. The boundary conditions of the original problem apply to the transformed problem segment by

segment: Dirichlet conditions on BC and EF, Neumann conditions elsewhere. Finite difference methods can be used to produce a solution.

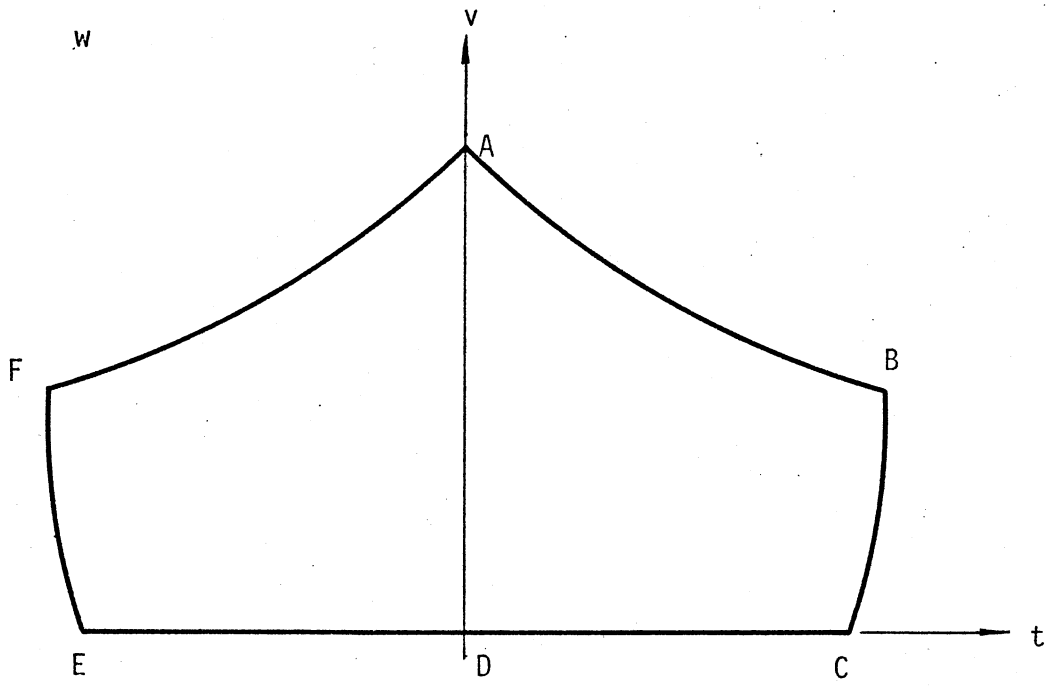


Figure 18. Tranformed L-Shaped Region

Several complications are introduced by the conformal mapping of (5.22). The curved boundaries of transformed solution domain require the use of irregular grids near the boundary in the finite difference process. The accuracy of the solution can be reduced (4) (11) (23). The Neumann conditions are particularly troublesome. Complicated interpolation schemes are required to implement them (4) (11). If the solution is to be used in the z domain the inverse mapping

$$z = w^{3/2} \tag{5.23}$$

can be used or points in the z domain can be transformed using (5.22). In either case, an interpolation scheme with accuracy equivalent to that of the solution must be used.

For a general re-entrant corner with internal angle $\alpha = \pi/m$ the form of (5.22) becomes

$$w = z^m = r^m e^{im\theta} \tag{5.24}$$

for z in polar coordinates.

Grid Refinement

The largest truncation term for the L-shaped region problem is $O(h^{2/3})$; it occurs at points adjacent to the re-entrant corner. The truncation error there can be made to approach the $O(h^4)$ error that exists at points far removed from the re-entrant corner. If a fine grid is used near the re-entrant corner, its spacing, h_d, must approach h^6 to give the desired truncation error.

The use of such a grid must be restricted to the immediate neighborhood of the re-entrant corner in order to keep storage to a minimum. It must be used in concert with coarser solution grid. By using a collection of grids with spacing h/2, h/4, ..., h_d, the solution grid can be connected to the fine grid. Figure 19 shows such a grid system.

The multigrid method can be adapted to use refined grids near a re-entrant corner. The grids can be fixed in place (5) or they can be generated as needed (6) (7). The adaptive multigrid process requires sophisticated routines for sensing error and managing the collection of grids. Near a feature such as a re-entrant corner as many as 20 grids might be used (7).

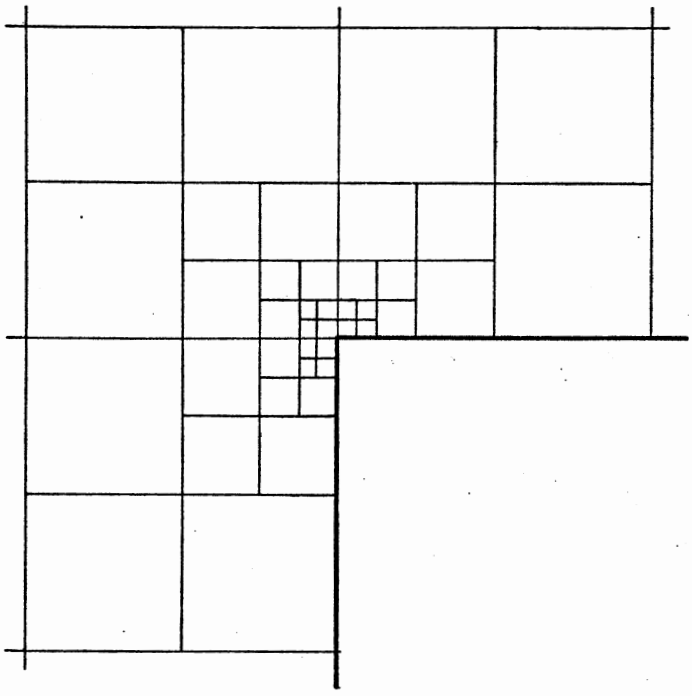


Figure 19. Refined Grids Near a Re-Entrant Corner

Here, grid refinement is used in conjunction with extrapolation to produce an accurate solution for the L-shaped region problem. As h goes to zero, solutions produced with finite difference methods approach the actual solution. By using a sequence of spacings, a convergent sequence of solutions can be produced. Aitken extrapolation can be used to give an estimate of the limit. Here, the sequence

$$h, h/2, h/4, \ldots \tag{5.25}$$

is used.

CHAPTER VI

RESULTS OF NUMERICAL EXPERIMENTS AND CONCLUSIONS

Performance Comparisons for Several Iterative Methods

Young's test problem (32) is an excellent means for comparing the iterative methods developed here. The dominant eigenvalues of the iteration matrices of the J and GS methods are known, hence, the optimum value of ω can be determined for the SOR and SSOR methods.

The test problem requires the solution of Laplace's equation on a square with sides of length π. There are Dirichlet conditions on the boundary of the square. The solution is set to zero on the boundary; the initial solution estimate is everywhere one. After each iteration of a particular method, the solution estimate serves as the error for the iteration since the solution must be everywhere zero.

The dominant eigenvalue of G_j is given by

$$\mu = \cos(h) \tag{6.1}$$

where h is the spacing of a uniform grid on the square (4) (11). Then, the dominant eigenvalue for G_g is

$$\Lambda = \cos^2(h) \tag{6.2}$$

from (3.53). Equation (4.7) for the optimum ω becomes

$$\omega_b = 2/(1 + \sin(h)) \tag{6.3}$$

Figure 20 shows the behavior of the residual vectors produced by the J, GS, SOR, and SSOR methods over 50 iterations for the test problem. The grid spacing is $h = \pi/32$. The norm of equation (3.44) is used as a measure of the residual vectors. The logarithm of the norm is plotted versus the iteration number in order to make its behavior visible over several orders of magnitude. Iterations of the SSOR method are counted as two iterations of the SOR method.

In Figure 20, the GS and J methods rapidly become asymptotic and exhibit the slow convergence predicted earlier. The lines representing the GS and J methods eventually cross. The SOR and SSOR methods make large initial reductions of their solution vectors towards zero. Once they become asymptotic they demonstrate superior performance in reducing the residuals to zero. The value of ω_b used for SOR and SSOR iterations is 1.821465 (from equation (6.3)).

Figure 21 shows the behavior of the error (solution) vectors over 50 iterations for the problem described above. Again, SOR and SSOR demonstrate superior convergence characteristics. From Figures 20 and 21, it is evident that SOR is a better solution method than SSOR. The symmetric iteration matrix of the SSOR method requires a performance sacrifice (32).

Table I gives comparisons between the knwon values of the dominant eigenvalues of the J and GS iteration matrices and experimentally determined values for the test problem. The experimental values are determined from the residual norm by using equations (4.3) and (4.6), Aitken extrapolation. The experimental values are given after 20 iterations and 100 iterations; the values of ω_b they would give are included in the table. It is apparent that the GS method affords better estimates of ω_b than the J method.

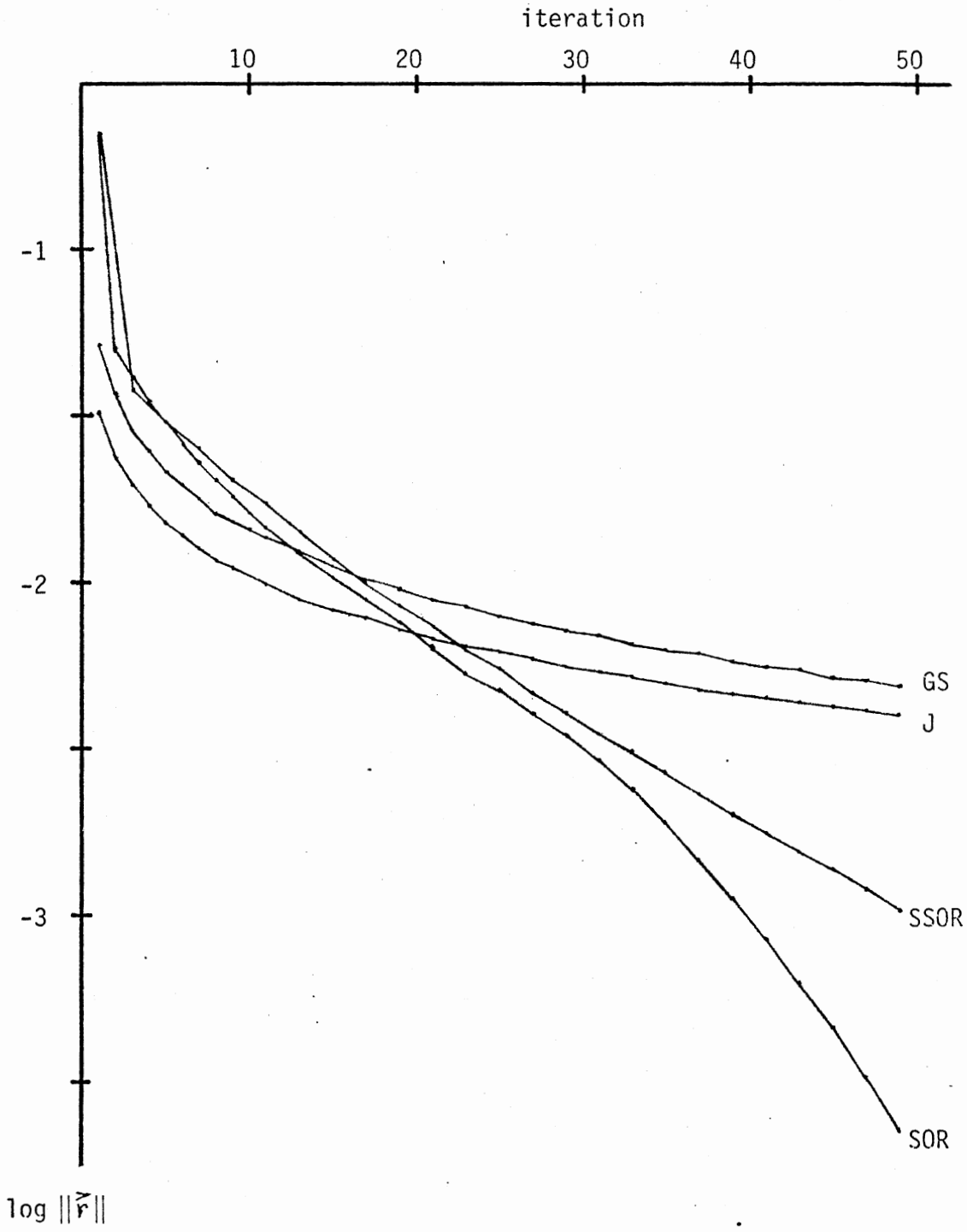


Figure 20. Behavior of the Residual Norms of the J, GS, SOR, and SSOR Methods for Young's Test Problem ($h = \pi/32$)

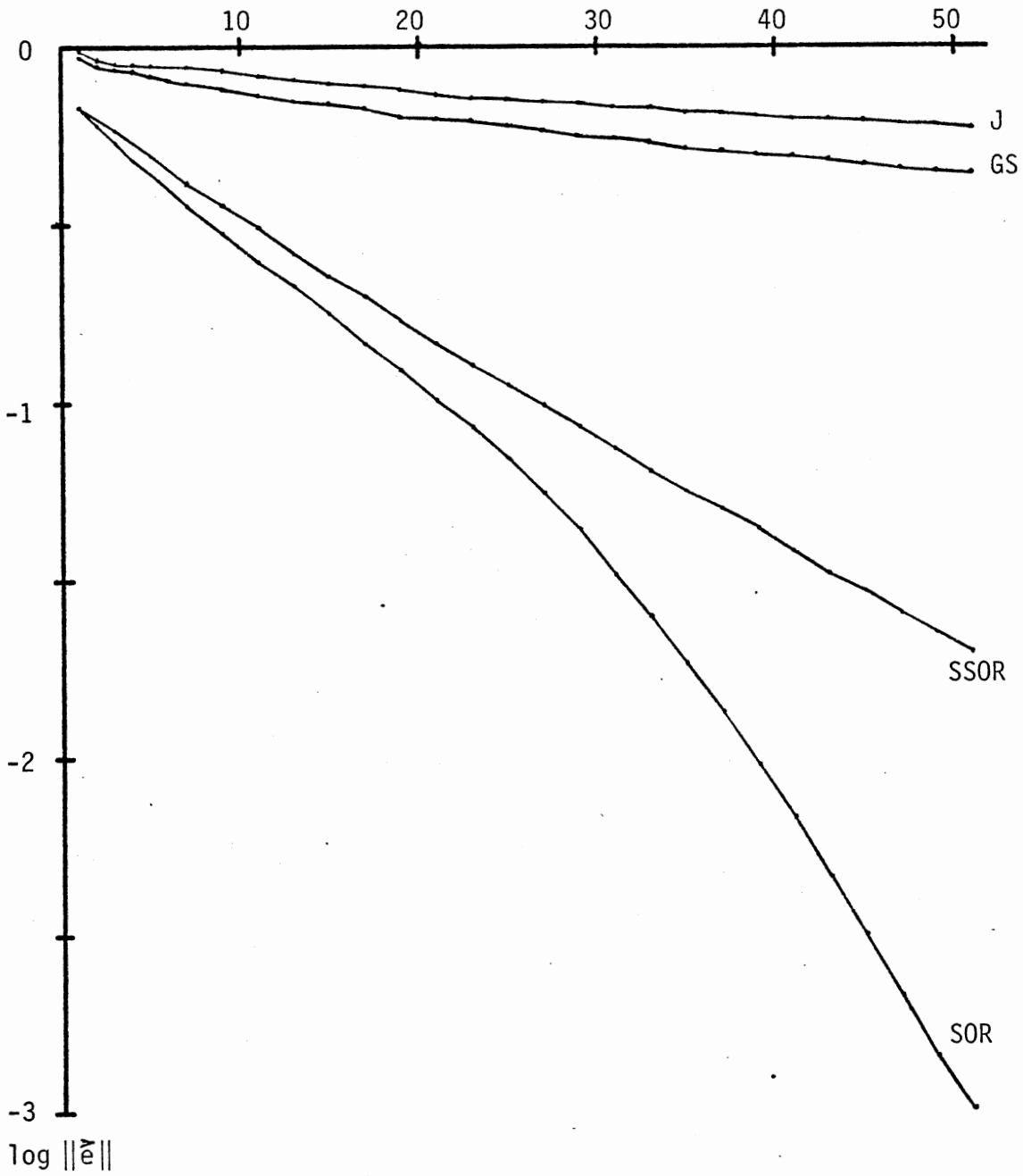


Figure 21. Behavior of the Error Norms of the J, GS, SOR, and SSOR Methods for the Test Problem ($h = \pi/32$)

TABLE I

KNOWN AND EXPERIMENTAL VALUES OF THE DOMINANT EIGENVALUES FOR THE J AND GS METHODS FOR THE TEST PROBLEM

	J	GS
Known S(G)	.9951847	.9903926
(ω_b)	(1.821465)	(1.821465)
S(G) from 20 iterations	.9820577	.9808125
(ω_b)	(1.682679)	(1.756668)
S(G) from 100 iterations	.9924346	.9894568
(ω_b)	(1.781302)	(1.813763)

Figure 22 shows the general characteristics of the iterates produced by the J, GS, and SSOR methods in a solution of the test problem. The sharp edges of the initial solution estimate are smoothed and the iterates take on a "pillow" shape. The solution tapers from a central maximum to zero on the boundary. For SOR, the maximum is skewed away from the center. The J and GS methods are particularly good at smoothing iterates. If their iterates are normalized to one following each iteration, they approach the function

$$f(x,y) = \sin(x) \sin(y), \tag{6.3}$$

an eigenfunction of the Laplacian differential operator.

The value of the central maximum of an iterate after a specified number of iterations is an additional indication of its effectiveness. The values range from .907 for the J method to $.545 \times 10^{-6}$ for SOR after 100 iterations. The exact value is zero.

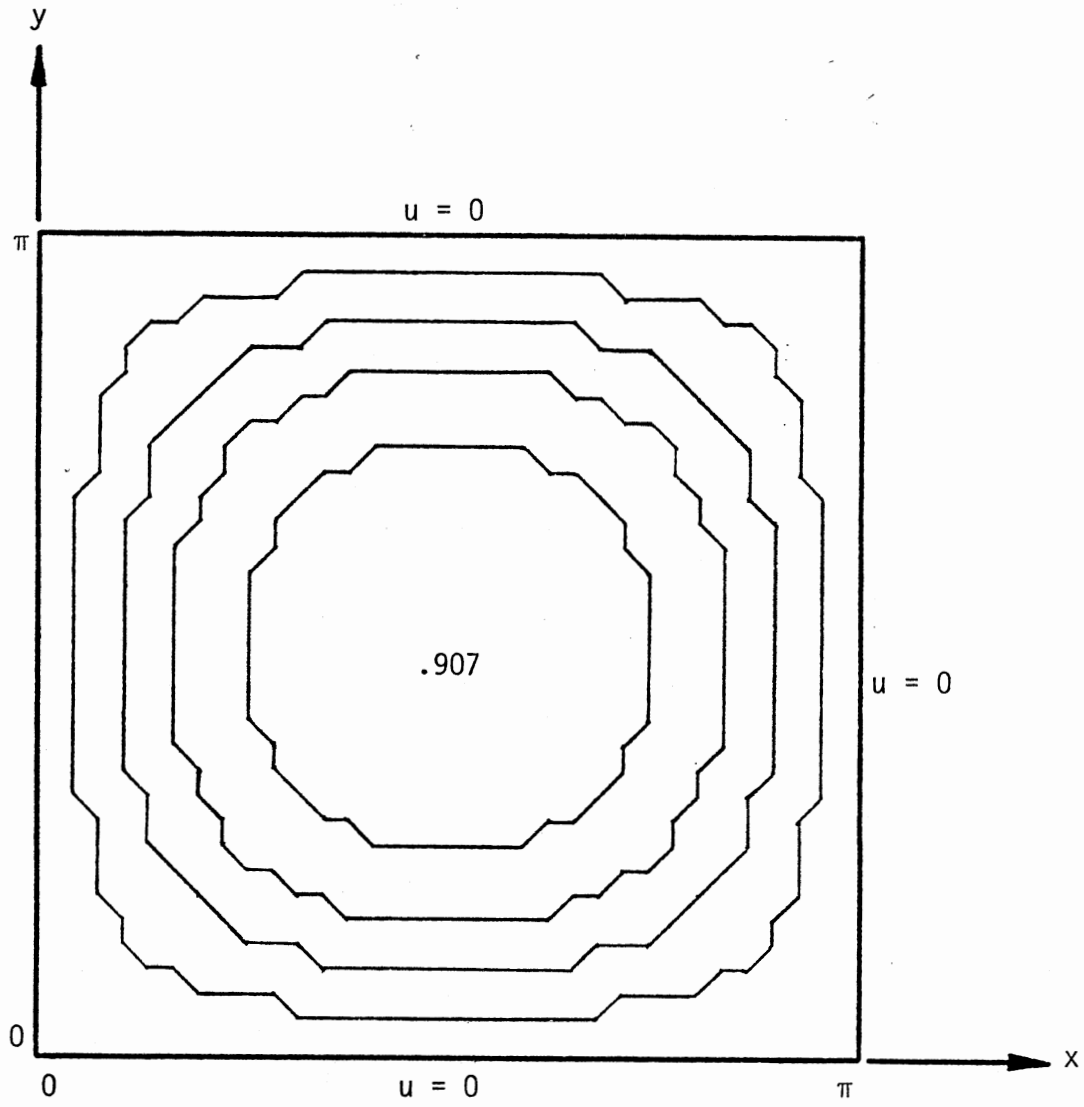


Figure 22. The Solution Vector of the J Method After 100 Iterations ($h = \pi/32$)

Figure 23 shows the residual norms of SOR, SSOR with vector Aitken acceleration (SSOR/VA), and the Cycle-C multigrid method. Dashed lines indicate extrapolations for SSOR/VA and coarse grid excursions for the multigrid method. The effort expended in coarse grid operations is measured according to iterations on the solution grid (identical to the SOR and SSOR grids). Figure 24 shows the error norms for the three methods.

Clearly, the Cycle-C multigrid method is superior to the SOR and SSOR/VA methods for the test problem. It meets its stopping criterion after the equivalent of 29 iterations. The SOR and SSOR/VA method show essentially the same performance over 100 iterations.

Table II gives a summary of the results for the test problem. Asymptotic rates of convergence, R(G), determined experimentally, are given for each method; actual rates are given when the dominant eigenvalue is known. Equation (3.46) is used to determine the rates. The value of the central maximum of the solution vector produced by each method is given, also. Each result is based on 100 iterations of the corresponding solution method (29 for the multigrid method). The exponential notation of FORTRAN is used in the table.

The experiments with the test problem indicate that the Cycle-C multigrid method should be the first choice for a solution method. The SSOR/VA method should be the second choice.

Analysis of the L-Shaped Region Problem

When no re-entrant corner or other type of singularity is present, the error in individual components of a solution of Laplace's equation using the five-point approximation, (3.10), is $O(h^2)$ (11). If the

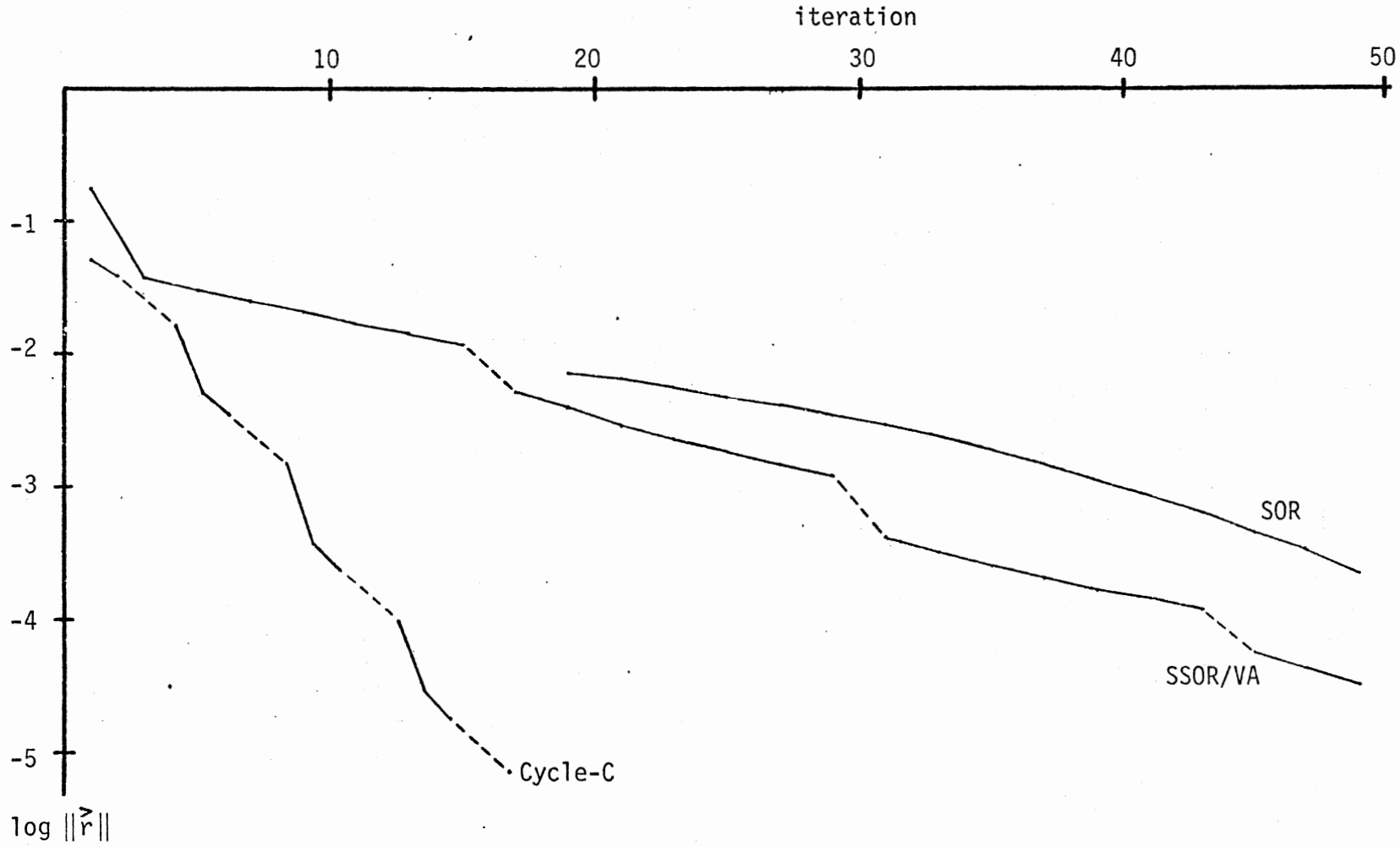


Figure 23. Behavior of the Residual Norms of the SOR, SSOR/VA, and Cycle-C Methods for the Test Problem ($h = \pi/32$)

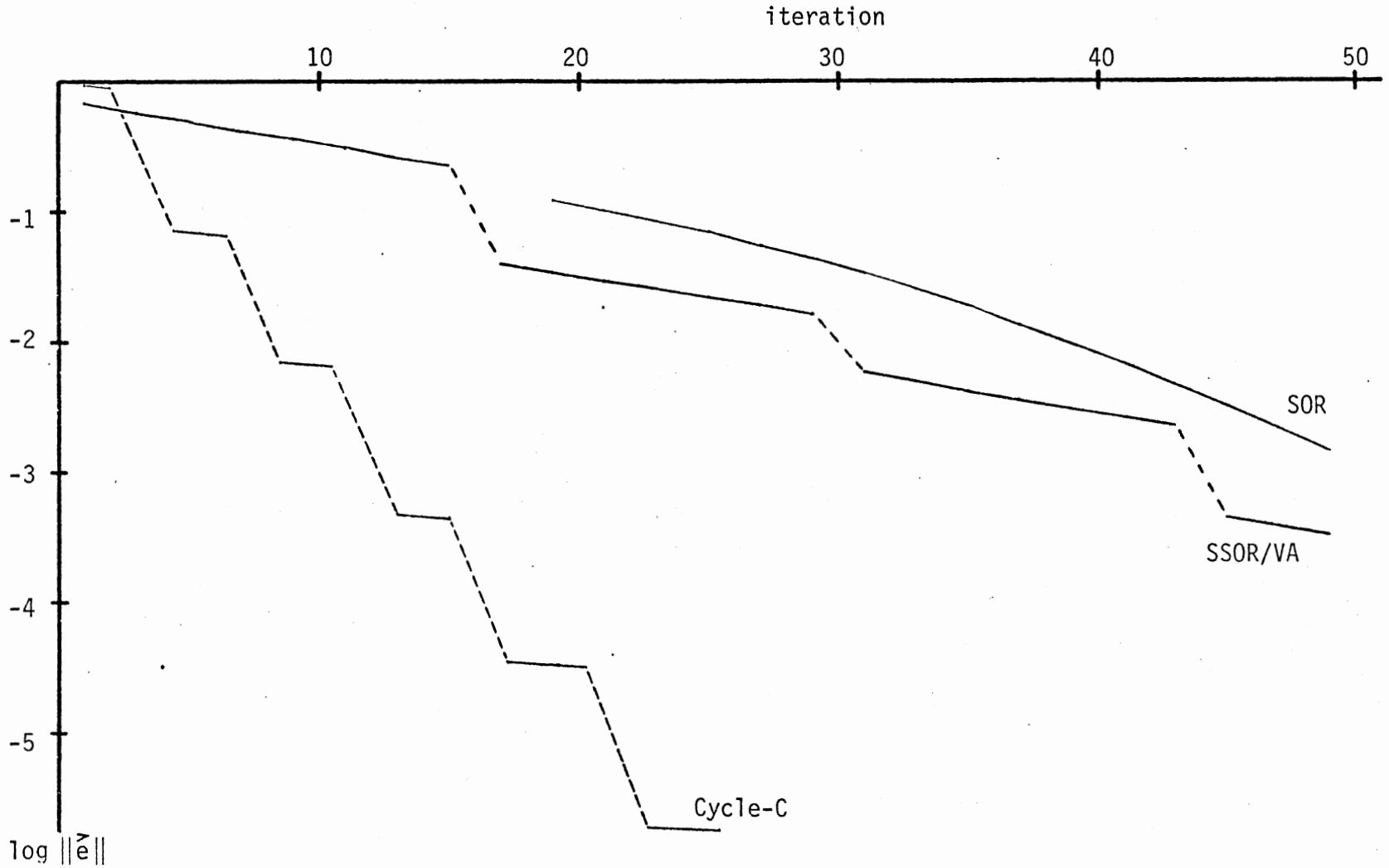


Figure 24. Behavior of the Error Norms of the SOR, SSOR/VA, and Cycle-C Methods for the Test Problem ($h = \pi/32$)

actual solution at a grid point with coordinates (x,y) is $u_\infty(x,y)$, the value given by a finite difference solution can be written

$$u_k(x,y) = u_\infty(x,y) + ah_k^2 + bh_k^4 + \ldots \qquad (6.4)$$

where h_k is the spacing of a uniform grid (11). By producing solutions for several values of h_k, the coefficients in (6.4) can be determined.

TABLE II

SUMMARY OF RESULTS FOR THE TEST PROBLEM

Method	R(G) (known)	R(G) (experimental)	max $\lvert e_i \rvert$
J	.2096309E-2	.3298102E-2	.9069986E0
GS	.4192613E-2	.4603162E-2	.6105773E0
SOR	.8541094E-1	.1306649E0	.5452358E-6
SSOR	--	.5590545E-1	.2024361E-2
SSOR/VA	--	.1520199E0	.3419504E-7
Cycle-C	--	.2589699E0	.2329918E-6

The effect of a re-entrant corner on the accuracy of finite difference solutions can be gauged by seeking a relation of the form (6.4). If it has the same powers in h, then the re-entrant corner has little influence.

