CONTINUOUS, LOW THRUST COPLANAR ORBIT TRANSFERS WITH VARYING ECCENTRICITY

THESIS

Gregory L. Beeker
Captain, USAF

AFIT/GA/AA/88D-01

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GA/AA/88D-01

CONTINUOUS, LOW THRUST COPLANAR ORBIT TRANSFERS WITH VARYING ECCENTRICITY

THESIS

Gregory L. Beeker
Captain, USAF

AFIT/GA/AA/88D-01

AFIT/GA/AA/88D-01

CONTINUOUS, LOW THRUST COPLANAR ORBIT TRANSFERS WITH VARYING ECCENTRICITY

THESIS

Presented to the Faculty of the School of Engineering

of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the

Requirements for the Degree of

Master of Science in Astronautical Engineering

Gregory L. Beeker, B.S.

Captain, USAF

December 1988

Preface

The purpose of this study was to explore the possible application of a thrust vector control law constraining either the radius of apogee or perigee to transfers between circular and eccentric orbits for continuous, low thrust spacecraft. I have tried to include as many details as possible in the analysis to aid anyone who may want to perform similar or follow-on research.

To prove the validity of the assumptions made in this study, I thought it necessary to perform a separate performance analysis of the more promising electrical propulsion systems (Appendix A). I am truly grateful to Mr Mike Patterson (NASA Lewis), Capt Wayne Schmidt (AFAL), and fellow Boilermaker, Mr Joe Cassady (Rocket Research Company) for providing me with the data I needed to put together a reasonably valid performance estimate of an Ammonia arcjet and Xenon ion propulsion system.

My sincere thanks goes to Dr William Wiesel, my thesis advisor, for his assistance and patience over the past several months. Also, I thank my wife, Mary, and my children, Bethany, Dustin, and Brandon, for their understanding and support during those many hours that I spent away from them over the past year.

I dedicate this thesis in memory of my grandparents, the late Ora and Rita Beeker.

TABLE OF CONTENTS

List of Figures

Figure		Page

List of Tables

Notation

Acronyms

GEO	geosynchonous orbit
IJK	Geocentric-Equatorial Coordinate System (earth centered)
LEO	low earth orbit
NASA	National Aeronautics and Space Administration
N_2H_4	Ammonia
PCU	Power Conditioning Unit
PPU	Power Processing Unit
PQW	Perifocal Coordinate System (earth centered)
RSW	Spacecraft Centered Coordinate System
SAFE	Solar Array Flight Experiment
TVS	Transfer Vehicle System
X_e	Xenon

Symbols

α	planar thrust control angle
α_i	planar control law for thrust vector
λ_i	Lagrange multiplier
$\mu_\oplus$	gravitational parameter
ν	true anomaly
θ	out-of-plane thrust control angle
A	acceleration
$\mathbf{A}$	acceleration Vector

Symbols (continued)

a_r	radial acceleration component
a_s	tangential acceleration component
a_v	normal acceleration component
g_c	gravitational constant (9.81 m/s^2)
h	specific angular momentum
I_{SP}	specific Impulse (sec)
J_i	Performance Index
m	mass (kg)
$\dot{m}$	mass flow rate (kg/sec)
$\dot{M}$	specific mass flow rate (1/sec)
n	mean anomaly
p	semi-latus rectum
r	distance between spacecraft center of mass and primary body (earth center)
r_a	radius of perigee
r_p	radius of apogee
$\mathbb{TP}$	orbital period
t	time
T	thrust vector
v	velocity (m/s)

Orbital Elements

a	semimajor axis
e	eccentricity
i	inclination
Ω	longitude of the ascending node
ω	argument of periapsis
M_o	mean anomaly at epoch

AFIT/GA/AA/88D-01

Abstract

The purpose of this study was to investigate the Δv requirements of a continuous, low thrust spacecraft performing coplanar orbit transfers that are constrained by either constant radius of apogee, perigee, or a combination of the two. The transfers were separated into two timescale problems. The fast timescale involved an optimization of the planar thrust control angle, α, to produce the maximum change in eccentricity or semimajor axis over a single revolution. The slow timescale applied the fast timescale results to complete the transfer through many revolutions about the primary body. The constrained radii control laws developed provide optimal circular-to-eccentric and eccentric-to-eccentric orbit transfers. However, when applied to circular-to-circular transfers, the resulting Δv is nearly twice that obtained using the most optimal continuous thrust control law ($\alpha = 0$), i.e. "spiral". Future recommended studies include the development of control laws to provide specified changes in apogee, perigee, and the argument of periapsis.

CONTINUOUS, LOW THRUST COPLANAR ORBIT TRANSFERS WITH VARYING ECCENTRICITY

I. Introduction

The United States' decision to construct a space station within the next decade has launched this country into a new era in which it will be accessing space more than ever before. With this new era comes an increasing need for an energy efficient orbit transfer vehicle to act as a freighter between low and high earth orbit - a vehicle vital in reducing the work loads of supporting expendable boosters and the Space Transportation System.

A transfer vehicle equipped with a continuous, low thrust, electric propulsion system, such as the ion systems currently in development at NASA Lewis Research Center or the arcjet systems at the Air Force Astronautics Laboratory, could provide for this need, greatly reducing fuel and support requirements (4; 6; 10:457-467). However, low thrust systems of this type require many revolutions to complete their transfer and are therefore limited to missions where extended transfer times (months) are acceptable. In addition, the electronics of the transfer vehicle and payload must be able to endure the extended exposure to the high radiation within the lower Van Allen belt.

Alfano (1) addressed the problem of locating an optimal thrust profile for noncoplanar transfers performed by a continuous, low thrust vehicle between circular orbits. However, the more complex problem of locating an optimal thrust profile for transfers between eccentric orbits has never been addressed, although there are many useful applications, such as, the placement of communications satellites into Molniya orbits.

To provide an initial investigation into the eccentric transfer problem, this research examines coplanar, eccentric transfers constrained by holding the radius of apogee or perigee *constant over each orbit*. The development of this problem is simplified by dividing the derivation between a fast and slow timescale, similar to that done by Alfano. The fast timescale problem involves an optimization of the change in the orbital elements over one revolution while implementing the constraint relation and holding the vehicle mass and acceleration constant. The slow timescale problem combines the fast timescale results with the changing mass and acceleration to complete the transfer over many revolutions.

The development of the fast timescale solution in chapter two shows this constraint relationship is unique since it provides two control laws for the planar thrust angle, α, (one for maximizing Δa and the other for maximizing Δe) which result in identical changes to the orbit's semimajor axis and eccentricity. Thus, either of these two control laws are applicable to any coplanar transfer, whether the transfer involves a change in eccentricity, the semimajor axis, or both. However, these results (and the slow timescale problem of chapter three) indicate that utilizing these control laws for transfers between coplanar circular orbits is much less efficient than the spiral ($\alpha = 0$) control law.

II. The Fast Timescale Problem

Problem Statement

The fast timescale analysis performed by Alfano (1:3-23) included an optimization of the out-of-plane thrust angle, θ, for each of the two orbital parameters involved, the change in semimajor axis, Δa, and the change in inclination, Δi, maximizing one parameter for any given value of the other. Optimization problems involving transfers between eccentric orbits become more complex due to the addition of the planar thrust angle, α, eccentricity, e, and argument of periapsis, ω. Thus, the fast timescale analysis includes an optimization of both α and θ for each of the four parameters involved, Δa, Δi, Δe, and $\Delta \omega$.

This problem can be simplified by considering only coplanar transfers and by placing no restrictions on $\Delta \omega$. This results in an optimization of α only (since $\theta = 0$) for the parameters Δa or Δe ($\Delta i = 0$). However, the application of the resulting control law would produce uncontrollable changes in the radius of perigee, possibly allowing an impact with the primary body. Therefore, the problem constraints should include the radius of perigee instead of the eccentricity or semimajor axis.

The coplanar transfer problem addressed by this study is constrained by holding the radius of apogee or perigee constant. Thus, the optimization is further simplified since the constraint relationship is no longer dependent on the total transfer and allows the Lagrange multiplier, λ, to be found in the fast, rather that the slow timescale problem. Given these constraints, the fast timescale problem involves an optimization of α in order to maximize the magnitude of Δa or Δe for one orbit about the primary body (Earth).

The formulation of the fast timescale problem is based on the following three assumptions:

First, the problem is restricted to two bodies, the earth and the spacecraft, which are modeled as point masses.

Second, since electric propulsion systems have very low propellant mass flow rates, the percent change in mass of the spacecraft over one orbit is small. Thus, the spacecraft's mass will be considered constant. Appendix A provides data in support of this assumption, showing that a 5000 kg transfer vehicle system (vehicle and payload) has a change in mass over one orbital period of less than 3% if equipped with the proposed arcjet propulsion system and less than .25% if equipped with the ion system.

Finally, the low thrust associated with electric propulsion systems produces such small accelerations on the spacecraft that the changes in the orbital elements occur very slowly. This leads to the assumption that these these parameters vary so little during the period of one orbit that they can be treated as constants. Appendix A also provides justification by showing the maximum acceleration as seen by the proposed TVS is approximately 4.5×10^{-3} m/s^2 using the arcjet system and 1.3×10^{-3} m/s^2 using the ion system.

Derivation

Coordinate System Definition. The coordinate system utilized throughout this analysis (2:397,398) is located at the center of mass of the transfer vehicle, with its principle axis, R (unit vector $\hat{R}$), located along the instantaneous radial vector, **r**. The second axis, S (unit vector $\hat{S}$), is located in the orbital plane perpendicular to R in the direction of increasing true anomaly, ν. The third axis, W (unit vector $\hat{W}$), is perpendicular to both R and S, forming an orthogonal triad where $\hat{W} \equiv \hat{R} \times \hat{S}$. Thus, this coordinate system is rotated by an angle ν with respect to the perifocal coordinate system. Figure 2-1 defines the geometric relationships among the spacecraft (RSW), perifocal (PQW), and geocentric-equatorial (IJK) frames (2:53-59).

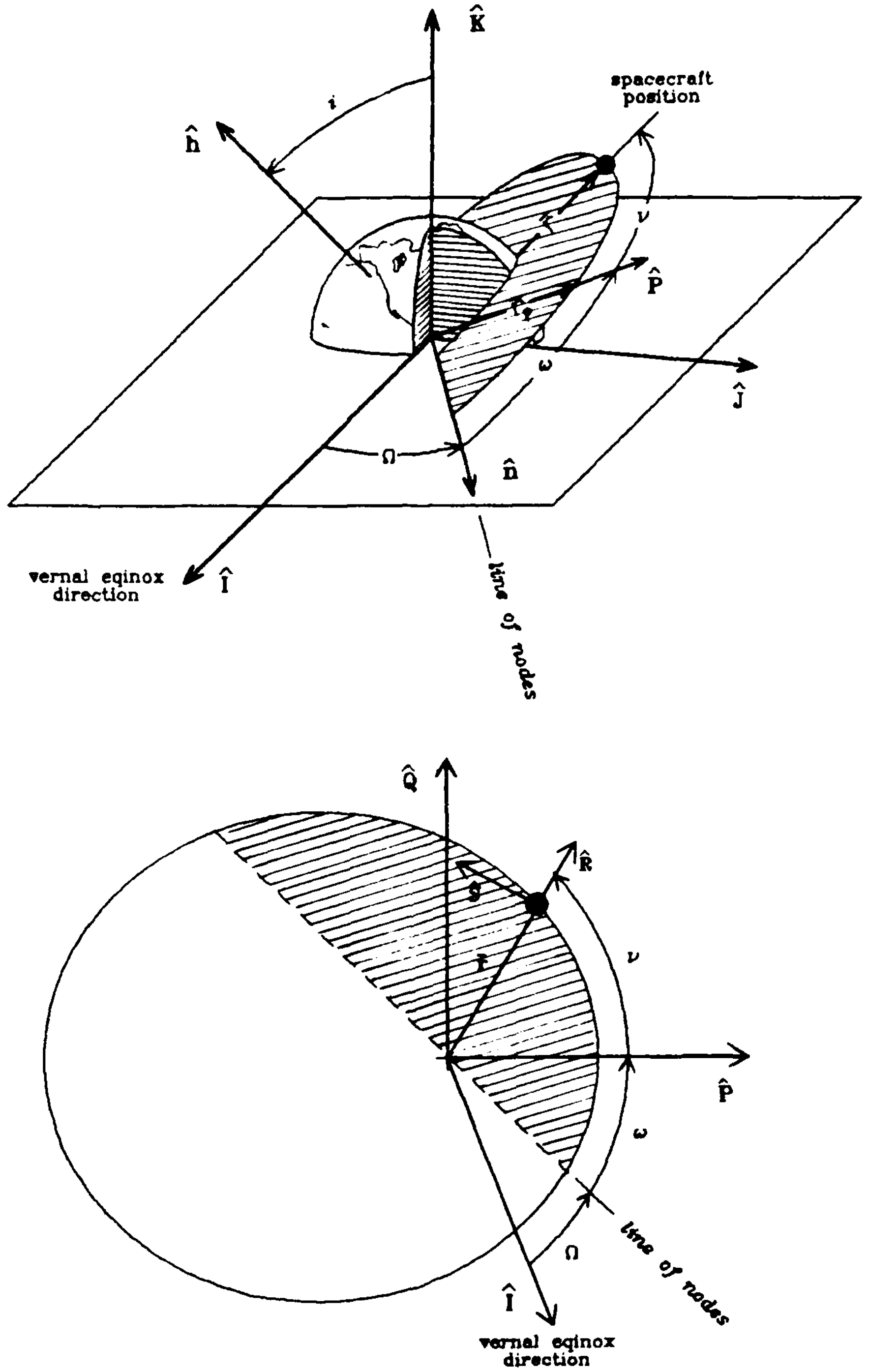


Figure 2-1. Relationships among the Spacecraft, Perifocal, and Geocentric-Equatorial Coordinate Frames

<u>Vehicle Thrust Vector.</u> The thrust vector, **T**, as defined in the RSW coordinate system, is directed at an angle θ with respect to the orbital plane formed by the radial unit vector, $\hat{R}$, and the tangential unit vector, $\hat{S}$ (Figure 2-2). The projection of T on the orbital plane is located at an angle α with respect to $\hat{S}$. The acceleration vector lies along the thrust vector and is defined as

$$\mathbf{A} = \frac{\mathbf{T}}{m_{TVS}} = A\,\hat{T} \equiv a_r\,\hat{R} + a_s\,\hat{S} + a_w\,\hat{W} \qquad (2\text{-}1)$$

where

$$a_r \equiv A \cos\theta \sin\alpha \qquad (2\text{-}2)$$

$$a_s \equiv A \cos\theta \cos\alpha \qquad (2\text{-}3)$$

$$a_w \equiv A \sin\theta \qquad (2\text{-}4)$$

$$0 \le \alpha \le 2\pi; \quad \frac{\pi}{2} \le \theta \le \frac{-\pi}{2}$$

and

$m \equiv$ mass of transfer vehicle system

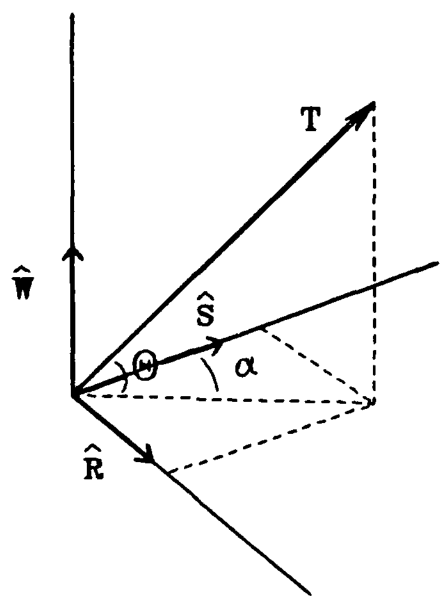


Figure 2-2. Thrust Vector Location Relative to the Orbital Frame

Perturbation Equations. In the defined coordinate system, the rate of change in the orbital elements can be found utilizing the Lagrange planetary equations in their acceleration component form (2:397-407):

$$\frac{da}{dt} = \frac{2e \sin \nu}{n \sqrt{1-e^2}} \alpha_r + \frac{2a \sqrt{1-e^2}}{nr} \alpha_s \tag{2-5}$$

$$\frac{de}{dt} = \frac{\sqrt{1-e^2} \sin \nu}{na} \alpha_r + \frac{\sqrt{1-e^2}}{na^2 e} \left[\frac{a^2\left(1-e^2\right)}{r} - r \right] a_s \tag{2-6}$$

$$\frac{di}{dt} = \frac{r \cos (\omega+\nu)}{na^2 \sqrt{1-e^2}} \alpha_w \tag{2-7}$$

$$\frac{d\Omega}{dt} = \frac{r \sin (\omega+\nu)}{na^2 \sqrt{1-e^2} \sin i} \alpha_w \tag{2-8}$$

$$\frac{d\omega}{dt} = - \frac{\sqrt{1-e^2} \cos \nu}{nae} \alpha_r + \frac{\sqrt{1-e^2} \sin \nu}{nae} \left(1 + \frac{1}{1 + e \cos \nu} \right) \alpha_s - \frac{r \cot i \sin (\omega+\nu)}{na^2\sqrt{1-e^2}} \alpha_w \tag{2-9}$$

$$\frac{dM_o}{dt} = -\frac{1}{na}\left[\frac{2r}{a} - \frac{1-e^2}{e}\cos\nu\right] a_r$$

$$- \frac{(1-e^2)\sin\nu}{nae}\left[1 + \frac{r}{a(1-e^2)}\right] a_s$$

$$+ \frac{3nt}{2a}\frac{da}{dt} \qquad (2\text{-}10)$$

Where:

$a_r \equiv$ *radial component of acceleration*

$a_s \equiv$ *tangential component of acceleration*

$a_w \equiv$ *normal component of acceleration*

Using the relation for the distance from the primary body (2:20,24)

$$r = \frac{p}{1 + e\cos\nu} = \frac{a(1-e^2)}{1 + e\cos\nu} \qquad (2\text{-}11)$$

the Lagrange equations can be expressed in terms of the orbital elements and the true anomaly only. Thus, substituting this relation, equations (2-5) thru (2-10) become

$$\frac{da}{dt} = \frac{2e\sin\nu}{n\sqrt{1-e^2}} a_r + \frac{2(1 + e\cos\nu)}{n\sqrt{1-e^2}} a_s \qquad (2\text{-}12)$$

$$\frac{de}{dt} = \frac{\sqrt{1-e^2}\sin\nu}{na} a_r$$

$$+ \frac{\sqrt{1-e^2}}{nae}\left[1 + e\cos\nu - \frac{(1-e^2)}{1 + e\cos\nu}\right] a_s \qquad (2\text{-}13)$$

$$\frac{di}{dt} = \frac{\sqrt{1-e^2}\cos(\omega+\nu)}{na(1 + e\cos\nu)} a_w \qquad (2\text{-}14)$$

$$\frac{d\Omega}{dt} = \frac{\sqrt{1-e^2}\ \sin\ (\omega+\nu)}{na\ (1\ +\ e\ \cos\ \nu)\ \sin\ i}\ a_v \tag{2-15}$$

$$\begin{aligned}\frac{d\omega}{dt} = &- \frac{\sqrt{1-e^2}\ \cos\ \nu}{nae}\ a_r \\ &+ \frac{\sqrt{1-e^2}\ \sin\ \nu\ (2\ +\ e\ \cos\ \nu)}{nae\ (1\ +\ e\ \cos\ \nu)}\ a_s \\ &- \frac{\sqrt{1-e^2}\ \cot\ i\ \sin\ (\omega+\nu)}{na\ (1\ +\ e\ \cos\ \nu)}\ a_v\end{aligned} \tag{2-16}$$

$$\begin{aligned}\frac{dMo}{dt} = &- \frac{1-e^2}{na}\left[\frac{2}{1\ +\ e\ \cos\ \nu} - \frac{\cos\ \nu}{e}\right] a_r \\ &- \frac{\left(1-e^2\right)\ \sin\ \nu}{nae\ (1\ +\ e\ \cos\ \nu)}\ (2\ +\ e\ \cos\ \nu)\ a_s \\ &+ \frac{3nt}{2a}\ \frac{da}{dt}\end{aligned} \tag{2-17}$$

The independent variable can be changed from time, t, to the true anomaly, ν, using the following relation for the angular momentum (2:17,28)

$$\begin{aligned} h &= |\mathbf{r} \times \mathbf{v}| = r^2 \frac{d\nu}{dt} \\ h &= na^2\sqrt{1-e^2}\end{aligned} \tag{2-18}$$

Solving for the time rate of change in the true anomaly

$$\begin{aligned}\frac{d\nu}{dt} &= \frac{na^2\sqrt{1-e^2}}{r^2} = \frac{na^2\sqrt{1-e^2}}{a^2\left(1-e^2\right)^2}\ (1\ +\ e\ \cos\ \nu)^2 \\ &= \frac{n\ (1\ +\ e\ \cos\ \nu)^2}{\left(1-e^2\right)^{3/2}}\end{aligned} \tag{2-19}$$

Multiplying equations (2-12) thru (2-19) by $\left(\frac{d\nu}{dt}\right)^{-1}$ results in the final form of the perturbation equations which define the changes in the orbital elements with respect to ν

$$\frac{da}{d\nu} = \frac{2e \sin \nu \left(1-e^2\right)}{n^2(1 + e \cos \nu)^2} \alpha_r + \frac{2\left(1-e^2\right)}{n^2(1 + e \cos \nu)} \alpha_s \qquad (2\text{-}20)$$

$$\frac{de}{d\nu} = \frac{\left(1-e^2\right)^2 \sin \nu}{n^2 a \ (1 + e \cos \nu)^2} \alpha_r + \frac{\left(1-e^2\right)^2}{n^2 ae \ (1 + e \cos \nu)} \left[1 - \frac{1-e^2}{(1 + e \cos \nu)^2}\right] \alpha_s \qquad (2\text{-}21)$$

$$\frac{di}{d\nu} = \frac{\left(1-e^2\right)^2 \cos \ (\omega+\nu)}{n^2 a \ (1 + e \cos \nu)^3} \alpha_w \qquad (2\text{-}22)$$

$$\frac{d\Omega}{d\nu} = \frac{\left(1-e^2\right)^2 \ \sin \ (\omega+\nu)}{n^2 a \ (1 + e \cos \nu)^3 \ \sin i} \alpha_w \qquad (2\text{-}23)$$

$$\frac{d\omega}{d\nu} = -\frac{\left(1-e^2\right)^2 \cos \nu}{n^2 ae \ (1 + e \cos \nu)^2} \alpha_r + \frac{\left(1-e^2\right)^2 \sin \nu \ (2 + e \cos \nu)}{n^2 ae \ (1 + e \cos \nu)^3} \alpha_s - \frac{\left(1-e^2\right)^2 \cot i \ \sin \ (\omega+\nu)}{n^2 a \ (1 + e \cos \nu)^3} \alpha_w \qquad (2\text{-}24)$$

$$\frac{dM_o}{d\nu} = -\frac{\left(1-e^2\right)^{5/2}}{n^2 a\ (1 + e\cos\nu)^2}\left[\frac{2}{1 + e\cos\nu} - \frac{\cos\nu}{e}\right]\alpha_r$$

$$-\frac{\left(1-e^2\right)^{5/2}\sin\nu}{n^2 ae\ (1 + e\cos\nu)^3}(2 + e\cos\nu)\ \alpha_s$$

$$+\frac{3nt}{2a}\frac{da}{d\nu} \tag{2-25}$$

Integrating equations (2-20) to (2-25) with respect to ν over one orbit (from 0 to 2π) will produce the changes in the orbital elements over one revolution of the transfer orbit. Thus, in integral form, these changes are

$$\Delta a = \int_0^{2\pi}\left[\frac{2e\sin\nu\left(1-e^2\right)}{n^2(1 + e\cos\nu)^2} A\cos\theta\sin\alpha + \frac{2\left(1-e^2\right)}{n^2(1 + e\cos\nu)} A\cos\theta\cos\alpha\right]d\nu \tag{2-26}$$

$$\Delta e = \int_0^{2\pi}\left[\frac{\left(1-e^2\right)^2\sin\nu}{n^2 a\ (1 + e\cos\nu)^2} A\cos\theta\sin\alpha + \left(1 - \frac{1-e^2}{(1 + e\cos\nu)^2}\right)\frac{\left(1-e^2\right)^2 A\cos\theta\cos\alpha}{n^2 ae\ (1 + e\cos\nu)}\right]d\nu \tag{2-27}$$

$$\Delta i = \int_0^{2\pi}\frac{\left(1-e^2\right)^2\cos(\omega+\nu)}{n^2 a\ (1 + e\cos\nu)^3} A\sin\theta\ d\nu \tag{2-28}$$

$$\Delta\Omega = \int_0^{2\pi}\frac{\left(1-e^2\right)^2\sin(\omega+\nu)}{n^2 a\ (1 + e\cos\nu)^3\sin i} A\sin\theta\ d\nu \tag{2-29}$$

$$\Delta\omega = \int_0^{2\pi} \left[- \frac{\left(1-e^2\right)^2 \cos \nu}{n^2 ae \, (1 + e \cos \nu)^2} A \cos \theta \sin \alpha \right.$$

$$+ \frac{\left(1-e^2\right)^2 \sin \nu \, (2 + e \cos \nu)}{n^2 ae \, (1 + e \cos \nu)^3} A \cos \theta \cos \alpha$$

$$\left. - \frac{\left(1-e^2\right)^2 \cot i \sin (\omega+\nu)}{n^2 a \, (1 + e \cos \nu)^3} A \sin \theta \right] d\nu \qquad (2-30)$$

$$\Delta M_o = \int_0^{2\pi} \left[- \frac{\left(1-e^2\right)^{5/2} A \cos \theta \sin \alpha}{n^2 a \, (1 + e \cos \nu)^2} \left(\frac{2}{1 + e \cos \nu} - \frac{\cos \nu}{e} \right) \right.$$

$$- \frac{\left(1-e^2\right)^{5/2} \sin \nu \, (2 + e \cos \nu)}{n^2 ae \, (1 + e \cos \nu)^3} A \cos \theta \cos \alpha$$

$$\left. + \frac{3nt}{2a} \frac{da}{d\nu} \right] d\nu \qquad (2-31)$$

where the components of acceleration have been expressed in terms of the control angles.