Here, the L-shaped region problem of Chapter V is used to model the effects of a re-entrant corner. Figure 25 shows a solution that results

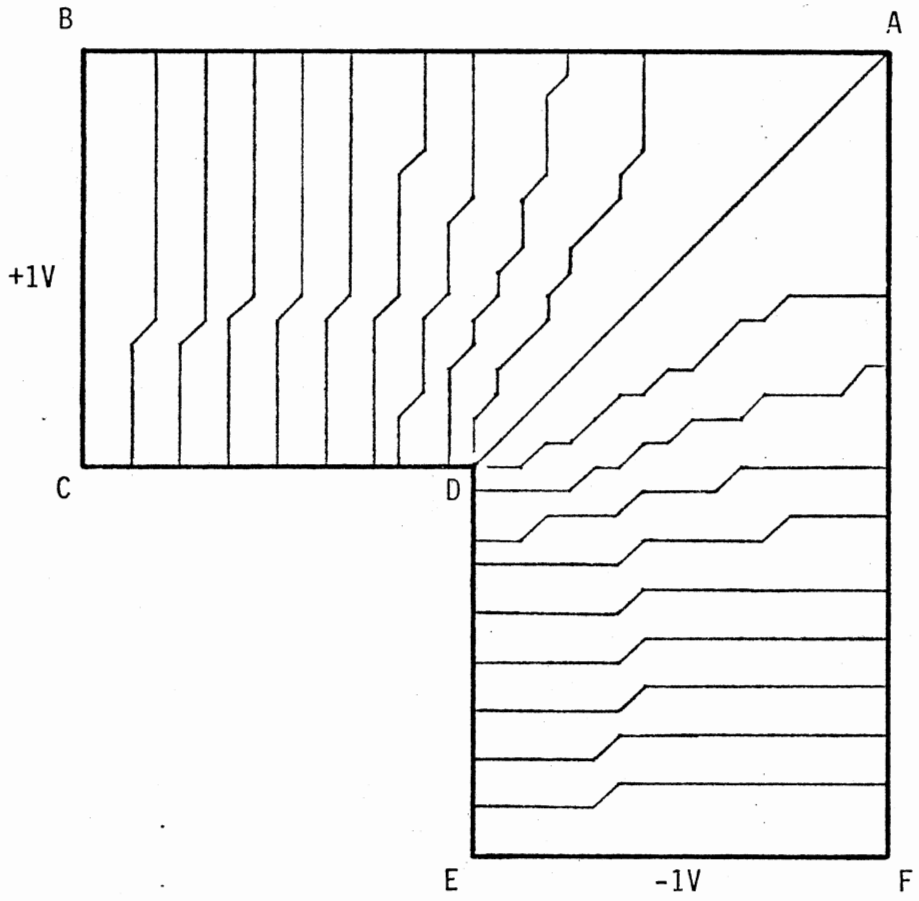


Figure 25. A Solution of the L-Shaped Region Problem Using SSOR/VA (20 Graduations)

from roughly 150 iterations of the SSOR/VA method; the segment AB has length 1.0 and h is 1/32. The solution is indicated by contours that range from -1.0 on EF to +1.0 on BC in 20 graduations. The solution is zero on the line of symmetry AD. From the Neumann conditions on BAF and CDE, each contour is tangent to the boundary at points of intersection.

The model used for the error in components of the solution is

$$u_k(x,y) = u_\infty(x,y) + ah_k^{\alpha} + bh_k^{\beta} . \tag{6.5}$$

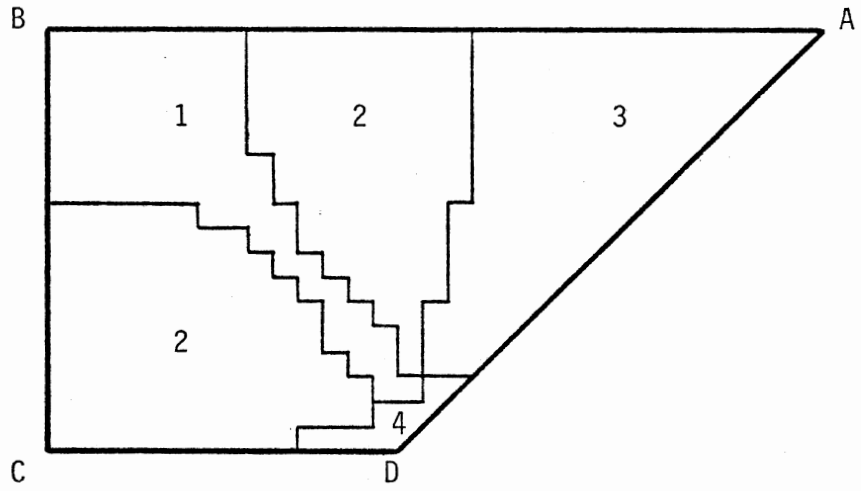
The terms in α and β are the dominant error terms. By producing solutions for the spacings h_0, h_1, h_2, and h_3 where $h_k = 2^{-k}h_0$, approximate values for α and β can be determined. Assuming that the term in α is dominant, the term in β can be dropped from (6.5). Then, writing the resulting equation for h_1, h_2, and h_3 gives

$$\alpha \cong \log [(u_3 - u_2)/(u_2 - u_1)]/\log (k) \tag{6.6}$$

at point (x,y) ($k = h_3/h_2$).

Solutions for $h_0 = 1/16$ through $h_3 = 1/128$ were produced in order to determine α and β. The residual norm for each solution was reduced to 10^{-14} by using double precision in a FORTRAN program.

Using (6.6) at points (x,y) where u_1, u_2, and u_3 exist gives the distribution of Figure 26. The segment AD is a line of symmetry for the distribution. Values of α range from 1.166 to 1.476. The greatest variance of values occurs about the re-entrant corner. The large number of values in the range 1.290 to 1.383 and the previous analysis of the L-shaped region problem (Chapter V) suggests that α is 4/3.



1--1.243 to 1.290 (77 points)
2--1.290 to 1.282 (189 points)
3--1.383 to 1.461 (115 points)
4--1.166 to 1.476 (10 points)

Figure 26. Distribution of Values of α

Using $\alpha = 4/3$ and both of the terms of (6.5) in h, an expression for β can be derived,

$$\beta = \log \left[\frac{k^{-\alpha}(u_3 - u_2) - (u_2 - u_1)}{k^{-\alpha}(u_2 - u_1) - (u_1 - u_0)} \right] \Big/ \log (k) \qquad (6.7)$$

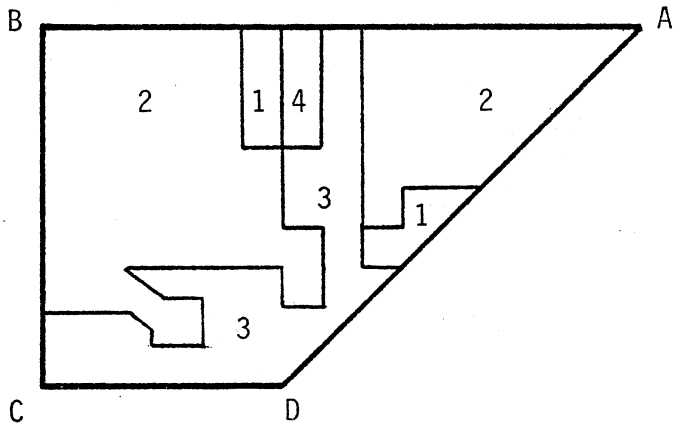
Applying (6.7) where u_0 through u_3 exist gives the distribution of Figure 27. The values range from 1.214 to 3.684. The largest number of values lie in the range 1.708 to 2.202 suggesting that β is 2 or 6/3.

Thus, it seems likely that the error components of the solution are given by

$$u_k(x,y) = u_\infty(x,y) + ah_k^{4/3} + bh_k^2 + ch_k^{8/3} + \ldots \qquad (6.8)$$

or $O(h^{4/3})$. This is serious degradation of the accuracy normally

afforded by finite difference methods. One of the techniques developed in Chapter V should be used to reduce the influence of the re-entrant corner.



1--1.214 to 1.708 (7 points)
2--1.708 to 2.202 (61 points)
3--2.202 to 2.860 (19 points)
4--2.820 to 3.684 (5 points)

Figure 27. Distribution of Values of β

The Multigrid Method for Re-Entrant Corners

It was hoped that the multigrid method could be used to develop solutions for the L-shaped region problem due to its speed and economy. It was found, though, that a single re-entrant corner could completely disrupt the Cycle-C process. As a result, the multigrid method had to be abandoned.

Figure 28 shows the behavior of the residual norm produced by iterations on the finest grid ($h = 1/32$); dashed lines indicate coarse grid excursions. After rapid initial progress toward a solution, the

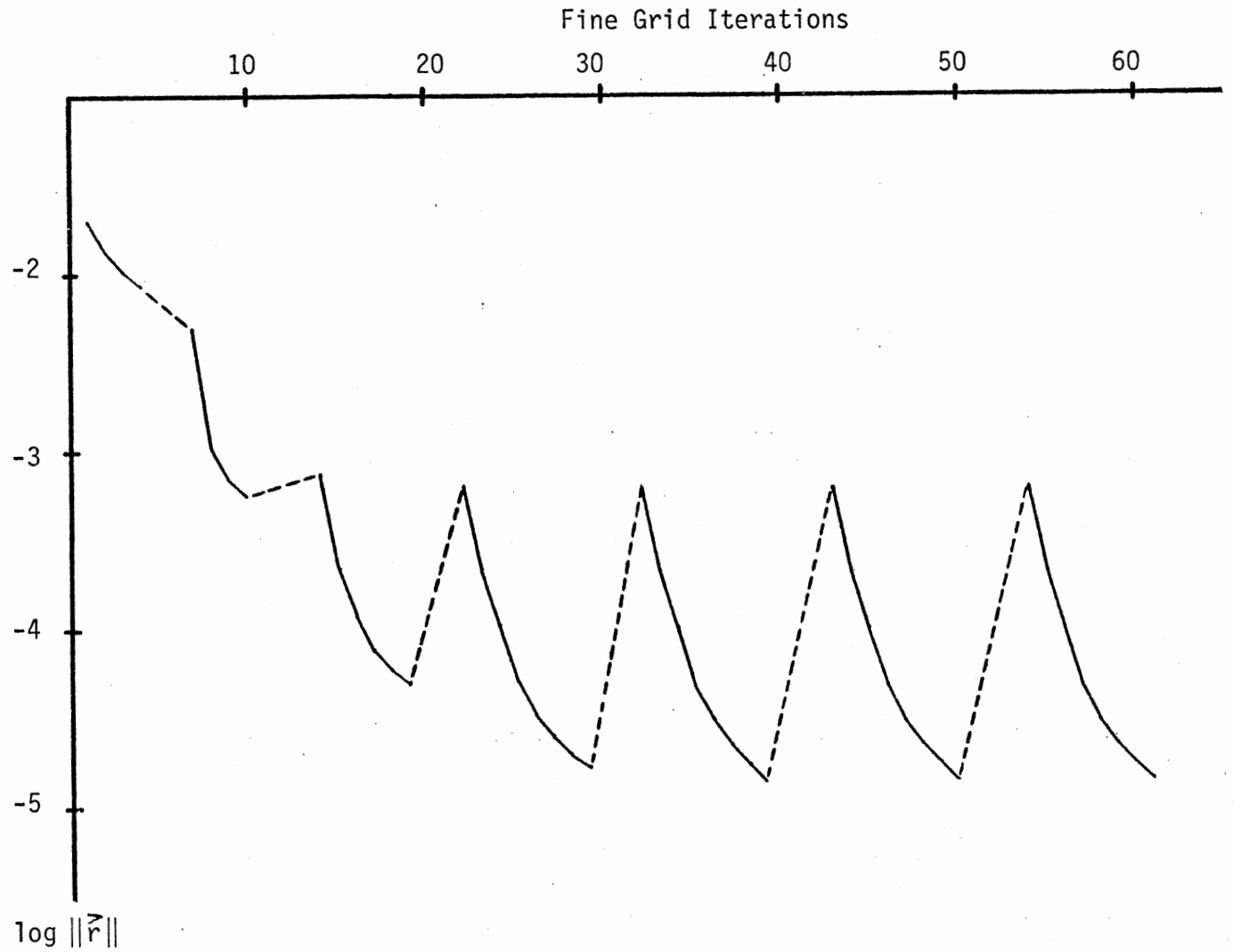


Figure 28. Behavior of the Multigrid Cycle-C Method for the L-Shaped Region Problem (h = 1/32)

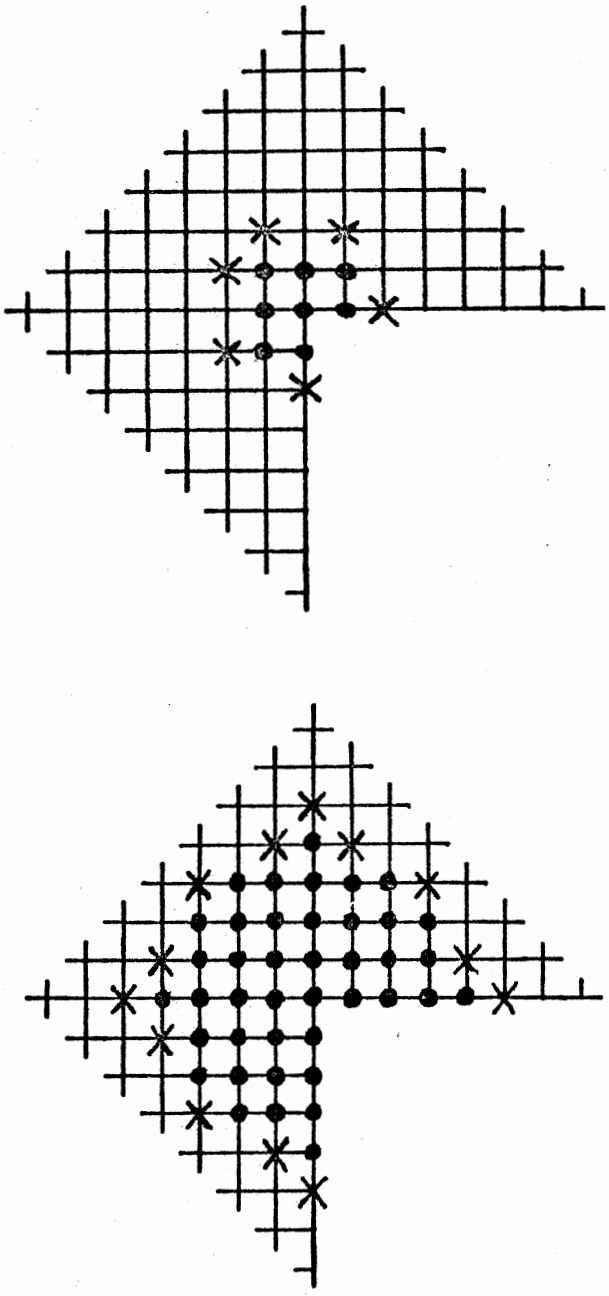
solution process enters a cyclic phase during which little progress is made. The operations on coarse grids to calculate corrections degrades to the point that large amounts of error are introduced into the solution on the fine grid. That error is quickly eliminated, but another coarse grid excursion reintroduces it.

The Behavior of Motz's Method for the L-Shaped Region Problem

Motz's method was used in conjunction with the SSOR/VA method in an attempt to restore the $O(h^2)$ discretization error when a re-entrant corner was present. The method was tested for accuracy and stability by using the L-shaped region problem with varying grid sizes. The use of extrapolation provided an additional test of stability.

Two sets of patterns of remote points and series replacement points were used. The simpler set of patterns involved 6 remote points and 8 series points. The more complex set had 13 remote points and 41 series points. The two sets are referred to as the 8-point and 41-point patterns, respectively. Figure 29 gives their organization. At the beginning of the forward and backward sweeps on an SSOR iteration, Motz's method was used to assign values to the series points. During an iteration, the series points were skipped.

The results of experiments with single precision arithmetic demonstrate limitations of Motz's method. The combined method converges (SSOR/VA and Motz), but progress slows when Motz's method becomes the dominant source of error. This occurs when the displacements in the series values and the quantities that make up the series values vary over the range of the single precision mantissa (seven digits). The



• - series point

x - remote point

Figure 29. The 8- and 41-Point (Series) Patterns for Use with Motz's Method

vector $\vec{d}_s$ made up of displacements in the series point vector, $\vec{u}_s$, is useful for observing the behavior of Motz's method,

$$\vec{d}_s^{\,n} = \vec{u}_s^{\,n} - \vec{u}_s^{\,n-1} \quad . \tag{6.9}$$

Figure 30 shows the norm of $\vec{d}_s$ for the 8-point pattern and the residual norm following the final vector Aitken extrapolation.

The displacement vector exhibits similar behavior when double precision arithmetic (16 digit mantissa) is used. Figure 31 shows the displacement norm for the 41-point pattern and the residual norm following the final extrapolation. Figure 32 indicates that the norm of $\vec{d}_s$ exhibits the characteristics observed earlier in the error and residual norms produced by the SSOR/VA method. A minimum of 10 iterations occurs between extrapolations.

An Accurate Solution for the L-Shaped Region Problem

An accurate solution is needed to gauge the effectiveness of Motz's method. It must be free of the error induced by the re-entrant corner. By using vector extrapolation, the uncorrected solutions used earlier to determine the exponents of (6.5) can yield an accurate solution. As mentioned in Chapter V, grid refinement produces solutions that converge to the actual solution. An extrapolation scheme based on equation (6.5) can be used to determine the components of the actual solution.

Solutions from grids with spacing $h_1 = 1/2$ through $h_7 = 1/128$ ($h_k = h_{k-1}/2$) were used to test the effects of extrapolation. Aitken extrapolation, equation (4.6), was used with corresponding components

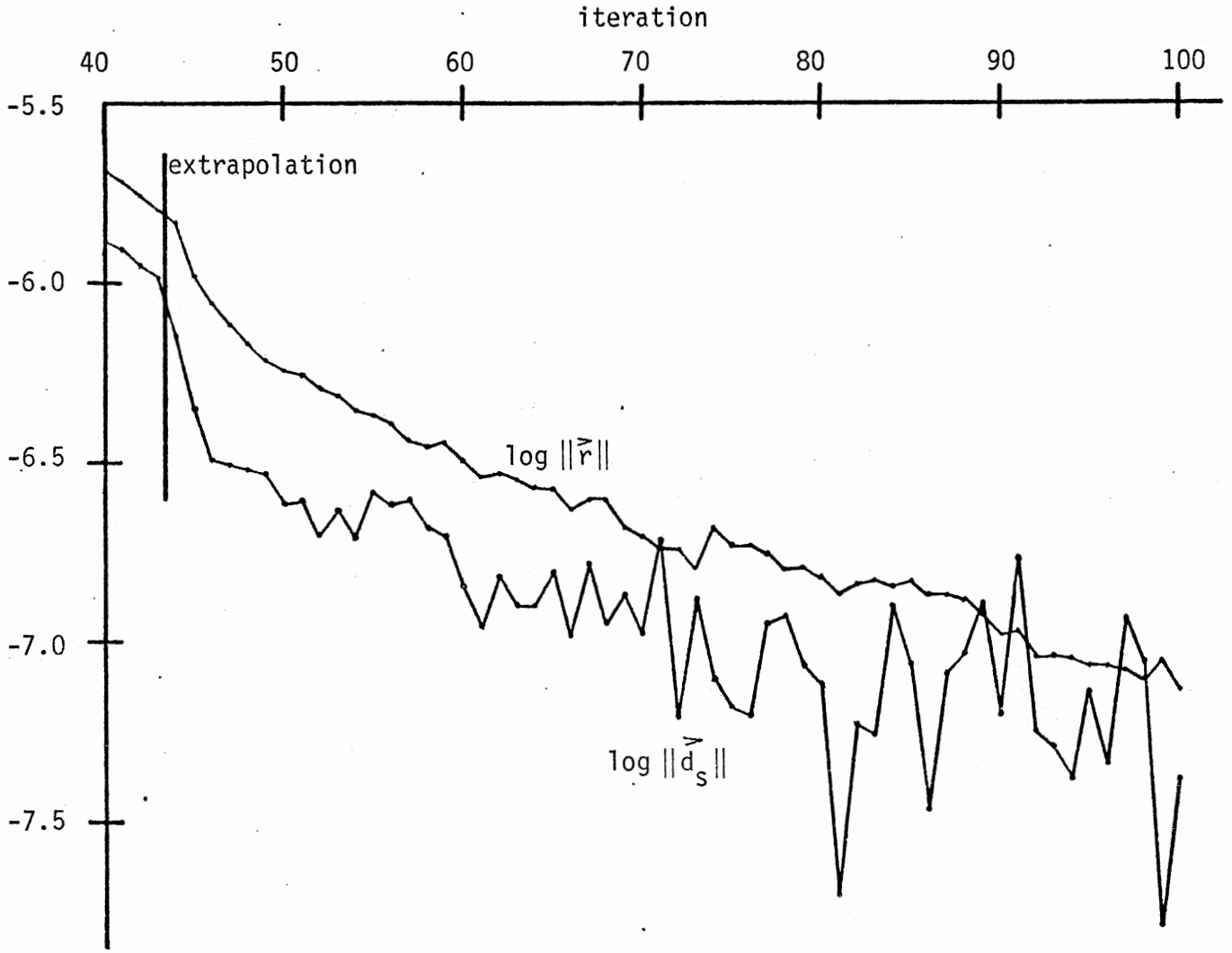


Figure 30. The Behavior of SSOR/VA and Motz's Method (8-Point Pattern) in Single Precision (Seven Digit Mantissa)

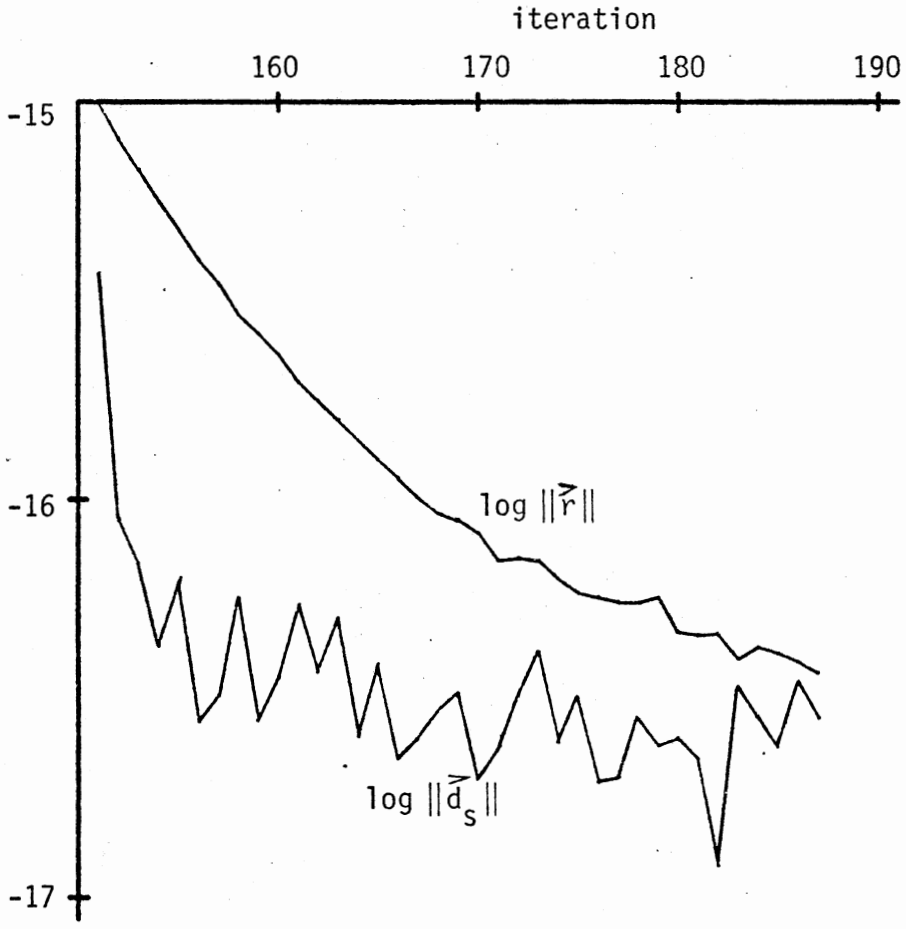


Figure 31. The Behavior of SSOR/VA and Motz's Method (41-Point Pattern) in Double Precision (Sixteen Digit Mantissa)

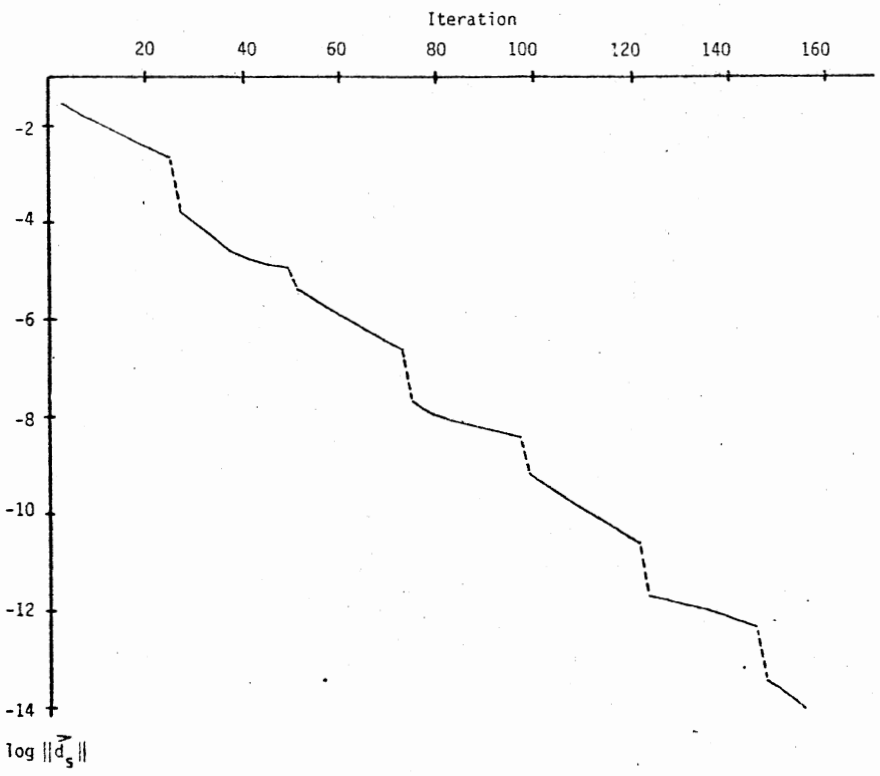


Figure 32. The Behavior of the Displacement Vector for SSOR/VA and Motz's Method (41-Point Pattern)

of the solutions for grids with spacing h_{k-2}, h_{k-1}, and h_k. The resulting vector of extrapolated components had spacing h_{k-2}. It was observed that an extrapolated solution for grid spacing h_{k-2} through h_k gave accuracy exceeding that of a solution for grid spacing h_{k+2}. Thus, the extrapolated solution for h_5 through h_7 had accuracy exceeding that of a solution produced with a grid of spacing 1/512. This extrapolated solution was chosen for comparison with the results from Motz's method.

Approximate values of ω_b were determined in producing the uncorrected solutions for the L-shaped region problem. They were used again for the solutions produced with Motz's method. The values of ω_b are given in Table III.

TABLE III

VALUES OF ω_b FOR THE L-SHAPED REGION PROBLEM

h	ω_b
1/16	1.659
1/32	1.811
1/64	1.893
1/128	1.930

Evaluation of Motz's Method

The 8-point and 41-point patterns of Figure 29 were used in testing Motz's method. L-Shaped region solutions were produced for the grid spacings $h_1 = 1/16$, $h_2 = 1/32$, and $h_3 = 1/64$. The convergence

criterion for each solution required that the residual error be reduced to 10×10^{-14}.

Figure 33 shows the residual norms for solutions produced by the combination of SSOR/VA and Motz's method with the 8-point pattern. The decrease in the rate of convergence with decreasing h is apparent. The solutions for h_1, h_2, and h_3 required 168, 296, and 684 iterations to meet the convergence criterion. Figure 34 shows the residual behavior for solutions produced with the 41 point pattern. The solutions for h_1, h_2, and h_3 required 156, 316, and 688 iterations, respectively. Thus, there is little difference in the effort required to produce a solution with one method or the other. These two trials and previous trials with varying h suggest that the number of iterations required to produce a solution is $O(h^{-1})$.

In order to compare the results of Motz's method and the exact solution, an error vector is needed,

$$\vec{e} = \vec{u}_e - \vec{u}_m, \tag{6.10}$$

where $\vec{u}_e$ is the exact solution and $\vec{u}_m$ results from Motz's method. The norm given by equation (3.44) can be used to measure the error vectors that result from comparison. It is scaled according to the number of components.

Tables IV, V, and VI give comparisons between the exact solution and the uncorrected solutions, the 8-point Motz solutions, and the 41-point Motz solutions.

Figure 35 gives the relationship between grid spacing and the error norm for the 8-point Motz solutions, the 41 point Motz solutions, and the uncorrected solutions. The important feature is that the 8-point and 41-point Motz solutions for h = 1/16 have accuracy equivalent

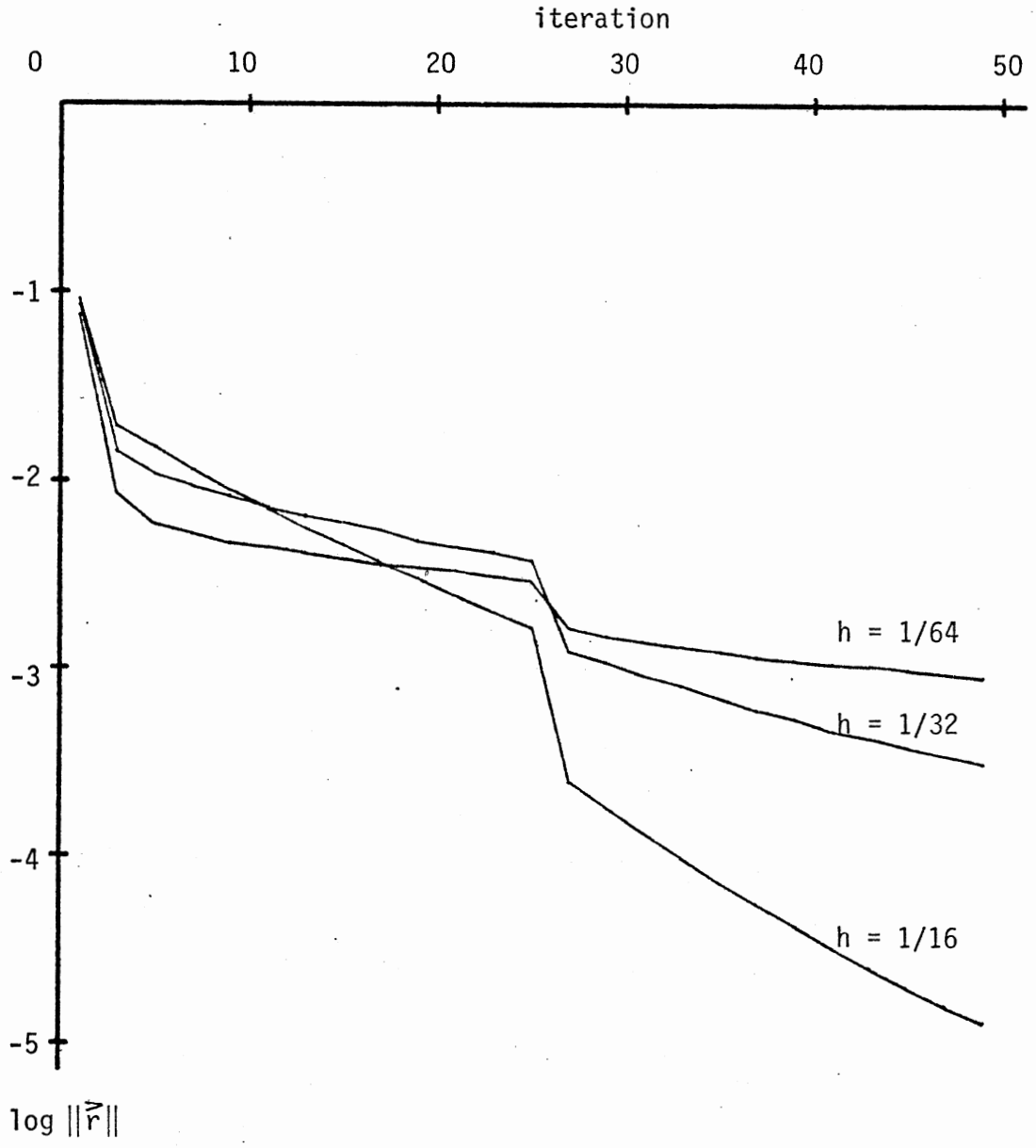


Figure 33. Behavior of Residual Norms for Solutions with 8-Point Motz and SSOR/VA

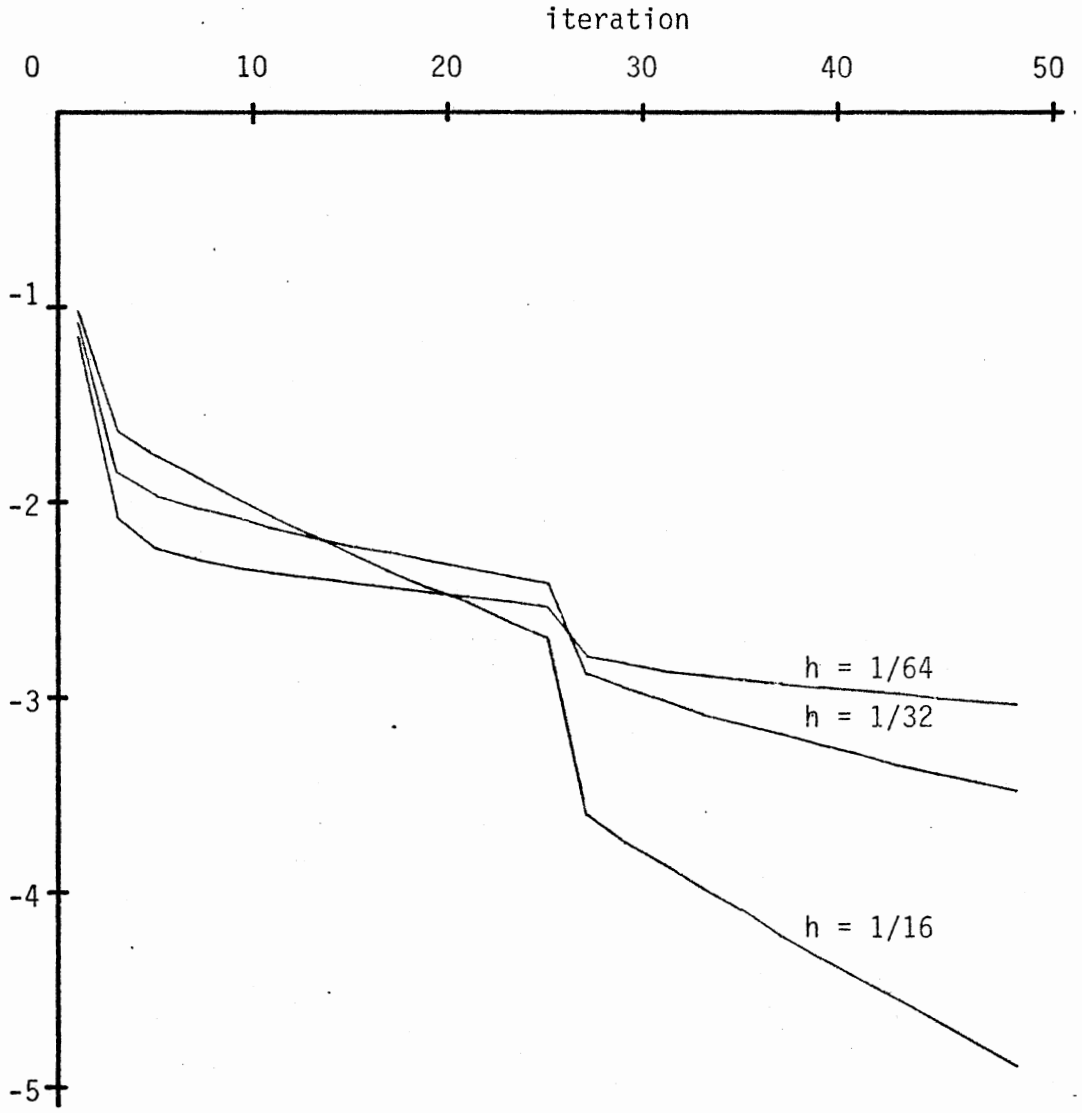


Figure 34. Behavior of Residual Norms for Solutions with 41-Point Motz and SSOR/VA

TABLE IV

COMPARISONS BETWEEN THE EXACT SOLUTION AND THE UNCORRECTED SSOR/VA SOLUTIONS

h	$\|\|\tilde{e}\|\|$	max $\|e_i\|$
1/8	.6510000E-2	.3082128E-1
1/16	.2646606E-2	.2058514E-1
1/32	.1049086E-2	.1373039E-1
1/64	.4161518E-3	.5684991E-2
1/128	.1653452E-3	.2343839E-2

TABLE V

COMPARISONS BETWEEN THE EXACT SOLUTION AND THE 8-POINT MOTZ SSOR/VA SOLUTIONS

h	$\|\|\tilde{e}\|\|$	max $\|e_i\|$
1/16	.1993717E-3	.1220212E-2
1/32	.6905748E-4	.7440496E-2
1/64	.2859288E-4	.9446983E-3

TABLE VI

COMPARISONS BETWEEN THE EXACT SOLUTION AND THE 41-POINT MOTZ SSOR/VA SOLUTIONS

h	$\|\|\tilde{e}\|\|$	max $\|e_i\|$
1/16	.9698017E-4	.2139818E-3
1/32	.3952888E-4	.4088687E-3
1/64	.1151425E-4	.4162266E-3

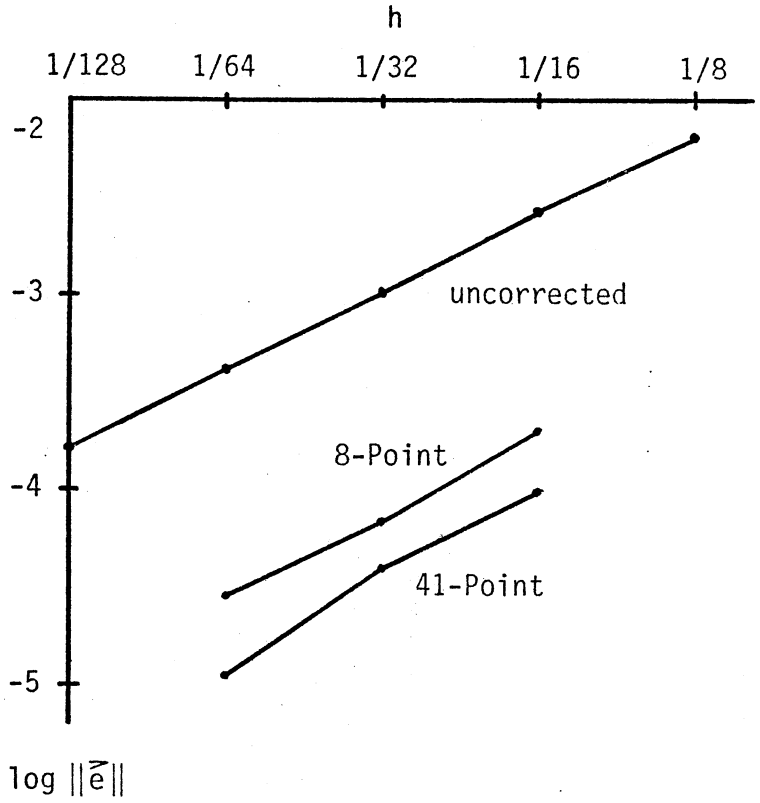


Figure 35. The Variation in the Error Norm with h for the 8-Point, 41-Point, and Uncorrected Solutions

to uncorrected solutions for h = 1/128 and h = 1/256, respectively. The solutions for h = 1/16 require approximately 160 iterations to converge while the solutions for h = 1/128 and h = 1/256 require 1,280 and 2,560 iterations, respectively.

Figures 36 and 37 demonstrate the effects that grid refinement and Motz's method have on the solution near the re-entrant corner. Both show the solution along the segment CD, with point D on the right of the diagram; distance is measured from D.

In Figure 36, the uncorrected solutions approach the extrapolated solution. The two lower lines are projected from points outisde the range of the diagram. Note that the slopes of the lines at D increase as h decreases, but they remain finite. As noted earlier, first derivatives of the solution become undefined at D. Also, the solutions for h = 1/16 and 1/64 are noticeably different than the extrapolated solution.

Figure 37 shows Motz 41-point solutions for h = 1/16 and h = 1/64. The two solutions coincide with the exact solution at corresponding grid points. Thus, Motz's method immediately eliminates much of the error due to the re-entrant corner.

The line segments connecting the solution values in Figure 36 indicate the need for a smooth interpolant near the re-entrant corner. An extremely useful feature of Motz's method is the built-in interpolating function. The solution near the re-entrant corner is of primary importance in a solution, and Motz's method provides an analytical approximation of it.

An attempt was made to determine if solutions from Motz's method had error that could be modeled by equation (6.5), and, if so, what

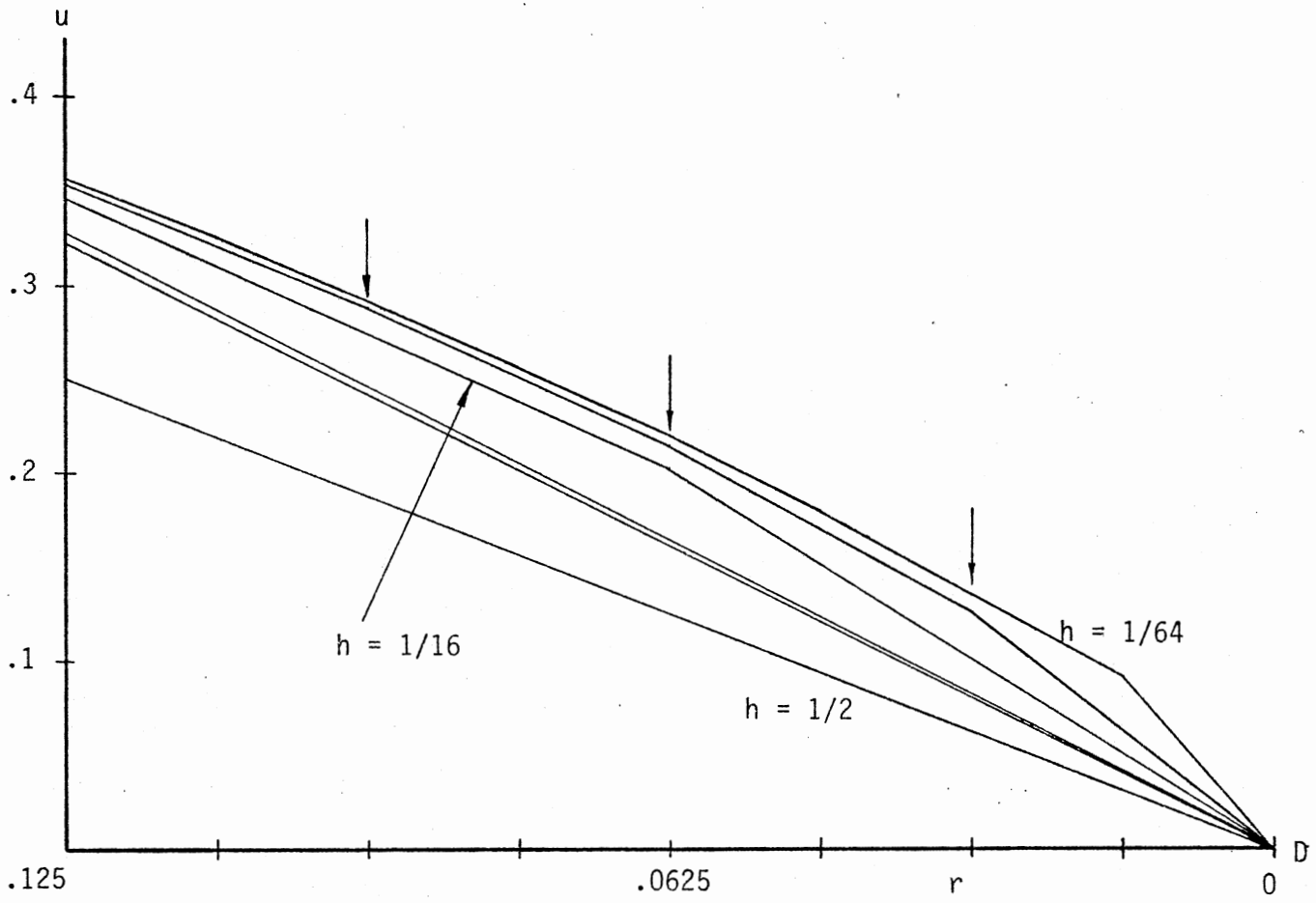


Figure 36. Uncorrected Solutions Along the Edge CD (Figure 28) for h = 1/2 through h = 1/64 (Vertical Arrows Give the Exact Solution)

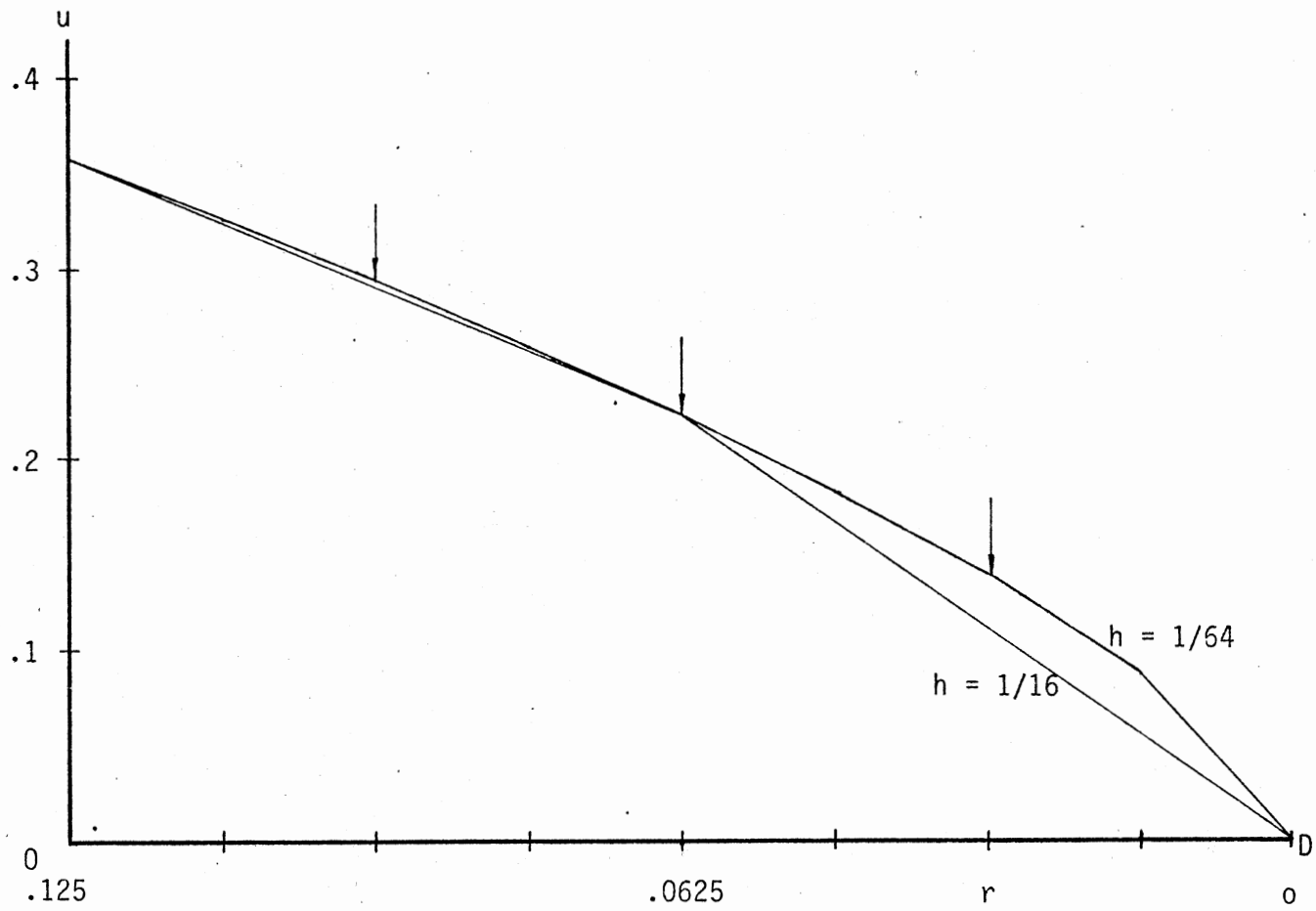


Figure 37. Motz 41-Point Solutions Along the Edge CD (Figure 28) for h = 1/16 and h = 1/64 (Vertical Arrows Indicate the Exact Solution)

the dominant power of h was. The process employed with the uncorrected solution was used with the 8-point and 41-point Motz solutions. Values for α were so widely scattered that no attempt was made to determine β. A method suggested by Reid and Walsh (23) was also used: Motz's method was used within a constant radius of the re-entrant corner for each grid spacing. This method was also unsuccessful.

The inability to determine the error behavior of Motz's method empirically is in keeping with its criticisms (30). No theory of its stability or error behavior exists. An examination of the Motz solutions shows that the h = 1/16 solutions agree with the exact solution to four digits; the h = 1/64 solutions agree to five digits. An optimum grid size seems to be h = 1/32. The density of solution values is adequate for interpolation, and only 110 iterations are required to reduce the residual norm to 1.0×10^{-5}. Single precision can be used throughout the solution process.

Conclusions and Suggestions for Further Study

As demonstrated, re-entrant corners seriously degrade the accuracy afforded by finite difference methods. Two of the methods used here, grid refinement and Motz's method, have been shown to reduce the effects of re-entrant corners. Each has limitations: grid refinement over the entire solution domain is not practical, and Motz's method is not well understood.

A more practical grid refinement scheme appears in Figure 19. It should be considered for study since it does not require large amounts of storage, and because it can be incorporated into multigrid formulations (5) (6) (7). Dr. D. D. Fisher of the Computing and Information

Sciences Department at Oklahoma State University has the program GRIDPACK, an adaptive multigrid routine capable of grid refinement near re-entrant corners and other types of singularities. GRIDPACK should be examined in future studies of numerical solution of PDES and re-entrant corners. It is an outgrowth of the work of Brandt and his associates (6) (7).

Motz's method appears to be well-suited to problems requiring a grid with moderate refinement. It is stable in the formulations used to date, including some that involve extrapolation. Motz's method should be examined as a means of correcting the multigrid Cycle-C process. The solution process on each grid is affected by a re-entrant corner. The routines for performing Motz's method in the program CORNER (Appendix A) could be adapted and used in the Cycle-C routines VBOUND (Appendix B).

BIBLIOGRAPHY

(1) Abramowitz, M. and Stegun, I. Handbook of Mathematical Functions. New York: Dover Publications, Inc., 1972.

(2) Aitken, A. C. "Studies in Practical Mathematics, II. The Evaluation of the Latent Roots and Latent Vectors of a Matrix." Proceedings of the Royal Society of Edinburgh, Vol. 57, Section A (1937), pp. 269-304.

(3) _____. "On the Iterative Solution of a System of Linear Equations." Proceedings of the Royal Society of Edinburgh, Vol. 63, Section A (1950), pp. 52-60.

(4) Ames, William F. Numerical Methods for Partial Differential Equations. 2nd Ed. New York: Academic Press, 1977.

(5) Brandt, Achi. "Multi-Level Adaptive Techniques (MLAT) for Fast Numerical Solution to Boundary-Value Problems." Lecture Notes in Physics: Proceedings of the Third International Conference on Numerical Methods in Fluid Mechanics, Vol. 1, Springer-Verlag, Berlin, 1973.

(6) _____. "Multi-Level Adaptive Solutions to Boundary-Value Problems." Mathematics of Computation, Vol. 31, No. 138 (1977), pp. 333-390.

(7) _____. "Multi-Level Adaptive Techniques (MLAT) for Partial Differential Equations: Ideas and Software." Mathematical Software III, John R. Rice (ed.). New York: Academic Press, 1977.

(8) Carre, B. A. "The Determination of the Optimum Accelerating Factor for Successive Over-Relaxation." Computer Journal, Vol. 4 (1961), pp. 73-77.

(9) Courrant, R., Friedrichs, K., and Lewy, H. "On the Partial Difference Equations of Mathematical Physics." IBM Journal of Research and Development, Vol. 11 (1967), pp. 213-247.

(10) Forsythe, G. E., Malcolm, M. A., and Moler, C. B. Computer Methods for Mathematical Computations. Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1977.

(11) Forsthye, George E. and Wasow, Wolfgang R. Finite-Difference Methods for Partial Differential Equations. New York: John Wiley and Sons, Inc., 1960.

(12) Golub, Gene H. and Varga, Richard S. "Chebyshev Semi-Iterative Methods, Successive Overrelaxation Iterative Methods, and Second Order Richardson Iterative Methods." Numerische Mathematic, Vol. 3 (1961), pp. 147-168.

(13) Hageman, L. A. and Kellogg, R. B. "Estimating Optimum Over-relaxation Parameters." Mathematics of Computation, Vol. 22 (1968), pp. 60-68.

(14) Hageman, L. A. and Young, D. M. Applied Iterative Methods. New York: Academic Press, 1981.

(15) Jackson, John D. Classical Electrodynamics, 2nd Ed. New York: John Wiley and Sons, Inc., 1975.

(16) Jennings, Alan. "Accelerating the Convergence of Matrix Iterative Processes." Journal of the Institute of Mathematics and Its Applications, Vol. 8 (1971), pp. 99-110.

(17) _____. Matrix Computations for Engineers and Scientists. New York: John Wiley and Sons, Inc., 1977.

(18) Kincaid, D. R., Grimes, R. G., Respess, J. R., and Young, D. M. "ITPACK2B: A FORTRAN Package for Solving Large Sparse Linear Systems by Adaptive Accelerated Iterative Methods." Report No. CNA-173, University of Texas Center for Numerical Analysis, 1981.

(19) Morse, Phillip M. and Feshbach, Herman. Methods of Theoretical Physics. New York: McGraw-Hill, 1953.

(20) Motz, H. "The Treatment of Singularities in Relaxation Methods." Quarterly Journal of Applied Mathematics, Vol. 4 (1946), pp. 371-377.

(21) Reid, J. K. "A Method for Finding the Optimum Successive Over-Relaxation Parameter." Computer Journal, Vol. 9 (1966), pp. 200-204.

(22) _____. "On the Method of Conjugate Gradients for the Solution of Large Sparse Systems of Linear Equations." In Large Sparse Sets of Linear Equations. New York: Academic Press, 1971.

(23) Reid, J. K. and Walsh, J. E. "An Elliptic Eigenvalue Problem for a Re-Entrant Region," Journal of the Society for Industrial and Applied Mathematics, Vol. 13, No. 3, 1965.

(24) Richardson, L. F. "The Approximate Arithmetical Solution by Finite Differences of Physical Problems Involving Differential Equations, With an Application to the Stresses in a Masonary Dam." Royal Society of London Philosophical Transactions, Vol. 210, Series A (1910).

(25) _____. "How to Solve Differential Equations Approximately by Arithmetic." The Mathematical Gazette, Vol. 12 (1925), pp. 415-421.

(26) Rigler, A. K. "Estimation of the Successive Over-Relaxation Factor." Mathematics of Computation, Vol. 19 (1965), pp. 302-307.

(27) Shortley, G. H. and Weller, R. "The Numerical Solution of LaPlace's Equation." Journal of Applied Physics, Vol. 9 (1938), pp. 334-348.

(28) Whiteman, J. R. "Singularities Due to Re-Entrant Corners in Harmonic Boundary Value Problems." Report No. MRC-TR-829, University of Wisconsin Mathematics Research Center, 1967.

(29) _____. "Treatment of Singularities in a Harmonic Mixed Boundary Value Problem by Dual Series Methods." Quarterly Journal of Mechanics and Applied Mathematics, Vol. 21 (1968).

(30) _____. "Numerical Solution of a Harmonic Mixed Boundary-Value Problem by the Extension of a Dual Series Method." Quarterly Journal of Mechanics and Applied Mathematics, Vol. 23, Pt. 3 (1970), pp. 449-455.

(31) Woods, L. C. "The Relaxation Treatment of Singular Points in Poisson's Equation." Quarterly Journal of Mechanics and Applied Mathematics, Vol. 6 (1953), pp. 161-185. New York: John Wiley and Sons, Inc., 1960.

(32) Young, David M. Iterative Solution of Large Linear Systems. New York: Academic Press, 1971.

(33) _____. "On the Accelerated SSOR Method for Solving Large Systems." Report No. CNA-92, University of Texas Center for Numerical Analysis, 1974.

APPENDIXES

APPENDIX A

THE PROGRAM CORNER

The program CORNER is included here. It will solve Laplace's equation on any domain that can be represented by a uniform grid. The boundary conditions can be Dirichlet, Neumann, or mixed. At present, the Neumann conditions are restricted to vanishing normal derivatives and boundaries that are parallel to lines of the grid. The solution method can be SOR or SSOR with Aitken acceleration. The GS method is available for the determination of the relaxation parameter for SOR and SSOR; Aitken extrapolation is used to speed its determination. Also, Motz's method is included for use with re-entrant corners with internal angles of 3 /2.

The solution grid is represented in Figure 3. The rows and columns must be numbered from one, though, in the description for CORNER. The program requires a row by row description of the grid; the location and type (Neumann or Dirichlet) must be given for each boundary point on a row of the grid.

Figure 8 shows four computational molecules for use with Neumann conditions. For CORNER, an integer code must be used to identify which molecule must be used on a particular boundary segment. Clockwise from the top the codes for the molecules of Figure 8 are 4, 2, 3, and 1. The corners require special molecules; clockwise from the upper right they are 8, 6, 5, 7. Dirichlet points have the code 0. The data required for the L-shaped region follow the program listing.

Motz's method can be used at several re-entrant corners in a solution domain. The patterns of remote and series points must be given. In the data following the listing, patterns used with the L-shaped region problem are given. They are defined for a corner in the orientation of Figure 17. The location and orientation of each