Since the thrust is constant, and the mass is assumed to be constant over one orbit, the vehicle acceleration may be moved outside the integrals. In addition, with the assumption that the orbital elements can be treated as constants over one orbit, they too can be moved outside the integrals, where possible. Thus, the equations for the change in the orbital elements become

$$\Delta a = \frac{A}{n^2} 2\left(1-e^2\right) \int_0^{2\pi} \left[\frac{e \sin \nu \sin \alpha(\nu)}{(1 + e \cos \nu)^2} \right.$$

$$\left. + \frac{\cos \alpha(\nu)}{(1 + e \cos \nu)} \right] \cos \theta(\nu) \, d\nu \qquad (2\text{-}32)$$

$$\Delta e = \frac{A \left(1-e^2\right)^2}{n^2 a e} \left[e \int_0^{2\pi} \left[\frac{\sin \nu \sin \alpha(\nu) \cos \theta(\nu)}{(1 + e \cos \nu)^2} \right] d\nu \right.$$

$$\left. + \int_0^{2\pi} \left[1 - \frac{1-e^2}{(1 + e \cos \nu)^2} \right] \left(\frac{\cos \alpha(\nu) \cos \theta(\nu)}{1 + e \cos \nu} \right) d\nu \right]$$

$$= \left(\frac{1-e^2}{2ae} \right) \Delta a - \frac{A \left(1-e^2\right)^3}{n^2 a e} \int_0^{2\pi} \frac{\cos \alpha(\nu) \cos \theta(\nu)}{(1 + e \cos \nu)^3} d\nu$$

(2-33)

$$\Delta i = \frac{A}{n^2 a} \left(1-e^2\right)^2 \int_0^{2\pi} \frac{\cos (\omega+\nu) \sin \theta(\nu)}{(1 + e \cos \nu)^3} d\nu \qquad (2\text{-}34)$$

$$\Delta \Omega = \frac{A \left(1-e^2\right)^2}{n^2 a \sin i} \int_0^{2\pi} \frac{\sin (\omega+\nu) \sin \theta(\nu)}{(1 + e \cos \nu)^3} d\nu \qquad (2\text{-}35)$$

$$\Delta \omega = \frac{A \left(1-e^2\right)^2}{n^2 ae} \int_0^{2\pi} \left[\left[- \frac{\cos \nu \sin \alpha(\nu)}{(1 + e \cos \nu)^2} \right. \right.$$

$$\left. + \frac{\sin \nu (2 + e \cos \nu)}{(1 + e \cos \nu)^3} \cos \alpha(\nu) \right] \cos \theta(\nu)$$

$$\left. - \frac{\cot i \sin (\omega+\nu)}{(1 + e \cos \nu)^3} \sin \theta(\nu) \right] d\nu \qquad (2\text{-}36)$$

$$\Delta M_o = \frac{A\left(1-e^2\right)^{5/2}}{n^2 ae} \int_0^{2\pi} - \left[\left(2e - \cos\nu\ (1 + e\cos\nu)\right) \sin\alpha(\nu) \right.$$

$$\left. + \sin\nu\ (2 + e\cos\nu)\cos\alpha(\nu) \right] \frac{\cos\theta(\nu)}{(1 + e\cos\nu)^{-3}}\, d\nu$$

$$+ \frac{3nt}{2a}\Delta a \qquad (2\text{-}37)$$

Since this study does not address noncoplanar transfers, the out-of-plane control angle, θ, will be considered to be zero making equation (2-34) for Δi and equation (2-35) for $\Delta\Omega$ also equal to zero. In addition, only the changes in the semimajor axis and eccentricity are applicable to the remaining derivation, eliminating the need of equation (2-36) for $\Delta\omega$ and equation (2-37) for ΔM_o.

<u>Nondimensional</u> <u>Form.</u> The dependence of equations (2-32) and (2-33) on the semimajor axis and acceleration can be removed by introducing the nondimensional variables

$$\Delta a^* = \frac{n^2}{A}\Delta a$$

$$\Delta e^* = \frac{n^2 a}{A}\Delta e \qquad (2\text{-}38)$$

resulting in

$$\Delta a^* = 2\left(1-e^2\right) \int_0^{2\pi} \left[\frac{e\sin\nu\sin\alpha(\nu)}{(1 + e\cos\nu)^2} + \frac{\cos\alpha(\nu)}{(1 + e\cos\nu)} \right] d\nu \qquad (2\text{-}39)$$

$$\Delta e^* = \left(\frac{1-e^2}{2e} \right)\Delta a^* - \frac{\left(1-e^2\right)^3}{e} \int_0^{2\pi} \frac{\cos\alpha(\nu)}{(1 + e\cos\nu)^3}\, d\nu \qquad (2\text{-}40)$$

Equations (2-39) and (2-40) represent the primary governing equations for the fast timescale problem. Singularities occur in them when the eccentricity is equal to zero or one, e.g., if $e = 0$ equation (2-40) has a singularity. If $e = 1$, both equations (2-39) and (2-40) each have a singularity when $\nu = \pi$. However, only the singularity at $e = 0$ is of concern since $e = 1$ represents a parabolic trajectory. These singularities are removed in the final algorithm of the following section by extrapolation of the data obtained from these equations.

Planar Thrust Angle Optimal Control Laws

Unconstrained Problem.

Before discussing an optimal control law for the constrained problem, we should first consider the unconstrained problem to gain insight into the maximum changes in the semimajor axis and eccentricity that are possible in the coplanar problem. While the unconstrained optimal control laws will provide useful comparison data, they are obviously impractical since they optimize one variable while allowing unrestrained changes in the others.

The optimal control laws for the planar thrust control angle, α, which maximize the change in the semimajor axis and the eccentricity over one orbit in the unconstrained case can be obtained using the performance indices given by (3:47-48)

$$J_{a_{uc}} = \Delta a = \int_0^{2\pi} \frac{da}{d\nu}\, d\nu \tag{2-41}$$

$$J_{e_{uc}} = \Delta e = \int_0^{2\pi} \frac{de}{d\nu}\, d\nu \tag{2-42}$$

The magnitude of these performance indices can be maximized by first taking their variation resulting in

$$\delta J_{a_{uc}} = \int_0^{2\pi} \frac{\partial}{\partial\alpha} \frac{da}{d\nu}\, \delta\alpha\, d\nu \tag{2-43}$$

$$\delta J_{e_{uc}} = \int_0^{2\pi} \frac{\partial}{\partial\alpha} \frac{de}{d\nu}\, \delta\alpha\, d\nu \tag{2-44}$$

where the variations in the derivatives of the semimajor axis and eccentricity derived from equations (2-20) and (2-21) are

$$\frac{\partial}{\partial\alpha}\frac{da}{d\nu} = \frac{2e \sin\nu \left(1-e^2\right)}{n^2 (1 + e\cos\nu)^2} A \cos\theta \cos\alpha - \frac{2\left(1-e^2\right)}{n^2 (1 + e\cos\nu)} A \cos\theta \sin\alpha \tag{2-45}$$

$$\frac{\partial}{\partial\alpha}\frac{de}{d\nu} = \frac{\left(1-e^2\right)^2 \sin\nu}{n^2 a\, (1 + e\cos\nu)^2} A \cos\theta \cos\alpha - \frac{\left(1-e^2\right)^2 A \cos\theta \sin\alpha}{n^2 ae\, (1 + e\cos\nu)} \left[1 - \frac{1-e^2}{(1 + e\cos\nu)^2} \right] \tag{2-46}$$

A stationary point (preferably a minimum or maximum) is obtained when the variance of the performance index equals zero ($\delta J = 0$) for all possible values of the optimizing variable ($\delta\alpha$). This is only true if the integrands of equations (2-43) and (2-44) are zero (3:49). Thus,

$$\frac{\partial}{\partial\alpha}\frac{da}{d\nu} = 0 = \frac{2e \sin\nu \left(1-e^2\right)}{n^2(1 + e\cos\nu)^2} A\cos\theta\cos\alpha - \frac{2\left(1-e^2\right)}{n^2(1 + e\cos\nu)} A\cos\theta\sin\alpha \qquad (2\text{-}47)$$

and

$$\frac{\partial}{\partial\alpha}\frac{de}{d\nu} = 0 = \frac{\left(1-e^2\right)^2 \sin\nu}{n^2 a\,(1 + e\cos\nu)^2} A\cos\theta\cos\alpha - \frac{\left(1-e^2\right)^2 A\cos\theta\sin\alpha}{n^2 ae\,(1 + e\cos\nu)}\left[1 - \frac{1-e^2}{(1 + e\cos\nu)^2}\right] \qquad (2\text{-}48)$$

Solving these two equations for $\alpha(\nu)$ yields the following two optimal control laws for the planar thrust angle in the unconstrained problem

$$\alpha(\nu)_{a_{uc}} = \tan^{-1}\left[\frac{(1 + e\cos\nu)\, e\sin\nu}{(1 + e\cos\nu)^2 - \left(1-e^2\right)}\right] \qquad (2\text{-}49)$$

$$\alpha(\nu)_{e_{uc}} = \tan^{-1}\left[\frac{e\sin\nu}{1 + e\cos\nu}\right] \qquad (2\text{-}50)$$

Equation (2-49) represents the optimal control law for maximizing the change in the semimajor axis and equation

(2-50) represents the optimal control law for maximizing the change in eccentricity. These two control laws are independent of the vehicle acceleration, the orbit semimajor axis, and the out of plane thrust control angle (θ), making them valid for any coplanar or noncoplanar transfer. Direct substitution into equations (2-39) and (2-40) will result in the maximum possible change in the magnitude of the nondimensional semimajor axis or eccentricity over one orbit of a coplanar transfer.

Constrained Problem. The fast timescale problem is constrained by forcing the distance between the primary focus (Earth's center) and the transfer orbit perigee or apogee position to remain constant. This distance is defined as (2:25)

$$r_{a,p} \equiv a\,(1 \pm e) \qquad (2\text{-}51)$$

where the "+" sign is used for the radius of apogee, r_a, and the "-" sign for the radius of perigee, r_p.

The time rate of change in $r_{a,p}$ is given by

$$\frac{dr_{a,p}}{dt} = \frac{da}{dt}(1 \pm e) \pm a\,\frac{de}{dt} \qquad (2\text{-}52)$$

As was done previously for the orbital element perturbation equations, the independent variable can be changed from t to ν by multiplying by $\left(\frac{d\nu}{dt}\right)^{-1}$

$$\frac{dr_{\alpha,p}}{d\nu} = \frac{da}{d\nu}(1 \pm e) \pm a \frac{de}{d\nu} \qquad (2\text{-}53)$$

Integrating this expression over one orbit results in

$$\Delta r_{\alpha,p} = \int_0^{2\pi} \left[\frac{da}{d\nu} (1 \pm e) \pm a \frac{de}{d\nu} \right] d\nu \qquad (2\text{-}54)$$

Applying the assumption that the orbital elements and acceleration are constant, equation (2-54) simplifies to

$$\Delta r_{\alpha,p} = \Delta a \; (1 \pm e) \pm a \; \Delta e \qquad (2\text{-}55)$$

This expression can be nondimensionalized in the same manner as done previously for Δa. Thus, a new nondimensional parameter can be defined by multiplying equation (2-55) by $\left[n^2/A\right]$ producing

$$\Delta r^*_{\alpha,p} = \frac{n^2}{A} \Delta r_{\alpha,p} = \frac{n^2}{A} \Delta a \; (1 \pm e) \pm \frac{n^2 a}{A} \Delta e \qquad (2\text{-}56)$$

By direct substitution of equation (2-38) all dependence on the vehicle acceleration and the orbit semimajor axis can be removed resulting in the final form of the constraint relationship:

$$\Delta r^*_{\alpha,p} = \Delta a^* \; (1 \pm e) \pm \Delta e^* \qquad (2\text{-}57)$$

The optimal control laws for α which will maximize the magnitude of the change in semimajor axis or eccentricity can be obtained by using the performance indices

$$J_a = \Delta a = \int_0^{2\pi} \frac{da}{d\nu}\, d\nu \qquad (2\text{-}41)$$

$$J_e = \Delta e = \int_0^{2\pi} \frac{de}{d\nu}\, d\nu \qquad (2\text{-}42)$$

However, we must now add the constraint relationship described for $r_{a,p}$, making these indices of the form (3:48)

$$J_{a_c} = \int_0^{2\pi} \left[\frac{da}{d\nu} + \lambda_a \left(\frac{dr_{a,p}}{d\nu} - \frac{\Delta r_{a,p}}{2\pi} \right) \right] d\nu \qquad (2\text{-}58)$$

$$J_{e_c} = \int_0^{2\pi} \left[\frac{de}{d\nu} + \lambda_e \left(\frac{dr_{a,p}}{d\nu} - \frac{\Delta r_{a,p}}{2\pi} \right) \right] d\nu \qquad (2\text{-}59)$$

where λ_a and λ_e are the Lagrange multipliers. Substituting equation (2-52) into these two performance index relations

$$J_{a_c} = \int_0^{2\pi} \left[\frac{da}{d\nu} + \lambda_a \left(\frac{da}{d\nu}(1 \pm e) \pm a\frac{de}{d\nu} - \frac{\Delta r_{a,p}}{2\pi} \right) \right] d\nu$$

$$= \int_0^{2\pi} \left(\left[1 + \lambda_a (1 \pm e)\right] \frac{da}{d\nu} + \lambda_a a \frac{de}{d\nu} - \frac{\Delta r_{a,p}}{2\pi} \right) d\nu$$

(2-60)

$$J_{e_c} = \int_0^{2\pi} \left[\frac{de}{d\nu} + \lambda_e \left(\frac{da}{d\nu}(1 \pm e) \pm a\frac{de}{d\nu} - \frac{\Delta r_{a,p}}{2\pi} \right) \right] d\nu$$

$$= \int_0^{2\pi} \left((1 \pm \lambda_e a) \frac{de}{d\nu} + \lambda_e (1 \pm e) \frac{da}{d\nu} - \frac{\Delta r_{a,p}}{2\pi} \right) d\nu$$

(2-61)

Taking the variance of equations (2-60) and (2-61), noting that $\Delta r_{\alpha,p}$ is constant (equal to zero), and setting the integrands to zero (3:49) yields

$$\left[1 + \lambda_a (1 \pm e)\right] \frac{\partial}{\partial\alpha} \frac{da}{d\nu} + \lambda_a a \frac{\partial}{\partial\alpha} \frac{de}{d\nu} = 0 \qquad (2\text{-}62)$$

$$(1 \pm \lambda_e a) \frac{\partial}{\partial\alpha} \frac{de}{d\nu} + \lambda_e (1 \pm e) \frac{\partial}{\partial\alpha} \frac{da}{d\nu} = 0 \qquad (2\text{-}63)$$

where the variations in the derivatives of the semimajor axis and eccentricity are defined in equations (2-45) and (2-46). Solving these equations for $\alpha(\nu)$ produces

$$\alpha(\nu)_i = \tan^{-1} \left[\frac{e \sin \nu \quad (1 + e \cos \nu) \quad F_{1i}}{(1 + e \cos \nu)^2 \; F_{1i} - F_{2i}} \right] \qquad (2\text{-}64)$$

where, for the control law $\alpha(\nu)_a$ which maximizes the change in semimajor axis, the functions F_1 and F_2 are

$$F_{1a} = 2e + \lambda_a \left[2e \; (1 \pm e) \pm \left(1 - e^2\right) \right] \qquad (2\text{-}65)$$

$$F_{2a} = \pm \left(1 - e^2\right)^2 \qquad (2\text{-}66)$$

and, for the control law $\alpha(\nu)_e$ which maximizes the eccentricity,

$$F_{1e} = \left(1 - e^2\right) + \lambda_e a \left[2e \; (1 \pm e) \pm \left(1 - e^2\right) \right] \qquad (2\text{-}67)$$

$$F_{2e} = (1 \pm \lambda_e a) \left(1 - e^2\right)^2 \qquad (2\text{-}68)$$

By introducing a new nondimensional parameter

$$\lambda_e^* = \lambda_e a$$

equations (2-67) and (2-68) can be rewritten as

$$F_{1e} = \left(1 - e^2\right) + \lambda_e^* \left[2e \ (1 \pm e) \pm \left(1 - e^2\right) \right] \qquad (2\text{-}69)$$

$$F_{2e} = (1 \pm \lambda_e^*) \left(1 - e^2\right)^2 \qquad (2\text{-}70)$$

Thus, with the constrained radii, two control laws have been found, one maximizing the change in semimajor axis [equations (2-64) thru (2-66)] and the other maximizing the change in eccentricity [equations (2-64), (2-69), and (2-70)], that are independent of the vehicle acceleration, orbit semimajor axis, and out-of-plane thrust control angle. The final solution to the fast timescale problem can be found by substituting these control laws into equations (2-39) and (2-40) to find values of λ_a and λ_e^* which will drive equation (2-57) for Δr_a^* or Δr_p^* to zero. The relationship between $\Delta r_{a,p}^*$ and $\Delta r_{a,p}$ given by (2-56) shows Δr_a and Δr_p will also be zero for these values of the Lagrange multipliers, thus satisfying the given constraint relationship.

With the constrained radii control laws derived, the definition of even and odd functions (5:475) can be applied to evaluate equations (2-36) and (2-37) for the change in the argument of periapsis, $\Delta\omega$, and mean anomaly at epoch,

ΔM_o. The constrained control laws are odd functions (refer to Figure 2-9). With θ set to zero, insertion of any of the control laws into equation (2-36) makes the remaining two terms odd. Since the integral of an odd function over its total period is zero, both terms are zero. Thus, over one orbit, $\Delta\omega = 0$. Insertion into equation (2-37) results in the first two terms being odd, leaving ΔM_o equal to $\frac{3nt}{2a}\,\Delta a$.

Resulting Algorithms and Solutions

Two computer programs were written that numerically solved equation (2-39) for Δa^* and equation (2-40) for Δe^*, determining $\Delta r^*_{\alpha,p}$ as a function of the Lagrange multipliers (for values of eccentricity ranging between zero and one). The first program utilized the constrained control law, α_a, given by equations (2-64) thru (2-66) to maximize Δa (Δa^*). The second program utilized the constrained control law, α_e, given by equations (2-64), (2-69), and (2-70) to maximize Δe (Δe^*). Both programs incorporate the composite Simpson's rule (13:156) to solve the integrals found in the equations for Δa^* and Δe^*. Thus, ignoring the truncation error, the equation used to solve each integral is given by

$$\text{Integral} = \frac{1}{3}\left(\Delta\nu\right)\Big\{I(\nu=0) + 4I(\nu=\Delta\nu) + 2I(\nu=2\Delta\nu) + 4I(\nu=3\Delta\nu) + 2I(\nu=4\Delta\nu) + \ldots + 2I(\nu=2(\pi-\Delta\nu)) + 4I(\nu=2\pi-\Delta\nu) + I(2\pi)\Big\} \qquad (2\text{-}71)$$

where

$$I \equiv \text{Integrand}$$

$$\Delta\nu = \frac{2\pi}{n} \quad (n \text{ even})$$

The value of n used in the above relation was set to 360 in each of the runs of these programs to allow the planar thrust angles (control laws) to be calculated at one degree intervals around the orbit. Higher values of n did not improve the accuracy within the limits of the machine used (≅ 1.E-16).

These two programs were used to obtain a rough estimate of the range of the Lagrange multipliers, λ_a and $\lambda^*_\bullet$ ($\lambda_i^{(*)}$) where the the functions Δr^*_a and Δr^*_p cross the λ^*_i axis (equal to zero). Sample data (plotted in Figures 2-3 thru 2-6) shows that the desired values of these parameters lie between ± 1. This range provided the initial guesses to the secant method routine of Program DELAMAX2 (Appendix B-1) and Program DELEMAX2 (Appendix B-2), expansions of the original two programs. Given two initial guesses in the vicinity of the solution the next guess of $\lambda_i^{(*)}$ is found from (13:265)

$$\lambda^{(*)}_{i_{j+1}} = \lambda^{(*)}_{i_j} - \Delta r^*_{a,p_j} \left[\frac{\lambda^{(*)}_{i_j} - \lambda^{(*)}_{i_{j-1}}}{\Delta r^*_{a,p_j} - \Delta r^*_{a,p_{j-1}}} \right] \qquad (2\text{-}72)$$

where the subscript j represents the current value of the parameter. Thus, with this routine, these two programs

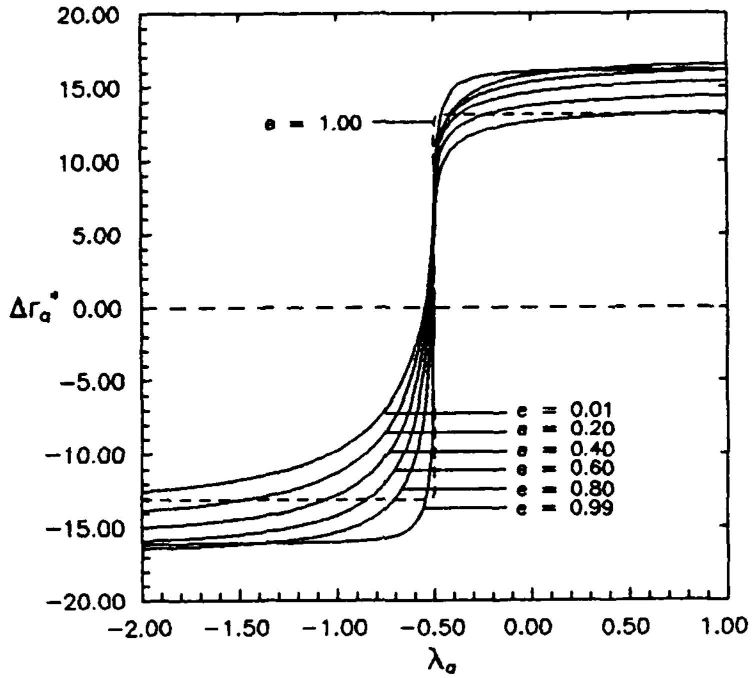


Figure 2-3. Nondimensional Change in Radius of Apogee as a Function of the Lagrange Multiplier (max Δa Control Law) at Various Eccentricity

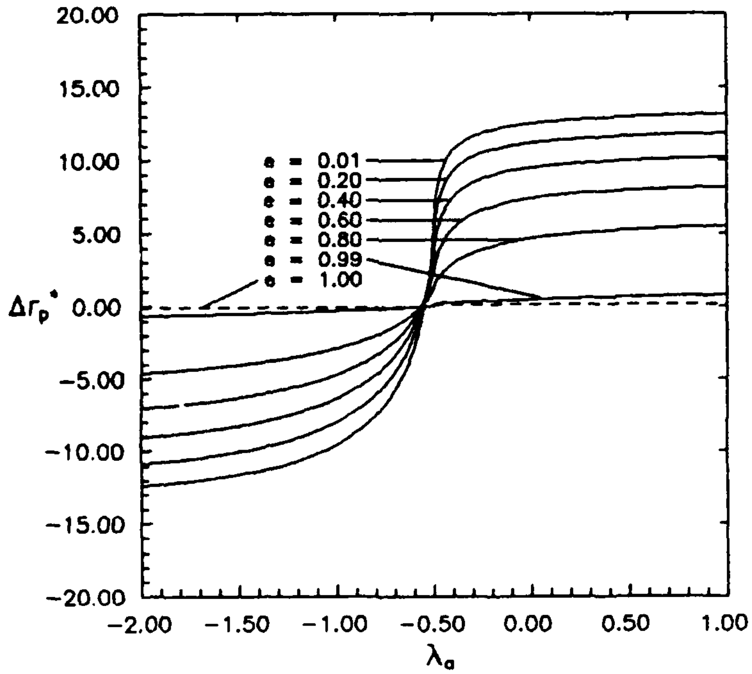


Figure 2-4. Nondimensional Change in Radius of Perigee as a Function of the Lagrange Multiplier (max Δa Control Law) at Various Eccentricity

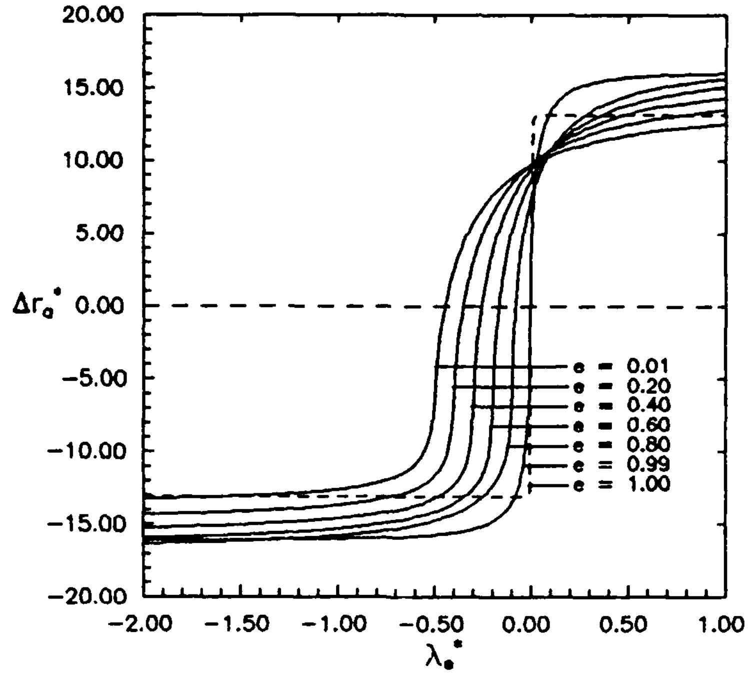


Figure 2-5. Nondimensional Change in Radius of Apogee as a Function of the Lagrange Multiplier (max Δe Control Law) at Various Eccentricity

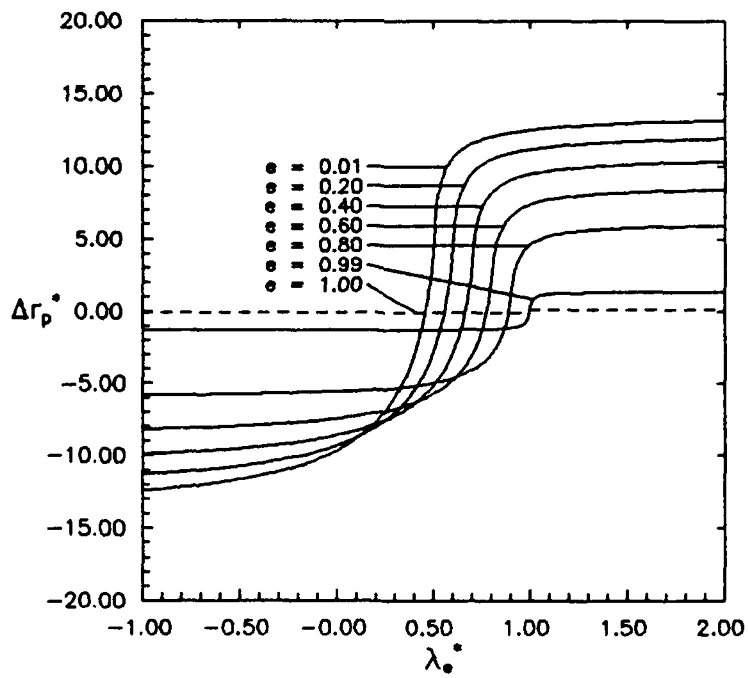


Figure 2-6. Nondimensional Change in Radius of Perigee as a Function of the Lagrange Multiplier (max Δe Control Law) at Various Eccentricity

solved the fast timescale problem, providing λ_α, λ^*_e, Δa^*, and Δe^* as functions of e, while holding Δr_α or Δr_p equal to zero. Resulting plots of these parameters are provided in Figures 2-7 thru 2-9.

The uniqueness of the particular constraints imposed is seen in the plots of Δa^*_α, $\pm\Delta e^*_\alpha$, $\pm\Delta a^*_e$, and Δe^*_e displayed in Figures 2-8 and 2-9. Notice that imposing each of the two constraints results in identical changes in the magnitudes of Δa^* and Δe^*, whether these changes were obtained from the control law α_α which maximizes Δa^* or the control law α_e which maximizes Δe^*. However, while the magnitudes of these resulting changes are the same for each of the two control

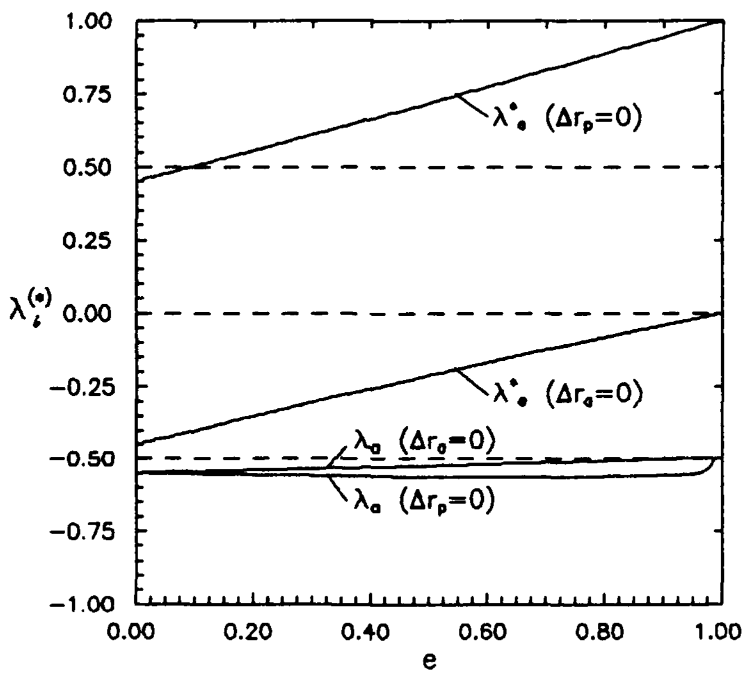


Figure 2-7. Lagrange Multiplier as a Function of Eccentricity for Constant Apogee and Perigee

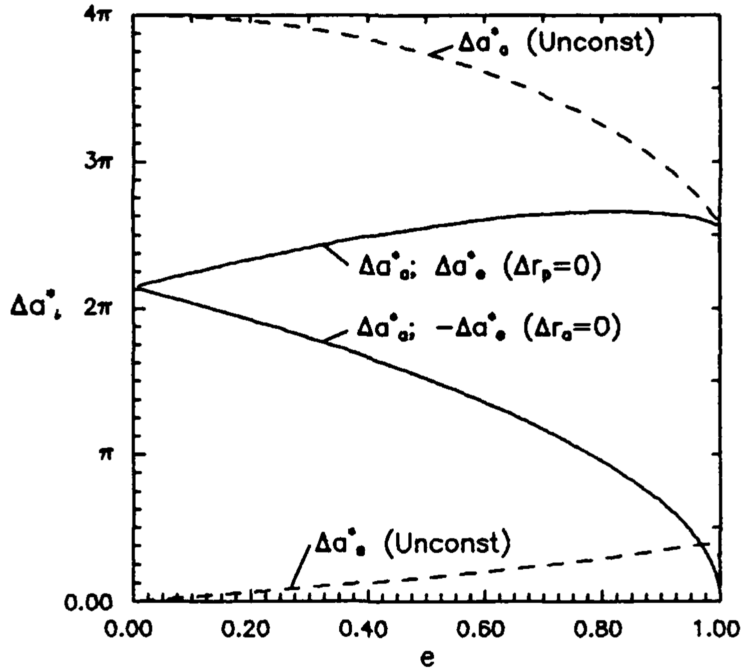


Figure 2-8. Nondimensional Change in Semimajor Axis as a Function of Eccentricity for Constant Apogee and Perigee

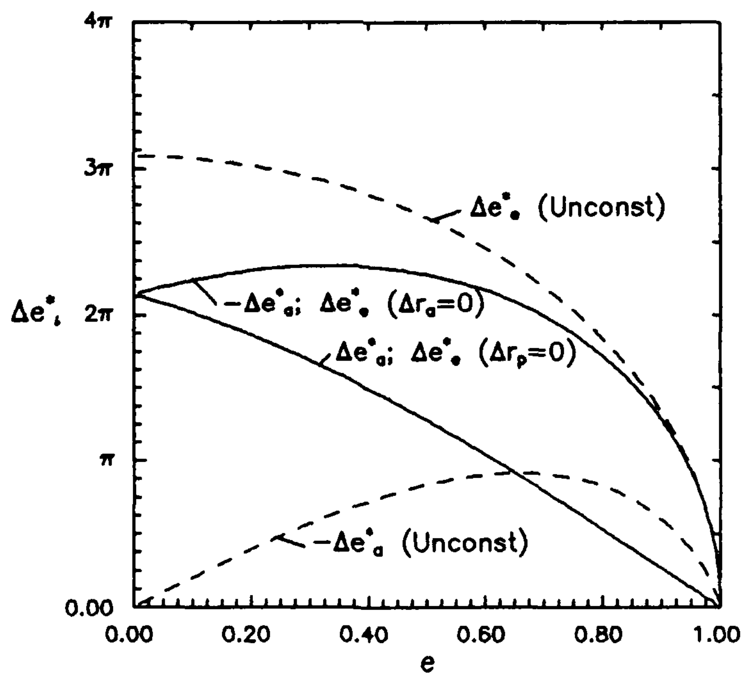


Figure 2-9. Change in Eccentricity Parameter as a Function of Eccenticity for Constant Apogee and Perigee

laws, when constrained by $\Delta r_a = 0$, α_a produces changes of opposite sign (negative) to those found using α_e.

Two similar programs incorporating the composite Simpson's rule algorithm, DELAMAXUC (Appendix B-3) and DELEMAXUC (Appendix B-4), provide the solutions to equations (2-39) and (2-40) for the unconstrained problem utilizing the control laws defined in equations (2-49) and (2-50). The first program (utilizing α_{auc}) provides the maximum possible changes in the nondimensional semimajor axis, Δa_a^*, and the associated residual changes in the eccentricity parameter, Δe_a^*, as functions of eccentricity. The second (utilizing α_{euc}) provides the maximum possible values of the changes in the eccentricity parameter, Δe_e^*, and the associated changes in the nondimensional semimajor axis, Δa_e^*, as functions of eccentricity.

The data obtained from these two programs is plotted in Figures 2-8 and 2-9 for comparison with the constrained results. The plots show the maximum obtainable changes in the nondimensional semimajor axis or eccentricity parameter are reduced as eccentricity is increased. In addition, it can be seen that utilizing α_{auc} to increase the semimajor axis of eccentric orbits results in negative values in Δe_a^* driving the eccentricity to zero. Once the orbit is circularized, Δe_a^* becomes zero and the maximum change in the nondimensional semimajor axis ($\Delta a_a^* = 4\pi$) is obtained. This maximum value is identical to the result obtained by Alfano for spiral transfers between coplanar circular orbits where both α and θ are constant and equal to zero. Thus, this control law is ideal to increase the semimajor axis while circularizing an eccentric orbit. There is never a threat of impact with the primary body, since the changes in Δr_a and Δr_p are both positive. On the other hand, utilizing this control law to decrease the semimajor axis (thrusting in the opposite direction) results in an uncontrollable increase in the eccentricity and decrease in Δr_a and Δr_p, making it impractical.

Utilizing α_{euc} to increase the eccentricity of a circular or eccentric orbit produces positive changes in the semimajor axis. However Δr_p is negative, leading to a possible impact with the primary body. Thrusting in the opposite direction decreases both the eccentricity and semimajor axis of an eccentric orbit. However, Δr_a is negative while Δr_p is positive, making an application to circularizing eccentric orbits possible if the uncontrollable changes in the semimajor axis were acceptable.

Unfortunately, the constraint relationship imposed in this study provides results significantly less than the optimal. Figure 2-8 shows when e = 0, the magnitude of change in the semimajor axis is slightly higher than one half ($\cong 2.2\pi$) of that obtained in the unconstrained case. As $e \rightarrow 1$, the magnitude of Δa^* constrained by $\Delta r_p = 0$ approaches the unconstrained value while the magnitude of

Δa^* constrained by $\Delta r_a = 0$ decreases to zero. Figure 2-9 shows similar results for Δe^*. When $e = 0$, the magnitude of change in the eccentricity is exactly the same as the change in semimajor axis and approximately two thirds of the unconstrained value. As $e \rightarrow 1$, the magnitude of Δe^* constrained by $\Delta r_a = 0$ approaches the unconstrained value while the magnitude of Δa^* constrained by $\Delta r_p = 0$ diverges toward zero.

The data from Figures 2-8 and 2-9 can be used in conjunction with the vehicle accelerations derived in Appendix A to validate that the changes in eccentricity and semimajor axis over one orbit are small. Utilizing the maximum values of Δa^* and Δe^* for the unconstrained control laws (4π and $\cong 3.1\pi$, respectively) and the maximum value of acceleration seen by the arcjet TVS at low earth orbit (LEO) and geosynchronous orbit (GEO), equation (2-38) for the dimensional values of Δa and Δe can now be solved:

$$\Delta a = \frac{A}{n^2}\Delta a^* = \frac{A\,a^3}{\mu_\oplus}\Delta a^* = \frac{A\,a^3}{3.986 \times 10^{14}\ m^3/s^2}(4\pi)$$
$$= \left(3.153 \times 10^{-14}\ m^{-2}\right) A\,a^3$$

$$\Delta e = \frac{A}{n^2 a}\Delta e^* = \frac{A\,a^2}{\mu_\oplus}\Delta e^* = \frac{A\,a^2}{3.986 \times 10^{14}\ m^3/s^2}(3.1\pi)$$
$$= \left(2.443 \times 10^{-14}\ m^{-2}\right) A\,a^2$$

At LEO ($a = 6.68 \times 10^6 m$, $A_{max} = .00446\ m/s^2$) these equations yield a change in semimajor axis of $\cong .6\%$ of its original value and a total change in eccentricity is ≅ .005. At GEO ($a = 4.22 \times 10^7$m, $A_{max} = .00257\ m/s^2$), the change in semimajor axis has increased to ≅ 14.5% of its original value and the total change in eccentricity is up to .112. Thus, as expected, the original assumption is valid near LEO, but becomes much less accurate as the semimajor axis increases to GEO.

The programs of Appendices B-1 and B-2 were also used to obtain the changes in the constrained planar thrust angle (control laws), α_i, as the spacecraft proceeds around an orbit of given eccentricity. Data for orbits with eccentricities of .01, .4, and .8 is plotted in Figure 2-10.

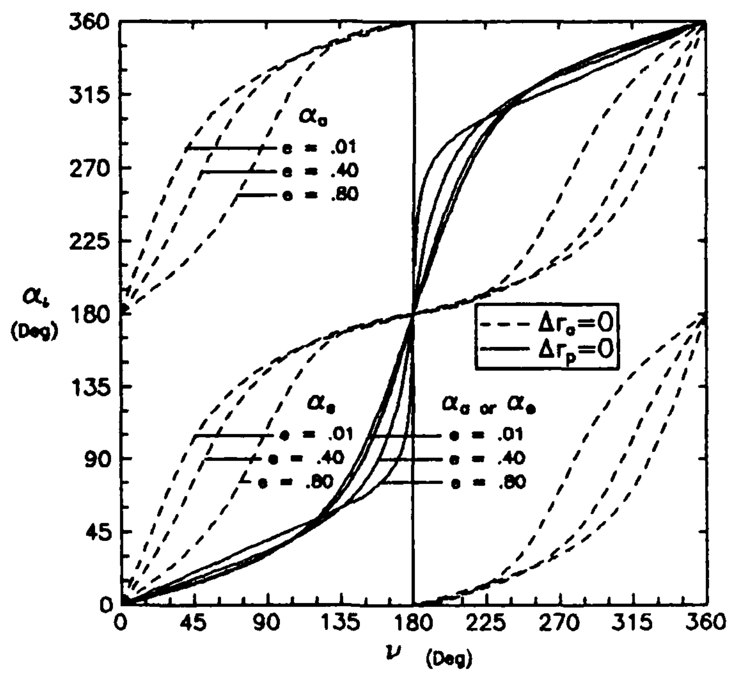
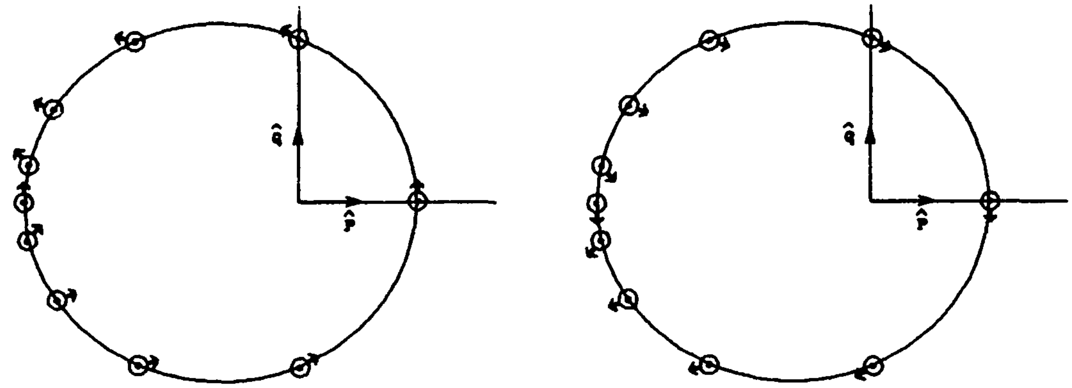


Figure 2-10. Constrained Thrust Vector Angle as a Function of the True Anomaly for Various Eccentricity

For the constraint relationship $\Delta r_p = 0$, the plots of α_a and α_e only differ slightly at e = .01 and become indistinguishable as the eccentricity becomes larger. When apogee is constrained ($\Delta r_a = 0$), the plots of α_a and α_e differ by a shift of ±180 degrees (again, slightly more for lower values of e). This "phase shift" explains the difference in signs of Δa^* and Δe^* obtained by these two control laws earlier (see Figures 2-8 and 2-9) when apogee was constrained. By simply shifting the curves for one of the control laws by 180° to more closely agree with the curves of the other we can obtain the change in sign of Δa^* and Δe^* needed to make the results match. Thus, even though the two control laws differ slightly at lower eccentricities, they do in fact produce identical results for the changes in these two parameters (magnitude and sign).

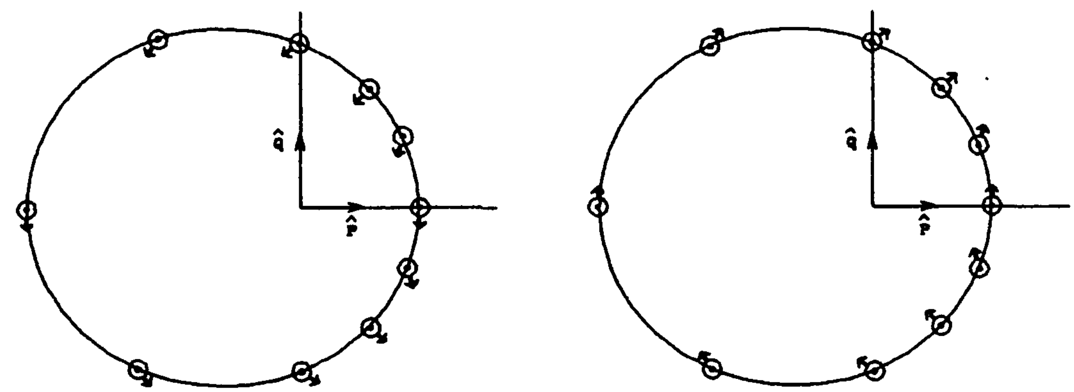
A final point regarding the plots of Δa^* and Δe^* in Figures 2-8 and 2-9 involves their significance when related to a desired transfer. For example, to increase the radius of a circular orbit (increase a) using the specified constraints, perigee is first held constant while pushing apogee out to the larger radius. This would result in an increase in eccentricity since we are basically "stretching" the original orbit from a circular to an elliptical shape. To complete the transfer, apogee is held constant while raising perigee to the final radius, decreasing the eccentricity until the orbit was circularized (e = 0). On the other hand, to decrease the radius of a circular orbit (decrease a), this procedure is reversed. First, apogee is held constant and perigee is lowered to the desired radius, increasing the eccentricity. Then, perigee is held constant while apogee is pushed down to the final radius, decreasing the eccentricity to zero. Therefore, it is expected that the control laws constrained by maintaining perigee constant will produce changes in the parameters Δa^* and Δe^* which are of the same sign (±), while the control laws constraining apogee produce changes of the opposite sign ($\pm\Delta a^*$ and $\mp\Delta e^*$). Figures 2-8 and 2-9 show this is exactly what has been found. For both control laws, constraining perigee results in positive changes in Δa^* and Δe^*. If α_a is put "in phase" with α_e, constraining apogee produces positive changes in Δa^* and negative changes in Δe^* for both control laws. Thrusting in the opposite direction (shifting α_i ±180°) changes the sign of all of the parameters making both Δa^* and Δe^* negative when constraining perigee, and Δa^* negative and Δe^* positive when constraining apogee.