re-entrant corner must be given. The four corners of a cross-shaped region give the four possible orientations. Clockwise from the upper right they are coded 2, 4, 3, 1 for CORNER.

```
C            CORNERTX,DMOIZ,FORT                                          00000100
C                                                                         00000200
C            DRIVER                                                       00000300
C                                                                         00000400
      IMPLICIT REAL*8 (A-H,O-Z)                                           00000500
      COMMON /TRACE/ IN,LP,NINT,NDBUG                                     00000600
      COMMON /CONTRL/ OMEGA,RESLMT,KSWEEP,MAXIT,KOMEGA                    00000700
      INTEGER LEFT,RIGHT                                                  00000800
      COMMON /BOUND/ XMIN,XMAX,YMIN,YMAX,H,                               00000900
     $     MAP(65,65),LEFT(101),RIGHT(101),NROW,NCOL,NPT                00001000
      COMMON /AITKEN/COSTH,SMAX,SMIN,KOUNT,KPER,METHD,                    00001100
     $     NPREP,NPER                                                   00001200
C                                                                         00001300
C                                                                         00001400
      IN=12                                                               00001500
      LP=6                                                                00001600
      NDBUG=0                                                             00001700
      NINT=20                                                             00001800
C                                                                         00001900
      COSTH=.995D0                                                        00002000
      SMAX=50.D0                                                          00002100
      SMIN=0.D0                                                           00002200
      KPER=0                                                              00002300
      NPER=1                                                              00002400
      METHD=3                                                             00002500
C                                                                         00002600
C                                                                         00002700
      CALL AITRAT                                                         00002800
C                                                                         00002900
      STOP                                                                00003000
      END                                                                 00003100
C                                                                         00003200
C                                                                         00003300
C                                                                         00003400
C                                                                         00003500
      FUNCTION F(X,Y,HTOL)                                                00003600
      IMPLICIT REAL*8 (A-H,O-Z)                                           00003700
C                                                                         00003800
      F=0.                                                                00003900
      IF(X.LT.HTOL)F=1.0D0                                                00004000
      IF(Y.LT.HTOL)F=-1.0D0                                               00004100
C                                                                         00004200
      RETURN                                                              00004300
      END                                                                 00004400
C                                                                         00004500
C                                                                         00004600
C                                                                         00004700
C                                                                         00004800
      SUBROUTINE AITRAT                                                   00004900
C                                                                         00005000
C                                                                         00005100
C                                                                         00005200
      IMPLICIT REAL*8 (A-H,O-Z)                                           00005300
      COMMON /TRACE/ IN,LP,NINT,NDBUG                                     00005400
      COMMON /CONTRL/ OMEGA,RESLMT,KSWEEP,MAXIT,KOMEGA                    00005500
      COMMON /STORE/ U(65,65),V(65,65)                                    00005600
      INTEGER LEFT,RIGHT                                                  00005700
      COMMON /BOUND/ XMIN,XMAX,YMIN,YMAX,H,                               00005800
     $     MAP(65,65),LEFT(101),RIGHT(101),NROW,NCOL,NPT                00005900
      INTEGER SPT,SROW,SCOL,EPT,EROW,ECOL                                 00006000
      COMMON /SNG/ SMAT(50,15),EMAT(15,15),EXPVAL(10,15),SCALE(15),       00006100
     $     B(15),SROW(100),SCOL(100),EROW(100),ECOL(100),SPT(10),         00006200
     $     EPT(10),IPVT(15),NSING                                         00006300
      COMMON /AITKEN/COSTH,SMAX,SMIN,KOUNT,KPER,METHD,                    00006400
     $     NPREP,NPER                                                   00006500
C                                                                         00006600
      KOUNT=0                                                             00006700
      NGREAS=0                                                            00006710
C                                                                         00006800
      READ(IN,5)MAXIT                                                     00006900
      RESLMT=1.0D-14                                                      00007000
      READ(IN,5)KSWEEP                                                    00007100
      READ(IN,5)KOMEGA                                                    00007200
      READ(IN,8)OMEGA                                                     00007300
      READ(IN,5)NPREP                                                     00007400
C                                                                         00007500
      WRITE(LP,40)MAXIT,KSWEEP,KOMEGA,NPREP,RESLMT,OMEGA                  00007600
C                                                                         00007700
C                                                                         00007800
      CALL BNDRY                                                          00007900
C                                                                         00008000
      CALL SETPIR                                                         00008100
C                                                                         00008200
      CALL SNGLOC                                                         00008300
C                                                                         00008400
      CALL PLOTB                                                          00008500
C                                                                         00008600
      CALL INIT                                                           00008700
C                                                                         00008800
C                                                                         00008900
      CALL PLOT(U)                                                        00009000
C                                                                         00009100
C                                                                         00009200
C                                                                         00009300
      IF(KOMEGA.NE.0)OMEGA=1.0                                            00009400
      RESPRV=100.D0                                                       00009500
      SPECT0=.100                                                         00009600
      SPECT1=.400                                                         00009700
C                                                                         00009800
      DO 400 J=1,MAXIT                                                    00009900
C                                                                         00010000
C                                                                         00010100
        CALL SWEEP(KSWEEP,OMEGA,RESAVE,RESLOG)                            00010200
C                                                                         00010300
        IF(KOMEGA.EQ.0)GO TO 380                                          00010400
C                                                                         00010500
C                                                                         00010600
          SPECT2=RESAVE/RESPRV                                            00010700
C                                                                         00010800
          SPECIN=SPECT2-(SPECT2-SPECT1)**2/                               00010900
     $         (SPECT0-2.D0*SPECT1+SPECT2)                                00011000
C                                                                         00011100
          SPECT0=SPECT1                                                   00011200
          SPECT1=SPECT2                                                   00011300
C                                                                         00011500
          OMEGA=2.D0/(1.D0+DSQRT(1.D0-SPECIN))                            00011600
C                                                                         00011700
C                                                                         00011800
  380   WRITE(LP,10)J,RESAVE,RESLOG                                       00011900
        IF(KOMEGA.NE.0)WRITE(LP,12)SPECIN,OMEGA                           00012000
C                                                                         00012100
C                                                                         00012200
        IF(RESAVE.LT.RESLMT) GO TO 500                                    00012300
C                                                                         00012310
        IF(NGREAS.GT.10)GO TO 500                                         00012320
C                                                                         00012330
```

```
            NCREAS=NCREAS+1                                              00012340
            IF(RESAVE.LT.RESPRV)NCREAS=0                                 00012350
C                                                                        00012360
C                                                                        00012400
            RESPRV=RESAVE                                                00012410
C                                                                        00012420
            IF(NDBUG.GE.50)CALL PLOT(U)                                  00012500
            IF(NDBUG.GE.50)CALL RESULT(16,18)                            00012600
C                                                                        00012700
            CALL VALFK(METHD,NPREP,NPER,KOUNT,KPER,COSTH,SMAX,SMIN,      00012800
     $           COSIN,SRAW,S)                                           00012900
C                                                                        00013000
            WRITE(LP,15)COSIN,SRAW,S                                     00013100
C                                                                        00013200
 400        CONTINUE                                                     00013300
C           WRITE(LP,20)                                                 00013400
C                                                                        00013500
C                                                                        00013600
 500        CONTINUE                                                     00013700
C                                                                        00013800
            CALL PLOT(U)                                                 00013900
C           CALL RESULT(1,NROW)                                          00014000
C                                                                        00014100
C                                                                        00014200
            STOP                                                         00014300
C                                                                        00014400
C                                                                        00014500
  2         FORMAT(10X,1215)                                             00014600
  8         FORMAT(10X,2D25.16)                                          00014700
 10         FORMAT(/6X,'RES.-NO,AVE,LOG',I5,3E24.16)                     00014800
 12         FORMAT(6X,'SPECIRAL RAD,OMEGA',2X,2E24.16)                   00014900
 15         FORMAT(6X,'COSIN,SRAW,S',8X,3E24.16)                         00015000
 20         FORMAT(6X,'LIMIT WAS NOT REACHED')                           00015100
 40         FORMAT(//6X,'-- ITRAT --'//11X,'MAXIT,KSWEEP,KOMEGA,NPREP',  00015200
     $        4I10//11X,'RES-LIMIT,OMEGA',2E24.16//)                    00015300
            END                                                          00015400
C                                                                        00015500
C                                                                        00015600
C                                                                        00015700
C                                                                        00015800
            SUBROUTINE BNDRY                                             00015900
C                                                                        00016000
            IMPLICIT REAL*8 (A-H,O-Z)                                    00016100
            INTEGER BLOC,BTYP,L,R                                        00016200
            DIMENSION BLOC(101),BTYP(101)                                00016300
            INTEGER LEFT,RIGHT                                           00016400
            COMMON /BOUND/ XMIN,XMAX,YMIN,YMAX,H,                        00016500
     $           MAP(65,65),LEFT(101),RIGHT(101),NROW,NCOL,NPT           00016600
            COMMON /TRACE/ IN,LP,NINT,NDBUG                              00016700
            DATA NEPAG/-3/                                               00016800
C                                                                        00016900
            NPT=0                                                        00017000
C                                                                        00017100
            READ(IN,25)NROW,NCOL                                         00017200
            READ(IN,30)XMIN,XMAX                                         00017300
            READ(IN,30)YMIN,YMAX                                         00017400
C                                                                        00017500
            H=(XMAX-XMIN)/DFLOAT(NCOL-1)                                 00017600
C                                                                        00017700
            WRITE(LP,40)NROW,NCOL,XMIN,XMAX,YMIN,YMAX,H                  00017800
C                                                                        00017900
            DO 200 J=1,NROW                                              00018000
C                                                                        00018100
                                                                         00018200
                                                                         00018300
            READ(IN,25)L,R,NBPT                                          00018400
C                                                                        00018500
            DO 120 K=L,R                                                 00018600
               MAP(J,K)=NFLAG                                            00018700
 120        CONTINUE                                                     00018800
C                                                                        00018900
            NPT=NPT+(R-L)+1                                              00019000
            LEFT(J)=L                                                    00019100
            RIGHT(J)=R                                                   00019200
C                                                                        00019300
C                                                                        00019400
            READ(IN,25)(BLOC(K),K=1,NBPT)                                00019500
            READ(IN,25)(BTYP(K),K=1,NBPT)                                00019600
C                                                                        00019700
            IF(NDBUG.GE.100)WRITE(LP,50)J,LEFT(J),RIGHT(J),NBPT,         00019800
     $           (BLOC(K),BTYP(K),K=1,NBPT)                              00019900
C                                                                        00020000
C                                                                        00020100
            DO 140 K=1,NBPT                                              00020200
               L=BLOC(K)                                                 00020300
               MAP(J,L)=BTYP(K)                                          00020400
 140        CONTINUE                                                     00020500
C                                                                        00020600
C                                                                        00020700
 200    CONTINUE                                                         00020800
C                                                                        00020900
C                                                                        00021000
C                                                                        00021100
            RETURN                                                       00021200
C                                                                        00021300
 25         FORMAT(10X,1215)                                             00021400
 30         FORMAT(10X,4E15.7)                                           00021500
 40         FORMAT(1H1,5X,'-- BNDRY --'/11X,'NROW,NCOL',2I5/             00021600
     $        10X,'XMIN,XMAX,YMIN,YMAX,H',5E15.7)                       00021700
 50         FORMAT(11X,'ROW,L,R,NBPT',4I5/                               00021800
     $        18X,'BLOC,BTYP'/(16X,2I5))                                00021900
C           END                                                          00022000
                                                                         00022100
C                                                                        00022200
C                                                                        00022300
C                                                                        00022400
C                                                                        00022500
            SUBROUTINE SETPIR                                            00022600
C                                                                        00022700
C                                                                        00022800
            IMPLICIT REAL*8 (A-H,O-Z)                                    00022900
            COMMON /TRACE/ IN,LP,NINT,NDBUG                              00023000
            INTEGER LEFT,RIGHT                                           00023100
            COMMON /BOUND/ XMIN,XMAX,YMIN,YMAX,H,                        00023200
     $           MAP(65,65),LEFT(101),RIGHT(101),NROW,NCOL,NPT           00023300
            INTEGER CPIMIN,CPIMAX                                        00023400
            COMMON /SIMAP/ KTOP(65),KBOT(65),CPIMIN(65),CPIMAX(65)       00023500
            DATA MAXROW/65/,MAXCOL/65/                                   00023600
C                                                                        00023700
C           SET UP COLUMN POINTERS FOR VALFK                             00023800
C                                                                        00023900
            IF(NDBUG.GE.100)WRITE(LP,10)                                 00024000
C                                                                        00024100
            KOVER=MAXROW+1                                               00024200
            KUNDER=0                                                     00024300
            DO 280 K=1,NCOL                                              00024400
C                                                                        00024500
               KTOP(K)=KOVER                                             00024600
               KBOT(K)=KUNDER                                            00024700
C                                                                        00024800
```

```
 280      CONTINUE                                                       00024900
C                                                                       00025000
C                                                                       00025100
         DO 400 J=1,NROW                                                00025200
C                                                                       00025300
              KL=LEFT(J)                                                00025400
              KR=RIGHT(J)                                               00025500
C                                                                       00025600
              DO 300 K=KL,KR                                            00025700
C                                                                       00025800
                   IF(KTOP(K).GT.J)KTOP(K)=J                            00025900
                   IF(KBOT(K).LT.J)KBOT(K)=J                            00026000
C                                                                       00026100
 300          CONTINUE                                                  00026200
C                                                                       00026300
 400      CONTINUE                                                       00026400
C                                                                       00026500
C                                                                       00026600
         DO 600 K=1,NCOL                                                00026700
C                                                                       00026800
              KOFFST=MAXROW*(K-1)                                       00026900
C                                                                       00027000
              CPIMIN(K)=KOFFST+KTOP(K)                                  00027100
              CPIMAX(K)=KOFFST+KBOT(K)                                  00027200
C                                                                       00027300
              IF(NDBUG.GE.100)WRITE(LP,20)KTOP(K),KBOT(K),CPIMIN(K),    00027400
     $                                    CPIMAX(K)                     00027500
C                                                                       00027600
 600      CONTINUE                                                       00027700
C                                                                       00027800
C                                                                       00027900
         RETURN                                                         00028000
C                                                                       00028100
  10     FORMAT(//6X,'-- SLTPTR --'/)                                   00028200
  20     FORMAT(11X,'KTOP,KBOT,CPIMIN,CPIMAX',4I5)                      00028300
C                                                                       00028400
         END                                                            00028500
C                                                                       00028600
C                                                                       00028700
C                                                                       00028800
C                                                                       00028900
         SUBROUTINE SNGLOC                                              00029000
C                                                                       00029100
         IMPLICIT REAL*8 (A-H,O-Z)                                      00029200
         INTEGER SPAIR,SPAIC,EPAIR,EPAIC                                00029300
         COMMON /PATTRN/ NS,SPATR(50),SPATC(50),NE,EPAIR(15),EPATC(15)  00029400
         INTEGER LEFT,RIGHT                                             00029500
         COMMON /BOUND/ XMIN,XMAX,YMIN,YMAX,H,                          00029600
     $        MAP(65,65),LEFT(101),RIGHT(101),NROW,NCOL,NPI             00029700
         INTEGER SPI,SROW,SCOL,EPI,EROW,ECOL                            00029800
         COMMON /SNG/ SMAT(50,15),EMAT(15,15),EXPVAL(10,15),SCALE(15),  00029900
     $        B(15),SROW(100),SCOL(100),EROW(100),ECOL(100),SPT(10),    00030000
     $        EPI(10),IPVT(15),NSING                                    00030100
         COMMON /TRACE/ IN,LP,NINI,NDBUG                                00030200
         INTEGER SR,SC,SNGTYP,SPOS,EPOS                                 00030300
         DIMENSION STHETA(50),SRAD(50),ETHETA(15),ERAD(15)              00030400
         INTEGER SFLAG,EFLAG,EMAXR,EMAXC,SMAXR,SMAXC                    00030500
         DATA SFLAG,EFLAG,EMAXR,EMAXC,SMAXR,SMAXC/-1,-2,15,15,50,15/,   00030600
     $        NFLAG/-3/                                                 00030700
C                                                                       00030800
         PI=DATAN(1.0D0)*4.D0                                           00030900
C                                                                       00031000
C                                                                       00031100
         READ(IN,10)NSING                                               00031200
         WRITE(LP,50)NSING                                              00031300
C                                                                       00031400
         IF(NSING.LT.1)RETURN                                           00031500
C                                                                       00031600
C             DETERMINE WHICH POINTS ARE TO BE REPLACED WITH SERIES     00031700
C             ESTIMATES AND WHICH WILL SERVE AS EXTREME POINTS.         00031800
C                                                                       00031900
         CALL SHAPE                                                     00032000
C                                                                       00032100
         SPOS=1                                                         00032200
         EPOS=1                                                         00032300
C                                                                       00032400
C                                                                       00032500
C                                                                       00032600
         DO 200 J=1,NSING                                               00032700
C                                                                       00032800
              READ(IN,10)SR,SC,SNGTYP                                   00032900
              IF(NDBUG.GE.100)WRITE(LP,60)J,SR,SC,SNGTYP                00033000
C                                                                       00033100
              SPI(J)=SPOS                                               00033200
              EPI(J)=EPOS                                               00033300
C                                                                       00033400
              WRITE(LP,70)                                              00033500
              CALL ROTRAN(SPAIC,SPAIR,SCOL(SPOS),SROW(SPOS),            00033600
     $                    NS,SC,SR,SNGTYP)                              00033700
C                                                                       00033800
              WRITE(LP,75)                                              00033900
              CALL ROTRAN(EPAIC,EPAIR,ECOL(EPOS),EROW(EPOS),            00034000
     $                    NE,SC,SR,SNGTYP)                              00034100
C                                                                       00034200
              SPOS=SPOS+NS                                              00034300
              EPOS=EPOS+NE                                              00034400
C                                                                       00034500
C                                                                       00034600
 200      CONTINUE                                                       00034700
C                                                                       00034800
C             UPDATE MAP TO REFLECT EXTREME AND SPECIAL POINTS           00034900
C                                                                       00035000
         LMAX=SPOS-1                                                    00035100
         DO 400 L=1,LMAX                                                00035200
              J=SROW(L)                                                 00035300
              K=SCOL(L)                                                 00035400
              MAP(J,K)=SFLAG                                            00035500
 400      CONTINUE                                                       00035600
C                                                                       00035700
C                                                                       00035800
         LMAX=EPOS-1                                                    00035900
         DO 500 L=1,LMAX                                                00036000
              J=EROW(L)                                                 00036100
              K=ECOL(L)                                                 00036200
              IF(MAP(J,K).EQ.NFLAG)MAP(J,K)=EFLAG                       00036300
 500      CONTINUE                                                       00036400
C                                                                       00036500
C             FORM THE POLAR COORDINATES OF THE SPECIAL AND              00036600
C             EXTREME POINT LOCATIONS                                   00036700
C                                                                       00036800
         CALL POLAR(SPATC,SPATR,STHETA,SRAD,NS,H)                       00036900
         CALL POLAR(EPAIC,EPAIR,ETHETA,ERAD,NE,H)                       00037000
C                                                                       00037100
C             FILL EMAT AND SMAT                                        00037200
C                                                                       00037300
         WRITE(LP,80)                                                   00037400
         CALL MATCAL(SMAT,SMAXR,SMAXC,NS,NE,SRAD,STHETA)                00037500
C                                                                       00037600
         WRITE(LP,85)                                                   00037700
         CALL MATCAL(EMAT,EMAXR,EMAXC,NE,NE,ERAD,ETHETA)                00037800
```

```
C
C
      CALL DECOMP(EMAXR,NE,EMAT,COND,IPVT,SCALE,B)
C
      WRITE(LP,30)COND,(SCALE(J),J=1,NE)
      CONDP1=COND+1.
C
      IF(CONDP1.NE.COND)GO TO 900
           WRITE(LP,40)
           STOP
C
C
  900 RETURN
C
   10 FORMAT(10X,1215)
   30 FORMAT(//11X,'CONDITION',E24.16/11X,'SCALE',4E24.16/
     $       (16X,4E24.16))
   40 FORMAT(//11X,'*** MATRIX IS SINGULAR TO WORKING PRECISION ***')
   50 FORMAT(//6X,'-- SNGLOC --'//11X,'NSING',I5)
   60 FORMAT(11X,'SING NO.,ROW,COL,SING TYPE',4I5)
   70 FORMAT(18X,'SROW,SCOL'/)
   75 FORMAT(18X,'EROW,ECOL'/)
   80 FORMAT(//41X,'SMAT')
   85 FORMAT(//41X,'EMAT')
C
      END
      SUBROUTINE SHAPE
C
      IMPLICIT REAL*8 (A-H,O-Z)
      INTEGER SPAIR,SPAIC,EPAIR,EPAIC
      COMMON /PATTRN/ NS,SPAIR(50),SPAIC(50),NE,EPAIR(15),EPAIC(15)
      COMMON /TRACE/IN,LP,NINT,NDBUG
C
      READ(IN,20)NS
      READ(IN,20)(SPAIC(J),J=1,NS)
      READ(IN,20)(SPAIR(J),J=1,NS)
      WRITE(LP,50)NS,(SPAIC(J),SPAIR(J),J=1,NS)
C
      READ(IN,20)NE
      READ(IN,20)(EPAIC(J),J=1,NE)
      READ(IN,20)(EPAIR(J),J=1,NE)
C
      WRITE(LP,60)NE,(EPAIC(J),EPAIR(J),J=1,NE)
      RETURN
C
   20 FORMAT(10X,1215)
   50 FORMAT(//6X,'-- SHAPE --'//11X,'REPLACEMENT PATTERN (',I5,
     $       ' POINTS)'/(16X,2I5))
   60 FORMAT(//11X,'EXTREME POINT PATTERN (',I5,' POINTS)'/
     $       (16X,2I5))
C
      END
C
C
C
C
C
C
C
C
      SUBROUTINE ROTRAN(IX,IY,NEWX,NEWY,NPT,LOCX,LOCY,STYP)
C
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON /TRACE/IN,LP,NINT,NDBUG
      DIMENSION IX(NPT),IY(NPT),NEWX(NPT),NEWY(NPT)
      INTEGER STYP,ROTATE
      DIMENSION ROTATE(4,2,2)
      DATA ROTATE/-1,0,0,1, 0,1,-1,0, 0,-1,1,0, -1,0,0,1/
C
      MX=ROTATE(STYP,1,1)
      MY=ROTATE(STYP,1,2)
      NX=ROTATE(STYP,2,1)
      NY=ROTATE(STYP,2,2)
C
C
      DO 400 J=1,NPT
C
           NEWX(J)=(IX(J)*MX+IY(J)*MY)+LOCX
           NEWY(J)=(IX(J)*NX+IY(J)*NY)+LOCY
C
           WRITE(LP,20)NEWY(J),NEWX(J)
   20      FORMAT(16X,2I5)
C
  400 CONTINUE
C
C
      RETURN
C
      END
C
C
C
C
      SUBROUTINE POLAR(IX,IY,THETA,RADIUS,NPT,N)
C
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON /TRACE/  IN,LP,NINT,NDBUG
      DIMENSION IX(NPT),IY(NPT),THETA(NPT),RADIUS(NPT)
C
      TWOPI=8.D0*DATAN(1.0D0)
C
      DO 200 J=1,NPT
C
           NX=IX(J)
           NY=IY(J)
C
           THETA(J)=0.0D0
           RADIUS(J)=0.0D0
C
           IF(NX.EQ.0 .AND. NY.EQ.0)GO TO 200
                THETA(J)=DATAN2(DFLOAT(NY),DFLOAT(NX))
                RADIUS(J)=DSQRT(DFLOAT(NX**2+NY**2))
                IF(THETA(J).LT.0.0D0)THETA(J)=TWOPI+THETA(J)
C
  200 CONTINUE
C
      IF(NDBUG.GT.50)WRITE(LP,50)(J,IX(J),IY(J),RADIUS(J),THETA(J),
     $       J=1,NPT)
C
      RETURN
C
   50 FORMAT(//6X,'-- POLAR --'/21X,'J,IX,IY,RADIUS,THETA'/
     $       (11X,3I5,2E24.16))
C
      END
C
C
C
C
      SUBROUTINE MATCAL(XMAT,MAXR,MAXC,NR,NC,XRAD,XTHETA)
```

```
C
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON /TRACE/IN,LP,NINI,NDBUG
      DIMENSION XMAT(MAXR,MAXC),XRAD(NR),XTHETA(NR)
      INTEGER LEFT,RIGHT
      COMMON /BOUND/ XMIN,XMAX,YMIN,YMAX,H,
     $      MAP(65,65),LEFT(101),RIGHT(101),NROW,NCOL,NPT
C
      C=2.D0/3.D0
      DO 600 J=1,NR
C
            THETA=XTHETA(J)
            RAD=XRAD(J)
C
            KMIN=1
            IF(RAD.GT..5D0)GO TO 480
C
                  KMIN=2
                  XMAT(J,1)=1.0D0
C
  480       RAD=RAD**C
C
            DO 500 K KMIN,NC
C
                  N=K-1
                  XMAT(J,K)=(RAD**N)*DCOS(C*DFLOAT(N)*THETA)
  500       CONTINUE
C
            WRITE(LP,20)J,(XMAT(J,K),K=1,NC)
  600 CONTINUE
C
      RETURN
C
   20       FORMAT(/4X,'ROW',I4,4E24.16/(11X,4E24.16))
C
      END
C
C
C
C
      SUBROUTINE INIT
C
C
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON /STORE/U(65,65),V(65,65)
      INTEGER LEFT,RIGHT
      COMMON /BOUND/ XMIN,XMAX,YMIN,YMAX,H,
     $      MAP(65,65),LEFT(101),RIGHT(101),NROW,NCOL,NPT
C
      HTOI=H**.5D0
C
      DO 200 J=1,NROW
C
            Y=YMIN+DFLOAT(J-1)*H
            KMIN=LEFT(J)
            KMAX=RIGHT(J)
C
            DO 180 K=KMIN,KMAX
C
                  X=XMIN+DFLOAT(K-1)*H
                  U(J,K)=F(X,Y,HTOI)
  180       CONTINUE
C
  200 CONTINUE
C
```

```
      RETURN
C
      END
C
C
C
      SUBROUTINE DECOMP (NDIM,N,A,COND,IPVT,SCALE,WORK)
C
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(NDIM,NDIM),WORK(N),IPVT(N),SCALE(N)
C
C  DECOMPOSES A REAL MATRIX BY GAUSSIAN ELIMINATION,
C  AND ESTIMATES THE CONDITION OF THE MATRIX.
C
C  -COMPUTER METHODS FOR MATHEMATICAL COMPUTATIONS-, BY G. E. FORSYTHE,
C       M. A. MALCOLM, AND C. B. MOLER (PRENTICE-HALL, 1977)
C
C  USE SUBROUTINE SOLVE TO COMPUTE SOLUTIONS TO LINEAR SYSTEMS.
C
C  INPUT..
C
C     NDIM = DECLARED ROW DIMENSION OF THE ARRAY CONTAINING  A
C     N    = ORDER OF THE MATRIX
C     A    = MATRIX TO BE TRIANGULARIZED
C
C  OUTPUT..
C
C     A  CONTAINS AN UPPER TRIANGULAR MATRIX  U  AND A PERMUTED
C       VERSION OF A LOWER TRIANGULAR MATRIX  I-L  SO THAT
C       (PERMUTATION MATRIX)*A = L*U
C
C     COND = AN ESTIMATE OF THE CONDITION OF  A .
C       FOR THE LINEAR SYSTEM  A*X = B , CHANGES IN  A  AND  B
C       MAY CAUSE CHANGES  COND  TIMES AS LARGE IN  X.
C       IF  COND+1.0 .EQ. COND , A IS SINGULAR TO WORKING
C       PRECISION.  COND  IS SET TO  1.0E+32  IF EXACT
C       SINGULARITY IS DETECTED.
C
C     IPVT       = THE PIVOT VECTOR
C       IPVT(K) = THE INDEX OF THE K-TH PIVOT ROW
C       IPVT(N) = (-1)**(NUMBER OF INTERCHANGES)
C
C  WORK SPACE..   THE VECTOR  WORK  MUST BE DECLARED AND INCLUDED
C       IN THE CALL.  ITS INPUT CONTENTS ARE IGNORED.
C       ITS OUTPUT CONTENTS ARE USUALLY UNIMPORTANT.
C
C  THE DETERMINANT OF  A  CAN BE OBTAINED ON OUTPUT BY
C     DET(A) = IPVT(N) * A(1,1) * A(2,2) * ... * A(N,N) .
C
C
C
      XABS(Z)=DABS(Z)
      XFLOAT(IZ)=DFLOAT(IZ)
C
C
C        PERFORM COLUMN SCALING
C
C
      CALL CSCALE(NDIM,N,A,SCALE)
C
      IPVT(N)=1
      IF(N .EQ. 1) GO TO 150
      NM1=N-1
C                        COMPUTE THE 1-NORM OF  A .
      ANORM=0.0D0
```

```
      DO 20 J=1,N                                                       00063900
        T=0.0D0                                                         00064000
        DO 10 I=1,N                                                     00064100
   10     T=T+XABS(A(I,J))                                             00064200
        IF(T .GT. ANORM) ANORM=T                                        00064300
   20   CONTINUE                                                        00064400
C                               DO GAUSSIAN ELIMINATION WITH PARTIAL    00064500
C                                    PIVOTING.                          00064600
        DO 70 K=1,NM1                                                   00064700
          KP1=K+1                                                       00064800
C                               FIND THE PIVOT.                         00064900
          M=K                                                           00065000
          DO 30 I=KP1,N                                                 00065100
            IF(XABS(A(I,K)) .GT. XABS(A(M,K))) M=I                      00065200
   30       CONTINUE                                                    00065300
          IPVT(K)=M                                                     00065400
          IF(M .NE. K) IPVT(N)=-IPVT(N)                                 00065500
          T=A(M,K)                                                      00065600
          A(M,K)=A(K,K)                                                 00065700
          A(K,K)=T                                                      00065800
C                               SKIP THE ELIMINATION STEP IF PIVOT IS ZERO.00065900
          IF(T .EQ. 0.0D0) GO TO 70                                     00066000
C                                                                       00066100
C                               COMPUTE THE MULTIPLIERS.                00066200
          DO 40 I=KP1,N                                                 00066300
   40       A(I,K)=-A(I,K)/T                                            00066400
C                               INTERCHANGE AND ELIMINATE BY COLUMNS.   00066500
          DO 60 J=KP1,N                                                 00066600
            T=A(M,J)                                                    00066700
            A(M,J)=A(K,J)                                               00066800
            A(K,J)=T                                                    00066900
            IF(T .EQ. 0.0D0) GO TO 60                                   00067000
            DO 50 I=KP1,N                                               00067100
              A(I,J)=A(I,J)+A(I,K)*T                                    00067200
   50       CONTINUE                                                    00067300
   60       CONTINUE                                                    00067400
   70     CONTINUE                                                      00067500
C                                                                       00067600
C  COND = (1-NORM OF A)*(AN ESTIMATE OF THE 1-NORM OF A-INVERSE)        00067700
C  THE ESTIMATE IS OBTAINED BY ONE STEP OF INVERSE ITERATION FOR THE    00067800
C  SMALL SINGULAR VECTOR.  THIS INVOLVES SOLVING TWO SYSTEMS            00067900
C  OF EQUATIONS,  (A-TRANSPOSE)*Y = E  AND  A*Z = Y  WHERE  E           00068000
C  IS A VECTOR OF +1 OR -1 COMPONENTS CHOSEN TO CAUSE GROWTH IN Y.      00068100
C  ESTIMATE = (1-NORM OF Z)/(1-NORM OF Y)                               00068200
C                                                                       00068300
C                               SOLVE  (A-TRANSPOSE)*Y = E .            00068400
        DO 190 K=1,N                                                    00068500
          T=0.0D0                                                       00068600
          IF(K .EQ. 1) GO TO 90                                         00068700
          KM1=K-1                                                       00068800
          DO 80 I=1,KM1                                                 00068900
   80       T=T+A(I,K)*WORK(I)                                          00069000
   90     EK=1.0                                                        00069100
          IF(T .LT. 0.0D0) EK=-1.0                                      00069200
          IF(A(K,K) .EQ. 0.0D0) GO TO 160                               00069300
  100     WORK(K)=-(EK+T)/A(K,K)                                        00069400
        DO 120 KB=1,NM1                                                 00069500
          K=N-KB                                                        00069600
          T=0.0D0                                                       00069700
          KP1=K+1                                                       00069800
          DO 110 I=KP1,N                                                00069900
  110       T=T+A(I,K)*WORK(K)                                          00070000
          WORK(K)=T                                                     00070100
          M=IPVT(K)                                                     00070200
          IF(M .EQ. K) GO TO 120                                        00070300
          T=WORK(M)                                                     00070400
          WORK(M)=WORK(K)                                               00070500
          WORK(K)=T                                                     00070600
  120     CONTINUE                                                      00070700
C                                                                       00070800
        YNORM=0.0D0                                                     00070900
        DO 130 I=1,N                                                    00071000
  130     YNORM=YNORM+XABS(WORK(I))                                     00071100
C                                                                       00071200
C                                 SOLVE  A*Z = Y .                      00071300
        CALL SOLVE (NDIM,N,A,WORK,IPVT,SCALE,0)                         00071400
C                                                                       00071500
        ZNORM=0.0D0                                                     00071600
        DO 140 I=1,N                                                    00071700
  140     ZNORM=ZNORM+XABS(WORK(I))                                     00071800
C                                                                       00071900
C                                 ESTIMATE THE CONDITION.               00072000
        COND=ANORM*ZNORM/YNORM                                          00072100
        IF(COND .LT. 1.0) COND=1.0                                      00072200
        RETURN                                                          00072300
C                                 1-BY-1 CASE..                         00072400
  150   COND=1.0D0                                                      00072500
        IF(A(1,1) .NE. 0.0D0) RETURN                                    00072600
C                                                                       00072700
C                                 EXACT SINGULARITY..                   00072800
  160   COND=1.0D32                                                     00072900
        RETURN                                                          00073000
        END                                                             00073100
C                                                                       00073200
C                                                                       00073300
C                                                                       00073400
C                                                                       00073500
      SUBROUTINE CSCALE(NDIM,N,A,SCALE)                                 00073600
C                                                                       00073700
      IMPLICIT REAL*8 (A-H,O-Z)                                         00073800
      DIMENSION A(NDIM,NDIM),SCALE(N)                                   00073900
C                                                                       00074000
C                                                                       00074100
C     COMPUTE COLUMN NORMS                                              00074200
C                                                                       00074300
      DO 400 JCOL=1,N                                                   00074400
C                                                                       00074500
        ASUM=0.0D0                                                      00074600
C                                                                       00074700
        DO 200 JROW=1,N                                                 00074800
          ASUM=ASUM+DABS(A(JROW,JCOL))                                  00074900
  200   CONTINUE                                                        00075000
C                                                                       00075100
        SFACT=DFLOAT(N)/ASUM                                            00075200
        SCALE(JCOL)=SFACT                                               00075300
C                                                                       00075400
        DO 300 JROW=1,N                                                 00075500
          A(JROW,JCOL)=A(JROW,JCOL)*SFACT                               00075600
  300   CONTINUE                                                        00075700
C                                                                       00075800
  400 CONTINUE                                                          00075900
C                                                                       00076000
      RETURN                                                            00076100
C                                                                       00076200
      END                                                               00076300
C                                                                       00076400
C                                                                       00076500
C                                                                       00076600
      SUBROUTINE SOLVE (NDIM,N,A,B,IPVT,SCALE,MODE)                     00076700
C                                                                       00076800
      IMPLICIT REAL*8 (A-H,O-Z)
```

```
      DIMENSION A(NDIM,NDIM),B(N),IPVT(N),SCALE(N)                      00076900
C                                                                       00077000
C  SOLVES A LINEAR SYSTEM,  A*X = B .                                   00077100
C  DO NOT SOLVE THE SYSTEM IF DECOMP HAS DETECTED SINGULARITY.          00077200
C                                                                       00077300
C  -COMPUTER METHODS FOR MATHEMATICAL COMPUTATIONS-, BY G. E. FORSYTHE, 00077400
C       M. A. MALCOLM, AND C. B. MOLER (PRENTICE-HALL, 1977)            00077500
C                                                                       00077600
C  INPUT..                                                              00077700
C                                                                       00077800
C     NDIM = DECLARED ROW DIMENSION OF ARRAY CONTAINING  A              00077900
C     N    = ORDER OF MATRIX                                            00078000
C     A    = TRIANGULARIZED MATRIX OBTAINED FROM SUBROUTINE DECOMP      00078100
C     B    = RIGHT HAND SIDE VECTOR                                     00078200
C     IPVT = PIVOT VECTOR OBTAINED FROM DECOMP                          00078300
C                                                                       00078400
C  OUTPUT..                                                             00078500
C                                                                       00078600
C     B = SOLUTION VECTOR,  X                                           00078700
C                                                                       00078800
C                              DO THE FORWARD ELIMINATION.              00078900
C                                                                       00079000
      IF(N .EQ. 1) GO TO 50                                             00079100
      NM1=N-1                                                           00079200
      DO 20 K=1,NM1                                                     00079300
        KP1=K+1                                                         00079400
        M=IPVT(K)                                                       00079500
        T=B(M)                                                          00079600
        B(M)=B(K)                                                       00079700
        B(K)=T                                                          00079800
        DO 10 I=KP1,N                                                   00079900
   10     B(I)=B(I)+A(I,K)*T                                           00080000
   20   CONTINUE                                                        00080100
C                                 NOW DO THE BACK SUBSTITUTION.         00080200
      DO 40 KB=1,NM1                                                    00080300
        KM1=N-KB                                                        00080400
        K=KM1+1                                                         00080500
        B(K)=B(K)/A(K,K)                                                00080600
        T=-B(K)                                                         00080700
        DO 30 I=1,KM1                                                   00080800
   30     B(I)=B(I)+A(I,K)*T                                           00080900
   40   CONTINUE                                                        00081000
   50 B(1)=B(1)/A(1,1)                                                  00081100
C                                                                       00081200
      IF(MODE.LT.1)GO TO 999                                            00081300
C                                                                       00081400
C                                                                       00081500
C           CORRECT FOR COLUMN SCALING                                  00081600
C                                                                       00081700
      DO 100 J=1,N                                                      00081800
        B(J)=B(J)*SCALE(J)                                              00081900
  100 CONTINUE                                                          00082000
C                                                                       00082100
C                                                                       00082200
  999 RETURN                                                            00082300
C                                                                       00082400
      END                                                               00082500
C                                                                       00082600
C                                                                       00082700
C                                                                       00082800
C                                                                       00082900
C           SUBROUTINE EXSAVE                                           00083000
C                                                                       00083100
C           IMPLICIT REAL*8 (A-H,O-Z)                                   00083200
C           COMMON /TRACE/IN,LP,NINT,NDBUG                              00083300
      INTEGER SPAIR,SPAIC,EPAIR,EPAIC                                   00083400
      COMMON /PATTRN/ NS,SPAIR(50),SPAIC(50),NE,EPAIR(15),EPAIC(15)     00083500
      COMMON /STORE/U(65,65),V(65,65)                                   00083600
      INTEGER SPT,SROW,SCOL,EPT,EROW,ECOL                               00083700
      COMMON /SNG/ SMAT(50,15),EMAT(15,15),EXPVAL(10,15),SCALE(15),     00083800
     $     B(15),SROW(100),SCOL(100),EROW(100),ECOL(100),SPT(10),       00083900
     $     EPT(10),IPVT(15),NSING                                       00084000
      INTEGER LEFT,RIGHT                                                00084100
      COMMON /BOUND/ XMIN,XMAX,YMIN,YMAX,H,                             00084200
     $     MAP(65,65),LEFT(101),RIGHT(101),NROW,NCOL,NPT                00084300
      INTEGER ROW,COL                                                   00084400
C                                                                       00084500
      IF(NSING.LT.1)RETURN                                              00084600
C                                                                       00084700
C                                                                       00084800
      DO 200 J=1,NSING                                                  00084900
C                                                                       00085000
           EPOS=EPT(J)                                                  00085100
           DO 100 K=1,NE                                                00085200
C                                                                       00085300
                ROW=EROW(EPOS)                                          00085400
                COL=ECOL(EPOS)                                          00085500
                EXPVAL(J,K)=U(ROW,COL)                                  00085600
                EPOS=EPOS+1                                             00085700
C                                                                       00085800
  100      CONTINUE                                                     00085900
C                                                                       00086000
  200 CONTINUE                                                          00086100
C                                                                       00086200
      IF(NDBUG.LE.100)GO TO 300                                         00086300
C                                                                       00086400
      DO 280 J=1,NSING                                                  00086500
C                                                                       00086600
           WRITE(LP,50)J,(EXPVAL(J,K),K=1,NE)                           00086700
  280 CONTINUE                                                          00086800
C                                                                       00086900
C                                                                       00087000
C                                                                       00087100
  300 RETURN                                                            00087200
C                                                                       00087300
   50 FORMAT(//6X,'-- EXSAVE --'/9X,'SING',40X,'EXT. PTS.'/             00087400
     $     6X,15,4E24.16/(11X,4E24.16))                                 00087500
C                                                                       00087600
      END                                                               00087700
C                                                                       00087800
C                                                                       00087900
C                                                                       00088000
C                                                                       00088100
      SUBROUTINE SNGCAL                                                 00088200
C                                                                       00088300
      IMPLICIT REAL*8 (A-H,O-Z)                                         00088400
      COMMON /TRACE/IN,LP,NINT,NDBUG                                    00088500
      INTEGER SPAIR,SPAIC,EPAIR,EPAIC                                   00088600
      COMMON /PATTRN/ NS,SPAIR(50),SPAIC(50),NE,EPAIR(15),EPAIC(15)     00088700
      INTEGER ROW,COL,SPOS                                              00088800
      COMMON /STORE/U(65,65),V(65,65)                                   00088900
      INTEGER SPT,SROW,SCOL,EPT,EROW,ECOL                               00089000
      COMMON /SNG/ SMAT(50,15),EMAT(15,15),EXPVAL(10,15),SCALE(15),     00089100
     $     B(15),SROW(100),SCOL(100),EROW(100),ECOL(100),SPT(10),       00089200
     $     EPT(10),IPVT(15),NSING                                       00089300
      INTEGER LEFT,RIGHT                                                00089400
      COMMON /BOUND/ XMIN,XMAX,YMIN,YMAX,H,                             00089500
     $     MAP(65,65),LEFT(101),RIGHT(101),NROW,NCOL,NPT                00089600
      DIMENSION USAVE(50)                                               00089700
      INTEGER EMAXR                                                     00089800
      DATA EMAXR/15/
```

```
C
      IF(NSING.LT.1)RETURN
C
      IF(NDBUG.GT.20)WRITE(LP,40)
C
      DO 400 J=1,NSING
C
           IF(NDBUG.GT.20)WRITE(LP,50)J
C
           DO 200 K=1,NE
                B(K)=EXPVAL(J,K)
 200       CONTINUE
C
C
           CALL SOLVE(EMAXR,NE,EMAT,B,IPVT,SCALE,1)
C
C          REPLACE EACH POINT ABOUT THE SINGULARITY WITH ITS
C          SERIES REPRESENTATION
C
           SPOS SPT(J)
C
           RESSUM=0.0D0
           DO 300 K=1,NS
C
                USUM=0.0D0
                ROW SROW(SPOS)
                COL SCOL(SPOS)
                SPOS SPOS+1
                DO 280 L=1,NE
C
                     USUM=USUM+B(L)*SMAT(K,L)
 280            CONTINUE
C
                RESSUM RESSUM+DABS(U(ROW,COL)-USUM)
                U(ROW,COL)=USUM
C
                USAVE(K)=USUM
C
 300       CONTINUE
C
           RESAVE=RESSUM/DFLOAT(NS)
           RESLOG=-1.0D70
           IF(RESAVE.GT.0.0D0)RESLOG=DLOG10(RESAVE)
           WRITE(LP,70)J,RESAVE,RESLOG
           IF(NDBUG.GT.20)WRITE(LP,60)(USAVE(K),K=1,NS)
C
C
 400  CONTINUE
C
C
      RETURN
C
  50  FORMAT(/11X,'SING NO.',I5)
  40  FORMAT(//6X,'-- SNGCAL --'//)
  60  FORMAT(2X,'SNG. PTS.',4E24.16/(11X,4E24.16))
  70  FORMAT(/11X,'SNG. RES.-NO,AVE,LOG',I5,2E24.16)
C
      END
C
C
C
C
      SUBROUTINE SWEEP(KSWEEP,OMEGA,RESAVE,RESLOG)
C
      IMPLICIT REAL*8 (A-H,O-Z)
```

```
      COMMON /TRACE/IN,LP,NINT,NDBUG
      INTEGER BPLOC,BPOINT
      COMMON /STORE/ U(65,65),V(65,65)
      INTEGER SPT,SROW,SCOL,EPT,EROW,ECOL
          COMMON /SNG/ SMAT(50,15),EMAT(15,15),EXPVAL(10,15),SCALE(15),
     $         B(15),SROW(100),SCOL(100),EROW(100),ECOL(100),SPT(10),
     $         EPT(10),IPVT(15),NSING
      INTEGER LEFT,RIGHT
          COMMON /BOUND/ XMIN,XMAX,YMIN,YMAX,H,
     $         MAP(65,65),LEFT(101),RIGHT(101),NROW,NCOL,NPT
C
      DIMENSION MDIR(8,4)
      DATA MDIR/1,1,1,-1,1,1,-1,-1,-1,-1,1,-1,1,1,-1,-1,
     $     1,-1,-1,-1,1,-1,1,-1,1,1,1,-1,1,-1/
      INTEGER EFLAG,DFLAG
      DATA EFLAG,DFLAG/-2,0/
C
      IF(NDBUG.GT.100)WRITE(LP,50)
C
      NSWEEP=1
      IF(KSWEEP.NE.1)NSWEEP=2
C
C*****CALL EXSAVE*************************************************
C
      DO 600 N=1,NSWEEP
C
C          SAVE THE VALUES OF THE EXTREME POINTS
C
           CALL EXSAVE
           CALL SNGCAL
C
C
           INC=1
           J=1
           IF(N.LT.2)GO TO 180
                INC=-1
                J=NROW
                WRITE(LP,80)RESAVE,RESLOG
C
C
C
 180       RESSUM=0.0D0
           RESMAX=0.0D0
C
           DO 400 JCOUNT=1,NROW
C
                KMAX=RIGHT(J)
                KMIN=LEFT(J)
C
                K=KMIN
                IF(N.EQ.2)K=KMAX
C
                DO 300 KCOUNT=KMIN,KMAX
C
                     USAVE=U(J,K)
                     MOLTYP=MAP(J,K)
                     IF(MOLTYP.LE.EFLAG)GO TO 200
C
C                         BOUNDARY POINT OR SPECIAL POINT
C
                    IF(MOLTYP.LE.DFLAG)GO TO 280
C
C
                    JT=J+MDIR(MOLTYP,1)
                    JB=J+MDIR(MOLTYP,2)
```

```
                        KL=K+MDIR(MOLTYP,3)                                     00102800
                        KR=K+MDIR(MOLTYP,4)                                     00102900
C                                                                               00103000
                        UNEW=.25D0*(U(JT,K)+U(JB,K)+U(J,KL)+U(J,KR))            00103100
                        GO TO 240                                               00103200
C                                                                               00103300
C                                                                               00103400
C                             NORMAL MOLECULE                                   00103500
C                                                                               00103600
 200                UNEW=.25D0*(U(J+1,K)+U(J-1,K)+U(J,K+1)+U(J,K-1))           00103700
C                                                                               00103800
C                                                                               00103900
 240                RES=UNEW-USAVE                                             00104000
                    RESSUM=RESSUM+DABS(RES)                                     00104100
C                                                                               00104200
                    U(J,K)=USAVE+OMEGA*RES                                      00104300
C                                                                               00104400
C                                                                               00104500
 280                K=K+INC                                                    00104600
C                                                                               00104700
 300            CONTINUE                                                       00104800
C                                                                               00104900
                J=J+INC                                                         00105000
C                                                                               00105100
 400        CONTINUE                                                           00105200
C                                                                               00105300
C                                                                               00105400
C*********CALL *SNGCAL ***************************************************** 00105500
C                                                                               00105600
          RESAVE=RESSUM/DFLOAT(NPT)                                             00105700
          RESLOG=DLOG10(RESAVE)                                                 00105800
C                                                                               00105900
C                                                                               00106000
 600  CONTINUE                                                                  00106100
C                                                                               00106200
C*****CALL *SNGCAL ***************************************************** 00106300
C                                                                               00106400
      RETURN                                                                    00106500
C                                                                               00106600
 50   FORMAT(//6X,'-- SWEEP --'//)                                              00106700
 80       FORMAT(/6X,'RES.-   AVE.,LOG,MAX,R,C',5X,3E24.16,2I5,5X,             00106800
     $          '(FORWARD SWEEP)')                                             00106900
C                                                                               00107000
      END                                                                       00107100
C                                                                               00107200
C                                                                               00107300
C                                                                               00107400
C                                                                               00107500
      SUBROUTINE VAITK (METHD,NPREP,NPER,                                       00107600
     *                  KOUNT,KPER,    COSTH,SMAX,SMIN,    COSIN,SRAW,S)      00107700
C                                                                               00107800
          IMPLICIT REAL*8 (A-H,O-Z)                                             00107900
          COMMON /STORE/ U(65,65),V(65,65)                                      00108000
          INTEGER LEFT,RIGHT                                                    00108100
          COMMON /BOUND/ XMIN,XMAX,YMIN,YMAX,H,                                 00108200
     $         MAP(65,65),LEFT(101),RIGHT(101),NROW,NCOL,NPT                    00108300
          DIMENSION X(4500),TA(4500),TB(4500)                                   00108400
          EQUIVALENCE (X(1),U(1,1))                                             00108500
          COMMON /TRACE/IN,IP,NINI,NDBUG                                        00108600
          INTEGER CPIMIN,CPIMAX                                                 00108700
          COMMON /SIMAP/ KTOP(65),KBOT(65),CPIMIN(65),CPIMAX(65)                00108800
C                                                                               00108900
C                                                                               00109000
C  VAITK 3.1        A.N.S.I. STANDARD BASIC FORTRAN        JULY 1981           00109100
C  SUBROUTINE VAITK PERFORMS DYNAMIC VECTOR AITKEN EXTRAPOLATION,                00109200
C  IN AN ATTEMPT TO ACCELERATE THE CONVERGENCE OF A GIVEN SLOWLY                 00109300
C  CONVERGENT VECTOR ITERATION SCHEME.                                           00109400
C  J. P. CHANDLER, COMPUTER SCIENCE DEPT., OKLAHOMA STATE UNIVERSITY             00109500
C                                                                               00109600
C  A. JENNINGS, J.INST.MATH.APPLIC. 8 (1971) 99-110                             00109700
C  E. F. BOYLE AND A. JENNINGS, INT.J.NUM.METH.ENG. 7 (1974) 232-245            00109800
C  H. K. CHENG AND M. M. HAFEZ, PAGES 101-122 IN                                00109900
C       -PADE APPROXIMANTS METHOD AND ITS APPLICATIONS TO MECHANICS-,           00110000
C       EDITED BY H. CABANNES (SPRINGER-VERLAG 1976)                            00110100
C  E. GEKELER, MATHEMATICS OF COMPUTATION 26 (1972) 427-436                     00110200
C  C. BREZINSKI AND A. C. RIEU, MATH. OF COMP. 28 (1974) 731-741                00110300
C  C. BREZINSKI, COMPUTING 14 (1975) 205-211                                    00110400
C  P. WYNN, MATH. OF COMP. 16 (1962) 301-322                                    00110500
C                                                                               00110600
C  INPUT QUANTITIES.....  X(*),NX,METHD,NPREP,NPER,                             00110700
C                            (KOUNT,KPER),      COSTH,SMAX,SMIN                00110800
C  OUTPUT QUANTITIES....  COSIN,SRAW,S,X(*)                                     00110900
C  COUNTERS.............  KOUNT,KPER                                           00111000
C  SCRATCH ARRAYS.......  TA(*),TB(*)                                           00111100
C                                                                               00111200
C     X(*)      --  INPUT VECTOR ITERATE.  ALSO RETURNS THE EXTRAPOLATED       00111300
C                     OUTPUT VECTOR.                                            00111400
C     NX        --  NUMBER OF COMPONENTS IN THE VECTORS X(*), TA(*),            00111500
C                     AND TB(*)                                                 00111600
C     TA(*)     --  SCRATCH VECTOR OF NX COMPONENTS                             00111700
C     TB(*)     --  SCRATCH VECTOR OF NX COMPONENTS                             00111800
C     METHD     --  =1 TO USE THE -EDM- METHOD OF JENNINGS,                     00111900
C                   =2 TO USE THE -SDM- METHOD (THIS IS THE FIRST STAGE        00112000
C                      OF THE VECTOR EPSILON ALGORITHM OF WYNN),                00112100
C                   =3 TO USE AN -SDN- (SECOND DIFFERENCE NORM) METHOD,         00112200
C                   =4 TO USE A -EDN- (DIFFERENCE OF FIRST DIFFERENCE          00112300
C                      NORMS) METHOD                                            00112400
C                   (SUGGESTION ...  TRY METHD=3, AND PERHAPS 2 AND 4)         00112500
C     NPREP     --  NUMBER OF ITERATES DISCARDED BEFORE EACH                    00112600
C                     EXTRAPOLATION BEGINS                                      00112700
C                   (SUGGESTION ...  SET NPREP=0, OR PERHAPS 1 OR 2)           00112800
C     NPER      --  =1 IF THE X(*) ITERATES ARE THOUGHT TO BE CONVERGING       00112900
C                     ALONG A LINE, EITHER MONOTONICALLY OR                     00113000
C                     ALTERNATING IN DIRECTION,                                 00113100
C                   =2 IF THE X(*) ITERATES ARE THOUGHT TO BE ZIGZAGGING       00113200
C                     WITH A PERIOD OF TWO ITERATIONS, ETC.                     00113300
C                   (USE THE SMALLEST VALUE OF NPER SUCH THAT COSIN             00113400
C                     APPROACHES +1 OR -1.  NPER=2 GIVES THE                    00113500
C                     -SQUARED- EXTRAPOLATION METHODS OF JENNINGS.)             00113600
C     KOUNT     --  ITERATION COUNTER                                           00113700
C     KPER      --  COUNTER FOR ZIGZAGGING                                      00113800
C     COSTH     --  EXTRAPOLATION IS DONE ONLY IF THE MAGNITUDE OF              00113900
C                     COSIN (SEE BELOW) IS .GT. COSTH                           00114000
C                     (SUGGESTION...  SET COSTH .95 OR LARGER)                  00114100
C     SMAX      --  UPPER LIMIT ON THE EXTRAPOLATION FACTOR (SEE BELOW)         00114200
C     SMIN      --  LOWER LIMIT ON THE EXTRAPOLATION FACTOR                     00114300
C     COSIN     --  RETURNS THE COSINE OF THE ANGLE BETWEEN THE TWO             00114400
C                     FIRST DIFFERENCE VECTORS                                  00114500
C     SRAW      --  RETURNS THE RAW VALUE OF THE EXTRAPOLATION FACTOR S,        00114600
C                     BEFORE COSTH, SMAX, AND SMIN ARE APPLIED                  00114700
C     S         --  RETURNS THE VALUE OF S ACTUALLY USED                        00114800
C                                                                               00114900
C  THE USER CALLS VAITK REPEATEDLY.  THE X(*) VECTORS ARE SUCCESSIVE            00115000
C  VECTORS FROM SOME ITERATION SCHEME HAVING ROUGHLY LINEAR CONVERGENCE.        00115100
C  THE COUNTERS KOUNT AND KPER MUST BE SET TO ZERO BEFORE THE FIRST             00115200
C  CALL TO VAITK FOR A GIVEN PROBLEM, AND NOT CHANGED UNTIL THE NEXT            00115300
C  PROBLEM IS TO BE STARTED, EXCEPT AS NOTED BELOW.                             00115400
C  ON CERTAIN CALLS TO VAITK, THE VECTOR X(*) WILL BE EXTRAPOLATED              00115500
C  INSIDE OF VAITK TO AN ESTIMATE OF THE LIMIT VECTOR TOWARD WHICH              00115600
C  THE ITERATES X(*) HAVE BEEN CONVERGING.  THIS IS DONE BY ADDING              00115700
```

```
C  TO X(*) A SCALAR MULTIPLE S OF THE RESULTANT CHANGE IN X(*)                00115800
C  DURING THE PREVIOUS NPER ITERATIONS.                                       00115900
C  IF AN ITERATION DIVERGES, IT IS PROBABLY BEST TO FORCE CONVERGENCE        00116000
C  BY APPLYING UNDER-RELAXATION TO IT, AND THEN APPLY WAITK TO THIS          00116100
C  CONVERGENT ITERATION.                                                     00116200
C                                                                            00116300
C  MONOTONIC BEHAVIOR OF THE ITERATION IS ASSOCIATED WITH VALUES OF          00116400
C  COSIN THAT ARE .GT. ZERO, AND OSCILLATORY BEHAVIOR WITH VALUES THAT       00116500
C  ARE .LT. ZERO.  IN ADDITION, WHEN THE VALUE OF COSIN IS NEAR 1.0          00116600
C  OR NEAR -1.0, THE VALUE OF S INDICATES THE BEHAVIOR OF SETS OF            00116700
C  NPER CONSECUTIVE ITERATIONS, AS FOLLOWS.....                              00116800
C                  S .LE. -1.0          MONOTONIC DIVERGENCE                 00116900
C       -1.0 .LT. S .LT. -0.5           OSCILLATORY DIVERGENCE               00117000
C                  S .EQ. -0.5          OSCILLATION                          00117100
C       -0.5 .LT. S .LT.  0.0           OSCILLATORY CONVERGENCE              00117200
C        0.0 .LE. S                     MONOTONIC CONVERGENCE                00117300
C  WHEN COSIN IS NEAR 1.0 OR -1.0, THE FIRST DIFFERENCE VECTOR FOR           00117400
C  SETS OF NPER CONSECUTIVE ITERATIONS IS APPROXIMATELY MULTIPLIED           00117500
C  BY S/(S+1) ON EACH SUCCESSIVE SET.                                        00117600
C                                                                            00117700
C  THE PURPOSE OF SMAX AND SMIN IS TO PREVENT A WILD VALUE OF S FROM         00117800
C  CAUSING AN ABSURD EXTRAPOLATION.  FOR A SEQUENCE THAT IS KNOWN            00117900
C  TO BE NONDIVERGENT, THE USER SHOULD PROBABLY SET SMIN=-0.5 .              00118000
C  SMAX MIGHT BE SET TO 10., OR 20., OR 50., OR ....                         00118100
C                                                                            00118200
C  TO IMPLEMENT DOUBLE PRECISION ACCUMULATION OF INNER PRODUCTS,             00118300
C  ACTIVATE THE FIRST DOUBLE PRECISION STATEMENT BELOW.                      00118400
C  TO IMPLEMENT COMPLETE DOUBLE PRECISION ARITHMETIC, ACTIVATE ALSO          00118500
C  THE SECOND DOUBLE PRECISION STATEMENT AND THE DOUBLE PRECISION            00118600
C  FORM OF THE STATEMENT FUNCTIONS BELOW.                                    00118700
C                                                                            00118800
C     DOUBLE PRECISION DOT,DXLSQ,DXSQ,DXDDX,DDXSQ,DXOLD,DX,DDX              00118900
C     DOUBLE PRECISION X,TA,TB,COSIN,SMAX,SMIN,COSIN,SRAW,S,                 00119000
C    X    ARG,QABS,DABS,QSQRT,DSQRT,RZERO,                                  00119100
C    X    TEMP,SXLSQ,SXSQ,SXDDX,SDXSQ,SXOLD,SX,DENOM                        00119200
C                                                                            00119300
C                                                                            00119400
C     QABS(ARG)=ABS(ARG)                                                     00119500
C     QSQRT(ARG)=SQRT(ARG)                                                   00119600
      QABS(ARG)=DABS(ARG)                                                    00119700
      QSQRT(ARG)=DSQRT(ARG)                                                  00119800
C                                                                            00119900
      RZERO=0.D0                                                             00120000
      COSIN=RZERO                                                            00120100
      SRAW=RZERO                                                             00120200
      S=RZERO                                                                00120300
      IF(KOUNT-NPREP)10,40,20                                                00120400
   10 KOUNT=KOUNT+1                                                          00120500
      GO TO 350                                                              00120600
   20 IF(KPER-(NPER-1))30,40,40                                              00120700
   30 KPER=KPER+1                                                            00120800
      GO TO 350                                                              00120900
   40 KPER=0                                                                 00121000
      KOUNT=KOUNT+1                                                          00121100
      IF(KOUNT-(NPREP+2))290,330,50                                          00121200
C                                                                            00121300
C                              ACCUMULATE ALL INNER PRODUCTS.                00121400
   50 DOT=RZERO                                                              00121500
      DXLSQ=RZERO                                                            00121600
      DXSQ=RZERO                                                             00121700
      DXDDX=RZERO                                                            00121800
      DDXSQ=RZERO                                                            00121900
      DO 60 I=1,NCOL                                                         00122000
C                                                                            00122100
            JMAX=OPTMAX(I)                                                   00122200
            JMIN=OPTMIN(I)                                                   00122300
C                                                                            00122400
            DO 60 J=JMIN,JMAX                                                00122500
C                                                                            00122600
                  DXOLD=TB(J)-TA(J)                                          00122700
                  DX=X(J)-TB(J)                                              00122800
                  DDX=DX-DXOLD                                               00122900
                  DOT=DOT+DXOLD*DX                                           00123000
                  DXLSQ=DXLSQ+DXOLD*DXOLD                                    00123100
                  DXSQ=DXSQ+DX*DX                                            00123200
                  DXDDX=DXDDX+DX*DDX                                         00123300
                  DDXSQ=DDXSQ+DDX*DDX                                        00123400
C                                                                            00123500
   60 CONTINUE                                                               00123600
      SXLSQ=DXLSQ                                                            00123700
      SXSQ=DXSQ                                                              00123800
      SXDDX=DXDDX                                                            00123900
      SDXSQ=DDXSQ                                                            00124000
C                              COMPUTE COSIN.                                00124100
      SXOLD=QSQRT(SXLSQ)                                                     00124200
      SX=QSQRT(SXSQ)                                                         00124300
      DENOM=SXOLD*SX                                                         00124400
      IF(DENOM)310,310,70                                                    00124500
   70 TEMP=DOT                                                               00124600
      COSIN=TEMP/DENOM                                                       00124700
C                              SELECT THE METHOD AND COMPUTE SRAW.           00124800
      IF(METHD-2)90,110,80                                                   00124900
   80 IF(METHD-4)130,170,170                                                 00125000
C                                                                            00125100
C                              METHD=1 ...    S = -(DX,DX)/(DDX,DX)          00125200
   90 IF(SXDDX)100,310,100                                                   00125300
  100 SRAW=-SXSQ/SXDDX                                                       00125400
      GO TO 210                                                              00125500
C                              METHD=2 ...   S = -(DX,DDX)/(DDX,DDX)         00125600
  110 IF(SDXSQ)120,310,120                                                   00125700
  120 SRAW=-SXDDX/SDXSQ                                                      00125800
      GO TO 210                                                              00125900
C                              METHD=3 ...   S = SIGN*NORM(DX)/NORM(DDX)     00126000
  130 IF(SDXSQ)140,310,140                                                   00126100
  140 SRAW=SX/QSQRT(SDXSQ)                                                   00126200
      IF(COSIN)160,150,150                                                   00126300
  150 IF(SXOLD-SX)160,210,210                                                00126400
  160 SRAW=-SRAW                                                             00126500
      GO TO 210                                                              00126600
C                              METHD=4 ...                                   00126700
C                                S = -NORM(DX)/(NORM(DX)-SIGN*NORM(DXOLD))   00126800
  170 IF(COSIN)180,190,190                                                   00126900
  180 SXOLD=-SXOLD                                                           00127000
  190 DENOM=SX-SXOLD                                                         00127100
      IF(DENOM)200,310,200                                                   00127200
  200 SRAW=-SX/DENOM                                                         00127300
C                              TEST FOR SUFFICIENT COLINEARITY.              00127400
  210 IF(QABS(COSIN)-COSTH)310,220,220                                       00127500
C                                                                            00127600
C                              APPLY THE CONSTRAINTS SMAX AND SMIN.          00127700
  220 S=SRAW                                                                 00127800
      IF(S-SMAX)240,240,230                                                  00127900
  230 S=SMAX                                                                 00128000
  240 IF(S-SMIN)250,260,260                                                  00128100
  250 S=SMIN                                                                 00128200
  260 IF(S)270,310,270                                                       00128300
C                              EXTRAPOLATE.                                  00128400
C                                                                            00128500
  270 DO 280 I=1,NCOL                                                        00128600
C                                                                            00128700
```

```
            JMAX=CPIMAX(L)                                                      00128800
            JMIN=CPIMIN(L)                                                      00128900
C                                                                               00129000
            DO 280 J=JMIN,JMAX                                                  00129100
C                                                                               00129200
                 X(J)=X(J)+S*(X(J)-TB(J))                                       00129300
C                                                                               00129400
  280       CONTINUE                                                          00129500
C                                                                               00129600
      KOUNT=1                                                                   00129700
C                                 SAVE X(*).                                    00129800
C                                                                               00129900
  290 DO 300 L=1,NCOL                                                           00130000
C                                                                               00130100
            JMAX=CPIMAX(L)                                                      00130200
            JMIN=CPIMIN(L)                                                      00130300
C                                                                               00130400
            DO 300 J=JMIN,JMAX                                                  00130500
C                                                                               00130600
                 TA(J)=X(J)                                                     00130700
C                                                                               00130800
  300       CONTINUE                                                          00130900
C                                                                               00131000
      GO TO 350                                                                 00131100
C                                 SHIFT THE STORED ITERATES BACK ONE PLACE.     00131200
C                                                                               00131300
  310 DO 320 L=1,NCOL                                                           00131400
C                                                                               00131500
            JMAX=CPIMAX(L)                                                      00131600
            JMIN=CPIMIN(L)                                                      00131700
C                                                                               00131800
            DO 320 J=JMIN,JMAX                                                  00131900
C                                                                               00132000
        TA(J)=TB(J)                                                             00132100
C                                                                               00132200
  320       CONTINUE                                                          00132300
C                                                                               00132400
C                                 SAVE X(*).                                    00132500
C                                                                               00132600
  330 DO 340 L=1,NCOL                                                           00132700
C                                                                               00132800
            JMAX CPIMAX(L)                                                      00132900
            JMIN=CPIMIN(L)                                                      00133000
C                                                                               00133100
            DO 340 J=JMIN,JMAX                                                  00133200
C                                                                               00133300
                 TB(J)=X(J)                                                     00133400
C                                                                               00133500
  340       CONTINUE                                                          00133600
C                                                                               00133700
C                                                                               00133800
  350 RETURN                                                                    00133900
C                                                                               00134000
      END                                                                       00134100
C                                                                               00134200
C                                                                               00134300
C                                                                               00134400
C                                                                               00134500
      SUBROUTINE PLOTB                                                          00134600
C                                                                               00134700
      IMPLICIT REAL*8 (A-H,O-Z)                                                 00134800
      COMMON /TRACE/ IN,LP,NINT,NDBUG                                           00134900
      INTEGER LEFT,RIGHT                                                        00135000
      COMMON /BOUND/ XMIN,XMAX,YMIN,YMAX,H,                                     00135100
     $     MAP(65,65),LEFT(101),RIGHT(101),NROW,NCOL,NPT                        00135200
      DIMENSION ICHAR(20),LINE(120)                                             00135300
      DATA IBLNK/' '/,ICHAR/'*','E','S','D','1','2','3','4','5','6',00135400
     $     '7','8','9','I','J','K','L','M'S'/,                                 00135500
     $     KOFFST/4/                                                            00135600
C                                                                               00135700
      WRITE(LP,90)                                                              00135800
      J=NROW                                                                    00135900
      DO 1000 L=1,NROW                                                          00136000
C                                                                               00136100
           KMIN=LEFT(J)                                                         00136200
           KMAX=RIGHT(J)                                                        00136300
C                                                                               00136400
           DO 900 K=KMIN,KMAX                                                   00136500
C                                                                               00136600
                KCHAR=MAP(J,K)+KOFFST                                           00136700
                LINE(K)=ICHAR(KCHAR)                                            00136800
C                                                                               00136900
  900      CONTINUE                                                            00137000
C                                                                               00137100
           WRITE(LP,95)(IBLNK,K=1,KMIN),(LINE(K),K=KMIN,KMAX)                   00137200
C                                                                               00137300
           J=J-1                                                                00137400
C                                                                               00137500
 1000 CONTINUE                                                                  00137600
C                                                                               00137700
C                                                                               00137800
      RETURN                                                                    00137900
C                                                                               00138000
   90 FORMAT('1H1,5X,'BOUNDARY PLOT')                                           00138100
   95 FORMAT(1X,120A1)                                                          00138200
C                                                                               00138300
      END                                                                       00138400
C                                                                               00138500
C                                                                               00138600
C                                                                               00138700
C                                                                               00138800
      SUBROUTINE PLOT(U)                                                        00138900
C                                                                               00139000
      IMPLICIT REAL*8 (A-H,O-Z)                                                 00139100
      INTEGER ROW,COL                                                           00139200
      DIMENSION U(65,65)                                                        00139300
      COMMON /TRACE/IN,LP,NINT,NDBUG                                            00139400
      INTEGER LEFT,RIGHT                                                        00139500
      COMMON /BOUND/ XMIN,XMAX,YMIN,YMAX,H,                                     00139600
     $     MAP(65,65),LEFT(101),RIGHT(101),NROW,NCOL,NPT                        00139700
      DIMENSION ICHAR(20),LINE(120)                                             00139800
      DATA ICHAR /'A','B','C','D','E','F','G','H','I','J',                      00139900
     $           'K','L','M','N','O','P','Q','R','S','T'/,                      00140000
     $     IBLNK/' '/                                                           00140100
C                                                                               00140200
C                                                                               00140300
      COL=LEFT(1)                                                               00140400
      ROW=1                                                                     00140500
      UMIN U(ROW,COL)                                                           00140600
      UMAX=UMIN                                                                 00140700
C                                                                               00140800
      DO 220 J=1,NROW                                                           00140900
C                                                                               00141000
           KMIN=LEFT(J)                                                         00141100
           KMAX=RIGHT(J)                                                        00141200
C                                                                               00141300
           DO 200 K=KMIN,KMAX                                                   00141400
C                                                                               00141500
                UVAL=U(J,K)                                                     00141600
C                                                                               00141700
```

```
                IF(UVAL.GT.UMAX)UMAX=UVAL                               00141800
                IF(UVAL.LT.UMIN)UMIN=UVAL                               00141900
  200         CONTINUE                                                  00142000
C                                                                       00142100
  220   CONTINUE                                                        00142200
C                                                                       00142300
C                                                                       00142400
        WRITE(LP,10)UMAX,UMIN,NINT                                      00142500
        UDIF=UMAX-UMIN                                                  00142600
        USCALE=DFLOAT(NINT)/UDIF                                        00142700
        J=NROW                                                          00142800
C                                                                       00142900
        DO 400 L=1,NROW                                                 00143000
C                                                                       00143100
             KMIN=LEFT(J)                                               00143200
             KMAX=RIGHT(J)                                              00143300
C                                                                       00143400
             DO 300 K=KMIN,KMAX                                         00143500
C                                                                       00143600
                  NCHAR=IDINT((U(J,K)-UMIN)*USCALE)+1                   00143700
                  IF(NCHAR.GT.NINT)NCHAR=NINT                           00143800
                  LINE(K)=ICHAR(NCHAR)                                  00143900
  300         CONTINUE                                                  00144000
C                                                                       00144100
             IF(NCOL.GT.25)WRITE(LP,20)(IBLNK,M=1,KMIN),                00144200
     $                          (LINE(K),K=KMIN,KMAX)                   00144210
             IF(NCOL.LE.25)WRITE(LP,21)(IBLNK,M=1,KMIN),                00144220
     $                          (LINE(K),K=KMIN,KMAX)                   00144230
             J=J-1                                                      00144300
C                                                                       00144400
  400   CONTINUE                                                        00144500
C                                                                       00144600
C                                                                       00144700
        UINC=UDIF/DFLOAT(NINT)                                          00144800
        U2=UMIN                                                         00144900
        WRITE(LP,25)                                                    00145000
C                                                                       00145100
        DO 600 J=1,NINT                                                 00145200
C                                                                       00145300
             U1=U2                                                      00145400
  21    FORMAT(1X,25A2)                                                 00145410
             U2=UMIN+UINC*DFLOAT(J)                                     00145500
C                                                                       00145600
             WRITE(LP,30)ICHAR(J),U1,U2                                 00145700
  600   CONTINUE                                                        00145800
C                                                                       00145900
C                                                                       00146000
        RETURN                                                          00146100
C                                                                       00146200
   10   FORMAT(//6X,'-- PLOT --'/11X,'UMAX,UMIN,NINT',2E15.7,I10/)      00146300
   20   FORMAT(1X,120A1)                                                00146400
   25   FORMAT(//21X,'CHARACTER RANGES'//)                              00146500
   30   FORMAT(11X,A1,20E15.7,'  TO ',E15.7)                            00146600
C                                                                       00146700
        END                                                             00146800
C                                                                       00146900
C                                                                       00147000
C                                                                       00147100
C                                                                       00147200
        SUBROUTINE RESULT(MINR,MAXR)                                    00147300
C                                                                       00147400
        IMPLICIT REAL*8 (A-H,O-Z)                                       00147500
        COMMON /STORE/U(65,65),V(65,65)                                 00147600
        COMMON /TRACE/ IN,LP,NINT,NDBUG                                 00147700
        INTEGER LEFT,RIGHT                                              00147800
        COMMON /BOUND/ XMIN,XMAX,YMIN,YMAX,H,                           00147900
     $       MAP(65,65),LEFT(101),RIGHT(101),NROW,NCOL,NPT              00148000
        DATA LHEX/11/                                                   00148100
C                                                                       00148200
        WRITE(LP,10)NROW,NCOL,XMIN,XMAX,YMIN,YMAX,H                     00148300
        WRITE(LHEX,12)NROW,NCOL                                         00148400
C                                                                       00148500
        DO 200 J=MINR,MAXR                                              00148600
C                                                                       00148700
             Y=YMIN+H*DFLOAT(J-1)                                       00148800
             KMIN=LEFT(J)                                               00148900
             KMAX=RIGHT(J)                                              00149000
C                                                                       00149100
             XLEFT=XMIN+H*DFLOAT(KMIN-1)                                00149200
             XRIGHT=XMIN+H*DFLOAT(KMAX-1)                               00149300
C                                                                       00149400
             WRITE(LP,20)J,Y,KMIN,XLEFT,KMAX,XRIGHT,                    00149500
     $            (U(J,K),K=KMIN,KMAX)                                  00149600
             WRITE(LHEX,30)(U(J,K),K=KMIN,KMAX)                         00149700
C                                                                       00149800
  200   CONTINUE                                                        00149900
C                                                                       00150000
        RETURN                                                          00150100
C                                                                       00150200
   20   FORMAT(/11X,'ROW',I4,4X,'Y=',E15.7,4X,'COLS.',I4,'( X=',        00150300
     $       E15.7,') TO',I4,'( X=',E15.7,')'/                          00150400
     $       (11X,4E24.16))                                             00150500
   10   FORMAT(1H1,5X,'-- RESULT --'/11X,'ROWS,COLS',2I5/               00150600
     $       11X,'XMIN,XMAX,YMIN,YMAX,H',5E15.7//)                      00150700
   12   FORMAT(2Z8)                                                     00150800
   30   FORMAT(5Z16)                                                    00150900
C                                                                       00151000
        END                                                             00151100
```

```
MAXIT*****  120                                                          00000100
KSWEEP        2                                                          00000200
KOMEGA        0                                                          00000300
OMEGA     1.65900                                                        00000400
NPRTP        10                                                          00000500
NROW,NCOL    17    17                                                    00000600
XMIN,XMAX           0.0         1.0                                      00000700
YMIN,YMAX           0.0         1.0                                      00000800
ROW1          9    17     9                                             00000900
LOC           9    10    11    12    13    14    15    16    17         00001000
TYPE          0     0     0     0     0     0     0     0     0         00001100
ROW2-8        9    17     2                                             00001200
LOC           9    17                                                  00001300
TYPE          1     2                                                  00001400
ROW2-8        9    17     2                                             00001500
LOC           9    17                                                  00001600
TYPE          1     2                                                  00001700
ROW2-8        9    17     2                                             00001800
LOC           9    17                                                  00001900
TYPE          1     2                                                  00002000
ROW2-8        9    17     2                                             00002100
LOC           9    17                                                  00002200
TYPE          1     2                                                  00002300
ROW2-8        9    17     2                                             00002400
LOC           9    17                                                  00002500
TYPE          1     2                                                  00002600
ROW2-8        9    17     2                                             00002700
LOC           9    17                                                  00002800
TYPE          1     2                                                  00002900
ROW2-8        9    17     2                                             00003000
LOC           9    17                                                  00003100
TYPE          1     2                                                  00003200
ROW 9         1    17     9                                             00003300
LOC           1     2     3     4     5     6     7     8    17         00003400
TYPE          0     3     3     3     3     3     3     3    2          00003500
ROW10-16      1    17     2                                             00003600
LOC           1    17                                                  00003700
TYP           0     2                                                  00003800
ROW10-16      1    17     2                                             00003900
LOC           1    17                                                  00004000
TYP           0     2                                                  00004100
ROW10-16      1    17     2                                             00004200
LOC           1    17                                                  00004300
TYP           0     2                                                  00004400
ROW10-16      1    17     2                                             00004500
LOC           1    17                                                  00004600
TYP           0     2                                                  00004700
ROW10-16      1    17     2                                             00004800
LOC           1    17                                                  00004900
TYP           0     2                                                  00005000
ROW10-16      1    17     2                                             00005100
LOC           1    17                                                  00005200
TYP           0     2                                                  00005300
ROW10-16      1    17     2                                             00005400
LOC           1    17                                                  00005500
TYP           0     2                                                  00005600
ROW 17        1    17    17                                             00005700
LOC           1     2     3     4     5     6     7     8     9    10    11    12  00005800
             13    14    15    16    17                                00005900
TYPE          0     4     4     4     4     4     4     4     4     4     4     4  00006000
              4     4     4     4     8                                 00006100
```

```
NSING         1                                                          00006200
NR            8                                                          00006300
COL/XCOOR    -1     0    -1     0     1    -1     0     1               00006400
ROW/YCOOR    -1    -1     0     0     0     1     1     1               00006500
NE            6                                                          00006600
COL/XCOOR     0    -2     2    -2    -1     1                           00006700
ROW/YCOOR    -2    -1     0     1     2     2                           00006800
R,C,TYPE      9     9     3                                             00006810
                                                                        00006900
                                                                        00007000
```