To better illustrate the final four control laws (α_i and $\alpha_i \pm 180°$ for each of the two constraints), Figures 2-11 (a) and 2-11 (b) provide sketches of the required thrust vector control for a single orbit utilizing the data (e = .40) provided in Figure 2-10. When perigee is constrained [Figure 2-11 (a)], the direction of the thrust vector changes very rapidly near apogee. When apogee is



$+\Delta a$, $+\Delta e$ $\qquad$ $-\Delta a$, $-\Delta e$

(a). Constant Perigee



$+\Delta a$, $-\Delta e$ $\qquad$ $-\Delta a$, $+\Delta e$

(b). Constant Apogee

Figure 2-11. Spacecraft Thrust Vector Control (e = .40) constrained [Figure 2-11 (b)], the rapid change occurs near perigee. These characteristic can also be seen in the slopes of the plots of α_i in Figure 2-10.

To complete the short timescale problem, a final program, Program INTERPO (Appendix B-5), was written to provide a solution for the $\lambda_i^{(*)}$, Δa^*, and Δe^* for any given value of eccentricity less than one. This includes solutions at e = 0 that were unobtainable in the previous programs due to the singularity in equation (2-40). A Newton formula (13:92-98) was incorporated to produce an interpolating or extrapolating n^{th} order polynomial to calculate the values of the parameters between the data points obtained from the earlier programs which solved equations (2-39) and (2-40). This program is utilized as a subroutine in the slow timescale solution algorithm to provide an efficient method of calculating Δa^* and Δe^* as functions of eccentricity.

III. The Slow Timescale Problem

Problem Statement

The slow timescale problem provides for the complete transfer of a spacecraft from an initial tc final orbit through many revolutions about the primary body. Changes in the orbital elements found in the fast timescale problem of chapter 2 are included, as well as the changes in the vehicle mass due to propellant expulsion.

Derivation

Perturbation Equations. The period of an elliptical orbit is found from (2:33)

$$\mathbb{TP} = \frac{2\pi}{n} \tag{3-1}$$

Given the low thrust assumption, the changes in the semi-major axis and eccentricity with respect to time can be approximated from the single orbit case of the fast time-scale solution by

$$\frac{da}{dt} = \pm \frac{\Delta a}{\mathbb{TP}} = \pm \left[\frac{\Delta a^* \; A}{n^2} \right] \left[\frac{n}{2\pi} \right] = \pm \frac{A}{2\pi n} \Delta a^* \tag{3-2}$$

$$\frac{de}{dt} = \pm \frac{\Delta e}{\mathbb{TP}} = \pm \left[\frac{\Delta e^* \; A}{n^2 a} \right] \left[\frac{n}{2\pi} \right] = \pm \frac{A}{2\pi n a} \Delta e^* \tag{3-3}$$

where the nondimensional parameters, Δa^* and Δe^*, are given in equations (2-39) and (2-40). The "sign" of each equation is dependant on the change in the unconstrained radii desired (increase or decrease) and the control law chosen (α_a or α_e). To remove the dependence of the above two equations on the vehicle acceleration, the independent variable can be changed from time to velocity:

$$\frac{da}{dv} = \left(\frac{dv}{dt}\right)^{-1}\left(\frac{da}{dt}\right) = \pm\left(\frac{1}{A}\right)\frac{A}{2\pi n}\,\Delta a^* = \pm\frac{\Delta a^*}{2\pi n} \qquad (3\text{-}4)$$

$$\frac{de}{dv} = \left(\frac{dv}{dt}\right)^{-1}\left(\frac{de}{dt}\right) = \pm\frac{\Delta e^*}{2\pi n a} \qquad (3\text{-}5)$$

Based on the initial semimajor axis, a_o, and initial mean anomaly, n_o, we can define three nondimensional variables:

$$\bar{a} \equiv \frac{a}{a_o}$$

$$\bar{n} \equiv \frac{n}{n_o} \qquad (3\text{-}6)$$

$$\bar{v} \equiv \frac{v}{a_o n_o}$$

where

$$n_o \equiv \sqrt{\frac{\mu_\oplus}{a_o^3}} \qquad (3\text{-}7)$$

Substituting these parameters into equations (3-4) and (3-5) produces the final form of the two governing equations:

$$\frac{d\bar{a}}{d\bar{v}} = \pm\left(\frac{\frac{1}{a_o}}{\frac{1}{a_o n_o}}\right)\frac{\Delta a^*}{2\pi n} = \pm\frac{\Delta a^*}{2\pi\bar{n}} = \pm\,\bar{a}^{3/2}\,\frac{\Delta a^*}{2\pi} \qquad (3\text{-}8)$$

and

$$\frac{de}{d\bar{v}} = \pm \left[\frac{1}{\frac{1}{a_o n_o}} \right] \frac{\Delta e^*}{2\pi na} = \pm \frac{\Delta e^*}{2\pi \bar{n}\bar{a}} = \pm \bar{a}^{1/2} \frac{\Delta e^*}{2\pi} \qquad (3\text{-}9)$$

In addition, substitution into equation (2-51) for the radius of apogee and perigee results in a new nondimensional form of the radii equation:

$$\bar{r}_{a,p} \equiv \frac{r_{a,p}}{a_o} = \bar{a} \, (1 \pm e) \qquad (3\text{-}10)$$

<u>Mass Flow Rate/Acceleration Relation.</u> The vehicle mass and acceleration are related through Newton's second law of motion (2:3-4):

$$T = m(t) \; A(t) \qquad (3\text{-}11)$$

where

$T \equiv$ vehicle thrust (constant)
$m(t) \equiv$ vehicle mass at time t
$A(t) \equiv$ vehicle acceleration at time t

At the beginning of the transfer ($t = t_{initial} = 0$), equation (3-11) can be written in terms of the initial vehicle mass and acceleration:

$$T = m_o A_o \qquad (3\text{-}12)$$

Additionally, the vehicle mass at time t can be expressed in terms of the initial mass and propellant mass flow rate, $\dot{m}$, using the relation

$$m(t) = m_o - \dot{m} \; t \qquad (3\text{-}13)$$

Assuming constant thrust implies the mass flow rate is also constant.

Combining equations (3-11) thru (3-13) the vehicle acceleration can now be modeled in the slow timescale problem as

$$A(t) = \frac{dv}{dt} = \frac{A_o}{(1 - \dot{M}t)} \qquad (3\text{-}14)$$

where $\dot{M}$ is the specific mass flow rate, $\dot{m}/m_o$.

<u>Total Transfer Velocity Change/Time Relation.</u> The total accumulated velocity change is found by integrating equation (3-14) between 0 and t_f resulting in

$$v(t_f) - v_o = \Delta v(t_f) = -\frac{A_o}{\dot{M}} \ln (1 - \dot{M}t_f) \qquad (3\text{-}15)$$

Solving this expression for the total transfer time provides the final equation

$$t_f = \left[1 - \exp\left(\frac{\Delta v}{I_{SP} g_c} \right) \right] \frac{1}{\dot{M}} \qquad (3\text{-}16)$$

where

$$I_{SP} g_c = \frac{A_o}{\dot{M}}$$

<u>Resulting Algorithms</u>

<u>Slow Timescale Constraint Application.</u> One method of completing the transfer of the slow timescale problem involves applying the results of the fast timescale problem for each constraint over an extended period to relocate the

perigee and apogee positions separately. Program TRANSMUL, Appendix C-1, was written to perform the complete transfer by constraining one radii (apogee or perigee) over many revolutions until the other radii matches the final orbit. For example, if final perigee is higher than that of the original orbit, it is first held constant while apogee is increased or decreased, as required, to its final value. The transfer is completed by constraining apogee as perigee is raised to its final position. On the other hand, if final perigee is lower, apogee is first constrained while perigee is lowered, then perigee is constrained while apogee is raised or lowered to complete the transfer. This procedure applies to any transfer, including circular-to-circular transfers. Thus, given the initial and final orbit semimajor axes and eccentricities, primary body gravitational parameter and radius, and vehicle parameters (specific impulse, initial mass, and propellant mass flow rate), this program calculates the total accumulated velocity change and time for the constrained radii transfer.

The program incorporates an ordinary differential equations integrator equipped with a fourth order predictor-corrector algorithm, to numerically solve equations (3-8) and (3-9). In addition, as stated in chapter two, the final algorithm of the fast timescale problem (Appendix B-5) has been incorporated to provide nondimensional changes in the orbital elements needed to solve equations (3-8) and (3-9).

Given the current value of the orbit eccentricity and the constraint condition (either constant r_a or r_p), values of Δa^* and Δe^* are generated from data obtained using the constrained control law that maximizes Δe (α_e). The other constrained control law that maximizes Δa (α_a) could have been used since it provides identical results.

As a comparison in circular-to-circular transfers, this program also provides the Δv requirements of spiral and Hohmann transfers. For planar spiral transfers, the velocity change is given simply by the difference between the velocities associated with each orbit (10:463):

$$\Delta v_{spiral} = |v_{cs1} - v_{cs2}| \qquad (3\text{-}17)$$

where (2:165-166)

$$v_{csi} = \sqrt{\frac{\mu_\oplus}{r_i}}$$

The subscripts (1) and (2) in the above relation refers to the initial and final orbit, respectively. For Hohmann transfers, the total velocity change is obtained from (2:163-166)

$$\Delta v_{Hohmann} = \Delta v_1 + \Delta v_2 \qquad (3\text{-}18)$$

where

$$\Delta v_i \equiv |v_i - v_{csi}|$$

$$v_i = \left[2\mu_\oplus \left(\frac{1}{r_i} - \frac{1}{r_1 + r_2} \right) \right]^{1/2}$$

and r is the orbit radius.

Fast Timescale Constraint Application. Figures 2-8 and 2-9 show when the eccentricity is zero, the changes in the nondimensional semimajor axes resulting from each constraint are equal in magnitude and sign while the changes in the eccentricity parameters are equal in magnitude, but opposite in sign. This means that if we apply one constraint for one orbit, then the other constraint during the next orbit, over two revolutions, the eccentricity can be held nearly constant (≅ 0) as the semimajor axis is increased or decreased. Program TRANSALT, Appendix C-2, utilizes this idea to complete the slow timescale transfer by applying the fast timescale results to two revolutions of the transfer at a time, instead of one as was done in the previous program. Perigee is constrained during one of the two revolutions and apogee during the other. All routines of the previous program have been incorporated, as well as the following modified versions of equations (4-8) and (4-9) to evaluate the average change in the orbital elements occurring during the two orbits:

$$\frac{d\bar{a}}{d\bar{v}} = \frac{(\bar{a})^{3/2}}{2\pi}\left[\frac{\pm \Delta a_p^* \mp \Delta a_\alpha^*}{2}\right] \qquad (3\text{-}19)$$

$$\frac{de}{d\bar{v}} = \frac{(\bar{a})^{1/2}}{2\pi}\left[\frac{\pm \Delta e_p^* \mp \Delta e_\alpha^*}{2}\right] \qquad (3\text{-}20)$$

where the subscripts (a) and (p) refer to constrained perigee or apogee, respectively. The "signs" in these relations assume the control law $\alpha_{\bullet}$ is utilized. The upper signs are used for transfers to higher orbits, while the lower signs are used for transfers to lower orbits. This combination of constraints keeps each of the revolutions nearly circular as the orbit radius is raised or lowered, making this routine valid for circular-to-circular transfers only.

As an example, consider a transfer to a higher orbit. During the first revolution of the transfer, perigee is constrained while apogee is raised. At the beginning of the second revolution, the constraint is switched making apogee constrained while perigee is raised. By the end of the second revolution, the orbit has been recircularized at a higher radius. This procedure is repeated until the radius matches that of the final desired orbit.

IV. Results

Circular-to-Circular Transfers

Computer runs were performed for ratios of the final to initial orbit semimajor axes between 0 and 100. Utilizing the data obtained, a comparison among the nondimensional Δv requirements for each of the constrained radii, spiral, and Hohmann transfers is shown in Figure 4-1 below. Sample data for the LEO to GEO transfer performed using the slow and fast timescale algorithms is provided in Appendices D-1 and D-2, respectively. As expected after reviewing the results of the fast timescale problem, the Δv requirements for the

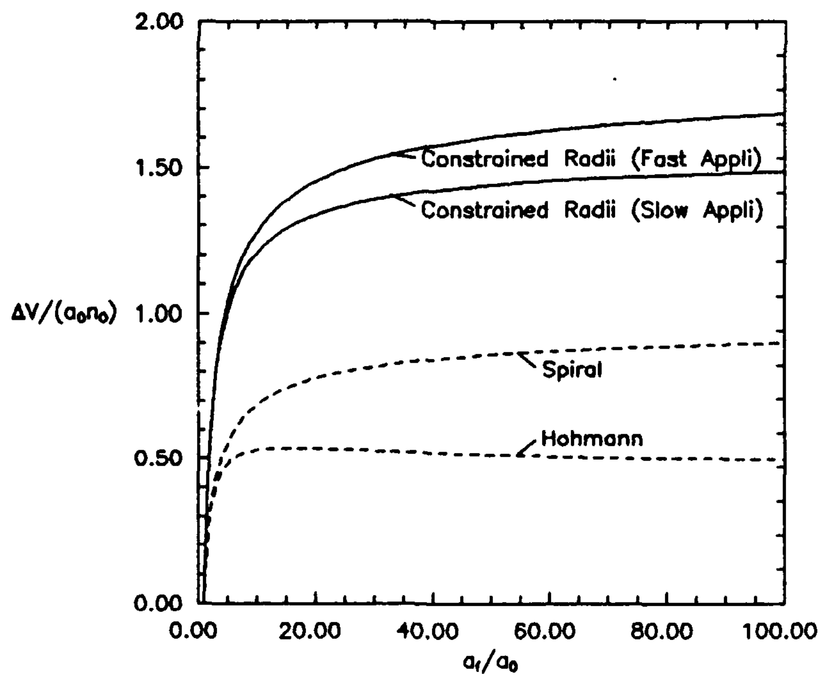


Figure 4-1 Nondimensional Total Accumulated Velocity Change for Ratios of Final to Initial Semimajor Axis

constrained radii transfers are higher than those of the spiral transfer.

The plots of Figure 4-1 show the Δv requirements resulting from applying each constraint separately during the slow timescale problem are slightly lower than those obtained by applying a combination of the constraints in the fast timescale problem. The reason for this difference can be seen by referring to the plots of Δa^* in Figure 2-8. While the fast timescale application keeps the eccentricity near zero, the slow timescale application allows the eccentricity to increase. Figure 2-8 shows higher eccentricity results in larger values of Δa^* when perigee is constrained and lower values of Δa^* when apogee is constrained. However, the data obtained using the slow timescale application shows apogee is constrained less than 30% of the total transfer time. This means that during nearly 70% of the total transfer, the change in semimajor axis per revolution is greater than that obtained from the transfer using the fast timescale application where the eccentricity is continually kept near zero. Thus, the application of the constraints separately in the slow timescale problem result in a quicker and less expensive transfer.

Figure 4-2 shows the resulting trajectory of a transfer between LEO and GEO using the slow timescale application of the constrained radii control law. Unfortunately, this transfer is still nearly 1.75 times more expensive to perform than a spiral transfer.

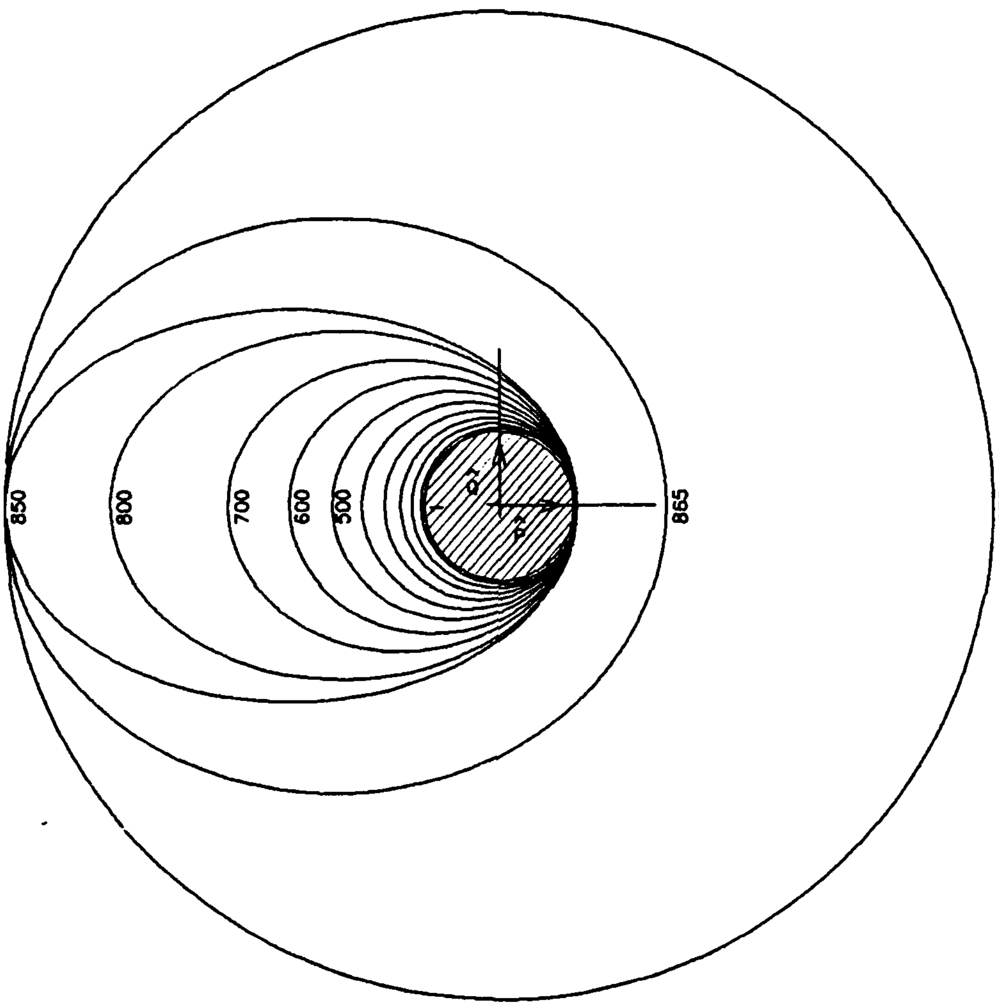


Figure 4-2. Trajectory of Transfer Between Leo (300 km) and GEO (35,863 km) using the Constrained Radii Control Law. Perigee is fixed as apogee is raised to GEO. Then, apogee is fixed as perigee is raised to complete the transfer to GEO.

Note: The revolution number is indicated where possible.

Circular-to-Eccentric Transfers

The most feasible application of the constrained radii transfer is to transfers between circular and eccentric orbits. One such transfer is that needed by communications satellites designed to provide coverage of the northern hemisphere (9:54.3.1-54.4.5). These satellites are placed into Molniya orbits, named after the Soviet communications satellites that first used them in the 60's. Molniya orbits are highly elliptical (e ≅ .73, depending on application) with a period of 12 hr and an inclination of 63.4 degrees.

The inclination of Molniya orbits is specifically chosen to maintain the argument of periapsis, ω, at 270 degrees, keeping apogee directly above the northern hemisphere. The major perturbation influencing satellites in earth orbit is due to the earth's oblateness, the J_2 term in the geopotential (14:46,86-91). By substituting the secular terms from the J_2 disturbing function into the disturbing function form of the Lagrange planetary equation for $\dot{\omega}$

$$\dot{\omega} = - \frac{3\ n\ J_2\ r_{\oplus}^2}{2\ a^2\ (1 - e^2)^2} \left[\frac{5}{2} \sin^2 i - 2 \right] \qquad (4\text{-}1)$$

it can be seen that by choosing an inclination of 63.4°, the right hand term becomes zero, driving $\dot{\omega}$ to zero. Thus, this critical inclination eliminates the influence of the earth's oblateness on the spacecraft.

A simulated transfer to a Molniya orbit was performed using the 5000 kg arcjet transfer vehicle (discussed in Appendix A). The transfer assumed an initial 807 km circular orbit (a = 7185 km) with inclination of 63.4°. Thrusting begins at $\omega = 270°$ using the constrained perigee control law ($\nu = 0$, initially) and continues until the semimajor axis and eccentricity are increased to the final Molniya orbit (a = 26,610 km and e = .73). Data obtained from the computer run is included in Appendix D-3. In addition, Figure 4-3 below provides a plot of the spacecraft trajectory obtained to illustrate the transfer performed.

The data obtained indicates the transfer to the Molniya orbit required a total Δv of 5.83 km/sec and took approxi-

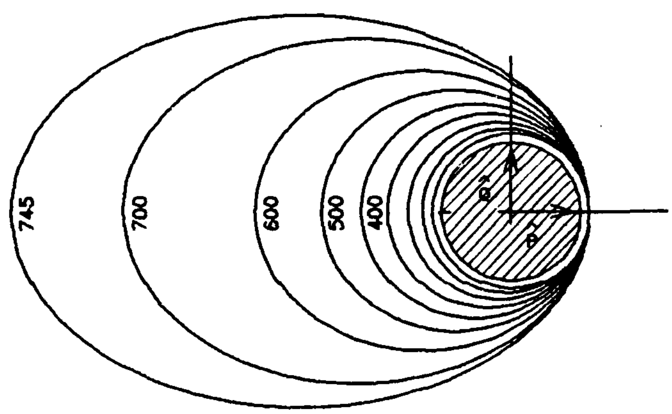


Figure 4-3. Spacecraft Trajectory for Transfer to Molniya Orbit using Constrained Radii Control Law

Note: The revolution number is indicated where possible.

mately 72.4 days to complete. With these values known, using equation (A-15) the total propellant used is

$$m_p = \dot{m}t = (.398 \times 10^{-3} \text{ kg/sec})(3.13 \times 10^{6} \text{ s}) = 1244 \text{ kg}$$

For comparison, consider executing this transfer with a chemical system. Only the first half of the Hohmann transfer needs to be performed, thus from equation (3-18) the total Δv is found by

$$\Delta v = |v_1 - v_{cs1}|$$

$$= \left| \left(\frac{\mu_{\oplus}}{r_p} \right)^{1/2} - \left[2\mu_{\oplus} \left(\frac{1}{r_p} - \frac{1}{r_p + r_a} \right) \right]^{1/2} \right|$$

where r_a and r_p are the final orbit radius of apogee and perigee, respectively. This equation yields a Δv of 2.35 km/sec. The total propellant needed can be found using equation (A-7)

$$m_p = m_T \left[1 - \exp\left(\frac{-\Delta v}{I_{sp} g_c} \right) \right]$$

Thus, assuming an I_{sp} of 300 sec, this equation gives a propellant mass of 2749 kg, over twice that needed by the arcjet propelled vehicle.

Eccentric-to-Eccentric Transfers

While the slow timescale constraint application algorithm was designed for application to eccentric-to-eccentric orbit transfers, it provides no control of ω other than keeping it constant. Thus, it is only useful for eccentric transfers which require no change in ω. The resulting trajectory for such a transfer would be similar to the circular-to-circular transfer shown in Figure 4-2.

V. Conclusions and Recommendations

Two optimal control laws for a continuous, low thrust spacecraft, each resulting in identical changes in the orbital elements have been derived. Constrained by constant radius of perigee or apogee, these control laws can be applied to provide an optimal coplanar circular-to-eccentric or eccentric-to-eccentric ($\Delta\omega = 0$) transfer, supplying needed control of the perigee and apogee heights. Application to circular-to-circular transfers is not feasible since it is much more expensive (larger Δv) and complex [$\alpha = \alpha(\nu)$] to perform than the optimal spiral control law ($\alpha = 0$).

The results of the application of the constrained radii control laws in the slow timescale problem inherently become less accurate as the semimajor axis increases. This is due to the associated increase in orbital period which slowly makes the change in semimajor axis more and more significant. Therefore, the assumption of the fast timescale which states that the changes in the orbital elements are small eventually becomes invalid. However, as was the case with the control law derived by Alfano (1:52), in actual application, a closed loop guidance scheme would be implemented during the latter part of the transfer using actual values of the orbital elements. This would also eliminate any errors accumulated during the earlier part of the transfer.

A recommendation to further this study is to add the change in the argument of periapsis, Δw, to the constraint relationship. This would allow for control of ω during transfers between planar eccentric orbits that is not included by this analysis. In addition, other approaches to optimizing the eccentric transfer problem should be considered in hopes of achieving better performance. For example, the development of an optimal control law to provide specified changes in perigee, apogee, and the argument of periapsis. However, successful completion of such analyses would depend on the complexity of the slow timescale problem.

Appendix A: Performance Analysis of Proposed Electrical Propulsion Systems for an Earth Orbiting Transfer Vehicle

The following analysis utilizes information provided in a paper presented to the 1985 Joint Army-Navy-NASA-Air Force Propulsion Meeting by Smith and Knowles discussing studies of electrical propulsion systems (10:457,463-467). Supplemental information on systems currently in development was provided by Rocket Research Company and two leading electric propulsion research centers: NASA Lewis Research Center for ion systems and the Air Force Astronautics Laboratory for arcjet systems.

The purpose of this analysis is to estimate the maximum acceleration placed on the transfer vehicle system (TVS) and the maximum percent change in mass of the TVS over one orbit. The analysis considers a typical transfer vehicle mission of delivering a payload from low earth orbit (LEO ~ 300 km) to geosynchronous orbit (GEO ~ 35,863 km), then returning to LEO in preparation for the next mission. For comparison, two types of electrical propulsion systems are considered for use by the transfer vehicle: an Ammonia (N_2H_4) arcjet system and a Xenon (Xe) ion system.

The Solar Array Flight Experiment (SAFE) flown on STS-41D has demonstrated that a solar power system can be built to provide a mass-to-power ratio of approximately 15 kg/kW (7). Based on current technology (8; 12), the arcjet propulsion system could be equipped with one 27 kW N_2H_4 thruster having a mass of approximately 4 kg, specific impulse (I_{SP}), of 863 sec, and an efficiency of 37%. With the given performance parameters, this thruster delivers 3.367 N of thrust (10:456,Table IV). It may be possible for the power conditioning unit (PCU) associated with this thruster to have an efficiency of 95% and a mass as little as 15 kg. Cooling of the PCU could be performed by heat exchange with the propellant, eliminating the need for radiators. Thus, assuming 10% of the available power is needed for supporting electronics and losses in the PCU, the total power requirement for the arcjet transfer vehicle is 30 kW. This makes the mass of the supporting power system approximately 450 kg. Allowing an additional 6 kg for connecting hardware, the total propulsion system mass is 25 kg and the dry mass of arcjet transfer vehicle, excluding fuel tanks and structure, becomes 475 kg.

For the transfer vehicle equipped with a Xe ion propulsion system, five 10.87 KW thrusters will be considered, each with an I_{SP} of 4267 sec and an efficiency of 75%, providing 389 mN thrust (6). With the associated power processing units (PPU) and interface hardware, the mass of this propulsion system is approximately 506 kg. Again, no radiators are considered, since the PPUs are assumed to provide adequate radiation exchange. With an additional 10% for electronics and PPU losses, the ion transfer vehicle requires 60 kW, making the power system mass approximately 900 kg. Thus, the dry mass of the ion transfer vehicle is approximately 1406 kg, again excluding fuel tanks and structure.

<u>TVS Mass Calculations.</u> To calculate the desired parameters we must first estimate the vehicle mass breakdown. For the first leg of the mission, the initial mass of the TVS leaving LEO is given by

$$m_I = m_{P_{GEO}} + m_{P_{LEO}} + m_L + m_V \qquad \text{(A-1)}$$

where

$m_{P_{GEO}}$ ≡ Mass of Propellant required for transfer to GEO

$m_{P_{LEO}}$ ≡ Mass of Propellant required for return transfer to LEO

m_L ≡ Mass of Payload

m_V ≡ Mass of Transfer Vehicle (propulsion system, power system, fuel tanks, structure, and electronics package)

After releasing the payload at GEO, at the beginning of the return leg to LEO, the total mass is

$$m_R = m_{P_{LEO}} + m_V \qquad \text{(A-2)}$$

The propellant masses can be found from the relation (11:137)

$$\Delta v = I_{SP}\, g_c \ln\left(\frac{m_T}{m_T - m_P}\right) \qquad \text{(A-3)}$$

where g_c is the gravitational constant (9.81 m/s^2) and Δv is the total change in velocity required between the two orbits. This velocity change can be obtained using the Edelbaum approximation (10:463) for a continuous spiral trajectory between two orbits

$$\Delta v = \sqrt{v_1^2 + v_2^2 - 2 v_1 v_2 \cos\left(\frac{\pi\ \Delta i}{2}\right)} \qquad \text{(A-4)}$$

which, for planar transfers (Δi=0), simplifies to

$$\Delta v = v_1 - v_2 \qquad \text{(A-5)}$$

However, the total velocity change required using the constrained radii control law discussed by this study is nearly 1.75 times higher than that given by the spiral transfer. Thus, the total Δv will be increased by this factor in further calculations.

The velocity of the transfer vehicle in circular orbit about the Earth is given by (2:34)

$$v = \sqrt{\frac{\mu_\oplus}{r_\oplus + h}} \qquad \text{(A-6)}$$

where

$$\mu_\oplus = 3.986012 \times 10^5\ \frac{km^3}{sec^2}$$

$$r_\oplus = 6378.145\ km$$

$$h \equiv \text{altitude}$$

Using the values of the altitude at LEO and GEO, the velocities are

$$v_{LEO} = 7725.76 \text{ m/s}$$

$$v_{GEO} = 3071.86 \text{ m/s}$$

and the magnitude of the velocity change required to transfer between LEO and GEO, including the scale factor for the constrained radii transfer, is

$$|\Delta v| = 1.75\ |v_{LEO} - v_{GEO}|$$

$$= 8144.33 \text{ m/s} \qquad (4653.90 \text{ m/s})$$

where the value in parenthesis is for the spiral transfer.

Solving equation (A-3) for the mass of the propellant

$$m_P = m_T \left[1 - \exp\left(\frac{-|\Delta v|}{I_{SP}\ g_c}\right) \right] \qquad \text{(A-7)}$$

Rewriting this equation in terms of the masses at the beginning of the initial transfer to GEO

$$m_{P_{GEO}} = m_I \left[1 - \exp\left(\frac{-|\Delta v|}{I_{SP}\ g_c}\right) \right] \qquad \text{(A-8)}$$

Assuming an initial mass of 5000 kg for both the ion and arcjet TVS, this equation yields

$$m_{p_{GEO}}^{arc} = 3089 \text{ kg} \qquad (2115 \text{ kg})$$

$$m_{p_{GEO}}^{ion} = 884 \text{ kg} \qquad (526 \text{ kg})$$

If an additional 10.5% of the propellant mass is assumed to be needed for the fuel tanks and structure of the arcjet propulsion package, and 14.5% for the ion, the transfer vehicle dry masses increase to

$$m_{v_1}^{arc} = 475 \text{ kg} + .105 \ (3089 \text{ kg})$$

$$= 799 \text{ kg} \qquad (697 \text{ kg})$$

$$m_{v_1}^{ion} = 1406 \text{ kg} + .145 \ (884 \text{ kg})$$

$$= 1534 \text{ kg} \qquad (1482 \text{ kg})$$

The difference between the two mass fractions arises due to the additional tank structure required to support the pressurized Xe.

Rewriting equation (A-7) in terms of the propellant required for the return trip to LEO and substituting equation (A-2) yields

$$m_{p_{LEO}} = m_R \left[1 - \exp\left(\frac{-|\Delta v|}{I_{SP}\ g_c}\right) \right]$$

$$= \left[m_{p_{LEO}} + m_v \right] \left[1 - \exp\left(\frac{-|\Delta v|}{I_{SP}\ g_c}\right) \right]$$

$$= m_v \left[\exp\left(\frac{|\Delta v|}{I_{SP} g_c}\right) - 1 \right] \qquad \text{(A-9)}$$

Since the additional tank mass required to support the return fuel has not yet been included in the vehicle mass, equation A-9 must be rewritten as

$$m_{p_{LEO}} = \left[m_{v_1} + x\, m_{p_{LEO}} \right] \left[\exp\left(\frac{|\Delta v|}{I_{sp} g_c}\right) - 1 \right]$$

$$= \frac{m_{v_1} \left[\exp\left(\frac{|\Delta v|}{I_{sp} g_c}\right) - 1 \right]}{\left[1 - x \left[\exp\left(\frac{|\Delta v|}{I_{sp} g_c}\right) - 1 \right] \right]} \qquad \text{(A-10)}$$

where the parameter x is equal to .105 for the arcjet system and .145 for the ion system.

Solving equation (A-10) for each system provides return propellant masses of

$$m_{p_{LEO}}^{arc} = 1557 \text{ kg} \qquad (553 \text{ kg})$$

$$m_{p_{LEO}}^{ion} = 340 \text{ kg} \qquad (177 \text{ kg})$$

making the final total transfer vehicle dry masses

$$m_v^{arc} = 799 \text{ kg} + .105\ (1557 \text{ kg})$$

$$= 963 \text{ kg} \qquad (755 \text{ kg})$$

$$m_v^{ion} = 1534 \text{ kg} + .145\ (340 \text{ kg})$$

$$= 1584 \text{ kg} \qquad (1508 \text{ kg})$$

The only unknown masses remaining are the those of the payloads. Solving equation (A-1) for the payload mass

$$m_L = m_I - m_{P_{GEO}} - m_{P_{LEO}} - m_V \qquad (A-11)$$

Substituting for the arcjet system

$$m_L^{arc} = -609 \text{ kg} \qquad (1577 \text{ kg})$$

and the ion system

$$m_L^{ion} = 2192 \text{ kg} \qquad (2789 \text{ kg})$$

Thus, it is obvious that the constrained radii transfer, with its much larger Δv requirement, cannot be performed by the arcjet TVS.

Table (A-1) provides a summary of the final breakdown of mass for each of the two 5000 kg transfer vehicle systems. With these quantities known, the maximum percent change in the vehicle mass over one orbit can now be estimated. This maximum change should occur at GEO, after the payload has been released, and the transfer vehicle is beginning its return trip to LEO. At this point, the orbital period is maximum (24 hr) and the TVS total mass reduced to the mass of the transfer vehicle and the fuel required for the return trip.

The mass expelled from the TVS over this orbit can be calculated knowing the mass flow rate (constant) and the orbital period ($\mathbb{TP}$). The mass flow rate of each system is given by (11:29)

$$\dot{m} = \frac{F}{g_c \, I_{SP}} \qquad (A-12)$$

where F is the total vehicle thrust. Using this equation, the mass flow rate of each TVS is

$$\dot{m}^{arc} = \frac{3.367\ N}{(9.81\ m/s^2)(863\ sec)} = .398 \times 10^{-3}\ kg/sec$$

$$\dot{m}^{ion} = \frac{5\ (389\ mN)}{(9.81\ m/s^2)(4267\ sec)} = .046 \times 10^{-3}\ kg/sec$$

TVS MASS BREAKDOWN	TRANSFER VEHICLE	
	ARCJET	ION
Transfer Vehicle Mass	969 kg (755 kg)	1584 kg (1508 kg)
Fuel Mass (Transfer to GEO)	* (2115 kg)	884 kg (526 kg)
Fuel Mass (Return to LEO)	* (553 kg)	340 kg (177 kg)
Payload Mass	* (1577 kg)	2192 kg (2789 kg)
Total Mass	5000 kg	5000 kg

Table A-1. Transfer Vehicle System Mass Distribution required for a Constrained Radii of Perigee and Apogee Transfer

Note: () refer to quantities for a spiral transfer

* Arcjet transfer vehicle incapable of mission using constrained radii control law

The change in mass (mass expelled) over one orbit can be obtained using

$$\Delta m = \dot{m}\,\mathbb{TP} \qquad \text{(A-13)}$$

where $\mathbb{TP}$ is the period of one orbit. Thus, for each propulsion system

$$\Delta m^{arc} = \dot{m}^{arc}\,\mathbb{TP} = (.000398\ \text{kg/sec})(24\ \text{hr})(3600\ \text{sec/hr})$$

$$= 34.36\ \text{kg}$$

$$\Delta m^{ion} = (.000046\ \text{kg/sec})(24\ \text{hr})(3600\ \text{sec/hr})$$

$$= 4.01\ \text{kg}$$

giving a final estimate of the maximum percent change in masses over one orbit as (spiral transfer only for arcjet TVS)

$$\frac{\Delta m^{arc}}{m_R}(100\%) = \frac{34.36\ \text{kg}}{(1308\ \text{kg})}(100\%) = (2.63\%)$$

$$\frac{\Delta m^{ion}}{m_R}(100\%) = \frac{4.01\ \text{kg}}{1924\ \text{kg}}(100\%) = .21\% \quad (.24\%)$$

Maximum Acceleration Calculations. The acceleration of the TVS is given by

$$A = \frac{F}{m} \qquad \text{(A-14)}$$

For comparison, we will calculate the anticipated maximum acceleration on the TVS at both LEO and GEO.

LEO. The maximum acceleration experienced by the TVS during the mission occurs upon final return of the TVS to LEO. At this point, since all fuel has been consumed, the TVS's mass has been reduced to that of the transfer vehicle dry weight. Thus, the instant the last of the fuel is consumed, the acceleration on the arcjet system (spiral transfer only) is

$$A^{arc}_{max} = \left[\frac{F}{m_V} \right]^{arc} = \frac{3.367 \text{ N}}{(755 \text{ kg})} = (.00446 \text{ m/s}^2)$$

$$= (.45 \times 10^{-3} \text{ g's})$$

and the ion system

$$A^{ion}_{max} = \left[\frac{F}{m_V} \right]^{ion} = \frac{5 \ (389 \text{ mN})}{1584 \text{ kg}} = .00123 \text{ m/s}^2 \quad (.00129 \text{ m/s}^2)$$

$$= .13 \times 10^{-3} \text{ g's} \quad (.13 \times 10^{-3} \text{ g's})$$

GEO. The largest acceleration experienced at GEO occurs just after the release of the payload, when the TVS begins the final return leg of the mission. For the arcjet propulsion system, this acceleration (spiral only) is

$$A^{arc}_{max} = \left[\frac{F}{m_R} \right]^{arc} = \frac{3.367 \text{ N}}{(1308 \text{ kg})} = (.00257 \text{ m/s}^2)$$

$$= (.26 \times 10^{-3} \text{g's})$$

and the ion system

$$A^{ion}_{max} = \left[\frac{F}{m_R} \right]^{ion} = \frac{5 \ (389 \text{ mN})}{1924 \text{ kg}} = .00101 \text{ m/s}^2 \quad (.00115 \text{ m/s}^2)$$

$$= .10 \times 10^{-3} \text{ g's} \quad (.12 \times 10^{-3} \text{ g's})$$

<u>TVS Transfer Times.</u> For a continuous thrust propulsion system, the mass flow rate is constant allowing the transfer time to be calculated using the relation

$$t = \frac{m_{exp}}{\dot{m}} \qquad \text{(A-15)}$$

where m_{exp} is the total mass expelled. Thus, the transfer times for each TVS can be estimated by simply dividing the total fuel mass expelled on each leg of the mission (LEO to GEO, and return to LEO) by the total mass flow rate. This results transfer times to GEO of

$$\left(t_{GEO}\right)^{arc} = \left(\frac{m_{p_{GEO}}}{\dot{m}}\right)^{arc}$$

$$= \frac{(2114\ \text{kg})}{.000398\ \text{kg/sec}} \left[\frac{1\ \text{hr}}{3600\ \text{sec}}\right]\left[\frac{1\ \text{day}}{24\ \text{hrs}}\right]$$

$$= (61.53\ \text{days, spiral only})$$

$$\left(t_{GEO}\right)^{ion} = \left(\frac{m_{p_{GEO}}}{\dot{m}}\right)^{ion}$$

$$= \frac{884\ \text{kg}}{.000046\ \text{kg/sec}} \left[\frac{1\ \text{hr}}{3600\ \text{sec}}\right]\left[\frac{1\ \text{day}}{24\ \text{hrs}}\right]$$

$$= 220.21\ \text{days} \qquad (131.05\ \text{days})$$

and return to LEO

$$\left(t_{LEO}\right)^{arc} = \left[\frac{m_{p_{LEO}}}{\dot{m}}\right]^{arc}$$

$$= \frac{(553\ kg)}{.000398\ kg/sec}\left[\frac{1\ hr}{3600\ sec}\right]\left[\frac{1\ day}{24\ hrs}\right]$$

$$= (16.10\ days,\ spiral\ only)$$

$$\left(t_{LEO}\right)^{ion} = \left[\frac{m_{p_{LEO}}}{\dot{m}}\right]^{ion}$$

$$= \frac{340\ kg}{.000046\ kg/sec}\left[\frac{1\ hr}{3600\ sec}\right]\left[\frac{1\ day}{24\ hrs}\right]$$

$$= 84.72\ days \qquad (44.17\ days)$$

Summary of Results. Table (A-2) provides an outline of the propulsion system and vehicle performance parameters, mass distribution, and transfer times for the N_2H_4 arcjet and Xe ion transfer vehicles. As a comparison, this data includes the results of utilizing the constrained radii of apogee/perigee transfer and the spiral transfer.

SYSTEM** PARAMETERS	CONSTRAINED RADII		SPIRAL	
	arcjet	ion	arcjet	ion
Thruster Perform.				
Specific Impulse	863 sec	4267 sec	863 sec	4267 sec
Efficiency	35 %	75 %	35 %	75 %
Power/Thruster	26.0 kW	10.87 kW	26.0 kW	10.87 kW
Thrust/Thruster	3.367 N	389 mN	3.367 N	389 mN
# of Thrusters	1	5	1	5
Power System				
Mass/Power	15 kg/kW	15 kg/kW	15 kg/kW	15 kg/kW
Mass Distribution				
Propulsion Sys	25 kg	506 kg	25 kg	506 kg
Power System	450 kg	900 kg	450 kg	900 kg
Tanks/Structure	*	178 kg	280 kg	102 kg
Fuel	*	1224 kg	2668 kg	703 kg
Payload	*	2192 kg	1577 kg	2789 kg
Total	*	5000 kg	5000 kg	5000 kg
Max %Δm/Orbit	*	0.21 %	2.63 %	0.24 %
Mass Flow Rate (km/sec)	.000398	.000046	.000398	.000046
Max Acceleration				
LEO (g's)	*	.00013	.00045	.00013
GEO (g's)	*	.00010	.00026	.00012
Transfer Time				
Leg 1: LEO-GEO	*	220 days	62 days	131 days
Leg 2: GEO-LEO	*	85 days	16 days	44 days

Table A-2. Performance Parameters and Mass Distribution of Ammonia Arcjet and Xenon Ion Transfer Vehicles for Constrained Radii and Spiral Tranfers.

* Arcjet TVS incapable of constrained radii transfer

** Total mission includes transfer of payload from LEO (300 km) to GEO (35,863 km) and return of "empty" transfer vehicle to LEO.