APPENDIX B

THE PROGRAM VBOUND

The program VBOUND is included here. It can be used to solve Poisson's equation on a solution domain with an irregular shape. A uniform grid is used to represent the irregular shape. The boundary conditions can be of Neumann (vanishing normal derivative) or Dirichlet type. The shape of the solution domain is specified in a manner similar to that used in the program CORNER; rows and columns of the grid are numbered from zero. Boundary conditions are specified in the same manner as for CORNER. In selecting fine and coarse grids, each must fully represent the shape of the solution domain.

```
C                                                                       00000900
C                                                                       00001000
C          CYCLE-C MULTIGRID SOLUTION OF POISSON EQUATION               00001100
C          ON A RECTANGLE.                                              00001200
C                                                                       00001300
      EXTERNAL G,R                                                      00001400
C                                                                       00001500
      COMMON /TRACE/ IN,LP,NDBUG,NPRINT,NINI                           00001600
      COMMON /CONTRL/ N,TOL,WMAX                                        00001700
C                                                                       00001800
      IN=10                                                             00001900
      LP=11                                                             00002000
      NDBUG=0                                                           00002100
C                                                                       00002200
      NPRINT=6                                                          00002300
      NINI=10                                                           00002400
C                                                                       00002500
C                                                                       00002800
      CALL DEFINE(G,R)                                                  00002900
C                                                                       00002910
C     CALL TEST(G,R)                                                    00003000
C     STOP                                                              00003100
C                                                                       00003200
C                                                                       00003300
C                                                                       00003400
      LP=6                                                              00003410
      CALL MULTIG                                                       00003500
C                                                                       00003600
C                                                                       00003700
      STOP                                                              00003800
      END                                                               00003900
C                                                                       00004000
C                                                                       00004100
C                                                                       00004200
C                                                                       00004300
      FUNCTION R(X,Y,HTOL)                                              00004400
C                                                                       00004500
C          RIGHT-HAND SIDE OF THE POISON EQUATION                       00004600
C                                                                       00004700
C                                                                       00004800
      R=0.                                                              00004900
      RETURN                                                            00005000
      END                                                               00005100
C                                                                       00005200
C                                                                       00005300
C                                                                       00005400
C                                                                       00005500
      FUNCTION G(X,Y,HTOL)                                              00005600
C                                                                       00005700
C          DIRICHLET BOUNDARY CONDITION AND INITIAL SOLUTION GUESS      00005800
C                                                                       00005900
      G=0.                                                              00006000
      IF(X.LT.HTOL)G=1.0                                                00006100
C                                                                       00006300
      RETURN                                                            00006400
      END                                                               00006500
C                                                                       00006600
C                                                                       00006700
C                                                                       00006800
C                                                                       00006900
C                                                                       00007000
```

```
      SUBROUTINE DEFINE(G,R)                                            00007100
C                                                                       00007200
C          DEFINE -- SET UP THE GRID SYSTEM                             00007300
C                                                                       00007400
C          INPUT PARAMETERS (ALL VALUES COME FROM CARD IMAGES -         00007500
C               COLUMNS 1-10 ARE RESERVED FOR A DESCRIPTION -           00007600
C               INTEGERS COME FROM FIELDS OF 5 COLUMNS, REALS FROM      00007700
C               FIELDS OF 15 WITH 7 DIGITS)                             00007800
C                                                                       00007900
C               WMAX -- THE TOTAL WORK TO BE ALLOWED ON ALL GRIDS       00007910
C                    BEFORE TERMINATION                                 00007920
C               TOL -- THE LIMIT THE RESIDUAL NORM MUST MEET BEFORE     00007930
C                    NORMAL TERMINATION                                 00007940
C               NLEVEL -- THE NUMBER GRIDS TO BE EMPLOYED IN THE        00008000
C                    MULTIGRID PROCESS                                  00008100
C               NCOLS -- LENGTH OF THE SOLUTION DOMAIN IN THE           00008200
C                    X-DIRECTION (UNITS OF H0)                          00008300
C               NROWS -- LENGTH IN THE Y-DIRECTION                      00008400
C               PXMIN -- LOWER BOUND ON X                               00008500
C               PXMAX -- UPPER BOUND ON X                               00008600
C               PYMIN -- LOWER BOUND ON Y                               00008700
C               PYMAX -- UPPER BOUND ON Y                               00008800
C                                                                       00008900
C          FUNCTIONS -                                                  00009000
C                                                                       00009100
C               G -- BOUNDARY VALUES OF THE SOLUTION DOMAIN             00009200
C                    AND INITIAL SOLUTION GUESS                         00009300
C               R -- RIGHT-HAND SIDE OF THE POISSON EQUATION            00009400
C                                                                       00009500
C                                                                       00009600
      EXTERNAL G,R                                                      00009700
      COMMON /TRACE/IN,LP,NDBUG,NPRINT,NINI                             00009800
      INTEGER SLVL,BLVL,RIGHT,RLOC,NLOC,SLOC,BNDLOC,BROW,BLOC,          00009900
     $     BNDTYP                                                       00010000
      COMMON /POINT/SLVL(20),BLVL(20),LEFT(400),RIGHT(400),             00010100
     $     LLOC(400),RLOC(400),NLOC(400),SLOC(400),BNDLOC(1000),        00010200
     $     BNDTYP(1000),BROW(400),BLOC(1000)                            00010300
      COMMON /SIZE/NROW(20),NCOL(20),NPOINT(20),HVAL(20),XMIN,XMAX,     00010400
     *     YMIN,YMAX                                                    00010500
      COMMON /CONTRL/ NLEVEL,TOL,WMAX                                   00010600
      DATA JSIZE/5000/                                                  00010700
      DATA IEVEL,JPRNT,JLEN,JPLOT,JSOL,JRHS/6*1/                        00010800
C                                                                       00010900
C                                                                       00010910
C          GET THE WORK AND RESIDUAL TOLERANCES                         00010920
C                                                                       00010930
      READ(IN,20)TOL,WMAX                                               00010940
C                                                                       00010950
C                                                                       00011000
C          GET POINTERS TO THE LEFT AND RIGHT SIDES OF THE SOLUTION     00011100
C          DOMAIN.  ALSO, GET LISTS OF THE LOCATION AND TYPE OF THE     00011200
C          BOUNDARY POINTS.                                             00011300
C                                                                       00011400
      CALL GETBND(NROWS,NCOLS,NBNDPT)                                   00011500
C                                                                       00011600
      WRITE(LP,50)NLEVEL,NROWS,NCOLS,NBNDPT,XMIN,XMAX,YMIN,YMAX         00011700
C                                                                       00011800
C                                                                       00011900
C          SET UP POINTERS FOR THE COARSER GRIDS                        00012000
C                                                                       00012100
      CALL GENPTR(NLEVEL,NROWS,NCOLS,NBNDPT)                            00012200
C                                                                       00012300
C                                                                       00012400
C          PRINT THE POINTERS FOR THE DIFFERENT GRIDS                   00012500
C                                                                       00012600
```

```
      CALL PPOINT(NLEVEL)                                               00012700
C                                                                       00012800
      CALL VERIFY(NLEVEL)                                               00012900
C                                                                       00013000
      CALL PREP(LEVEL,G,R)                                              00013100
C                                                                       00013200
      CALL RESULT(LEVEL,JPRNT,JLEN,JPLOT,JSOL,JRHS)                    00013300
C                                                                       00013400
      RETURN                                                            00013500
C                                                                       00013600
   20 FORMAT(10X,E20.7)                                                 00013610
   50 FORMAT(1H1,5X,'-- FINE GRID DEFINITION --'//                      00013700
     $      11X,'NLEVEL,NROWS,NCOLS,NBNDPT',4I10//                     00013800
     $      11X,'XMIN,XMAX,YMIN,YMAX',4E20.7)                          00013900
      END                                                               00014000
C                                                                       00014100
C                                                                       00014200
C                                                                       00014300
C                                                                       00014400
C                                                                       00014500
      SUBROUTINE GETBND(NROWS,NCOLS,NBNDPT)                             00014600
C                                                                       00014700
C     GETBND - FETCH A LIST OF VALUES TO BE PLACED IN KARRAY            00014800
C                                                                       00014900
C      AN INPUT RECORD WILL CONTAIN UP TO 12 FIELDS                     00015000
C           COLS 1 - 10 DESCRIPTION OF CARD CONTENTS                    00015100
C           COLS 11 - 15, 16 - 20, ... , 66-70 DATA FIELDS              00015200
C                                                                       00015300
      COMMON /TRACE/IN,LP,NDBUG,NPRINT,NINI                             00015400
      INTEGER SLVL,BLVL,RIGHT,RLOC,NLOC,SLOC,BNDLOC,BROW,BLOC,          00015500
     $     BNDTYP                                                       00015600
      COMMON /POINT/SLVL(20),BLVL(20),LEFT(400),RIGHT(400),             00015700
     $     LLOC(400),RLOC(400),NLOC(400),SLOC(400),BNDLOC(1000),        00015800
     $     BNDTYP(1000),BROW(400),BLOC(1000)                            00015900
      COMMON /SIZE/NROW(20),NCOL(20),NPOINT(20),HVAL(20),XMIN,XMAX,     00016000
     *     YMIN,YMAX                                                    00016100
      COMMON /CONTRL/ NLEVEL,IOL,WMAX                                   00016200
      INTEGER RI                                                        00016300
C                                                                       00016400
      IF(NDBUG.NE.0)WRITE(LP,40)                                        00016500
C                                                                       00016600
C                                                                       00016700
      READ(IN,10)NLEVEL                                                 00016800
      WRITE(LP,50)NLEVEL                                                00016900
C                                                                       00017000
      READ(IN,10)NROWS,NCOLS                                            00017100
      WRITE(LP,60)NROWS,NCOLS                                           00017200
C                                                                       00017300
      READ(IN,20)XMIN,XMAX                                              00017400
      READ(IN,20)YMIN,YMAX                                              00017500
      WRITE(LP,70)XMIN,XMAX,YMIN,YMAX                                   00017600
C                                                                       00017700
      KMIN=1                                                            00017800
      NBNDPT=0                                                          00017900
      DO 200 J=1,NROWS                                                  00018000
C                                                                       00018100
           READ(IN,10)LEFT,RI,NBP                                       00018200
           WRITE(LP,75)LEFT,RI,NBP                                      00018300
C                                                                       00018400
           NBNDPT=NBNDPT+NBP                                            00018500
C                                                                       00018600
           LEFT(J)=LEFT                                                 00018700
           RIGHT(J)=RI                                                  00018800
C                                                                       00018900
           KMAX=KMIN+NBP-1                                              00019000
           READ(IN,10)(BLOC(K),K=KMIN,KMAX)                             00019100
           READ(IN,10)(BNDTYP(K),K=KMIN,KMAX)                           00019200
C                                                                       00019300
           IF(NDBUG.LT.100)GO TO 180                                    00019400
C                                                                       00019500
                WRITE(LP,80)(BLOC(K),K=KMIN,KMAX)                       00019600
                WRITE(LP,85)(BNDTYP(K),K=KMIN,KMAX)                     00019700
C                                                                       00019800
  180      KMIN=KMAX+1                                                  00019900
C                                                                       00020000
  200 CONTINUE                                                          00020100
C                                                                       00020200
C                                                                       00020300
      IF(NDBUG.GE.100)WRITE(LP,90)NBNDPT                                00020400
C                                                                       00020500
      RETURN                                                            00020600
C                                                                       00020700
   10 FORMAT(10X,1215)                                                  00020800
   20 FORMAT(10X,4E15.7)                                                00020900
   40 FORMAT(//6X,'-- GETBND --')                                       00021000
   50 FORMAT(/11X,'NLEVEL',I10)                                         00021100
   60 FORMAT(/11X,'NROWS,NCOLS',2I10)                                   00021200
   70 FORMAT(/11X,'XMIN,XMAX',2E15.7//11X,'YMIN,YMAX',2E15.7)           00021300
   75 FORMAT(11X,'LEFT,RIGHT,NBPT',3I10)                                00021400
   80 FORMAT(/11X,'BLOC ',1615/(16X,1615))                              00021500
   85 FORMAT(/11X,'BTYP ',1615/(16X,1615))                              00021600
   90 FORMAT(/11X,'NBNDPT',I10)                                         00021700
C                                                                       00021800
C                                                                       00021900
      END                                                               00022000
C                                                                       00022100
C                                                                       00022200
C                                                                       00022300
C                                                                       00022400
C                                                                       00022500
      SUBROUTINE GENPIR(NLEVEL,NR,NC,NBNDPT)                            00022600
C                                                                       00022700
      INTEGER SLVL,BLVL,RIGHT,RLOC,NLOC,SLOC,BNDLOC,BROW,BLOC,          00022800
     $     BNDTYP                                                       00022900
      COMMON /SIZE/NROW(20),NCOL(20),NPOINT(20),HVAL(20),XMIN,XMAX,     00023000
     *     YMIN,YMAX                                                    00023100
      INTEGER BPREV,BPT,SPT,UPT                                         00023200
      COMMON /POINT/SLVL(20),BLVL(20),LEFT(400),RIGHT(400),             00023300
     $     LLOC(400),RLOC(400),NLOC(400),SLOC(400),BNDLOC(1000),        00023400
     $     BNDTYP(1000),BROW(400),BLOC(1000)                            00023500
      COMMON /TRACE/ IN,LP,NDBUG,NPRINT,NINI                            00023600
C                                                                       00023700
C                                                                       00023800
C     DETERMINE THE NUMBER OF ROWS AND COLUMNS FOR THE                  00023900
C     COARSER GRIDS                                                     00024000
C                                                                       00024100
      IF(NDBUG.GT.0)WRITE(LP,10)NLEVEL,NR,NC,NBNDPT                     00024200
   10 FORMAT(//6X,'-- GENPIR --'//11X,'NLEVEL,NR,NC,NBNDPT',4I10/)      00024300
      NCOL(1)=NC                                                        00024400
      NROW(1)=NR                                                        00024500
      XDIF=XMAX-XMIN                                                    00024600
      HVAL(1)=XDIF/FLOAT(NCOL(1)-1)                                     00024700
      JLEVEL=NLEVEL                                                     00024800
C                                                                       00024900
      DO 200 J=2,JLEVEL                                                 00025000
C                                                                       00025100
           K=J-1                                                        00025200
           IF(NCOL(K).LT.3 .OR. NROW(K).LT.3) GO TO 400                 00025300
           IF(MOD(NCOL(K),2).NE.0 .AND. MOD(NROW(K),2).NE.0)            00025400
     $          GO TO 160                                               00025500
```

```
C                                                                       00025600
C                 TOO MANY LEVELS OR IMPROPER CONFIGURATION OF GRIDS    00025700
C                                                                       00025800
 100              NLEVEL=K                                              00025900
                  WRITE(LP,20)NLEVEL                                    00026000
 20     FORMAT(1X,'### INCONSISTENT REQUEST FOR LEVELS -',I5,          00026100
     $         ' USED ###')                                            00026200
                  GO TO 240                                             00026300
C                                                                       00026400
C                                                                       00026500
 160              NROW(J)=NROW(J-1)/2+1                                 00026600
                  NCOL(J)=NCOL(J-1)/2+1                                 00026700
                  HVAL(J)=XDIF/FLOAT(NCOL(J)-1)                         00026800
C                                                                       00026900
 200          CONTINUE                                                  00027000
C                                                                       00027100
 240      CONTINUE                                                      00027200
C                                                                       00027300
C                                                                       00027400
C              PRODUCE POINTERS TO THE LEFT AND RIGHT ELEMENTS OF       00027500
C              EACH ROW FOR EACH GRID.                                  00027600
C                                                                       00027700
C                                                                       00027800
C               AUXILLIARY POINTERS FOR THE FINE GRID                   00027900
C                                                                       00028000
         SPT=1                                                          00028100
         UPT=1                                                          00028200
         KMIN=1                                                         00028300
         KMAX=NROW(1)                                                   00028400
         SLVL(1)=SPT                                                    00028500
C                                                                       00028600
         DO 400 KROW=KMIN,KMAX                                          00028700
              LLOC(KROW)=UPT                                            00028800
              UPT=UPT+(RIGHT(KROW)-LEFT(KROW))                          00028900
              RLOC(KROW)=UPT                                            00029000
              UPT=UPT+1                                                 00029100
 400      CONTINUE                                                      00029200
         SPT=KMAX+1                                                     00029300
C                                                                       00029400
C              POINTERS FOR THE COARSER GRIDS                           00029500
C                                                                       00029600
       DO 1000 LEVEL=2,NLEVEL                                           00029700
              SLVL(LEVEL)=SPT                                           00029800
              KMIN=SLVL(LEVEL-1)                                        00029900
              KMAX=SLVL(LEVEL)-1                                        00030000
C                                                                       00030100
C                                                                       00030200
              DO 600 K=KMIN,KMAX,2                                      00030300
C                                                                       00030400
                   LEFT(SPT)=LEFT(K)/2                                  00030500
                   RIGHT(SPT)=RIGHT(K)/2                                00030600
                   LLOC(SPT)=UPT                                        00030700
                   UPT=UPT+RIGHT(SPT)-LEFT(SPT)                         00030800
                   RLOC(SPT)=UPT                                        00030900
                   UPT=UPT+1                                            00031000
                   SPT=SPT+1                                            00031100
 600           CONTINUE                                                 00031200
C                                                                       00031300
 1000     CONTINUE                                                      00031400
         SLVL(NLEVEL+1)=SPT                                             00031500
C                                                                       00031600
C                                                                       00031700
C              DETERMINE THE NUMBER OF POINTS IN EACH GRID              00031800
C                                                                       00031900
         DO 1100 LEVEL=1,NLEVEL                                         00032000
C                                                                       00032100
           LMIN=SLVL(LEVEL)                                             00032200
           LMAX=SLVL(LEVEL+1)-1                                         00032300
C                                                                       00032400
           NPOINT(LEVEL)=(RLOC(LMAX)-LLOC(LMIN))+1                      00032500
C                                                                       00032600
 1100    CONTINUE                                                       00032700
C                                                                       00032800
C              FOR THE LEFT-HAND ELEMENT OF EACH ROW OF A GRID PRODUCE A00032900
C              POINTER TO THE NORTH AND SOUTH ELEMENTS OF THE           00033000
C              LAPLACIAN (CENTERED ABOUT THAT ELEMENT)                  00033100
C                                                                       00033200
         SPT=SPT-1                                                      00033300
C                                                                       00033400
         DO 1400 LEVEL=1,NLEVEL                                         00033500
C                                                                       00033600
C              GENERATE POINTERS FOR GRID LEVEL                         00033700
C                                                                       00033800
           LMIN=SLVL(LEVEL)+1                                           00033900
           LMAX=SLVL(LEVEL+1)-2                                         00034000
C                                                                       00034100
C                GIVE NULL VALUES TO POINTERS THAT WOULD POINT TO       00034200
C                ROWS OUTSIDE THE CURRENT GRID                          00034300
C                                                                       00034400
           SLOC(LMIN-1)=-1                                              00034500
           NLOC(LMAX+1)=-1                                              00034600
C                                                                       00034700
C                GENERATE THE REMAINING POINTERS FOR GRID LEVEL         00034800
C                                                                       00034900
           NLOC(LMIN-1)=LLOC(LMIN)+(LEFT(LMIN-1)-LEFT(LMIN))            00035000
           SLOC(LMAX+1)=LLOC(LMAX)+(LEFT(LMAX+1)-LEFT(LMAX))            00035100
C                                                                       00035200
           DO 1200 LPT=LMIN,LMAX                                        00035300
C                                                                       00035400
                SLOC(LPT)=LLOC(LPT-1)+(LEFT(LPT)-LEFT(LPT-1))           00035500
                NLOC(LPT)=LLOC(LPT+1)+(LEFT(LPT)-LEFT(LPT+1))           00035600
C                                                                       00035700
 1200      CONTINUE                                                     00035800
C                                                                       00035900
C                                                                       00036000
 1400    CONTINUE                                                       00036100
C                                                                       00036200
C                                                                       00036300
C                                                                       00036400
C              SET UP THE LISTS OF BOUNDARY POINT LOCATION AND TYPE     00036500
C              FOR EACH GRID                                            00036600
C                                                                       00036700
         BLVL(1)=1                                                      00036800
         BROW(1)=1                                                      00036900
C                                                                       00037000
C                                                                       00037100
C              DETERMINE WHERE THE ROWS ARE IN THE LIST OF              00037200
C              BOUNDARY LOCATIONS FOR THE FINEST GRID                   00037300
C                                                                       00037400
         JMAX=NROW(1)+1                                                 00037500
         BPT=1                                                          00037600
         BLOC(NBNDPT+1)=-1                                              00037700
         BPREV=BLOC(1)+1                                                00037800
C                                                                       00037900
         DO 1800 JROW=1,JMAX                                            00038000
C                                                                       00038100
 1500      CONTINUE                                                     00038200
C                                                                       00038300
C                                                                       00038400
           IF(BPREV.GT.BLOC(BPT))GO TO 1600                             00038500
```

```
                BPREV=BLOC(BPI)                                         00038600
                BNDLOC(BPI)=BPREV                                       00038700
                BPI=BPI+1                                               00038800
                GO TO 1500                                              00038900
C                                                                       00039000
 1600         BROW(JROW)=BPI                                            00039100
              BPREV=BLOC(BPI)                                           00039200
              BNDLOC(BPI)=BPREV                                         00039300
              BPI=BPI+1                                                 00039400
C                                                                       00039500
 1800     CONTINUE                                                      00039600
C                                                                       00039700
C                                                                       00039800
C             PRODUCE THE LISTS FOR THE COARSER GRIDS                   00039900
C                                                                       00040000
         JPI=NROW(1)+1                                                  00040100
         BPI=NBNDPT+1                                                   00040200
C                                                                       00040300
         DO 2200 LEVEL=2,NLEVEL                                         00040400
C                                                                       00040500
              BLVL(LEVEL)=JPI                                           00040600
C                                                                       00040700
              KMIN=BLVL(LEVEL-1)                                        00040800
              KMAX=JPI-1                                                00040900
C                                                                       00041000
              DO 2100 KPI=KMIN,KMAX,2                                   00041100
                   BROW(JPI)=BPI                                        00041200
                   JPI=JPI+1                                            00041300
                   LMIN=BROW(KPI)                                       00041400
                   LMAX=BROW(KPI+1)-1                                   00041500
C                                                                       00041600
                   DO 2000 LPI=LMIN,LMAX                                00041700
C                                                                       00041800
                        IF(MOD(BNDLOC(LPI),2).EQ.1) GO TO 2000          00041900
C                                                                       00042000
                             BNDLOC(BPI)=BNDLOC(LPI)/2                  00042100
                             BNDTYP(BPI)=BNDTYP(LPI)                    00042200
                             BPI=BPI+1                                  00042300
 2000                   CONTINUE                                        00042400
C                                                                       00042500
 2100              CONTINUE                                             00042600
 2200         CONTINUE                                                  00042700
C                                                                       00042800
         BNDLOC(BPI)=-1                                                 00042900
         BROW(JPI)=BPI                                                  00043000
         BLVL(NLEVEL+1)=JPI                                             00043100
C                                                                       00043200
C                                                                       00043300
C             CONVERT THE ROW AND COLUMN INFORMATION OF THE             00043400
C             BOUNDARY POINT LOCATIONS INTO ABSOLUTE LOCATIONS          00043500
C                                                                       00043600
         KMIN=1                                                         00043700
         KMAX=JPI-1                                                     00043800
C                                                                       00043900
         DO 2600 KPI=KMIN,KMAX                                          00044000
C                                                                       00044100
              LMIN=BROW(KPI)                                            00044200
              LMAX=BROW(KPI+1)-1                                        00044300
              ICOL=LEFT(KPI)                                            00044400
              IPOS=LLOC(KPI)                                            00044500
C                                                                       00044600
              DO 2400 LPI=LMIN,LMAX                                     00044700
                   BNDLOC(LPI)=IPOS+(BNDLOC(LPI)-ICOL)                  00044800
 2400         CONTINUE                                                  00044900
C                                                                       00045000
 2600     CONTINUE                                                      00045100
C                                                                       00045200
C                                                                       00045300
C                                                                       00045400
C                                                                       00045500
         RETURN                                                         00045600
         END                                                            00045700
                                                                        00045800
C                                                                       00045900
C                                                                       00046000
C                                                                       00046100
C                                                                       00046200
C                                                                       00046300
         SUBROUTINE PPOINT(NLEVEL)                                      00046400
C                                                                       00046500
C             PPOINT -- PRINT THE DIFFERENT SETS OF POINTERS FOR        00046600
C                  THE VARIOUS GRIDS                                    00046700
C                                                                       00046800
         INTEGER SIZE,BLVL,RIGHT,RLOC,NLOC,SLOC,BNDLOC,BROW,BLOC,       00046900
        $     BNDTYP                                                    00047000
         COMMON /POINT/SIZE(20),BLVL(20),LEFT(400),RIGHT(400),          00047100
        $     LLOC(400),RLOC(400),NLOC(400),SLOC(400),BNDLOC(1000),     00047200
        $     BNDTYP(1000),BROW(400),BLOC(1000)                         00047300
         COMMON /SIZE/NROW(20),NCOL(20),NPOINT(20),HVAL(20),XMIN,XMAX,  00047400
        *     YMIN,YMAX                                                 00047500
         COMMON /TRACE/IN,LP,NDBUG,NPRINT,NINT                          00047600
C                                                                       00047700
C                                                                       00047800
C             PRINT THE NUMBER OF ROWS AND COLUMNS IN EACH GRID.  ALSO, 00047900
C             GIVE THE NUMBER OF POINTS IN EACH GRID AND ITS SPACING.   00048000
C                                                                       00048100
         WRITE(LP,8)(NROW(J),NCOL(J),NPOINT(J),HVAL(J),J=1,NLEVEL)      00048200
C                                                                       00048300
         KLEVEL=1                                                       00048400
         IF(NDBUG.NE.0)KLEVEL=NLEVEL                                    00048500
C                                                                       00048600
         DO 1000 K=1,KLEVEL                                             00048700
C                                                                       00048800
              H=HVAL(K)                                                 00048900
              WRITE(LP,5)K,H                                            00049000
C                                                                       00049100
              LMIN=SIZE(K)                                              00049200
              LMAX=SIZE(K+1)-1                                          00049300
C                                                                       00049400
              IF(NDBUG.EQ.0)WRITE(LP,10)(LEFT(L),RIGHT(L),L=            00049500
        $          LMIN,LMAX)                                          00049600
              IF(NDBUG.NE.0)WRITE(LP,20)(LEFT(L),RIGHT(L),LLOC(L),      00049700
        $          RLOC(L),SLOC(L),NLOC(L),L=LMIN,LMAX)                00049800
C                                                                       00049900
C                                                                       00050000
              JROW=0                                                    00050100
              LMIN=BLVL(K)                                              00050200
              LMAX=BLVL(K+1)-1                                          00050300
              WRITE(LP,30)                                              00050400
C                                                                       00050500
              DO 400 L=LMIN,LMAX                                        00050600
C                                                                       00050700
                   JMIN=BROW(L)                                         00050800
                   JMAX=BROW(L+1)-1                                     00050900
                   Y=YMIN+H*FLOAT(JROW)                                 00051000
C                                                                       00051100
                   WRITE(LP,40)JROW,Y                                   00051200
                   IF(K.EQ.1)WRITE(LP,50)(BLOC(J),J=JMIN,JMAX)          00051300
                   IF(NDBUG.NE.0)WRITE(LP,60)(BNDLOC(J),J=JMIN,JMAX)    00051400
                   WRITE(LP,70)(BNDTYP(J),J=JMIN,JMAX)                  00051500
                   JROW=JROW+1
```

```
C
 400          CONTINUE
C
 1000     CONTINUE
C
          RETURN
C
 5        FORMAT(1H1,'-- LEVEL',I5,' POINTERS --'/11X,'(H=',E15.7,' )')
 8        FORMAT(//11X,'NROW',6X,'NCOL',4X,'NPOINT',10X,'H'//
     $         (6X,3I10,I20.7))
 10       FORMAT(//12X,'LEFT',5X,'RIGHT'/(11X,I5,5X,I5))
 20       FORMAT(//12X,'LEFT',5X,'RIGHT',6X,'ILOC',6X,'RLOC',6X,'SLOC',
     $         6X,'NLOC'/(11X,6(I5,5X)))
 30       FORMAT(//6X,'BOUNDARY VALUES')
 40       FORMAT(/11X,'-- ROW',I5,5X,'Y',E15.7,' --')
 50       FORMAT(11X,'BLOC  ',2015/(17X,2015))
 60       FORMAT(11X,'BNDLOC',2015/(17X,2015))
 70       FORMAT(11X,'BNDTYP',2015/(17X,2015))
C
          END
C
C
C
C
C
          SUBROUTINE VERIFY(NLEVEL)
C
C              VERIFY -- PRINT DIAGRAMS OF THE GRIDS
C
C
          INTEGER CHAR(20),DASH,BLANK,LINE(115),BPOS,BPI,BTYP,RPI
          INTEGER SLVL,BLVL,RIGHT,RLOC,NLOC,SLOC,BNDLOC,BROW,BLOC,
     $         BNDTYP
          COMMON /IPAGE/IN,LP,NDBUG,NPRINT,NINI
          COMMON /POINT/SLVL(20),BLVL(20),LEFT(400),RIGHT(400),
     $         ILOC(400),RLOC(400),NLOC(400),SLOC(400),BNDLOC(1000),
     $         BNDTYP(1000),BROW(400),BLOC(1000)
          COMMON /SIZE/NROW(20),NCOL(20),NPOINT(20),HVAL(20),XMIN,XMAX,
     *         YMIN,YMAX
          DATA CHAR/'0','1','2','3','4','5','6','7','8','9','A','B','C',
     $         'D',6*'S'/,
     $         BLANK/' '/,DASH/'-'/
C
          WRITE(LP,10)
C
          DO 1000 JLEVEL=1,NLEVEL
C
               WRITE(LP,20)JLEVEL
C
               KPI=SLVL(JLEVEL+1)-1
               RPI=BLVL(JLEVEL+1)-1
               KMAX=NROW(JLEVEL)
C
               DO 800 KROW=1,KMAX
C
                    BPI=BROW(RPI)
                    BPOS=BNDLOC(BPI)
C
                    NBLNK=LEFT(KPI)+1
                    NCHAR=RIGHT(KPI)-LEFT(KPI)+1
                    LPI=ILOC(KPI)
C
                         PRODUCE A LINE OF PRINT
C
                    DO 600 LPOS=1,NCHAR
C
                         LINE(LPOS)=DASH
                         IF(BPOS.NE.LPI) GO TO 500
C
                              BTYP=BNDTYP(BPI)
                              LINE(LPOS)=CHAR(BTYP+1)
                              BPI=BPI+1
                              BPOS=BNDLOC(BPI)
C
                         LPI=LPI+1
C
                    CONTINUE
C
                    LROW=KMAX-KROW
                    WRITE(LP,30)LROW,(BLANK,N=1,NBLNK),
     $                   (LINE(N),N=1,NCHAR)
C
                    RPI=RPI-1
                    KPI=KPI-1
C
               CONTINUE
C
C
 1000     CONTINUE
C
          RETURN
C
 10       FORMAT(1H1,6X,'GRID DIAGRAMS'//11X,'MOLECULES --'//
     $              /21X,'X    X   X         X    X          X'
     $              /21X,'1X   X2  X3X  X4X  5X  X6  7X  X8  X9X'
     $              /21X,'X    X       X          X    X    X'
     $              //21X,'A,B,C,D -- HOECKER MOLECULE'//)
 20       FORMAT(///11X,'LEVEL',I5//1X,'ROW')
 30       FORMAT(1X,I5,115A1)
C
          END
C
C
C
C
C
          SUBROUTINE PREP(LEVEL,G,R)
C
C              PREP -- ASSIGN INITIAL VALUES TO THE SOLUTION VECTOR
C                   (FINE GRID) AND FORM THE CORRESPONDING RIGHT-HAND
C                   SIDE
C
          INTEGER SLVL,BLVL,RIGHT,RLOC,NLOC,SLOC,BNDLOC,BROW,BLOC,
     $         BNDTYP
          COMMON /POINT/SLVL(20),BLVL(20),LEFT(400),RIGHT(400),
     $         ILOC(400),RLOC(400),NLOC(400),SLOC(400),BNDLOC(1000),
     $         BNDTYP(1000),BROW(400),BLOC(1000)
          COMMON /SIZE/NROW(20),NCOL(20),NPOINT(20),HVAL(20),XMIN,XMAX,
     *         YMIN,YMAX
          COMMON /STORE/U(5000),F(5000)
C
C
          JROW=0
          NCOL=NCOL(LEVEL)
C
          JMIN=SLVL(LEVEL)
          JMAX=SLVL(LEVEL+1)-1
          H=HVAL(LEVEL)
          HTOL=.5*H
C
C              PRODUCE VALUES BY ROWS
```

```
C
      DO 400 JPT=JMIN,JMAX                                              00064600
C                                                                       00064700
            Y=YMIN+H*FLOAT(JROW)                                        00064800
            JROW=JROW+1                                                 00064900
C                                                                       00065000
            KMIN=LLOC(JPT)                                              00065100
            KMAX=RLOC(JPT)                                              00065200
            JCOL=LEFT(JPT)                                              00065300
C                                                                       00065400
            DO 300 KPT=KMIN,KMAX                                        00065500
C                                                                       00065600
                  X=XMIN+H*FLOAT(JCOL)                                  00065700
                  JCOL=JCOL+1                                           00065800
C                                                                       00065900
                  U(KPT)=G(X,Y,HTOL)                                    00066000
                  F(KPT)=R(X,Y,HTOL)                                    00066100
C                                                                       00066200
  300       CONTINUE                                                    00066300
C                                                                       00066400
  400 CONTINUE                                                          00066500
C                                                                       00066600
C                                                                       00066700
      RETURN                                                            00066800
C                                                                       00066900
      END                                                               00067000
C                                                                       00067100
C                                                                       00067200
C                                                                       00067300
C                                                                       00067400
C                                                                       00067500
C                                                                       00067600
      SUBROUTINE TEST(G,R)                                              00067700
C                                                                       00067800
      EXTERNAL G,R                                                      00067900
C                                                                       00068000
      INTEGER NOPRNT,PRNT,SHORT,LONG,NOPLOT,PLOT,NOSVEC,SVEC,           00068100
     *      NORHS,RHS                                                   00068200
      COMMON /CONTRL/ NLEVEL,TOL,WMAX                                   00068300
C                                                                       00068400
      DATA NOPRNT,SHORT,NOPLOT,NOSVEC,NORHS/5*0/,                       00068500
     *      PRNT,LONG,PLOT,SVEC,RHS/5*1/                                00068600
C                                                                       00068700
C                                                                       00068800
      LEVEL=1                                                           00068900
      CALL PREP(LEVEL,G,R)                                              00069000
      CALL RESULT(LEVEL,NOPRNT,LONG,PLOT,SVEC,RHS)                      00069100
      CALL RELAX(LEVEL,X,Y,Z)                                           00069200
      CALL RESULT(LEVEL,PRNT,LONG,PLOT,SVEC,RHS)                        00069300
      CALL RESCAL(LEVEL)                                                00069400
      CALL RESULT(LEVEL+1,PRNT,LONG,PLOT,SVEC,RHS)                      00069500
C                                                                       00069600
C                                                                       00069700
      RETURN                                                            00069800
      END                                                               00069900
C                                                                       00070000
C                                                                       00070100
C                                                                       00070200
C                                                                       00070300
C                                                                       00070400
      SUBROUTINE MULTIG                                                 00070500
C                                                                       00070600
C     MULTIG -- THIS ROUTINE PERFORMS THE MULTIGRID PROCESS             00070700
C          DESIGNATED CYCLE-C ON A GRID OF ARBITRARY SHAPE              00070800
C          WITH UNIFORM SPACING.  THE ALGORITHM IS OUTLINED             00070900
C          IN THE ARTICLE BY ACHI BRANDT, MATHEMATICS OF                00071000
C          COMPUTATION, APRIL, 1977, PAGES 333-390.                     00071100
C          DIRICHLET PROBLEMS MAY BE SOLVED WITH NO                     00071200
C          RESTRICTIONS.  NEUMANN AND MIXED NEUMANN-DIRICHLET           00071300
C          PROBLEMS MAY BE SOLVED WITH THE RESTRICTION OF ZERO          00071400
C          NORMAL DERIVATIVES.  THIS SET OF ROUTINES MAY BE             00071500
C          MODIFIED TO ACCEPT NEUMANN OR MIXED PROBLEMS IN              00071600
C          WHICH EITHER THE X OR Y COMPONENT IS ZERO AT EVERY            00071700
C          BOUNDARY POINT.                                              00071800
C                                                                       00071900
C                                                                       00072000
C PARAMETERS --                                                         00072100
C      NLEVEL -- THE NUMBER GRIDS TO BE EMPLOYED IN THE                 00072200
C           MULTIGRID PROCESS                                           00072300
C      TOL -- TOLERANCE ON THE ERROR ON GRID M                          00072400
C      WMAX -- MAXIMUM WORK ALLOWED FOR SOLUTION PROCESS                00072500
C                                                                       00072600
C                                                                       00072700
C VARIABLES AND ARRAYS                                                  00072800
C      U -- STORAGE FOR GRIDS 1 THROUGH M AND ACCOMPANYING              00072900
C           RIGHT-HAND SIDES                                            00073000
C      KLEVEL -- THE CURRENT LEVEL                                      00073100
C      WORK -- THE TOTAL EFFORT EXPENDED IN THE SOLUTION                00073200
C           PROCESS                                                     00073300
C      DELTA -- THE UPPER LIMIT ON THE RATIO OF ERROR ON                00073400
C           GRID K TO THAT ON GRID K+1 IF INJECTION IS TO               00073500
C           OCCUR                                                       00073600
C      EPS(K) -- LIST OF THE ERROR LIMITS THAT MUST BE MET              00073700
C           TO GET BACK TO GRID M BY INJECTION FROM GRID K              00073800
C      ETA -- THE CONVERGENCE FACTOR (CURRENT ERROR DIVIDED             00073900
C           BY PREVIOUS ERROR) MUST BE LESS THAN ETA FOR                00074000
C           RELAXATION SWEEPS TO CONTINUE                               00074100
C                                                                       00074200
C COMMON STORAGE --                                                     00074300
C                                                                       00074400
C           /TRACE/                                                     00074500
C      IN - INPUT DEVICE                                                00074600
C      LP -- OUTPUT DEVICE                                              00074700
C      NDBUG -- SWITCH THAT CONTROLS THE DEBUG TRACE                    00074800
C                0 -- OFF                                               00074900
C                ANYTHING ELSE -- ON                                    00075000
C      NPRINT -- THE NUMBER OF ROWS OF THE SOLUTION VECTOR              00075100
C           AND RIGHT-HAND SIDE TO BE PRINTED WHEN THE                  00075200
C           TRACE IS ON                                                 00075300
C      NINT -- THE NUMBER OF INTERVALS TO BE USED IN PLOTS              00075400
C                                                                       00075500
C           /SIZE/                                                      00075600
C      NROW(*) -- THE NUMBER OF ROWS IN EACH GRID                       00075700
C      NCOL(*) -- THE NUMBER OF COLUMNS IN EACH GRID                    00075800
C      HVAL(*) -- THE SPACING OF EACH GRID (UNIFORM)                    00075900
C      XMIN,XMAX,YMIN,YMAX -- SAME AS PXMIN,ETC.                        00076000
C                                                                       00076100
C           /STORE/                                                     00076200
C      U -- STORAGE FOR GRIDS 1 THROUGH M AND ACCOMPANYING              00076300
C           RIGHT-HAND SIDES                                            00076400
C      F(*) -- STORAGE FOR THE RIGHT-HAND SIDE                          00076500
C                                                                       00076600
C USE OF MULTIG --                                                      00076700
C                                                                       00076800
C      MULTIG REQUIRES THREE ADDITIONAL ROUTINES FOR                    00076900
C      OPERATION.  ONE FUNCTION MUST GIVE THE BOUNDARY                  00077000
C      CONDITIONS OF THE SOLUTION DOMAIN AND AN ESTIMATE                00077100
C      OF THE SOLUTION.  ANOTHER MUST GIVE THE POISSON                  00077200
C      EQUATION RIGHT-HAND SIDE.  A MAIN ROUTINE IS NEEDED              00077300
C      TO CALL MULTIG AND GIVE VALUES TO THE PARAMETERS                 00077400
C      IT REQUIRES.  THE NAMES OF THE TWO FUNCTIONS MUST BE             00077500
```

```
C                 PASSES TO MULTIG THROUGH THE ARGUMENT LIST BY USING 00077600
C                 EXTERNAL.                                           00077700
C                                                                     00077800
C                 A PARTIAL TRACE IS ALWAYS GIVEN ALONG WITH THE      00077900
C                 SOLUTION VECTOR.  AN EXPANDED TRACE CAN BE OBTAINED 00078000
C                 BY SETTING NDBUG TO 0 AND SPECIFYING WITH NPRINT    00078100
C                 THE NUMBER OF ROWS OF THE SOLUTION AND RIGHT-HAND   00078200
C                 SIDE VECTORS TO BE PRINTED.  THIS TRACE WILL BE     00078300
C                 VOLUMINOUS IF MORE THAN 4 GRIDS ARE USED AND        00078400
C                 NPRINT IS LARGER THAN 5.                            00078500
C                                                                     00078600
C                                                                     00078700
C                                                                     00078800
      COMMON /TRACE/IN,IP,NODBUG,NPRINT,NINT                          00078900
      COMMON /CONTRL/ KLEVEL,TOL,WMAX                                 00079000
      COMMON /SIZE/NROW(20),NCOL(20),NPOINT(20),HVAL(20),XMIN,XMAX,   00079100
     *     YMIN,YMAX                                                  00079200
      INTEGER NOPRNT,PRNT,SHORT,LONG,NOPLOT,PLOT,NOSVEC,SVEC,         00079300
     *     NORHS,RHS                                                  00079400
      DIMENSION EPS(20),ERRLIM(20)                                    00079500
      DATA NOPRNT,SHORT,NOPLOT,NOSVEC,NORHS/5*0/,                     00079600
     *     PRNT,LONG,PLOT,SVEC,RHS/5*1/                               00079700
C                                                                     00079800
C                                                                     00079900
      EPS(1)=TOL                                                      00080000
      KLEVEL=1                                                        00080100
      MAXLVL=KLEVEL                                                   00080200
      ERRLIM(1)=1.E30                                                 00080300
      ERR=1.E30                                                       00080400
      WORK=0.                                                         00080500
      DELTA=.2                                                        00080600
      ETA=.8                                                          00080700
C                                                                     00080800
C                                                                     00080900
      WRITE(IP,10)MAXLVL,TOL,WMAX                                     00081000
C                                                                     00081100
      WRITE(IP,5)                                                     00081200
      WRITE(IP,40)                                                    00081300
C                                                                     00081400
C                                                                     00081500
C           PERFORM THE MULTIGRID PROCESS - RELAXATION SWEEPS,        00081600
C           CALCULATION OF RESIDUALS, AND INJECTION OF RESIDUALS      00081700
C                                                                     00081800
C                                                                     00081900
 200  CONTINUE                                                        00082000
C                                                                     00082100
C                                                                     00082200
           ERRPRV=ERR                                                 00082300
C                                                                     00082400
C                PERFORM A GAUSS-SEIDEL RELAXATION SWEEP              00082500
C                                                                     00082600
           CALL RELAX(KLEVEL,RESSUM,DIFMAX,ERR,UAV)                   00082700
           IF(NDBUG.NE.0)CALL RESULT(KLEVEL,NOPRNT,SHORT,PLOT,        00082800
     $                               SVEC,NORHS)                      00082900
C                                                                     00083000
           WORK=WORK+4.**(1-KLEVEL)                                   00083100
           IF(NDBUG.NE.0)WRITE(IP,40)                                 00083200
           ERRLOG=-1.0E70                                             00083205
           IF(ERR.GT.0.0)ERRLOG=ALOG10(ERR)                           00083210
           WRITE(IP,15)KLEVEL,DIFMAX,ERR,ERRLOG,WORK                  00083300
C                                                                     00083400
C                                                                     00083500
C                DETERMINE IF THE WORK LIMIT HAS BEEN EXCEEDED        00083600
C                                                                     00083700
C                                                                     00083800
```

```
      IF(WORK.LT.WMAX) GO TO 220                                      00083900
C                                                                     00084000
C               LIMIT EXCEEDED - PRINT SOLUTION VECTOR AND QUIT       00084100
C                                                                     00084200
           WRITE(IP,20)                                               00084300
           KLEVEL=1                                                   00084400
           CALL RESULT(KLEVEL,PRNT,SHORT,PLOT,SVEC,NORHS)             00084500
           RETURN                                                     00084600
C                                                                     00084700
C               DETERMINE IF CONVERGENCE IS FAST ENOUGH               00084800
C                                                                     00084900
  220 IF(ERR.LT.EPS(KLEVEL)) GO TO 240                                00085000
C                                                                     00085100
C               CONTINUE THE RELAXATION SWEEPS IF THE PRESENT         00085200
C               GRID IS THE COARSEST GRID OR IF THE RATE OF           00085300
C               CONVERGENCE IS WITHIN THE DESIRED RANGE               00085400
C               OR IF THE ERROR IS CONVERGING AS REQUIRED             00085500
C                                                                     00085600
           IF(KLEVEL.EQ.MAXLVL .OR. ERR/ERRPRV .LT.ETA)               00085700
     $          GO TO 200                                            00085800
C                                                                     00085900
C                                                                     00086000
C                 DETERMINE IF THE PROCESS ON THE COARSER GRIDS     00086100
C                 IS REDUCING THE ERROR ON THE PRESENT GRID          00086200
C                                                                     00086300
           IF(ERR.LT.ERRLIM(KLEVEL))GO TO 230                         00086400
C                                                                     00086500
C                    ELIMINATE THE COARSER GRIDS FROM USE           00086600
C                                                                     00086700
                 NXTLVL=KLEVEL+1                                      00086710
                 WRITE(IP,50)NXTLVL,MAXLVL                            00086720
                 MAXLVL=KLEVEL                                        00086800
                 GO TO 200                                            00086900
C                                                                     00087000
C                                                                     00087100
C                                                                     00087200
C               THE CONVERGENCE IS SLOW - TAKE THE RESIDUALS          00087300
C               THAT WOULD RESULT FROM ANOTHER RELAXATION SWEEP       00087400
C               TO A COARSER GRID.                                    00087500
C                                                                     00087600
C                                                                     00087700
  230      ERRLIM(KLEVEL)=ERR                                         00087800
           ERRLIM(KLEVEL+1)=1.E30                                     00087900
           CALL RESCAL(KLEVEL)                                        00088000
           KLEVEL=KLEVEL+1                                            00088100
           IF(NDBUG.NE.0)CALL RESULT(KLEVEL,PRNT,SHORT,PLOT,          00088200
     $                               NOSVEC,NORHS)                    00088300
           EPS(KLEVEL)=DELTA*ERR                                      00088400
           CALL ZERO(KLEVEL)                                          00088500
           ERR=1.0E30                                                 00088600
           GO TO 200                                                  00088700
C                                                                     00088800
C                                                                     00088900
C                                                                     00089000
C                                                                     00089100
C               THE ERROR FOR THIS GRID IS WITHIN SPECIFIED           00089200
C               TOLERANCES.  IF THE CURRENT GRID IS NOT THE           00089300
C               FINE GRID, INJECT THE SOLUTION OF THE CURRENT         00089400
C               RESIDUAL EQUATION INTO THE NEXT FINER GRID.           00089500
C                                                                     00089600
  240 IF(KLEVEL.NE.1) GO TO 250                                       00089700
C                                                                     00089800
C                    THE RESIDUAL ON THE FINE GRID IS                 00089900
C                    WITHIN THE SPECIFIED TOLERANCE SO QUIT           00090000
C                                                                     00090100
```

```
      JCMAX=RLOC(JROW)                                                  00096700
C                                                                       00096800
      BPT=BROW(IROW)                                                    00096900
      BPOS=BNDLOC(BPT)                                                  00097000
C                                                                       00097100
      DO 4000 JC=JCMIN,JCMAX                                            00097200
C                                                                       00097300
            UOLD=U(JC)                                                  00097400
C                                                                       00097500
C           CHECK FOR BOUNDARY POINT                                    00097600
C                                                                       00097700
            IF(BPOS.EQ.JC) GO TO 60                                     00097800
C                                                                       00097900
C                 OTHERWISE USE A 4-PT MOLECULE                         00098000
C                                                                       00098100
            U(JC)=.25*(U(JC-1)+U(JC+1)+U(JN)+U(JS)-F(JC))               00098200
            GO TO 4800                                                  00098300
C                                                                       00098400
C                                                                       00098500
 60         BTYP=BNDTYP(BPT)                                            00098600
            BPT=BPT+1                                                   00098700
            BPOS=BNDLOC(BPT)                                            00098800
C                                                                       00098900
C           CHECK FOR DIRICHLET POINT                                   00099000
C                                                                       00099100
            IF(BTYP.EQ.0) GO TO 4800                                    00099200
C                                                                       00099300
C           SELECT THE INDICATED COMPUTATIONAL                          00099400
C           MOLECULE                                                    00099500
C                                                                       00099600
C           X    X   X         X    X          X                       00099700
C           1X   X2  X3X  X4X  5X   X6  7X   X8  X9X                   00099800
C           X    X        X          X    X    X                       00099900
C                                                                       00100000
C           10,11,12,13 HOFCKER MOLECULE                                00100100
C                                                                       00100200
C                                                                       00100300
            GO TO(100,200,300,400,500,600,700,800,900,                  00100400
     $           1000,1100,1200,1300),BTYP                              00100500
C                                                                       00100600
C                                                                       00100700
 100        U(JC)=.25*(U(JN)+U(JS)+U(JC+1)+U(JC+1)-F(JC))               00100800
            GO TO 4800                                                  00100900
C                                                                       00101000
C                                                                       00101100
 200        U(JC)=.25*(U(JN)+U(JS)+U(JC-1)+U(JC-1)-F(JC))               00101200
            GO TO 4800                                                  00101300
C                                                                       00101400
C                                                                       00101500
 300        U(JC)=.25*(U(JN)+U(JN)+U(JC+1)+U(JC-1)-F(JC))               00101600
            GO TO 4800                                                  00101700
C                                                                       00101800
C                                                                       00101900
 400        U(JC)=.25*(U(JS)+U(JS)+U(JC+1)+U(JC-1)-F(JC))               00102000
            GO TO 4800                                                  00102100
C                                                                       00102200
C                                                                       00102300
 500        U(JC)=.25*(U(JN)+U(JN)+U(JC+1)+U(JC+1)-F(JC))               00102400
            GO TO 4800                                                  00102500
C                                                                       00102600
C                                                                       00102700
 600        U(JC)=.25*(U(JN)+U(JN)+U(JC-1)+U(JC-1)-F(JC))               00102800
            GO TO 4800                                                  00102900
C                                                                       00103000
C                                                                       00103100
```

```
                  CALL RESULT(KLEVEL,PRNT,LONG,PLOT,SVEC,NORHS)         00090200
                  RETURN                                                00090300
C                                                                       00090400
C                                                                       00090500
C                                                                       00090600
C                 INJECT THE RESIDUAL - INTERPOLATE IT AND              00090700
C                 ADD TO THE NEXT FINER GRID                            00090800
C                                                                       00090900
 250        CALL INTADD(KLEVEL)                                         00091000
            KLEVEL=KLEVEL-1                                             00091100
            IF(NDBUG.GE.100)CALL RESULT(KLEVEL,PRNT,SHORT,PLOT,         00091200
     $                          SVEC,NORHS)                             00091300
            ERR=1.0E30                                                  00091400
            GO TO 200                                                   00091500
C                                                                       00091600
C                                                                       00091700
C                                                                       00091800
C                                                                       00091900
  5         FORMAT(1H1)                                                 00092000
 10         FORMAT(1H1,5X,'-- MULTIG --'//                              00092100
     *           11X,'MAXLVL,TOL,WMAX',I5,2E20.7)                       00092200
 15         FORMAT(6X,I5,4E15.7)                                        00092300
 20         FORMAT(1H0,10X,'-- WORK LIMIT EXCEEDED --')                 00092400
 40         FORMAT(/8X,'LEVEL,DIFMAX,ERR,ERRLOG,WORK')                  00092500
 50       FORMAT(/11X,'DROPPING GRIDS ',I5,' TO ',I5/)                  00092650
C                                                                       00092700
C                                                                       00092800
C                                                                       00092900
      END                                                               00093000
C                                                                       00093100
C                                                                       00093200
C                                                                       00093300
C                                                                       00093400
C                                                                       00093500
      SUBROUTINE RELAX(LEVEL,RESSUM,DIFMAX,ERR,UAV)                     00093600
C                                                                       00093700
C           RELAX -- PERFORM A RELAXATION SWEEP ON GRID LEVEL           00093800
C                                                                       00093900
      INTEGER BPT,BPOS,BTYP                                             00094000
      INTEGER SLVL,BLVL,RIGHT,RLOC,NLOC,SLOC,BNDLOC,BROW,BLOC,          00094100
     $     BNDTYP                                                       00094200
      COMMON /TRACE/IN,LP,NDBUG,NPRINT,NINT                             00094300
      COMMON /POINT/SLVL(20),BLVL(20),LEFT(400),RIGHT(400),             00094400
     $     LLOC(400),RLOC(400),NLOC(400),SLOC(400),BNDLOC(1000),        00094500
     $     BNDTYP(1000),BROW(400),BLOC(1000)                            00094600
      COMMON /SIZE/NROW(20),NCOL(20),NPOINT(20),NVAL(20),XMIN,XMAX,     00094700
     *     YMIN,YMAX                                                    00094800
      COMMON /STORE/U(5000),F(5000)                                     00094900
C                                                                       00095000
C                                                                       00095100
      IF(NDBUG.NE.0)WRITE(LP,10)LEVEL                                   00095200
C                                                                       00095300
      USUM=0.                                                           00095400
      RESSUM=0.                                                         00095500
      DIFMAX=0.                                                         00095600
C                                                                       00095700
      IROW=BLVL(LEVEL)                                                  00095800
      JMIN=SLVL(LEVEL)                                                  00095900
      JMAX=SLVL(LEVEL+1)-1                                              00096000
C                                                                       00096100
      DO 6000 JROW=JMIN,JMAX                                            00096200
C                                                                       00096300
            JN=NLOC(JROW)                                               00096400
            JCMIN=LLOC(JROW)                                            00096500
            JS=SLOC(JROW)                                               00096600
```

```
  700                     U(JC)=.25*(U(JS)+U(JS)+U(JC+1)+U(JC+1)-F(JC))   00103200
                               GO TO 4800                                00103300
C                                                                        00103400
C                                                                        00103500
  800                     U(JC)=.25*(U(JS)+U(JS)+U(JC-1)+U(JC-1)-F(JC))   00103600
                               GO TO 4800                                00103700
C                                                                        00103800
C                                                                        00103900
  900                     U(JC)=.25*(U(JN)+U(JS)+U(JC+1)+U(JC-1)-F(JC))   00104000
                               GO TO 4800                                00104100
C                                                                        00104200
C                                                                        00104300
 1000                     U(JC)=(U(JC-1)+U(JN)+2.*(U(JC+1)+U(JS))-        00104400
     $                           F(JC))/6.                               00104500
                               GO TO 4800                                00104600
C                                                                        00104700
C                                                                        00104800
 1100                     U(JC)=(U(JC+1)+U(JN)+2.*(U(JC-1)+U(JS))-        00104900
     $                           F(JC))/6.                               00105000
                               GO TO 4800                                00105100
C                                                                        00105200
C                                                                        00105300
 1200                     U(JC)=(U(JC-1)+U(JS)+2.*(U(JC+1)+U(JN))-        00105400
     $                           F(JC))/6.                               00105500
                               GO TO 4800                                00105600
C                                                                        00105700
C                                                                        00105800
 1300                     U(JC)=(U(JC+1)+U(JS)+2.*(U(JC-1)+U(JN))-        00105900
     $                           F(JC))/6.                               00106000
                               GO TO 4800                                00106100
C                                                                        00106200
C                                                                        00106300
 4800                     JN=JN+1                                         00106400
                          JS=JS+1                                         00106500
C                                                                        00106600
                          DIFF=ABS(U(JC)-UOLD)                            00106700
                          RESSUM=RESSUM+DIFF                              00106800
                          USUM=USUM+ABS(U(JC))                            00106900
                          IF(DIFMAX.LT.DIFF)DIFMAX=DIFF                   00107000
C                                                                        00107100
 4000           CONTINUE                                                 00107200
C                                                                        00107300
C                                                                        00107400
               LROW=LROW+1                                               00107500
C                                                                        00107600
C                                                                        00107700
 6000     CONTINUE                                                       00107800
C                                                                        00107900
C              DETERMINE THE AVERAGE RESIDUAL                            00108000
C                                                                        00108100
          ERR=RESSUM/FLOAT(NPOINT(LEVEL))                                00108200
          UAV=USUM/FLOAT(NPOINT(LEVEL))                                  00108300
C                                                                        00108400
C                                                                        00108500
          RETURN                                                         00108600
C                                                                        00108700
  10      FORMAT(//6X,'--- RELAX --- (LEVEL',I5,' )')                    00108800
C                                                                        00108900
C                                                                        00109000
          END                                                            00109100
C                                                                        00109200
C                                                                        00109300
C                                                                        00109400
C                                                                        00109600
C                                                                        00109700
```

```
          SUBROUTINE RESCAL(LEVEL)                                       00109800
C                                                                        00109900
C         RELAX -- PERFORM A RELAXATION SWEEP ON GRID LEVEL              00110000
C              IN ORDER TO CALCULATE THE COARSE RESIDUAL                 00110100
C                                                                        00110200
          INTEGER BPT,BPOS,BTYP,RSWA,RSWB,CPT,SPT,TPT                    00110300
          INTEGER SLVL,BLVL,RIGHT,RLOC,NLOC,SLOC,BNDLOC,BROW,BLOC,       00110400
     $         BNDTYP                                                    00110500
          COMMON /TRACE/IN,LP,NDBUG,NPRINT,NINT                          00110600
          COMMON /POINT/SLVL(20),BLVL(20),LEFT(400),RIGHT(400),          00110700
     $         LLOC(400),RLOC(400),NLOC(400),SLOC(400),BNDLOC(1000),     00110800
     $         BNDTYP(1000),BROW(400),BLOC(1000)                         00110900
          COMMON /SIZE/NROW(20),NCOL(20),NPOINT(20),HVAL(20),XMIN,XMAX,  00111000
     *         YMIN,YMAX                                                 00111100
          COMMON /STORE/U(5000),F(5000)                                  00111200
          DIMENSION USAVE(2,100)                                         00111300
          DATA SCALE/4.0/                                                00111400
C                                                                        00111500
C                                                                        00111600
          IF(NDBUG.NE.0)WRITE(LP,10)LEVEL                                00111700
C                                                                        00111800
C                                                                        00111900
C                                                                        00112000
C                                                                        00112100
          LROW=BLVL(LEVEL+1)                                             00112200
          JROW=SLVL(LEVEL)                                               00112300
          KMAX=SLVL(LEVEL+2)-1                                           00112400
          KMIN=SLVL(LEVEL+1)                                             00112500
C                                                                        00112600
C                                                                        00112700
C                                                                        00112800
          DO 6000 KROW=KMIN,KMAX                                         00112900
C                                                                        00113000
               JN=NLOC(JROW)                                             00113100
               JS=SLOC(JROW)                                             00113200
               JC=LLOC(JROW)                                             00113300
C                                                                        00113400
               BPT=BROW(LROW)                                            00113500
               BPOS=BNDLOC(BPT)                                          00113600
C                                                                        00113700
               KCMIN=LLOC(KROW)                                          00113800
               KCMAX=RLOC(KROW)                                          00113900
C                                                                        00114000
               DO 4000 KC=KCMIN,KCMAX                                    00114100
C                                                                        00114200
C                                                                        00114300
C                   CHECK FOR A BOUNDARY POINT                           00114400
C                                                                        00114500
                    IF(BPOS.EQ.KC) GO TO 60                              00114600
C                                                                        00114700
C                        OTHERWISE USE A 4-PT MOLECULE                   00114800
C                                                                        00114900
                    UVAL=(U(JC-1)+U(JC+1)+U(JN)+U(JS)-F(JC))             00115000
                    GO TO 4800                                           00115100
C                                                                        00115200
C                                                                        00115300
 60                 BTYP=BNDTYP(BPT)                                     00115400
                    BPT=BPT+1                                            00115500
                    BPOS=BNDLOC(BPT)                                     00115600
C                                                                        00115700
C                        CHECK FOR DIRICHLET POINT                       00115800
C                                                                        00115900
                    IF(BTYP.NE.0)GO TO 90                                00116000
C                                                                        00116100
                         UVAL=U(JC)                                      00116200
```

```
                GO TO 4800                                              00116300
C                                                                       00116400
C                                                                       00116500
C                                                                       00116600
C               SELECT THE INDICATED COMPUTATIONAL                      00116700
C               MOLECULE                                                00116800
C                                                                       00116900
C               X    X   X        X    X           X                   00117000
C               1X  X2  X3X  X4X  5X  X6  7X  X8  X9X                   00117100
C               X    X        X          X   X   X                      00117200
C                                                                       00117300
C               10,11,12,13 HOECKER MOLECULE                            00117400
C                                                                       00117500
C                                                                       00117600
  90                                                                    00117700
     $          GO TO(100,200,300,400,500,600,700,800,900,             00117800
                1000,1100,1200,1300),BTYP                               00117900
C                                                                       00118000
C                                                                       00118100
 100            UVAL=(U(JN)+U(JS)+U(JC+1)+U(JC+1)-F(JC))                00118200
                GO TO 4800                                              00118300
C                                                                       00118400
C                                                                       00118500
 200            UVAL=(U(JN)+U(JS)+U(JC-1)+U(JC-1)-F(JC))                00118600
                GO TO 4800                                              00118700
C                                                                       00118800
C                                                                       00118900
 300            UVAL=(U(JN)+U(JN)+U(JC+1)+U(JC-1)-F(JC))                00119000
                GO TO 4800                                              00119100
C                                                                       00119200
C                                                                       00119300
 400            UVAL=(U(JS)+U(JS)+U(JC+1)+U(JC-1)-F(JC))                00119400
                GO TO 4800                                              00119500
C                                                                       00119600
C                                                                       00119700
 500            UVAL=(U(JN)+U(JN)+U(JC+1)+U(JC+1)-F(JC))                00119800
                GO TO 4800                                              00119900
C                                                                       00120000
C                                                                       00120100
 600            UVAL=(U(JN)+U(JN)+U(JC-1)+U(JC-1)-F(JC))                00120200
                GO TO 4800                                              00120300
C                                                                       00120400
C                                                                       00120500
 700            UVAL=(U(JS)+U(JS)+U(JC+1)+U(JC+1)-F(JC))                00120600
                GO TO 4800                                              00120700
C                                                                       00120800
C                                                                       00120900
 800            UVAL=(U(JS)+U(JS)+U(JC-1)+U(JC-1)-F(JC))                00121000