Appendix B-1: Program DELAMAX2

```
C     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C     *                                                               *
C     *                       PROGRAM DELAMAX2                        *
C     *                                                               *
C     *                     Capt Gregory Beeker                       *
C     *              Air Force Institute of Technology                *
C     *                          May 1988                             *
C     *                                                               *
C     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
C      The following program solves for the maximum change in the
C      magnitude of the nondimensional semimajor axis for the contin-
C      uous thrust orbit transfer problem in which the distance to
C      perigee (Rp) or apogee (Ra) is held fixed.  Simpson's rule
C      is used in subroutine integrate to solve the integral
C      equations for the change in the nondimensional semimajor
C      axis (delta a*) and eccentricity (delta e*) given values of
C      a, e, nu, and the Lagrange multiplier, lambda.  The secant
C      method is used to find the value of lambda which will drive
C      the value of delta-Rp or delta-Rp to zero.
C
C
      IMPLICIT DOUBLE PRECISION (A-H,L-Z)
      DIMENSION ALPHAI(361),NUI(361)
      COMMON E,DELR,DELA,DELE,LAMBDA,IDIV,SIGN,ALPHAI,NUI
C
C     * INPUT INITIAL DATA *
C     **********************
C
      WRITE (*,*) 'Please enter the initial and final values'
      WRITE (*,*) 'of the eccentricity, e, dear.'
      READ (*,*)  E0,EF
      WRITE (*,*) 'Enter the number of steps between the initial'
      WRITE (*,*) 'and final values of eccentricity.'
      READ (*,*) INUME
      WRITE (*,*) 'Do you wish to keep the distance to '
      WRITE (*,*) 'apogee constant, or the distance to'
      WRITE (*,*) 'perigee constant ?'
      WRITE (*,*) '(Type: -1.0 for perigee; 1.0 for apogee)'
      READ (*,*) SIGN
      WRITE (*,*) 'Enter the number of pieces Subroutine'
      WRITE (*,*) 'Integrate is to dissect the integals into.'
      WRITE (*,*) '(Must be an even number, 360 maximum)'
      READ (*,*) IDIV
      WRITE (*,*) 'Enter the initial and second guess of the '
      WRITE (*,*) 'parameter lambda.'
      READ (*,*) LAMBDA0,LAMBDA1
```

```
      WRITE (*,*) 'Enter the maximum number of iterations to be'
      WRITE (*,*) 'performed by secant method routine.'
      READ (*,*) IMAX
      WRITE (*,*) 'Enter the tolerance of delta-Ra/p.'
      READ (*,*) TOLR
      WRITE (*,*) 'What type of print out do you wish?'
      WRITE (*,*) '(0 for document data, 1 for plot data)'
      READ (*,*) IPNT
      WRITE (*,*) 'If document data was selected, do you wish'
      WRITE (*,*) 'to print the thrust angle around the orbit?'
      WRITE (*,*) '(0 - no, 1 - yes)'
      READ (*,*) ITANG
C
C     * Print Initial Data *
C     **********************
C
      IF ((IPNT.EQ.1).AND.(SIGN.LE.0.)) THEN
        OPEN (UNIT=11, STATUS = 'NEW',FILE = 'APLAM.DAT')
        OPEN (UNIT=12, STATUS = 'NEW',FILE = 'APDELA.DAT')
        OPEN (UNIT=13, STATUS = 'NEW',FILE = 'APDELE.DAT')
        ENDIF
C
      IF ((IPNT.EQ.1).AND.(SIGN.GT.0.)) THEN
        OPEN (UNIT=11, STATUS = 'NEW',FILE = 'AALAM.DAT')
        OPEN (UNIT=12, STATUS = 'NEW',FILE = 'AADELA.DAT')
        OPEN (UNIT=13, STATUS = 'NEW',FILE = 'AADELE.DAT')
        ENDIF
C
      IF (IPNT.EQ.0) THEN
        OPEN (UNIT=10, STATUS = 'NEW',FILE = 'AMAX2.OUT')
        WRITE (10,12) IMAX
        WRITE (10,15) IDIV
        WRITE (10,*)
        IF (SIGN.GT.0) WRITE (10,18) TOLR
        IF (SIGN.LT.0) WRITE (10,19) TOLR
        WRITE (10,*)
        WRITE (10,25)
        ENDIF
C
      WRITE (6,12) IMAX
      WRITE (6,15) IDIV
      WRITE (6,*)
      IF (SIGN.GT.0) WRITE (6,18) TOLR
      IF (SIGN.LT.0) WRITE (6,19) TOLR
      WRITE (6,*)
      WRITE (6,25)
C
   12 FORMAT (3X,'Max # of iterations to be performed ',
     +            'by Secant Method Routine is ',I4)
   15 FORMAT (3X,'Number of divisions for each integral is ',I4)
   18 FORMAT (3X,'Apogee Distance Fixed with a tolerance of ',E8.1)
   19 FORMAT (3X,'Perigee Distance Fixed with a tolerance of ',E8.1)
```

```
   25 FORMAT (6X,'e',13X,'Lambda',16X,'Delta a*',14X,'Delta e*')
C
      ESTEP=(EF-E0)/INUME
      E=E0
C
C     * BEGIN INITIAL LOOP WHICH INCREMENTS ECCENTRICITY *
C     ****************************************************
C
      LAMBDA=LAMBDA0
      CALL INTEGRATE
      IF (ABS(DELR).LE.TOLR) GO TO 45
      DELRM1=DELR
      LAMDM1=LAMBDA
      LAMBDA=LAMBDA1
   30 CALL INTEGRATE
      IF (ABS(DELR).LE.TOLR) GO TO 45
C
C     * BEGIN ITERATIONS OF SECANT METHOD *
C     *************************************
C
      DO 40 I=1,IMAX
        LAMDP1=LAMBDA-DELR*(LAMBDA-LAMDM1)/(DELR-DELRM1)
        DELRM1=DELR
        LAMDM1=LAMBDA
        LAMBDA=LAMDP1
        CALL INTEGRATE
        IF (ABS(DELR).LE.TOLR) GO TO 45
   40   CONTINUE
C
      IF (IPNT.EQ.0) THEN
        WRITE (10,*)
        WRITE (10,42) E,TOLR
        WRITE (10,*)
        ENDIF
C
      WRITE (6,*)
      WRITE (6,42) E,TOLR
      WRITE (6,*)
C
   42 FORMAT (3X,'e = ',F8.4,3X,'Secant Method did not converge',
     +           ' within given tolerance of ',E15.7)
      GO TO 60
C
C
C     * PRINT FINAL DATA *
C     ********************
C
   45 WRITE (6,48) E,LAMBDA,DELA,DELE
      IF (IPNT.EQ.0) WRITE (10,48) E,LAMBDA,DELA,DELE
      IF (IPNT.EQ.1) THEN
        WRITE (11,49) E,LAMBDA
        WRITE (12,49) E,DELA
```

```
        WRITE (13,49) E,DELE
        ENDIF
      IF ((IPNT.EQ.0).AND.(ITANG.EQ.1)) THEN
        WRITE (6,*)
        WRITE (10,*)
        WRITE (6,50)
        WRITE (10,50)
        WRITE (6,*)
        WRITE (10,*)
        DO 46 I=1,(IDIV+1)
          WRITE (6,51) NUI(I), ALPHAI(I)
          WRITE (10,51) NUI(I), ALPHAI(I)
 46       CONTINUE
        WRITE (6,*)
        WRITE (10,*)
        ENDIF
C
 48   FORMAT (3X,F6.4,3X,E20.13,3X,E20.13,3X,E20.13)
 49   FORMAT (3X,E20.13,3X,E20.13)
 50   FORMAT (7X,'NU',8X,'ALPHA')
 51   FORMAT (3X,F7.2,5X,F7.2)
C
      DIFF=EF-E
      IF (ABS(DIFF).LE.1E-10) GO TO 60
C
      E=E+ESTEP
      GO TO 30
C
 60   STOP
      END
C
C
C
C
C
C     **********************************************************
C                          SUBROUTINE INTEGRATE
C     **********************************************************
C
      SUBROUTINE INTEGRATE
      IMPLICIT DOUBLE PRECISION (A-H,L-Z)
      DIMENSION ALPHAI(361),NUI(361)
      COMMON E,DELR,DELA,DELE,LAMBDA,IDIV,SIGN,ALPHAI,NUI
C
      NU=0.
      AINT1=0.0
      AINT2=0.0
      EINT=0.0
      ESQU=1.-E**2
      PI=DBLE(ACOS(-1.0))
C     WRITE (*,*) PI
      DELNU=2.*PI/IDIV
```

```
      F1=2.*E+LAMBDA*(2.*E*(1.+SIGN*E)+SIGN*ESQU)
      F2=SIGN*LAMBDA*ESQU**2
C     WRITE (*,*) F1,F2
C
      DO 100  I=1, (IDIV+1)
        CNU=1.+E*DCOS(NU)
        SNU=DSIN(NU)
C
C       * CALCULATE THRUST VECTOR ANGLE *
C       *********************************
C
        NUMER=E*SNU*CNU*F1
        DENOM=CNU**2*F1-F2
C       WRITE (*,*) NUMER,DENOM
C
        IF ((DABS(NUMER).LE.1.E-15).AND.(DABS(DENOM)
     +         .LE.1.E-15)) THEN

          WRITE (6,80)
          IF (IPNT.EQ.0) WRITE (10,80)
   80     FORMAT (3X,'The denominator argument of the',
     +            'ARCTAN function was equal to or near 0.')
          WRITE (6,85) NUMER,DENOM,F1,F2,NU
          IF (IPNT.EQ.0) WRITE (10,85) NUMER,DENOM,F1,F2,NU
   85     FORMAT (3X,'NUM = ',E15.8,3X,'DEN = ',E15.8,3X,
     +            'F1 = ',E15.8,3X,'F2 = ',E15.8,3X,
     +            'NU = ',E15.8,' Rad')
          ALPHA=0.0
          GO TO 88
          ENDIF
C
        ALPHA=DATAN2(NUMER,DENOM)
   88   ALPHAI(I)=180.*ALPHA/PI
        NUI(I)=180.*NU/PI
        IF (ALPHAI(I).LT.0.) ALPHAI(I)=360.+ALPHAI(I)
        CALPHA=DCOS(ALPHA)
        SALPHA=DSIN(ALPHA)
C
C
C       * CALCULATE INTEGRALS OF DELTA A AND DELTA E *
C       **********************************************
C
        CONST=2.0
        IF ((I/2*2).EQ.I) CONST=4.0
        IF ((I.EQ.1).OR.(I.EQ.(IDIV+1))) CONST=1.0
        AINT1=AINT1+CONST*SNU/CNU**2*SALPHA
        AINT2=AINT2+CONST*CALPHA/CNU
        EINT=EINT+CONST*CALPHA/CNU**3
C
        NU=NU+DELNU
C
  100   CONTINUE
```

```
C
C
      AINT1=DELNU/3.*AINT1
      AINT2=DELNU/3.*AINT2
      EINT=DELNU/3.*EINT
C
C     * CALCULATE VALUES OF DELTA-A, DELTA-E, AND DELTA-Ra/p *
C     ********************************************************
C
      DELA=2.*ESQU*(E*AINT1+AINT2)
      DELE=ESQU/2./E*DELA-ESQU**3/E*EINT
      DELR=DELA*(1.+SIGN*E)+SIGN*DELE
C
      RETURN
      END
```

Appendix B-2: Program DELEMAX2

```
C     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C     *                                                               *
C     *                      PROGRAM DELEMAX2                         *
C     *                                                               *
C     *                    Capt Gregory Beeker                        *
C     *              Air Force Institute of Technology                *
C     *                         May 1988                              *
C     *                                                               *
C     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
C      The following program solves for the maximum change in the
C      magnitude of the nondimensional eccentricity for the contin-
C      uous thrust orbit transfer problem in which the distance to
C      perigee (Rp) or apogee (Ra) is held fixed.  Simpson's rule
C      is used in subroutine integrate to solve the integral
C      equations for the change in the nondimensional semimajor
C      axis (delta a*) and eccentricity (delta e*) given values of
C      a, e, nu, and the nondimensional Lagrange multiplier, lambda*.
C      The secant method is used to find the value of lambda* which
C      will drive the value of delta-Rp or delta-Rp to zero.
C
C
      IMPLICIT DOUBLE PRECISION (A-H,L-Z)
      DIMENSION ALPHAI(361),NUI(361)
      COMMON E,DELR,DELA,DELE,LAMBDA,IDIV,SIGN,ALPHAI,NUI
C
C     * INPUT INITIAL DATA *
C     **********************
C
      WRITE (*,*) 'Please enter the initial and final values'
      WRITE (*,*) 'of the eccentricity, e, dear.'
      READ (*,*)  EO,EF
      WRITE (*,*) 'Enter the number of steps between the initial'
      WRITE (*,*) 'and final values of eccentricity.'
      READ (*,*) INUME
      WRITE (*,*) 'Do you wish to keep the distance to '
      WRITE (*,*) 'apogee constant, or the distance to'
      WRITE (*.*) 'perigee constant ?'
      WRITE (*,*) '(Type: -1.0 for perigee; 1.0 for apogee)'
      READ (*,*) SIGN
      WRITE (*,*) 'Enter the number of pieces Subroutine'
      WRITE (*,*) 'Integrate is to dissect the integrals into.'
      WRITE (*,*) '(Must be an even number, 360 maximum)'
      READ (*,*) IDIV
      WRITE (*,*) 'Enter the initial and second guess of the '
      WRITE (*,*) 'parameter lambda* (Lagrange multiplier,'
      WRITE (*,*) 'lambda, times the semimajor axis, a).'
      READ (*,*) LAMBDAO,LAMBDA1
```

```
      WRITE (*,*) 'Enter the maximum number of iterations to be'
      WRITE (*,*) 'performed by secant method routine.'
      READ (*,*) IMAX
      WRITE (*,*) 'Enter the tolerance of delta-Ra/p.'
      READ (*,*) TOLR
      WRITE (*,*) 'What type of print out do you wish?'
      WRITE (*,*) '(0 for document data, 1 for plot data)'
      READ (*,*) IPNT
      WRITE (*,*) 'If document data was selected, do you wish'
      WRITE (*,*) 'to print the thrust angle around the orbit?'
      WRITE (*,*) '(0 - no, 1 - yes)'
      READ (*,*) ITANG
C
C     * Print Initial Data *
C     **********************
C
      IF ((IPNT.EQ.1).AND.(SIGN.LE.0.)) THEN
        OPEN (UNIT=11, STATUS = 'NEW',FILE = 'PLAM.DAT')
        OPEN (UNIT=12, STATUS = 'NEW',FILE = 'PDELA.DAT')
        OPEN (UNIT=13, STATUS = 'NEW',FILE = 'PDELE.DAT')
        ENDIF
C
      IF ((IPNT.EQ.1).AND.(SIGN.GT.0.)) THEN
        OPEN (UNIT=11, STATUS = 'NEW',FILE = 'ALAM.DAT')
        OPEN (UNIT=12, STATUS = 'NEW',FILE = 'ADELA.DAT')
        OPEN (UNIT=13, STATUS = 'NEW',FILE = 'ADELE.DAT')
        ENDIF
C
      IF (IPNT.EQ.0) THEN
        OPEN (UNIT=10, STATUS = 'NEW',FILE = 'EMAX2.OUT')
        WRITE (10,12) IMAX
        WRITE (10,15) IDIV
        WRITE (10,*)
        IF (SIGN.GT.0) WRITE (10,18) TOLR
        IF (SIGN.LT.0) WRITE (10,19) TOLR
        WRITE (10,*)
        WRITE (10,25)
        ENDIF
C
      WRITE (6,12) IMAX
      WRITE (6,15) IDIV
      WRITE (6,*)
      IF (SIGN.GT.0) WRITE (6,18) TOLR
      IF (SIGN.LT.0) WRITE (6,19) TOLR
      WRITE (6,*)
      WRITE (6,25)
C
   12 FORMAT (3X,'Max # of iterations to be performed ',
     +            'by Secant Method Routine is ',I4)
   15 FORMAT (3X,'Number of divisions for each integral is ',I4)
   18 FORMAT (3X,'Apogee Distance Fixed with a tolerance of ',E8.1)
   19 FORMAT (3X,'Perigee Distance Fixed with a tolerance of ',E8.1)
```

```
   25 FORMAT (6X,'e',13X,'Lambda*',16X,'Delta a*',14X,'Delta e*')
C
      ESTEP=(EF-E0)/INUME
      E=E0
C
C     * BEGIN INITIAL LOOP WHICH INCREMENTS ECCENTRICITY *
C     ****************************************************
C
      LAMBDA=LAMBDA0
      CALL INTEGRATE
      IF (ABS(DELR).LE.TOLR) GO TO 45
      DELRM1=DELR
      LAMDM1=LAMBDA
      LAMBDA=LAMBDA1
   30 CALL INTEGRATE
      IF (ABS(DELR).LE.TOLR) GO TO 45
C
C     * BEGIN ITERATIONS OF SECANT METHOD *
C     *************************************
C
      DO 40 I=1,IMAX
        LAMDP1=LAMBDA-DELR*(LAMBDA-LAMDM1)/(DELR-DELRM1)
        DELRM1=DELR
        LAMDM1=LAMBDA
        LAMBDA=LAMDP1
        CALL INTEGRATE
        IF (ABS(DELR).LE.TOLR) GO TO 45
   40   CONTINUE
C
      IF (IPNT.EQ.0) THEN
        WRITE (10,*)
        WRITE (10,42) E,TOLR
        WRITE (10,*)
        ENDIF
C
      WRITE (6,*)
      WRITE (6,42) E,TOLR
      WRITE (6,*)
C
   42 FORMAT (3X,'e = ',F8.4,3X,'Secant Method did not converge',
     +           ' within given tolerance of ',E15.7)
      GO TO 60
C
C
C     * PRINT FINAL DATA *
C     ********************
C
   45 WRITE (6,48) E,LAMBDA,DELA,DELE
      IF (IPNT.EQ.0) WRITE (10,48) E,LAMBDA,DELA,DELE
      IF (IPNT.EQ.1) THEN
        WRITE (11,49) E,LAMBDA
        WRITE (12,49) E,DELA
```

```
          WRITE (13,49) E,DELE
          ENDIF
        IF ((IPNT.EQ.0).AND.(ITANG.EQ.1)) THEN
          WRITE (6,*)
          WRITE (10,*)
          WRITE (6,50)
          WRITE (10,50)
          WRITE (6,*)
          WRITE (10,*)
          DO 46 I=1,(IDIV+1)
            WRITE (6,51) NUI(I), ALPHAI(I)
            WRITE (10,51) NUI(I), ALPHAI(I)
   46       CONTINUE
          WRITE (6,*)
          WRITE (10,*)
          ENDIF
C
   48 FORMAT (3X,F6.4,3X,E20.13,3X,E20.13,3X,E20.13)
   49 FORMAT (3X,E20.13,3X,E20.13)
   50 FORMAT (7X,'NU',8X,'ALPHA')
   51 FORMAT (3X,F7.2,5X,F7.2)
C
      DIFF=EF-E
      IF (ABS(DIFF).LE.1E-10) GO TO 60
C
      E=E+ESTEP
      GO TO 30
C
   60 STOP
      END
C
C
C
C
C
C     **********************************************************
C                          SUBROUTINE INTEGRATE
C     **********************************************************
C
      SUBROUTINE INTEGRATE
      IMPLICIT DOUBLE PRECISION (A-H,L-Z)
      DIMENSION ALPHAI(361),NUI(361)
      COMMON E,DELR,DELA,DELE,LAMBDA,IDIV,SIGN,ALPHAI,NUI
C
      NU=0.
      AINT1=0.0
      AINT2=0.0
      EINT=0.0
      ESQU=1.-E**2
      PI=DBLE(ACOS(-1.0))
C     WRITE (*,*) PI
      DELNU=2.*PI/IDIV
```

```
      F1=ESQU+LAMBDA*(2.*E*(1.+SIGN*E)+SIGN*ESQU)
      F2=(1.+SIGN*LAMBDA)*ESQU**2
C     WRITE (*,*) F1,F2
C
      DO 100  I=1, (IDIV+1)
        CNU=1.+E*DCOS(NU)
        SNU=DSIN(NU)
C
C       * CALCULATE THRUST VECTOR ANGLE *
C       *********************************
C
        NUMER=E*SNU*CNU*F1
        DENOM=CNU**2*F1-F2
C       WRITE (*,*) NUMER,DENOM
C
        IF ((DABS(NUMER).LE.1.E-15).AND.(DABS(DENOM)
     +        .LE.1.E-15)) THEN

          WRITE (6,80)
          IF (IPNT.EQ.0) WRITE (10,80)
   80     FORMAT (3X,'The denominator argument of the',
     +            'ARCTAN function was equal to or near 0.')
          WRITE (6,85) NUMER,DENOM,F1,F2,NU
          IF (IPNT.EQ.0) WRITE (10,85) NUMER,DENOM,F1,F2,NU
   85     FORMAT (3X,'NUM = ',E15.8,3X,'DEN = ',E15.8,3X,
     +            'F1 = ',E15.8,3X,'F2 = ',E15.8,3X,
     +            'NU = ',E15.8,' Rad')
          ALPHA=0.0
          GO TO 88
          ENDIF
C
        ALPHA=DATAN2(NUMER,DENOM)
   88   ALPHAI(I)=180.*ALPHA/PI
        NUI(I)=180.*NU/PI
        IF (ALPHAI(I).LT.0.) ALPHAI(I)=360.+ALPHAI(I)
        CALPHA=DCOS(ALPHA)
        SALPHA=DSIN(ALPHA)
C
C
C       * CALCULATE INTEGRALS OF DELTA A AND DELTA E *
C       **********************************************
C
        CONST=2.0
        IF ((I/2*2).EQ.I) CONST=4.0
        IF ((I.EQ.1).OR.(I.EQ.(IDIV+1))) CONST=1.0
        AINT1=AINT1+CONST*SNU/CNU**2*SALPHA
        AINT2=AINT2+CONST*CALPHA/CNU
        EINT=EINT+CONST*CALPHA/CNU**3
C
        NU=NU+DELNU
C
  100   CONTINUE
```

```
C
C
      AINT1=DELNU/3.*AINT1
      AINT2=DELNU/3.*AINT2
      EINT=DELNU/3.*EINT
C
C     * CALCULATE VALUES OF DELTA-A, DELTA-E, AND DELTA-Ra/p *
C     ********************************************************
C
      DELA=2.*ESQU*(E*AINT1+AINT2)
      DELE=ESQU/2./E*DELA-ESQU**3/E*EINT
      DELR=DELA*(1.+SIGN*E)+SIGN*DELE
C
      RETURN
      END
```

Appendix B-3: Program DELAMAXUC

```
C     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C     *                                                               *
C     *                     PROGRAM DELAMAXUC                         *
C     *                                                               *
C     *                  Capt Gregory Beeker                          *
C     *            Air Force Institute of Technology                  *
C     *                      August 1988                              *
C     *                                                               *
C     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
C      The following program solves for the maximum change in the
C      magnitude of the nondimensional semimajor axis for the contin-
C      uous thrust orbit transfer problem.  Simpson's rule is used
C      in subroutine integrate to solve the integral equations for
C      the change in nondimensional semimajor axis (delta a*) and
C      eccentricity (delta e*) given values of a, e, and nu.
C
C
      IMPLICIT DOUBLE PRECISION (A-H,L-Z)
      DIMENSION ALPHAI(361),NUI(361)
      COMMON E,DELRA,DELRP,DELA,DELE,IDIV,ALPHAI,NUI
C
C     * INPUT INITIAL DATA *
C     **********************
C
      WRITE (*,*) 'Please enter the initial and final values'
      WRITE (*,*) 'of the eccentricity, e.'
      READ (*,*)  E0,EF
      WRITE (*,*) 'Enter the number of steps between the initial'
      WRITE (*,*) 'and final values of eccentricity.'
      READ (*,*) INUME
      WRITE (*,*) 'Enter the number of pieces Subroutine'
      WRITE (*,*) 'Integrate is to dissect the integrals into.'
      WRITE (*,*) '(Must be an even number, 360 maximum)'
      READ (*,*) IDIV
      WRITE (*,*) 'What type of print out do you wish?'
      WRITE (*,*) '(0 for document data, 1 for plot data)'
      READ (*,*) IPNT
      WRITE (*,*) 'If document data was selected, do you wish'
      WRITE (*,*) 'to print the thrust angle around the orbit?'
      WRITE (*,*) '(0 - no, 1 - yes)'
      READ (*,*) ITANG
C
C     * Print Initial Data *
C     **********************
C
      IF (IPNT.EQ.1) THEN
```

```
        OPEN (UNIT=12, STATUS = 'NEW',FILE = 'UCADELA.DAT')
        OPEN (UNIT=13, STATUS = 'NEW',FILE = 'UCADELE.DAT')
        OPEN (UNIT=14, STATUS = 'NEW',FILE = 'UCADELRA.DAT')
        OPEN (UNIT=15, STATUS = 'NEW',FILE = 'UCADELRP.DAT')
        ENDIF
C
      IF (IPNT.EQ.0) THEN
        OPEN (UNIT=10, STATUS = 'NEW',FILE = 'UCAMAXF.OUT')
        WRITE (10,15) IDIV
        WRITE (10,*)
        WRITE (10,25)
        ENDIF
C
      WRITE (6,15) IDIV
      WRITE (6,*)
      WRITE (6,25)
C
   15 FORMAT (3X,'Number of divisions for each integral is ',I4)
   25 FORMAT (6X,'e',13X,'Delta a*',14X,'Delta e*',14X,'Delta ra*',
     +       15X,'Delta rp*')
C
      ESTEP=(EF-E0)/INUME
      E=E0
C
C
   30 CALL INTEGRATE
C
C
C     * PRINT FINAL DATA *
C     ********************
C
   45 WRITE (6,48) E,DELA,DELE,DELRA,DELRP
      IF (IPNT.EQ.0) WRITE (10,48) E,DELA,DELE,DELRA,DELRP
      IF (IPNT.EQ.1) THEN
        WRITE (12,49) E,DELA
        WRITE (13,49) E,DELE
        WRITE (14,49) E,DELRA
        WRITE (15,49) E,DELRP
        ENDIF
C
      IF ((IPNT.EQ.0).AND.(ITANG.EQ.1)) THEN
        WRITE (6,*)
        WRITE (10,*)
        WRITE (6,50)
        WRITE (10,50)
        WRITE (6,*)
        WRITE (10,*)
        DO 46 I=1,(IDIV+1)
          WRITE (6,51) NUI(I), ALPHAI(I)
          WRITE (10,51) NUI(I), ALPHAI(I)
   46     CONTINUE
        WRITE (6,*)
```

```
          WRITE (10,*)
          ENDIF
C
   48 FORMAT (3X,F6.4,3X,E20.13,3X,E20.13,3X,E20.13,3X,E20.13)
   49 FORMAT (3X,E20.13,3X,E20.13)
   50 FORMAT (7X,'NU',8X,'ALPHA')
   51 FORMAT (3X,F7.2,5X,F7.2)
C
      DIFF=EF-E
      IF (ABS(DIFF).LE.1E-10) GO TO 60
C
      E=E+ESTEP
      GO TO 30
C
   60 STOP
      END
C
C
C
C
C
C     ***********************************************************
C                        SUBROUTINE INTEGRATE
C     ***********************************************************
C
      SUBROUTINE INTEGRATE
      IMPLICIT DOUBLE PRECISION (A-H,L-Z)
      DIMENSION ALPHAI(361),NUI(361)
      COMMON E,DELRA,DELRP,DELA,DELE,IDIV,ALPHAI,NUI
C
      NU=0.
      AINT1=0.0
      AINT2=0.0
      EINT=0.0
      ESQU=1.-E**2
      PI=DBLE(ACOS(-1.0))
C     WRITE (*,*) PI
      DELNU=2.*PI/IDIV
      DO 100  I=1, (IDIV+1)
        CNU=1.+E*DCOS(NU)
        SNU=DSIN(NU)
C
C       * CALCULATE THRUST VECTOR ANGLE *
C       *********************************
C
        NUMER=E*SNU
        DENOM=CNU
C       WRITE (*,*) NUMER,DENOM
C
        IF ((DABS(NUMER).LE.1.E-15).AND.(DABS(DENOM)
     +       .LE.1.E-15)) THEN
```

```
          WRITE (6,80)
          IF (IPNT.EQ.0) WRITE (10,80)
   80     FORMAT (3X,'The denominator argument of the',
     +            'ARCTAN function was equal to or near 0.')
          WRITE (6,85) NUMER,DENOM,NU
          IF (IPNT.EQ.0) WRITE (10,85) NUMER,DENOM,NU
   85     FORMAT (3X,'NUM = ',E15.8,3X,'DEN = ',E15.8,3X,
     +            'NU = ',E15.8,' Rad')
          ALPHA=0.0
          GO TO 88
          ENDIF
C
        ALPHA=DATAN2(NUMER,DENOM)
   88   ALPHAI(I)=180.*ALPHA/PI
        NUI(I)=180.*NU/PI
        IF (ALPHAI(I).LT.0.) ALPHAI(I)=360.+ALPHAI(I)
        CALPHA=DCOS(ALPHA)
        SALPHA=DSIN(ALPHA)
C
C
C       * CALCULATE INTEGRALS OF DELTA A AND DELTA E *
C       **********************************************
C
        CONST=2.0
        IF ((I/2*2).EQ.I) CONST=4.0
        IF ((I.EQ.1).OR.(I.EQ.(IDIV+1))) CONST=1.0
        AINT1=AINT1+CONST*SNU/CNU**2*SALPHA
        AINT2=AINT2+CONST*CALPHA/CNU
        EINT=EINT+CONST*CALPHA/CNU**3
C
        NU=NU+DELNU
C
  100   CONTINUE
C
C
      AINT1=DELNU/3.*AINT1
      AINT2=DELNU/3.*AINT2
      EINT=DELNU/3.*EINT
C
C     * CALCULATE VALUES OF DELTA-A, DELTA-E, AND DELTA-Ra/p *
C     ********************************************************
C
      DELA=2.*ESQU*(E*AINT1+AINT2)
      DELE=ESQU/2./E*DELA-ESQU**3/E*EINT
      DELRA=DELA*(1.+E)+DELE
      DELRP=DELA*(1.-E)-DELE
C
      RETURN
      END
```

Appendix B-4: Program DELEMAXUC

```
C     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C     *                                                               *
C     *                     PROGRAM DELEMAXUC                         *
C     *                                                               *
C     *                   Capt Gregory Beeker                         *
C     *            Air Force Institute of Technology                  *
C     *                       August 1988                             *
C     *                                                               *
C     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
C      The following program solves for the maximum change in the
C      magnitude of the nondimensional eccentricity for the contin-
C      uous thrust orbit transfer problem.  Simpson's rule is used
C      in subroutine integrate to solve the integral equations for
C      the change in nondimensional semimajor axis (delta a*) and
C      eccentricity (delta e*) given values of a, e, and nu.
C
C
      IMPLICIT DOUBLE PRECISION (A-H,L-Z)
      DIMENSION ALPHAI(361),NUI(361)
      COMMON E,DELRA,DELRP,DELA,DELE,IDIV,ALPHAI,NUI
C
C     * INPUT INITIAL DATA *
C     **********************
C
      WRITE (*,*) 'Please enter the initial and final values'
      WRITE (*,*) 'of the eccentricity, e, dear.'
      READ (*,*)  E0,EF
      WRITE (*,*) 'Enter the number of steps between the initial'
      WRITE (*,*) 'and final values of eccentricity.'
      READ (*,*) INUME
      WRITE (*,*) 'Enter the number of pieces Subroutine'
      WRITE (*,*) 'Integrate is to dissect the integrals into.'
      WRITE (*,*) '(Must be an even number, 360 maximum)'
      READ (*,*) IDIV
      WRITE (*,*) 'What type of print out do you wish?'
      WRITE (*,*) '(0 for document data, 1 for plot data)'
      READ (*,*) IPNT
      WRITE (*,*) 'If document data was selected, do you wish'
      WRITE (*,*) 'to print the thrust angle around the orbit?'
      WRITE (*,*) '(0 - no, 1 - yes)'
      READ (*,*) ITANG
C
C     * Print Initial Data *
C     **********************
C
      IF (IPNT.EQ.1) THEN
```

```
        OPEN (UNIT=12, STATUS = 'NEW',FILE = 'UCEDELA.DAT')
        OPEN (UNIT=13, STATUS = 'NEW',FILE = 'UCEDELE.DAT')
        OPEN (UNIT=14, STATUS = 'NEW',FILE = 'UCEDELRA.DAT')
        OPEN (UNIT=15, STATUS = 'NEW',FILE = 'UCEDELRP.DAT')
        ENDIF
C
      IF (IPNT.EQ.0) THEN
        OPEN (UNIT=10, STATUS = 'NEW',FILE = 'UCEMAXF.OUT')
        WRITE (10,15) IDIV
        WRITE (10,*)
        WRITE (10,25)
        ENDIF
C
      WRITE (6,15) IDIV
      WRITE (6,*)
      WRITE (6,25)
C
   15 FORMAT (3X,'Number of divisions for each integral is ',I4)
   25 FORMAT (6X,'e',13X,'Delta a*',14X,'Delta e*',14X,'Delta ra*',
     +       15X,'Delta rp*')
C
      ESTEP=(EF-E0)/INUME
      E=E0
C
C
   30 CALL INTEGRATE
C
C
C     * PRINT FINAL DATA *
C     ********************
C
   45 WRITE (6,48) E,DELA,DELE,DELRA,DELRP
      IF (IPNT.EQ.0) WRITE (10,48) E,DELA,DELE,DELRA,DELRP
      IF (IPNT.EQ.1) THEN
        WRITE (12,49) E,DELA
        WRITE (13,49) E,DELE
        WRITE (14,49) E,DELRA
        WRITE (15,49) E,DELRP
        ENDIF
C
      IF ((IPNT.EQ.0).AND.(ITANG.EQ.1)) THEN
        WRITE (6,*)
        WRITE (10,*)
        WRITE (6,50)
        WRITE (10,50)
        WRITE (6,*)
        WRITE (10,*)
        DO 46 I=1,(IDIV+1)
          WRITE (6,51) NUI(I), ALPHAI(I)
          WRITE (10,51) NUI(I), ALPHAI(I)
   46     CONTINUE
        WRITE (6,*)
```

```
          WRITE (10,*)
          ENDIF
C
   48 FORMAT (3X,F6.4,3X,E20.13,3X,E20.13,3X,E20.13,3X,E20.13)
   49 FORMAT (3X,E20.13,3X,E20.13)
   50 FORMAT (7X,'NU',8X,'ALPHA')
   51 FORMAT (3X,F7.2,5X,F7.2)
C
      DIFF=EF-E
      IF (ABS(DIFF).LE.1E-10) GO TO 60
C
      E=E+ESTEP
      GO TO 30
C
   60 STOP
      END
C
C
C
C
C
C     **********************************************************
C                        SUBROUTINE INTEGRATE
C     **********************************************************
C
      SUBROUTINE INTEGRATE
      IMPLICIT DOUBLE PRECISION (A-H,L-Z)
      DIMENSION ALPHAI(361),NUI(361)
      COMMON E,DELRA,DELRP,DELA,DELE,IDIV,ALPHAI,NUI
C
      NU=0.
      AINT1=0.0
      AINT2=0.0
      EINT=0.0
      ESQU=1.-E**2
      PI=DBLE(ACOS(-1.0))
C     WRITE (*,*) PI
      DELNU=2.*PI/IDIV
      DO 100  I=1, (IDIV+1)
        CNU=1.+E*DCOS(NU)
        SNU=DSIN(NU)
C
C       * CALCULATE THRUST VECTOR ANGLE *
C       *********************************
C
        NUMER=E*SNU*CNU
        DENOM=CNU**2-ESQU
C       WRITE (*,*) NUMER,DENOM
C
        IF ((DABS(NUMER).LE.1.E-15).AND.(DABS(DENOM)
     +        .LE.1.E-15)) THEN
```

```
            WRITE (6,80)
            IF (IPNT.EQ.0) WRITE (10,80)
   80       FORMAT (3X,'The denominator argument of the',
     +              'ARCTAN function was equal to or near 0.')
            WRITE (6,85) NUMER,DENOM,NU
            IF (IPNT.EQ.0) WRITE (10,85) NUMER,DENOM,NU
   85       FORMAT (3X,'NUM = ',E15.8,3X,'DEN = ',E15.8,3X,
     +              'NU = ',E15.8,' Rad')
            ALPHA=0.0
            GO TO 88
            ENDIF
C
          ALPHA=DATAN2(NUMER,DENOM)
   88     ALPHAI(I)=180.*ALPHA/PI
          NUI(I)=180.*NU/PI
          IF (ALPHAI(I).LT.0.) ALPHAI(I)=360.+ALPHAI(I)
          CALPHA=DCOS(ALPHA)
          SALPHA=DSIN(ALPHA)
C
C
C         * CALCULATE INTEGRALS OF DELTA A AND DELTA E *
C         **********************************************
C
          CONST=2.0
          IF ((I/2*2).EQ.I) CONST=4.0
          IF ((I.EQ.1).OR.(I.EQ.(IDIV+1))) CONST=1.0
          AINT1=AINT1+CONST*SNU/CNU**2*SALPHA
          AINT2=AINT2+CONST*CALPHA/CNU
          EINT=EINT+CONST*CALPHA/CNU**3
C
          NU=NU+DELNU
C
  100     CONTINUE
C
C
        AINT1=DELNU/3.*AINT1
        AINT2=DELNU/3.*AINT2
        EINT=DELNU/3.*EINT
C
C       * CALCULATE VALUES OF DELTA-A, DELTA-E, AND DELTA-Ra/p *
C       ********************************************************
C
        DELA=2.*ESQU*(E*AINT1+AINT2)
        DELE=ESQU/2./E*DELA-ESQU**3/E*EINT
        DELRA=DELA*(1.+E)+DELE
        DELRP=DELA*(1.-E)-DELE
C
        RETURN
        END
```

Appendix B-5: Program INTERPO

```
C     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C     *                                                             *
C     *                      PROGRAM INTERPO                        *
C     *                                                             *
C     *                   Capt Gregory Beeker                       *
C     *             Air Force Institute of Technology               *
C     *                        June 1988                            *
C     *                                                             *
C     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
C     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C     *  The following program utilizes the Newton formula to       *
C     *  interpolate/extrapolate from the given data sets           *
C     *  utilizing an nth degree interpolating polynomial.          *
C     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
C
      IMPLICIT DOUBLE PRECISION (A-H,L-Z)
      DIMENSION E(100),LAMBDA(100),DELA(100),DELE(100)
      CHARACTER LAMBDAT *10
      CHARACTER DELADAT *10
      CHARACTER DELEDAT *10
      CHARACTER OUTFILE *10
C
C     * INPUT INITIAL DATA *
C     **********************
C
      WRITE (*,*) 'Please specify the name of the Lambda*'
      WRITE (*,*) 'data file to be used.'
      READ (*,'(A)') LAMBDAT
      WRITE (*,*) 'Please specify the name of the Delta a*'
      WRITE (*,*) 'data file to be used.'
      READ (*,'(A)') DELADAT
      WRITE (*,*) 'Please specify the name of the Delta e*'
      WRITE (*,*) 'data file to be used.'
      READ (*,'(A)') DELEDAT
      OPEN (UNIT=15, STATUS = 'OLD',FILE = LAMBDAT)
      OPEN (UNIT=16, STATUS = 'OLD',FILE = DELADAT)
      OPEN (UNIT=17, STATUS = 'OLD',FILE = DELEDAT)
      WRITE (*,*) 'Is this data based on keeping the distance to '
      WRITE (*,*) 'apogee constant, or the distance to'
      WRITE (*,*) 'perigee constant ?'
      WRITE (*,*) '(0 - apogee; 1 - perigee)'
      READ (*,*) ID
      WRITE (*,*) 'Please specify the number of data points contained'
      WRITE (*,*) 'in each of the data files.'
      READ (*,*) IDATA
```

```
      WRITE (*,*) 'Please specify the min and max values of '
      WRITE (*,*) 'eccentricity for the range to be evaluated.'
      READ (*,*) EMIN, EMAX
      WRITE (*,*) 'Please enter the number of values of eccentricity'
      WRITE (*,*) 'to be evaluated within the specified range'
      WRITE (*,*) '(excluding the initial value).'
      READ (*,*) IE
      WRITE (*,*) 'Please specify the degree of the interpolating'
      WRITE (*,*) 'polynomial to be used (cannot exceed n-1, where'
      WRITE (*,*) 'n is the number of data points.)'
      READ (*,*) IDEG
      REWIND (15)
      REWIND (16)
      REWIND (17)
      DO 5, I=1,IDATA
        READ (15,*) E(I),LAMBDA(I)
        READ (16,*) E(I),DELA(I)
        READ (17,*) E(I),DELE(I)
    5   CONTINUE
      WRITE (*,*) 'Please specify the name of the output file.'
      READ (*,'(A)') OUTFILE
C
      OPEN (UNIT=10, STATUS = 'NEW',FILE = OUTFILE)
C
      IF (ID.EQ.0) THEN
        WRITE (6,18)
        WRITE (10,18)
        ENDIF
      IF (ID.EQ.1) THEN
        WRITE (6,19)
        WRITE (10,19)
        ENDIF
      WRITE (6,*)
      WRITE (10,*)
      WRITE (6,20) IDEG
      WRITE (10,20) IDEG
C
   18 FORMAT (3X,'Apogee Distance Fixed')
   19 FORMAT (3X,'Perigee Distance Fixed')
   20 FORMAT (3X,'Interpolating/Extrapolating Polynomials are of ',
     +           'degree ',I2)
C
      WRITE (6,*)
      WRITE (10,*)
      WRITE (6,110)
      WRITE (10,110)
      WRITE (6,*)
      WRITE (10,*)
C
      DELTAE=(EMAX-EMIN)/IE
      IEP1=IE+1
      EBAR=EMIN
```

```
C
C
C     * EVALUATE FUNCTION AT SPECIFIED VALUES OF ECCENTRICITY *
C     *********************************************************
C
      DO 100 ICOUNT=1,IEP1
C
C     * Check location of data point *
C
        DO 30 I=1,IDATA
          DIFF=EBAR-E(I)
          IF (ABS(DIFF).LE.1.E-15) THEN
            LAM=LAMBDA(I)
            DA=DELA(I)
            DE=DELE(I)
            GO TO 40
            ENDIF
          IF ((I.EQ.1).AND.(DIFF.LT.0.)) THEN
            IMIN=1
            IMAX=IMIN+IDEG
            GO TO 35
            ENDIF
          IF ((I.EQ.IDATA).AND.(DIFF.GT.0.)) THEN
            IMAX=IDATA
            IMIN=IMAX-IDEG
            GO TO 35
            ENDIF
          ILEFT=I
          IF (DIFF.LT.0.) GO TO 31
   30     CONTINUE
C
   31   IDEGP1=IDEG+1
        IDEGL=IDEGP1/2
        IMIN=ILEFT+1-IDEGL
        IMAX=IMIN+IDEG
        IF (IMIN.LT.1) THEN
          IMIN=1
          IMAX=IMIN+IDEG
          ENDIF
        IF (IMAX.GT.IDATA) THEN
          IMAX=IDATA
          IMIN=IMAX-IDEG
          ENDIF
C
C     * Evaluate Polynomial *
C
   35   CALL INTERP(IDLG,IMIN,IMAX,EBAR,E,LAMBDA,DELA,DELE,LAM,DA,DE)
C
C
C     * PRINT DATA *
C     **************
C
```

```
 40   WRITE (6,120) EBAR,LAM,DA,DE
      WRITE (10,120) EBAR,LAM,DA,DE
C
C
      EBAR=EBAR+DELTAE
100   CONTINUE
C
C
110 FORMAT (13X,'e',19X,'Lambda*',15X,'Delta a*',15X,'Delta e*')
120 FORMAT (3X,E20.13,3X,E20.13,3X,E20.13,3X,E20.13)
C
    CLOSE (10)
    CLOSE (15)
    CLOSE (16)
    CLOSE (17)
C
    STOP
    END
C
C
C
C
C     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C     *                    SUBROUTINE INTERP                          *
C     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
C     *     The following algorithm evaluates an interpolating        *
C     *                  nth degree polynomial                        *
C
C     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
      SUBROUTINE INTERP(IDEG,IMIN,IMAX,EBAR,E,LAMBDA,
     +                  DELA,DELE,LAM,DA,DE)
C
      IMPLICIT DOUBLE PRECISION (A-H,L-Z)
      DIMENSION E(100),LAMBDA(100),DELA(100),DELE(100)
      DIMENSION X(100),D(100),P(100)
C
      IDEGP1=IDEG+1
      DO 200 I=1,3
        DO 150 J=IMIN,IMAX
          K=J-IMIN+1
          IF (I.EQ.1) D(K)=LAMBDA(J)
          IF (I.EQ.2) D(K)=DELA(J)
          IF (I.EQ.3) D(K)=DELE(J)
          X(K)=E(J)
150       CONTINUE
        DO 155 K=1,IDEG
          DO 155 J=(K+1),IDEGP1
            J1=IDEGP1-J+(K+1)
            D(J1)=(D(J1)-D(J1-1))/(X(J1)-X(J1-K))
155         CONTINUE
```

```
C
      Z=EBAR-X(1)
      P(1)=D(1)+D(2)*Z
      DO 160 J=2,IDEG
        Z=Z*(EBAR-X(J))
        P(J)=P(J-1)+D(J+1)*Z
  160   CONTINUE
C
      IF (I.EQ.1) LAM=P(IDEG)
      IF (I.EQ.2) DA=P(IDEG)
      IF (I.EQ.3) DE=P(IDEG)
C
  200 CONTINUE
C
    RETURN
    END
```