                GO TO 4800                                              00121100
C                                                                       00121200
C                                                                       00121300
 900            UVAL=(U(JN)+U(JS)+U(JC+1)+U(JC-1)-F(JC))                00121400
                GO TO 4800                                              00121500
C                                                                       00121600
C                                                                       00121700
1000            UVAL=U(JC-1)+U(JN)+2.*(U(JC+1)+U(JS))-F(JC)             00121800
                GO TO 4700                                              00121900
C                                                                       00122000
C                                                                       00122100
1100            UVAL=U(JC+1)+U(JN)+2.*(U(JC-1)+U(JS))-F(JC)             00122200
                GO TO 4700                                              00122300
C                                                                       00122400
C                                                                       00122500
1200            UVAL=U(JC-1)+U(JS)+2.*(U(JC+1)+U(JN))-F(JC)             00122600
                GO TO 4700                                              00122700
C                                                                       00122800
1300            UVAL=U(JC+1)+U(JS)+2.*(U(JC-1)+U(JN))-F(JC)             00122900
                GO TO 4700                                              00123000
C                                                                       00123100
 4700           F(KC)=SCALE*(6.*U(JC)-UVAL)                             00123200
                GO TO 4900                                              00123300
C                                                                       00123400
 4800           F(KC)=SCALE*(4.*U(JC)-UVAL)                             00123500
C                                                                       00123600
C                                                                       00123700
 4900           JC=JC+2                                                 00123800
                JN=JN+2                                                 00123900
                JS=JS+2                                                 00124000
C                                                                       00124100
 4000           CONTINUE                                                00124200
C                                                                       00124300
C                                                                       00124400
                JROW=JROW+2                                             00124500
                LROW=LROW+1                                             00124600
C                                                                       00124700
 6000   CONTINUE                                                        00124800
C                                                                       00124900
C                                                                       00125000
        RETURN                                                          00125100
C                                                                       00125200
  10    FORMAT(//6X,'--- RESCAL --- (LEVEL',I5,' )')                    00125300
C                                                                       00125400
C                                                                       00125500
        END                                                             00125600
C                                                                       00125700
C                                                                       00125800
C                                                                       00125900
C                                                                       00126000
                                                                        00126100
        SUBROUTINE ZERO(LEVEL)                                          00126200
C                                                                       00126300
C       ZERO -- ZERO GRID LEVEL                                         00126400
C                                                                       00126500
        INTEGER SLVL,BLVL,RIGHT,RLOC,NLOC,SLOC,BNDLOC,BROW,BLOC,        00126600
     $      BNDTYP                                                      00126700
        COMMON /POINT/SLVL(20),BLVL(20),LEFT(400),RIGHT(400),           00126800
     $      LLOC(400),RLOC(400),NLOC(400),SLOC(400),BNDLOC(1000),       00126900
     $      BNDTYP(1000),BROW(400),BLOC(1000)                           00127000
        COMMON /SIZE/NROW(20),NCOL(20),NPOINT(20),HVAL(20),XMIN,XMAX,   00127100
     *      YMIN,YMAX                                                   00127200
        COMMON /STORE/U(5000),F(5000)                                   00127300
        COMMON /TRACE/IN,LP,NDBUG,NPRINT,NINI                           00127400
C                                                                       00127500
C                                                                       00127600
        IF(NDBUG.NE.0)WRITE(LP,10)LEVEL                                 00127700
C                                                                       00127800
        JMIN=SLVL(LEVEL)                                                00127900
        JMAX=SLVL(LEVEL+1)-1                                            00128000
C                                                                       00128100
        KMIN=LLOC(JMIN)                                                 00128200
        KMAX=RLOC(JMAX)                                                 00128300
C                                                                       00128400
        DO 100 K=KMIN,KMAX                                              00128500
C                                                                       00128600
            U(K)=0.                                                     00128700
C                                                                       00128800
 100    CONTINUE                                                        00128900
C                                                                       00129000
        RETURN                                                          00129100
C                                                                       00129200
  10    FORMAT(//6X,'--- ZERO --- (LEVEL',I5,' )')                      00129300
```

```
C                                                                       00129400
      END                                                               00129500
C                                                                       00129600
C                                                                       00129700
C                                                                       00129800
C                                                                       00129900
C                                                                       00130000
      SUBROUTINE INTADD(LEVEL)                                          00130100
C                                                                       00130200
C           INTADD -- INTERPOLATE THE SOLUTION OF THE RESIDUAL          00130300
C                EQUATION ON THE COARSE GRID (LEVEL) AND ADD IT         00130400
C                TO THE NEXT FINEST GRID.                               00130500
C                                                                       00130600
C                                                                       00130700
      INTEGER SLVL,BLVL,RIGHT,RLOC,NLOC,SLOC,BNDLOC,BROW,BLOC,          00130800
     $     BNDTYP                                                       00130900
      INTEGER BPTTOP,BPTBOT,BPOST,BPOSB,CHECKB                          00131000
      COMMON /STORE/U(5000),F(5000)                                     00131100
      COMMON /POINT/SLVL(20),BLVL(20),LEFT(400),RIGHT(400),             00131200
     $     TLOC(400),RLOC(400),NLOC(400),SLOC(400),BNDLOC(1000),        00131300
     $     BNDTYP(1000),BROW(400),BLOC(1000)                            00131400
      COMMON /SIZE/NROW(20),NCOL(20),NPOINT(20),HVAL(20),XMIN,XMAX,     00131500
     *     YMIN,YMAX                                                    00131600
      COMMON /TRACE/ IN,LP,NDBUG,NPRINT,NINT                            00131700
      COMMON /DPOINT/JSPTR,KSPTR,NR,MAXROW,NSEG,KSEG(3),                00131800
     $     JBOTL(3),JBOTR(3),JTOPL(3),KBOTL(3),KTOPL(3)                 00131900
C                                                                       00132000
C                                                                       00132100
      IF(NDBUG.NE.0)WRITE(LP,10)LEVEL                                   00132200
C                                                                       00132300
      JSPTR=SLVL(LEVEL)                                                 00132400
      KSPTR=SLVL(LEVEL-1)                                               00132500
      IROW=BLVL(LEVEL-1)                                                00132600
C                                                                       00132700
      MAXROW=NROW(LEVEL)                                                00132800
C                                                                       00132900
C                                                                       00133000
      DO 1000 NR=1,MAXROW                                               00133100
         CALL DECIDE                                                    00133200
C                                                                       00133300
            BPTTOP=BROW(IROW+1)                                         00133400
            BPTBOT=BROW(IROW)                                           00133500
            BPOST=BNDLOC(BPTTOP)                                        00133600
            BPOSB=BNDLOC(BPTBOT)                                        00133700
            IROW=IROW+2                                                 00133800
C                                                                       00133900
         DO 800 JSEG=1,NSEG                                             00134000
C                                                                       00134100
            JBMIN=JBOTL(JSEG)                                           00134200
            JBMAX=JBOTR(JSEG)-1                                         00134300
            JT=JTOPL(JSEG)                                              00134400
C                                                                       00134500
            KB=KBOTL(JSEG)                                              00134600
            KT=KTOPL(JSEG)                                              00134700
C                                                                       00134800
C                                                                       00134900
C                                                                       00135000
C                                                                       00135100
            IF(KSEG(JSEG).EQ.1)GO TO 500                                00135200
C                                                                       00135300
C                              TOP EDGE (2 POINTS)                      00135400
C                                                                       00135500
               DO 400 JB=JBMIN,JBMAX                                    00135600
C                                                                       00135700
                  IF(KB.NE.BPOSB)GO TO 320                              00135800
C                                                                       00135900
                  IF(CHECKB(BPOSB,BPTBOT).EQ.0)GO TO 330                00136000
C                                                                       00136100
C                                                                       00136200
C                                                                       00136300
  320             U(KB)=U(KB)+U(JB)                                     00136400
  330             KB=KB+1                                               00136500
C                                                                       00136600
                  IF(KB.NE.BPOSB)GO TO 340                              00136700
C                                                                       00136800
                  IF(CHECKB(BPOSB,BPTBOT).EQ.0)GO TO 350                00136900
C                                                                       00137000
C                                                                       00137100
C                                                                       00137200
  340             U(KB)=U(KB)+.5*(U(JB)+U(JB+1))                        00137300
  350             KB=KB+1                                               00137400
C                                                                       00137500
  400          CONTINUE                                                 00137600
C                                                                       00137700
               GO TO 800                                                00137800
C                                                                       00137900
C                                                                       00138000
C                                                                       00138100
  500       CONTINUE                                                    00138200
C                                                                       00138300
C                    4 POINTS                                           00138400
C                                                                       00138500
               DO 600 JB=JBMIN,JBMAX                                    00138600
C                                                                       00138700
                  IF(KB.NE.BPOSB)GO TO 520                              00138800
C                                                                       00138900
                  IF(CHECKB(BPOSB,BPTBOT).EQ.0)GO TO 530                00139000
C                                                                       00139100
C                                                                       00139200
C                                                                       00139300
  520             U(KB)=U(KB)+U(JB)                                     00139400
  530             KB=KB+1                                               00139500
C                                                                       00139600
                  IF(KB.NE.BPOSB)GO TO 540                              00139700
C                                                                       00139800
                  IF(CHECKB(BPOSB,BPTBOT).EQ.0)GO TO 550                00139900
C                                                                       00140000
C                                                                       00140100
C                                                                       00140200
  540             U(KB)=U(KB)+.5*(U(JB)+U(JB+1))                        00140300
  550             KB=KB+1                                               00140400
C                                                                       00140500
C                                                                       00140600
C                                                                       00140700
                  IF(KT.NE.BPOST)GO TO 560                              00140800
C                                                                       00140900
                  IF(CHECKB(BPOST,BPTTOP).EQ.0)GO TO 570                00141000
C                                                                       00141100
C                                                                       00141200
C                                                                       00141300
  560             U(KT)=U(KT)+.5*(U(JB)+U(JT))                          00141400
  570             KT=KT+1                                               00141500
C                                                                       00141600
C                                                                       00141700
C                                                                       00141800
C                                                                       00141900
                  U(KT)=U(KT)+.25*(U(JB)+U(JB+1)+                       00142000
     $                 U(JT)+U(JT+1))                                   00142100
                  KT=KT+1                                               00142200
                  JT=JT+1                                               00142300
```

```
C                                                                       00142400
 600                  CONTINUE                                          00142500
C                                                                       00142600
C                                                                       00142700
C                                                                       00142800
C                         TAKE CARE OF ANY POINT ON THE RIGHT OF A FINE 00142900
C                         GRID SEGMENT THAT HAS NOT BEEN CHANGED        00143000
C                                                                       00143100
                     IF(KT.NE.BPOST)GO TO 620                           00143200
C                                                                       00143300
                     IF(CHECKB(BPOST,BPITOP).EQ.0)GO TO 800             00143400
C                                                                       00143500
C                                                                       00143600
 620                 U(KT)=.5*(U(JB)+U(JT))                             00143700
C                                                                       00143800
C                                                                       00143900
 800              CONTINUE                                              00144000
C                                                                       00144100
C                                                                       00144200
C                         TAKE CARE OF THE BOTTOM POINT ON THE RIGHT SIDE OF 00144300
C                         THE LAST SEGMENT                              00144400
C                                                                       00144500
C                                                                       00144600
               IF(KB.NE.BPOSB)GO TO 820                                 00144700
C                                                                       00144800
               IF(CHECKB(BPOSB,BPIBOT).EQ.0)GO TO 840                   00144900
C                                                                       00145000
C                                                                       00145100
 820           U(KB)=U(KB)+U(JB)                                        00145200
C                                                                       00145300
C                    MOVE ROW POINTERS                                  00145400
C                                                                       00145500
 840           JSPIR=JSPIR+1                                            00145600
               KSPIR=KSPIR+2                                            00145700
C                                                                       00145800
C                                                                       00145900
 1000      CONTINUE                                                     00146000
C                                                                       00146100
C                                                                       00146200
C                                                                       00146300
          RETURN                                                        00146400
C                                                                       00146500
  10       FORMAT(//6X,'--- INIADB --- (LEVEL',I5,' )')                 00146600
C                                                                       00146700
           END                                                          00146800
C                                                                       00146900
C                                                                       00147000
C                                                                       00147100
C                                                                       00147200
C                                                                       00147300
           FUNCTION CHECKB(BPOS,BPI)                                    00147400
C                                                                       00147500
C                    DETERMINE THE TYPE OF THE BOUNDARY POINT AT BPOS   00147600
C                                                                       00147700
           INTEGER CHECKB,BPOS,BPI                                      00147800
           INTEGER SLVL,BLVL,RIGHT,RLOC,NLOC,SLOC,BNDLOC,BROW,BLOC,     00147900
     $          BNDTYP                                                 00148000
           COMMON /POINT/SLVL(20),BLVL(20),LEFT(400),RIGHT(400),        00148100
     $          LLOC(400),RLOC(400),NLOC(400),SLOC(400),BNDLOC(1000),  00148200
     $          BNDTYP(1000),BROW(400),BLOC(1000)                      00148300
C                                                                       00148400
C                                                                       00148500
           CHECKB=BNDTYP(BPI)                                           00148600
           BPI=BPI+1                                                    00148700
           BPOS=BNDLOC(BPI)                                             00148800
C                                                                       00148900
           RETURN                                                       00149000
           END                                                          00149100
C                                                                       00149200
C                                                                       00149300
C                                                                       00149400
C                                                                       00149500
C                                                                       00149600
           SUBROUTINE DECIDE                                            00149700
C                                                                       00149800
           DIMENSION LTAB(3,3)                                          00149900
           INTEGER SLVL,BLVL,RIGHT,RLOC,NLOC,SLOC,BNDLOC,BROW,BLOC,     00150000
     $          BNDTYP                                                 00150100
           COMMON /POINT/SLVL(20),BLVL(20),LEFT(400),RIGHT(400),        00150200
     $          LLOC(400),RLOC(400),NLOC(400),SLOC(400),BNDLOC(1000),  00150300
     $          BNDTYP(1000),BROW(400),BLOC(1000)                      00150400
           COMMON /DPOINT/JSPIR,KSPIR,NROW,MAXROW,NSEG,KSEG(3),         00150500
     $          JBOTL(3),JBOTR(3),JTOPL(3),KBOTL(3),KTOPL(3)           00150600
           DATA LTAB/1,3,1,1,3,1,2,4,2/                                 00150700
C                                                                       00150800
           JBL=LEFT(JSPIR)                                              00150900
           JBR=RIGHT(JSPIR)                                             00151000
C                                                                       00151100
C                    CHECK FOR THE TOP ROW OF THE COARSE GRID           00151200
C                                                                       00151300
           IF(NROW.NE.MAXROW)GO TO 80                                   00151400
C                                                                       00151500
                JTL=-1                                                  00151600
                JTR=-1                                                  00151700
                GO TO 500                                               00151800
C                                                                       00151900
C                                                                       00152000
C                                                                       00152100
  80       JTL=LEFT(JSPIR+1)                                            00152200
           JTR=RIGHT(JSPIR+1)                                           00152300
C                                                                       00152400
           LROW=1                                                       00152500
           IF(JBL.LT.JTL)LROW=2                                         00152600
           IF(JBL.GT.JTL)LROW=3                                         00152700
C                                                                       00152800
           LCOL=1                                                       00152900
           IF(JBR.LT.JTR)LCOL=2                                         00153000
           IF(JBR.GT.JTR)LCOL=3                                         00153100
C                                                                       00153200
           N=LTAB(LROW,LCOL)                                            00153300
C                                                                       00153400
C                                                                       00153500
           GO TO (100,200,300,400),N                                    00153600
C                                                                       00153700
C                    *------*    OR    *------*------*                 00153800
C                    *------*          *------*                        00153900
C                                                                       00154000
C                    *------*------*------*    OR    *------*------*   00154100
C                           *------*                       *------*     00154200
C                                                                       00154300
 100           NSEG=1                                                   00154400
               KSEG(1)=1                                                00154500
               JBOTL(1)=LLOC(JSPIR)                                     00154600
               JBOTR(1)=RLOC(JSPIR)                                     00154700
               KBOTL(1)=LLOC(KSPIR)                                     00154800
               JTOPL(1)=NLOC(JSPIR)                                     00154900
               KTOPL(1)=NLOC(KSPIR)                                     00155000
               GO TO 600                                                00155100
C                                                                       00155200
C                    *------*------*       OR   *------*               00155300
```

```
C                    *------*------*      *------*------*
C
 200          NSEG=2
              KSLG(1)=1
              JBOTL(1)=LLOC(JSPIR)
              JBOTR(1)=RLOC(JSPIR)-(JBR-JTR)
              KBOTL(1)=LLOC(KSPIR)
              JTOPL(1)=NLOC(JSPIR)
              KTOPL(1)=NLOC(KSPIR)
C
              KSLG(2) 0
              JBOTL(2)=JBOTR(1)
              JBOTR(2)=JBR
              KBOTL(2)=RLOC(KSPIR)-(JBR-JTR)*2
              KTOPL(2)=-1
              JTOPL(2)=-1
              GO TO 600
C
C
C                    *------*  OR          *------*------*
C                *------*------*          *------*------*
C
 300          NSEG=2
              KSLG(1)=0
              JBOTL(1)=LLOC(JSPIR)
              JBOTR(1)=SLOC(JSPIR+1)
              KBOTL(1)=LLOC(KSPIR)
              JTOPL(1)=-1
              KTOPL(1)=-1
C
              KSLG(2)=1
              JBOTL(2)=JBOTR(1)
              JBOTR(2)=RLOC(JSPIR)
              JTOPL(2)=LLOC(JSPIR+1)
              KBOTL(2)=SLOC(KSPIR+1)
              KTOPL(2)=LLOC(KSPIR+1)
              GO TO 600
C
C
C                    *------*
C                *------*------*------*
C
 400          NSLG=3
              KSLG(1) 0
              JBOTL(1)=LLOC(JSPIR)
              JBOTR(1)=SLOC(JSPIR+1)
              KBOTL(1)=LLOC(KSPIR)
              JTOPL(1)=-1
              KTOPL(1)=-1
              KSLG(2)=1
              JBOTL(2)=JBOTR(1)
              JBOTR(2)=RLOC(JSPIR)-(JBR-JTR)
              JTOPL(2)=LLOC(JSPIR+1)
              KBOTL(2)=SLOC(KSPIR+1)
              KTOPL(2)=LLOC(KSPIR+1)
C
              KSLG(3)=0
              JBOTL(3)=JBOTR(2)
              JBOTR(3)=RLOC(JSPIR)
              KBOTL(3)=RLOC(KSPIR)-(JBR-JTR)*2
              JTOPL(3)=-1
              KTOPL(3) -1
              GO TO 600
C
C
```

```
C
C
C
C                    *------*
C
 500          NSEG=1
              KSEG(1)=0
              JBOTL(1)=LLOC(JSPIR)
              JBOTR(1)=RLOC(JSPIR)
              KBOTL(1)=LLOC(KSPIR)
              JTOPL(1)=-1
              KTOPL(1)=-1
              GO TO 600
C
C
C
 600          RETURN
              END
C
C
C
C
C
              SUBROUTINE RESULT(LEVEL,JPRNT,JLEN,JPLOT,JSOL,JRHS)
C
C             RESULT -- PRINT ALL OR PART OF THE SOLUTION VECTOR
C
C                  LEVEL -- GRID TO BE USED
C                  JPRNT --
C                       0 - DO NOT PRINT EITHER THE SOLUTION VECTOR OR
C                           THE RIGHT-HAND SIDE
C                       1 - ENABLE PRINTING OF THE SOLUTION VECTOR OR
C                           THE RIGHT-HAND SIDE
C                  JLEN --
C                       0 - PRINT THE NUMBER OF ROWS GIVEN BY NPRINT
C                       1 - PRINT ALL ROWS OF THE GRID
C                  JPLOT --
C                       0 - DO NOT PLOT EITHER THE SOLUTION VECTOR OR
C                           THE RIGHT-HAND SIDE
C                       1 - ENABLE PRINTING OF THE SOLUTION VECTOR OR
C                           THE RIGHT-HAND SIDE
C                  JSOL --
C                       0 - DO NOT PRINT OR PLOT THE SOLUTION VECTOR
C                       1 - PRINT OR PLOT THE SOLUTION VECTOR IF
C                           ENABLED
C                  JRHS --
C                       0 - DO NOT PRINT OR PLOT THE RIGHT-HAND SIDE
C                       1 - PRINT THE RIGHT-HAND SIDE AND GIVE A PLOT
C                           IF PLOTS ARE ENABLED
C
              INTEGER SLVL,BLVL,RIGHT,RLOC,NLOC,SLOC,BNDLOC,BROW,BLOC,
     $             BNDTYP
              COMMON /TRACE/IN,LP,NDBUG,NPRINT,NINI
              COMMON /POINT/SLVL(20),BLVL(20),LEFT(400),RIGHT(400),
     $             LLOC(400),RLOC(400),NLOC(400),SLOC(400),BNDLOC(1000),
     $             BNDTYP(1000),BROW(400),BLOC(1000)
              COMMON /SIZE/NROW(20),NCOL(20),NPOINT(20),NVAL(20),XMIN,XMAX,
     *             YMIN,YMAX
              COMMON /STORE/U(5000),I(5000)
              DATA JSIZE/5000/
C
C
              WRITE(LP,10)LEVEL
C
C
```

```
      IF(JPRNT.EQ.0)GO TO 240                                           00168400
C                                                                       00168500
C                                                                       00168600
            KMAX=NROW(LEVEL)                                            00168700
            IF(JLEN.EQ.0)KMAX=AMINO(KMAX,NPRINT)                        00168800
C                                                                       00168900
            KMIN=SLVL(LEVEL)                                            00169000
            KMAX=KMAX+KMIN-1                                            00169100
            JROW=0                                                      00169200
            H=HVAL(LEVEL)                                               00169300
C                                                                       00169400
            DO 200 KPT=KMIN,KMAX                                        00169500
C                                                                       00169600
                  Y=YMIN+H*FLOAT(JROW)                                  00169700
C                                                                       00169800
                  WRITE(LP,20)JROW,LEFT(KPT),RIGHT(KPT),Y               00169900
C                                                                       00170000
                  JMIN=LLOC(KPT)                                        00170100
                  JMAX=RLOC(KPT)                                        00170200
C                                                                       00170300
                  IF(JSOL.NE.0)WRITE(LP,30)(U(J),J=JMIN,JMAX)           00170400
                  IF(JRHS.NE.0)WRITE(LP,40)(F(J),J=JMIN,JMAX)           00170500
                  JROW=JROW+1                                           00170600
C                                                                       00170700
  200       CONTINUE                                                    00170800
C                                                                       00170900
C                                                                       00171000
  240    CONTINUE                                                       00171100
C                                                                       00171200
C                                                                       00171300
C                                                                       00171400
      IF(JPLOT.EQ.0)GO TO 300                                           00171500
C                                                                       00171600
            IF(JSOL.EQ.0)GO TO 280                                      00171700
C                                                                       00171800
C                                                                       00171900
                  WRITE(LP,50)                                          00172000
                  CALL PLOT(LEVEL,U,JSIZE)                              00172100
C                                                                       00172200
  280       IF(JRHS.EQ.0) GO TO 300                                     00172300
C                                                                       00172400
                  WRITE(LP,60)                                          00172500
                  CALL PLOT(LEVEL,F,JSIZE)                              00172600
C                                                                       00172700
C                                                                       00172800
C                                                                       00172900
  300   RETURN                                                          00173000
C                                                                       00173100
C                                                                       00173200
   10   FORMAT(////6X,'--- RESULT --- (LEVEL',I5,' )')                  00173300
C                                                                       00173400
   20   FORMAT(/11X,'ROW',I5,5X,'LEFT COL',I5,5X,'RIGHT COL',I5,        00173500
     *       5X,'Y=',E15.7)                                             00173600
   30   FORMAT(/16X,'SOLUTION VECTOR'/(11X,6E15.7))                     00173700
   40   FORMAT(/16X,'RIGHT-HAND SIDE'/(11X,6E15.7))                     00173800
   50   FORMAT(1HI,5X,'SOLUTION VECTOR')                                00173900
   60   FORMAT(1HI,5X,'RIGHT-HAND SIDE')                                00174000
C                                                                       00174100
      END                                                               00174200
C                                                                       00174300
C                                                                       00174400
C                                                                       00174500
C                                                                       00174600
C                                                                       00174700
      SUBROUTINE PLOT(LEVEL,Z,JSIZE)                                    00174800
C                                                                       00174900
C                                                                       00175000
C           PLOT -- GIVE A CONTOUR PLOT OF THE SOLUTION VECTOR OF       00175100
C                GRID LEVEL                                             00175200
C                                                                       00175300
      INTEGER SLVL,BLVL,RIGHT,RLOC,NLOC,SLOC,BNDLOC,BROW,BLOC,          00175400
     $        BNDTYP                                                    00175500
      COMMON /TRACE/IN,LP,NDBUG,NPRINT,NINI                             00175600
      COMMON /POINT/SLVL(20),BLVL(20),LEFT(400),RIGHT(400),             00175700
     $        LLOC(400),RLOC(400),NLOC(400),SLOC(400),BNDLOC(1000),     00175800
     $        BNDTYP(1000),BROW(400),BLOC(1000)                         00175900
      COMMON /SIZE/NROW(20),NCOL(20),NPOINT(20),HVAL(20),XMIN,XMAX,     00176000
     *        YMIN,YMAX                                                 00176100
      DIMENSION Z(JSIZE)                                                00176200
      INTEGER LINE(120),CHAR(20),BLANK                                  00176300
      DATA CHAR/'A','B','C','D','E','F','G','H','J','K','N','O',       00176400
     $     'P','R','S','T','U','W','X','Z'/                             00176500
      DATA BLANK/' '/,MAXINT/20/,ZLIMB/1.0E-20/,ZLIMT/1.0E+20/          00176600
C                                                                       00176700
      JMIN=SLVL(LEVEL)                                                  00176800
      JMAX=SLVL(LEVEL+1)-1                                              00176900
C                                                                       00177000
      KMIN=LLOC(JMIN)                                                   00177100
      KMAX=RLOC(JMAX)                                                   00177200
C                                                                       00177300
      ZMAX=Z(KMIN)                                                      00177400
      ZMIN=ZMAX                                                         00177500
C                                                                       00177600
      DO 200 K=KMIN,KMAX                                                00177700
C                                                                       00177800
            ZVAL=Z(K)                                                   00177900
            IF(ZMIN.GT.ZVAL)ZMIN=ZVAL                                   00178000
            IF(ZMAX.LT.ZVAL)ZMAX=ZVAL                                   00178100
C                                                                       00178200
  200       CONTINUE                                                    00178300
C                                                                       00178400
C                                                                       00178500
      ZDIF=ZMAX-ZMIN                                                    00178600
      NINI=AMINO(NINI,MAXINT)                                           00178700
      ZFACT=0.0                                                         00178800
      IF(ZDIF.LT.ZLIMT .AND. ZDIF.GT.ZLIMB)ZFACT=FLOAT(NINI)/ZDIF       00178900
C                                                                       00179000
      LEN=AMAXO(NROW(LEVEL),NCOL(LEVEL))                                00179100
      JPT=JMAX                                                          00179200
      LMAX=NROW(LEVEL)                                                  00179300
C                                                                       00179400
      WRITE(LP,10)LEVEL                                                 00179500
C                                                                       00179600
      DO 400 L=1,LMAX                                                   00179700
C                                                                       00179800
            NBLANK=LEFT(JPT)+1                                          00179900
            NCHAR=RIGHT(JPT)-LEFT(JPT)+1                                00180000
            KMIN=LLOC(JPT)                                              00180100
            KMAX=RLOC(JPT)                                              00180200
            NPT=1                                                       00180300
C                                                                       00180400
            DO 300 KPT=KMIN,KMAX                                        00180500
C                                                                       00180600
                  NUM=AINT((Z(KPT)-ZMIN)*ZFACT)+1                       00180700
                  IF(NUM.GT.NINI)NUM=NINI                               00180800
                  LINE(NPT)=CHAR(NUM)                                   00180900
                  NPT=NPT+1                                             00181000
  300       CONTINUE                                                    00181100
C                                                                       00181200
            IF(LEN.LT.33)WRITE(LP,20)(BLANK,N=1,NBLANK),                00181300
     $           (LINE(N),N=1,NCHAR)
```

```
C                                                                       00181400
            IF(LEN.GE.33)WRITE(LP,30)(BLANK,N=1,NBLANK),                00181500
     $           (LINE(N),N=1,NCHAR)                                    00181600
C                                                                       00181700
C                                                                       00181800
            JPT=JPT-1                                                   00181900
C                                                                       00182000
  400   CONTINUE                                                        00182100
C                                                                       00182200
C           PRINT THE RANGES OF THE CONTOURS                            00182300
C                                                                       00182400
        WRITE(LP,40)ZMIN,ZMAX                                           00182500
  407   ZLEN=ZDIF/FLOAT(KINT)                                           00182700
C                                                                       00182800
        Z0=ZMIN                                                         00182900
        DO 600 NUM=1,KINT                                               00183000
C                                                                       00183100
            Z1=FLOAT(NUM)*ZLEN+ZMIN                                     00183200
            WRITE(LP,50)CHAR(NUM),Z0,Z1                                 00183300
            Z0=Z1                                                       00183400
C                                                                       00183500
  600   CONTINUE                                                        00183600
C                                                                       00183700
C                                                                       00183800
C                                                                       00183900
        RETURN                                                          00184000
C                                                                       00184100
   10   FORMAT(//6X,'-- PLOT -- (LEVEL',I5,')')                        00184200
   20   FORMAT(/1X,60A2)                                               00184300
   30   FORMAT(1X,120A1)                                               00184400
   40   FORMAT(//16X,'CONTOUR RANGES'//21X,'ZMIN,ZMAX',2E20.7)         00184500
   50   FORMAT(16X,A1,5X,E20.7,' --',E20.7)                            00184600
C                                                                       00184700
        END                                                             00184800
```

VITA

David Paul Williams

Candidate for the Degree of

Master of Science

Thesis: SOLUTION METHODS FOR RE-ENTRANT CORNERS IN ELLIPTIC PARTIAL DIFFERENTIAL EQUATIONS

Major Field: Computing and Information Sciences

Biographical:

Personal Data: Born in Durant, Oklahoma, February 4, 1956, the son of Mr. and Mrs. David L. Williams.

Education: Graduated from Durant High School, Durant, Oklahoma, in May, 1974; received the Bachelor of Science degree in Physics and Mathematics from Southeastern Oklahoma State University in May, 1978; attended The University of Texas at Austin, August, 1978 through May, 1979; completed requirements for the Master of Science degree at Oklahoma State University in May, 1983.

Professional Experience: Graduate teaching assistant, Department of Physics, The University of Texas at Austin, 1978-1979; Graduate teaching assistant, Department of Computing and Information Sciences, Oklahoma State University, 1979-1981; Research assistant, Department of Electrical Engineering, 1980-1981; member of the Association for Computing Machinery.