Appendix C-1: <u>Program TRANSMUL</u>

```
C     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C     *                                                               *
C     *                      Program TRANSMUL                         *
C     *                                                               *
C     *                   Captain Gregory Beeker                      *
C     *              Air Force Institute of Technology                *
C     *                         July 1988                             *
C     *                                                               *
C     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
C     The following program was written to solve the long timescale
C     problem to determine the total transfer delta v and transfer
C     time of a spacecraft traveling between two planar orbits.  The
C     program takes a spacecraft from an initial orbit to a final
C     orbit though many revolutions while constraining one of the
C     two radii (perigee (rp) or apogee (ra)).  Subroutine Haming is
C     incorporated to solve the differential equations for the non-
C     dimensional changes in eccentricity and semimajor axis
C     (dabar/dVbar and de/dVbar).  In addition, Program INTERPO, which
C     provided the final solutions to the fast timescale problem, is
C     also incorporated as a subroutine to provide the values of delta
C     a* and delta e* as functions of eccentricity.
C
C
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DOUBLE PRECISION ISP,MU,MANOMO,MANOM,MDOTBAR,MDOT,MO
      DIMENSION X(4),VAR1(4),VAR2(4),VAR3(4),VAR4(4),VAR5(4)
      COMMON /HAM/ VBAR,Y(42,4),F(42,4),ERR(42),N,H,MODE
      COMMON /RH/ RABAR(4),RPBAR(4),ICASE,ICNT,MANOMO,AO,MDOTBAR,
     +            ACCELO,PI2
C
C
C     * INITIALIZATION OF PARAMETERS FOR HAMING *
C     *******************************************
C
      N=4
      MODE=0
C
C
C     * INITIAL DATA ENTRY *
C     **********************
C
C     * Primary Body Parameters
C
      WRITE (*,*) 'Do you wish to change the value of the'
      WRITE (*,*) 'gravitational parameter and radius of the'
```

```
      WRITE (*,*) 'primary body (currently set to those of'
      WRITE (*,*) 'the Earth)?   (0 - yes; 1 - no)'
      READ (*,*) IP
C
      IF (IP.EQ.0) THEN
        WRITE (*,*) 'Enter the new values of mu (km3/sec2) and r (km).'
        READ (*,*) MU,R
        ENDIF
      IF (IP.EQ.1) THEN
        MU=3.986012D5
        R=6378.145D0
        ENDIF
C
C     * Orbit Data
C
      WRITE (*,*) 'Specify the type of program run (0 or 1).'
      WRITE (*,*)
      WRITE (*,*) '         0 - Single Transfer'
      WRITE (*,*) '         1 - Multiple Transfers (Includes plot'
      WRITE (*,*) '             data file of af/a0 vs delta-V total)'
      READ (*,*) ITYP
C
      WRITE (*,*) 'Enter the eccentricity and semimajor axis (km)'
      WRITE (*,*) 'of the initial orbit.'
      READ (*,*) E0,A0
C
    2 IF (A0.LE.R) THEN
        WRITE(*,*) 'Initial orbit intersects the surface of the'
        WRITE(*,*) 'primary body.  Please select a new value.'
        READ (*,*) A0
        GOTO 2
        ENDIF
C
      WRITE (*,*) 'Enter the eccentricity and semimajor axis (km)'
      WRITE (*,*) 'of the final orbit.'
      READ (*,*) EF,AF
C
    3 IF (AF.LE.R) THEN
        WRITE(*,*) 'Final orbit intersects the surface of the'
        WRITE(*,*) 'primary body.  Please select a new value.'
        READ (*,*) AF
        GOTO 3
        ENDIF
C
      IF (ITYP.EQ.1) THEN
        WRITE (*,*) 'Specify the number of transfers between'
        WRITE (*,*) 'the initial and final orbits.'
        READ (*,*) NTRANS
        ENDIF
C
C     * Vehicle Parameters
C
```

```
      M0=2000
      MDOT=4.0D-4
      MDOTBAR=MDOT/M0
      ISP=5000
C
      WRITE (*,*) 'Do you wish to change the value of the vehicle mass'
      WRITE (*,*) 'and mass flow rate parameters?  (0 - yes; 1 - no)'
      WRITE (*,*)
      WRITE (*,*) 'Current Settings:'
      WRITE (*,11) M0
      WRITE (*,12) MDOT
      WRITE (*,13) MDOTBAR
      READ (*,*) IM
C
      IF (IM.EQ.0) THEN
        WRITE (*,*) 'Enter the new values of the mass flow rate (kg/s)'
        WRITE (*,*) 'and initial vehicle mass (kg).'
        READ (*,*) MDOT,M0
        MDOTBAR=MDOT/M0
        WRITE (*,13) MDOTBAR
        ENDIF
C
      WRITE (*,*) 'Do you wish to change the value of propulsion'
      WRITE (*,14) ISP
      WRITE (*,*) '(0 - yes; 1 - no)'
      READ(*,*)IISP
C
      IF (IISP.EQ.0) THEN
        WRITE (*,*) 'Enter the new value of Isp.'
        READ (*,*) ISP
        ENDIF
C
      ACCEL0=ISP*9.81*MDOTBAR/1000.
C
C     * Integration Data
C
      WRITE (*,*) 'Enter the step size of the independent'
      WRITE (*,*) 'variable, V-bar.'
      READ (*,*) H
      WRITE (*,*) 'Enter the maximum number of iterations to be'
      WRITE (*,*) 'performed by Haming on each half transfer.'
      READ (*,*) IMAX
C
C     * Output Parameters
C
      IF (ITYP.EQ.1) THEN
        IOUT=2
        OPEN (UNIT = 10, STATUS = 'NEW', FILE = 'EDELVBAR')
        OPEN (UNIT = 11, STATUS = 'NEW', FILE = 'SDELVBAR')
        OPEN (UNIT = 12, STATUS = 'NEW', FILE = 'HDELVBAR')
        GO TO 8
        ENDIF
```

```
C
      WRITE (*,*) 'Enter output code.'
      WRITE (*,*) '(0 for screen, 1 for file and screen)'
      READ (*,*) IOUT
C
    8 IF (IOUT.GE.1) THEN
        WRITE (*,*) 'Specify the name of the data output file.'
        READ (*,'(A)') TRANSOUT
        OPEN (UNIT = 9, STATUS = 'NEW', FILE = TRANSOUT)
        ENDIF
C
      IPNT=0
      IF (IOUT.LE.1) THEN
        WRITE (*,*) 'Enter the number of steps between each'
        WRITE (*,*) 'data printout.'
        READ (*,*) IPNT
        ENDIF
C
C
C
C     * INITIALIZE  ORBIT PARAMETERS (NONDIMENSIONAL) *
C     *************************************************
C
C     * State Vector
C
      Y2F=AF/A0
      Y2FCK=Y2F
C
      IF (ITYP.EQ.1) THEN
        DELAF=(AF-A0)/NTRANS
        Y2F=1.D0
        ENDIF
C
   10 IF (ITYP.EQ.1) Y2F=Y2F+DELAF/A0
C
C
C     * e (de/dvbar)
      Y(1,1)=E0
C     * a bar, dabar/dvbar
      Y(2,1)=A0/A0
C     * t (dt/dvbar)
      Y(3,1)=0.0
C     * nu (dnu/dvbar)
      Y(4,1)=0.0
C
C
C     * Initial Mean Anomaly
C
      MANOMO=DSQRT(MU/A0**3)
C
      PI=DBLE(ACOS(-1.))
      PI2=2.0*PI
```

```
C
C
C     * Initialize ra* & rp*
C
      RABAR(1)=Y(2,1)*(1.D0+E0)
      RABARI=RABAR(1)
      RABARF=Y2F*(1.D0+EF)
      RPBAR(1)=Y(2,1)*(1.D0-E0)
      RPBARI=RPBAR(1)
      RPBARF=Y2F*(1.D0-EF)
C
C
C
C     * PRINT INITIAL DATA *
C     **********************
C
      IF (IOUT.EQ.1) THEN
        WRITE (9,*) '  Orbit Data'
        WRITE (9,*) '  **********'
        WRITE (9,*)
        WRITE (9,15) E0,A0,Y(2,1)
        WRITE (9,16) EF,AF,Y2FCK
        WRITE (9,*)
        WRITE (9,*)
        WRITE (9,*) '  Primary Body Data'
        WRITE (9,*) '  *****************'
        WRITE (9,*)'                                                   3    2'
        WRITE (9,17) MU,R
        WRITE (9,*)
        WRITE (9,*)
        WRITE (9,*) '  Transfer Vehicle Data'
        WRITE (9,*) '  *********************'
        WRITE (9,*)
        WRITE (9,18) M0,ISP
        WRITE (9,19) MDOT,MDOTBAR
        WRITE (9,*)
        WRITE (9,*)
        ENDIF
C
      IF (ITYP.EQ.1) THEN
        WRITE (*,*) '  *********************'
        WRITE (*,20) Y2F
        WRITE (*,*) '  *********************'
        WRIT  (9,*) '  *********************'
        WRITE (9,20) Y2F
        WRITE (9,*) '  *********************'
        ENDIF
C
      WRITE (*,*)
      WRITE (*,21)
      WRITE (*,*)
C
```

```
      IF (IOUT.GE.1) THEN
        WRITE (9,*)
        WRITE (9,21)
        WRITE (9,*)
        ENDIF
C
   11 FORMAT (3X,'Vehicle Initial Mass:  ',F9.2,' kg')
   12 FORMAT (3X,'Propellant Mass Flow Rate:  ',E13.7,' kg/sec')
   13 FORMAT (3X,'Nondimensional Propellant Mass Flow Rate:  ',E13.7,
     +         '/sec')
   14 FORMAT (1X,'system Isp currently set at ',F6.1,' sec ?')
   15 FORMAT (3X,'Initial Orbit Eccentricity:  ',F6.3,6X,'Initial ',
     +         'Orbit Semimajor Axis:  ',F10.2,' km  (',E13.7,')')
   16 FORMAT (3X,'Final Orbit Eccentricity:    ',F6.3,6X,'Final ',
     +         'Orbit Semimajor Axis:    ',F10.2,' km  (',E13.7,')')
   17 FORMAT (3X,'Gravitational Parameter:  ',E13.6,' km /sec ',6X,
     +         'Radius:  ',F9.3,' km')
   18 FORMAT (3X,'Total Initial Mass:  ',F9.2,' kg',6X,'Specific ',
     +         'Impulse:  ',F9.2,' sec')
   19 FORMAT (3X,'Propellant Mass Flow Rate:  ',E13.7,'kg/sec (',
     +         E13.7,' /sec)')
   20 FORMAT (3X,'AF/A0 = ',E13.7)
   21 FORMAT (3X,'Rev',5X,'Time (sec)',4X,'NU (Rad)',7X,'V-bar',14X,
     +         'e'13X,'a-bar'12X,'ra-bar',11X,'rp-bar')
C
C
C     * BEGIN ITERATIONS OF STATE VECTOR *
C     ************************************
C
      NREV=1
      Y4NU=Y(4,1)
      VBAR=0.D0
C
C     * Identify Initial Transfer Procedure
C
C     * CASE 11 - Increasing a and e
C     * CASE 12 - Increasing a and Decreasing e
C     * CASE 21 - Decreasing a and Increasing e
C     * CASE 22 - Decreasing a and e
C
      FLAG=0
      FLAG1=-1
      DIFFA=DABS(RABARF-RABARI)
      DIFFP=DABS(RPBARF-RPBARI)
      ICASE=21
      IF ((RPBARF.GT.RPBARI).OR.(DIFFP.LT.1.E-3)) THEN
        FLAG1=1
        IF (DIFFP.LT.1.E-3) FLAG=FLAG+1
        ICASE=11
        IF (RABARF.LT.RABARI) ICASE=ICASE+11
        IF (DIFFA.LT.1.E-3) GO TO 70
        ENDIF
```

```
C
   25 NXT=0
      ICNT=0
      CALL HAMING(NXT)
C
      IF ((NXT.EQ.0).AND.(H.LT.1.D-6)) THEN
        WRITE (*,*) 'H decreased to ',H,'.'
        WRITE (*,*) 'Haming still refused to initialize.'
        STOP
        ENDIF
C
      IF (NXT.EQ.0) THEN
        H=H/1.1
        IPNT=30
        GOTO 25
        ENDIF
C
      WRITE (*,130) NREV,Y(3,1),Y4NU,VBAR,Y(1,1),Y(2,1),
     +              RABAR(1),RPBAR(1)
      IF (IOUT.GE.1) WRITE (9,130) NREV,Y(3,1),Y4NU,VBAR,Y(1,1),
     +               Y(2,1),RABAR(1),RPBAR(1)
C
   50 DO 100 I=1,IMAX
C
        CALL HAMING(NXT)
C
        NREV=1+INT(Y(4,NXT)/PI2)
        Y4NU=Y(4,NXT)-FLOAT(NREV-1)*PI2
C
        IF (ICASE.EQ.11) DIFF=RABARF-RABAR(NXT)
        IF (ICASE.EQ.22) DIFF=RABAR(NXT)-RABARF
        IF (ICASE.EQ.12) DIFF=RPBARF-RPBAR(NXT)
        IF (ICASE.EQ.21) DIFF=RPBAR(NXT)-RPBARF
        IF (DABS(DIFF).LT.1.D-13) GO TO 110
        IF (DIFF.GT.0.D0) GO TO 90
C
C       WRITE (*,*)
C       WRITE (*,130) VBAR,Y(1,NXT),Y(2,NXT),RABAR(NXT),RPBAR(NXT)
C       IF (IOUT.EQ.1) THEN
C         WRITE (9,*)
C         WRITE (9,130) VBAR,Y(1,NXT),Y(2,NXT),RABAR(NXT),RPBAR(NXT)
C         ENDIF
C
        CALL HAMING(NXT)
C
C       WRITE (*,*)
C       WRITE (*,130) VBAR,Y(1,NXT),Y(2,NXT),RABAR(NXT),RPBAR(NXT)
C       WRITE (*,*)
C       IF (IOUT.EQ.1) THEN
C         WRITE (9,*)
C         WRITE (9,130) VBAR,Y(1,NXT),Y(2,NXT),RABAR(NXT),RPBAR(NXT)
C         WRITE (9,*)
```

```
C          ENDIF
C
        NXT1=NXT
        NXT=1
        VBAR1=VBAR
        DO 55 J=1,4
          J1=5-J
          IF ((ICASE.EQ.11).OR.(ICASE.EQ.22)) X(J1)=RABAR(NXT1)
          IF ((ICASE.EQ.12).OR.(ICASE.EQ.21)) X(J1)=RPBAR(NXT1)
          VAR1(J1)=VBAR1
          VAR2(J1)=Y(1,NXT1)
          VAR3(J1)=Y(2,NXT1)
          VAR4(J1)=Y(3,NXT1)
          VAR5(J1)=Y(4,NXT1)
          VBAR1=VBAR1-H
          NXT1=NXT1-1
          IF (NXT1.EQ.0) NXT1=4
   55     CONTINUE
        IF ((ICASE.EQ.11).OR.(ICASE.EQ.22)) XBAR=RABARF
        IF ((ICASE.EQ.12).OR.(ICASE.EQ.21)) XBAR=RPBARF
        CALL INTERP3 (XBAR,X,VAR1,P)
        VBAR=P
        CALL INTERP3 (XBAR,X,VAR2,P)
        Y(1,NXT)=P
        CALL INTERP3 (XBAR,X,VAR3,P)
        Y(2,NXT)=P
        CALL INTERP3 (XBAR,X,VAR4,P)
        Y(3,NXT)=P
        CALL INTERP3 (XBAR,X,VAR5,P)
        Y(4,NXT)=P
        IF ((ICASE.EQ.11).OR.(ICASE.EQ.22)) RABAR(NXT)=RABARF
        IF ((ICASE.EQ.12).OR.(ICASE.EQ.21)) RPBAR(NXT)=RPBARF
        NREV=1+INT(Y(4,NXT)/PI2)
        Y4NU=Y(4,NXT)-FLOAT(NREV-1)*PI2
C
   70   FLAG=FLAG+1
        IF (FLAG.EQ.2) GOTO 80
C
        ICASE=12
        IF (FLAG1.LT.0) THEN
          ICASE=11
          IF (RABARF.LT.RABARI) ICASE=ICASE+11
          IF (DIFFA.LT.1.E-3) GO TO 70
          ENDIF
C
        GO TO 25
C
C
   80   WRITE (*,130) NREV,Y(3,NXT),Y4NU,VBAR,Y(1,NXT),Y(2,NXT),
     +                RABAR(NXT),RPBAR(NXT)
        WRITE (*,*)
        IF (IOUT.GE.1) THEN
```

```
          WRITE (9,130) NREV,Y(3,NXT),Y4NU,VBAR,Y(1,NXT),Y(2,NXT),
     +                  RABAR(NXT),RPBAR(NXT)
          WRITE (9,*)
          ENDIF
        GO TO 150
C
C
 90     IF (IPNT.EQ.0) GO TO 100
        IF ((I/IPNT*IPNT).EQ.I) THEN
          WRITE (*,130) NREV,Y(3,NXT),Y4NU,VBAR,Y(1,NXT),Y(2,NXT),
     +                  RABAR(NXT),RPBAR(NXT)
          IF (IOUT.EQ.1) WRITE (9,130) NREV,Y(3,NXT),Y4NU,VBAR,
     +                         Y(1,NXT),Y(2,NXT),RABAR(NXT),RPBAR(NXT)
          ENDIF
C
C
100     CONTINUE
C
C
      WRITE (*,*) 'Maximum number of iterations reached.'
      WRITE (*,*) 'Program Terminated'
      GOTO 170
C
C
C
110   Y(1,1)=Y(1,NXT)
      Y(2,1)=Y(2,NXT)
      Y(3,1)=Y(3,NXT)
      Y(4,1)=Y(4,NXT)
      RABAR(1)=RABAR(NXT)
      RPBAR(1)=RPBAR(NXT)
C
      GO TO 70
C
C
130   FORMAT (3X,I3,3X,E14.7,3X,F6.3,3X,E14.7,3X,E14.7,3X,
     +        E14.7,3X,E14.7,3X,E14.7)
135   FORMAT (3X,'de/dVbar = ',E20.13)
C
150   CONTINUE
C
C
C     * TRANSFER TOTAL DELTA-V AND TIME CALCULATION *
C     ***********************************************
C
      DELV=VBAR*A0*MANOMO
      DELVM=DELV*1000.
      TOF=1./MDOTBAR*(1.-DEXP(-DELV/(9.81D-3*ISP)))/3600./24.
      TOFY=TOF/365.25
C
C
C     * Comparison Data  (Circular Orbits Only) *
```

```
C     * * * * * * * * * * * * * * * * * * * * * *
C
      IF ((E0.LT.1.E-6).AND.(EF.LT.1.E-6)) GO TO 170
C
C     * Spiral Transfer
C
      VCS1=DSQRT(MU/A0)
      VCS2=DSQRT(MU/A0/Y2F)
C
      DELVSP=VCS1-VCS2
      DELVSM=DELVSP*1000.
      VSPBAR=DELVSP/A0/MANOMO
      TOFSP=1./MDOTBAR*(1.-DEXP(-DELVSP/(9.81D-3*ISP)))/3600./24.
      TOFSPY=TOFSP/365.25
C
C
C     * Hohmann Transfer
C
      V1=DSQRT(2*MU*(1./A0-1./(A0+A0*Y2F)))
      V2=DSQRT(2*MU*(1./A0/Y2F-1./(A0+A0*Y2F)))
      DELVHM=(V1-VCS1)+(VCS2-V2)
      DELVHMM=DELVHM*1000.
      VHMBAR=DELVHM/A0/MANOMO
C
C
      WRITE (*,*)
      WRITE (*,160) DELV,DELVM
      WRITE (*,165) TOF,TOFY
      WRITE (*,*)
      WRITE (*,*)
      WRITE (*,*) '  Comparison Transfer Data'
      WRITE (*,*) '  ************************'
      WRITE (*,*)
      WRITE (*,*) '  Spiral Transfer'
      WRITE (*,*)
      WRITE (*,160) DELVSP,DELVSM
      WRITE (*,165) TOFSP,TOFSPY
      WRITE (*,*)
      WRITE (*,*) '  Hohmann Transfer'
      WRITE (*,*)
      WRITE (*,160) DELVHM,DELVHMM
      WRITE (*,*)
      WRITE (*,*)
      WRITE (*,*)
C
      IF (IOUT.GE.1) THEN
        WRITE (9,*)
        WRITE (9,160) DELV,DELVM
        WRITE (9,165) TOF,TOFY
        WRITE (9,*)
        WRITE (9,*)
        WRITE (9,*) '  Comparison Transfer Data'
```

```
        WRITE (9,*) '  ************************'
        WRITE (9,*)
        WRITE (9,*) '  Spiral Transfer'
        WRITE (9,*)
        WRITE (9,160) DELVSP,DELVSM
        WRITE (9,165) TOFSP,TOFSPY
        WRITE (9,*)
        WRITE (9,*) '  Hohmann Transfer'
        WRITE (9,*)
        WRITE (9,160) DELVHM,DELVHMM
        WRITE (9,*)
        WRITE (9,*)
        WRITE (9,*)
        ENDIF
C
      IF (ITYP.EQ.1) THEN
        WRITE (10,*) Y2F,VBAR
        WRITE (11,*) Y2F,VSPBAR
        WRITE (12,*) Y2F,VHMBAR
        ENDIF
C
  160 FORMAT (3X,'Total Transfer Delta-V:  ',F9.4,' km/s  (',
     +        F8.2,' m/s)')
  165 FORMAT (3X,'Total Transfer Time:  ',F9.3,' days   (',
     +        F6.2,' yr)')
C
C
      DIFFYF=Y2FCK-Y2F
      IF (DABS(DIFFYF).GT.1.E-13) GOTO 10
C
C
  170 CONTINUE
C
C
      IF (IOUT.GE.1) CLOSE (9)
      IF (ITYP.EQ.1) THEN
        CLOSE (10)
        CLOSE (11)
        CLOSE (12)
        ENDIF
C
C
      STOP
      END
C
C
C
C     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C     *                         SUBROUTINE RHS                          *
C     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
      SUBROUTINE RHS(NXT)
```

```
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DOUBLE PRECISION MANOMO,MDOTBAR
      COMMON /HAM/ VBAR,Y(42,4),F(42,4),ERR(42),N,H,MODE
      COMMON /RH/ RABAR(4),RPBAR(4),ICASE,ICNT,MANOMO,A0,MDOTBAR,
     +             ACCELO,PI2
C
C     * Specify ra = constant (ID=0) or rp = constant (ID=1)
C
      IF ((ICASE.EQ.11).OR.(ICASE.EQ.22)) ID=1
      IF ((ICASE.EQ.12).OR.(ICASE.EQ.21)) ID=0
C
      CALL INTERPO(Y(1,NXT),DELA,DELE,ID,ICNT)
      ICNT=ICNT+1
C
C     * Define Derivatives
C
      IF ((ICASE.EQ.11).OR.(ICASE.EQ.21)) SIGN=1.0
      IF ((ICASE.EQ.12).OR.(ICASE.EQ.22)) SIGN=-1.0
C
      F(1,NXT)=Y(2,NXT)**.5*SIGN*DELE/PI2
      F(2,NXT)=Y(2,NXT)**1.5*SIGN*DELA/PI2
      F(3,NXT)=(1.-MDOTBAR*Y(3,NXT))*MANOMO*A0/ACCELO
C
      Y4NU1=Y(4,NXT)-FLOAT(INT(Y(4,NXT)/PI2))
C
      F(4,NXT)=MANOMO/(Y(2,NXT)*(1.-Y(1,NXT)**2))**1.5*
     +          (1.+Y(1,NXT)*DCOS(Y4NU1))**2*F(3,NXT)
C
C     * Define Changes in ra-bar and rp-bar
C
      RABAR(NXT)=Y(2,NXT)*(1.+Y(1,NXT))
      RPBAR(NXT)=Y(2,NXT)*(1.-Y(1,NXT))
C
      RETURN
      END
C
C
      SUBROUTINE HAMING(NXT)
C VERSION OF 11/20/1987
C PURPOSE
C   HAMING IS AN ORDINARY DIFFERENTIAL EQUATIONS INTEGRATOR
C   IT IS A FOURTH ORDER PREDICTOR-CORRECTOR ALGORITHM WHICH
C   MEANS THAT IT CARRIES THE LAST FOUR VALUES OF THE STATE
C   VECTOR, AND EXTRAPOLATES THESE VALUES TO OBTAIN A PREDICTED
C   NEXT VALUE (THE PREDICTION STEP) AND EVALUATES THE EQUATIONS
C   OF MOTION AT THE PREDICTED POINT, AND THEN CORRECTS THE
C   EXTRAPOLATED POINT USING A HIGHER ORDER POLYNOMIAL (THE
C   CORRECTION STEP).
C INPUT
C   NXT --  IN THE CALL SPECIFIES WHICH OF THE FOUR VALUES OF
C           THE STATE VECTOR IS THE CURRENT ONE.  NXT IS UPDATED
C           BY HAMING AUTOMATICALLY, BUT MUST BE SET TO ZERO ON
```

```
C           THE FIRST CALL.
C CALL ROUTINES
C   RHS(NXT)
C REFERENCES
C   WILLIAM WEISEL
C PROGRAMMER
C   RODNEY D. BAIN
C PROGRAM MODIFICATIONS
C   NONE
C COMMENTS
C   TOL       -- IS HAMING'S START UP TOLERANCE ... SET TO REASONABLE VALUE
C                AS NECESSARY
C   THE COMMON BLOCK CONTAINS:
C     VBAR       -- IS THE INDEPENDENT VARIABLE (OFTEN TIME)
C     Y(42,4) -- IS THE STATE VECTOR, 4 COPIES OF IT, WITH NXT POINTING
C                POINTING TO THE CURRENT ONE, THE LIMIT OF 42 EQUATIONS
C                OF MOTION CAN BE CHANGED.
C     F(42,4) -- ARE THE EQUATIONS OF MOTION EVALUATED AT THE SAME TIMES
C                AS THE STATE VECTOR Y ... IT IS THE JOB OF SUBROUTINE
C                RHS TO CALCULATE THESE.
C     ERR(42) -- IS AN ESTIMATE OF THE ONE-STEP INTEGRATION ERROR
C     N       -- IS THE NUMBER OF ODES ... LIMIT IS 42 UNLESS YOU CHANGE
C                THE COMMON BLOCK
C     H       -- IS THE TIMESTEP ... ONE CALL TO HAMING INCREMENTS X BY H
C     MODE    -- IS ZERO FOR EOM ONLY, 1 FOR EOM AND EOV
C   THE USER MJST SUPPLY A MAIN PROGRAM, AND THE SUBROUTINE RHS(NXT) WHICH
C   EVALUATES THE EQUATIONS OF MOTION.
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      COMMON /HAM/ VBAR,Y(42,4),F(42,4),ERR(42),N,H,MODE
      DATA ZERO,ONE,TWO,THREE,FOUR/0.D0,1.D0,2.D0,3.D0,4.D0/
      TOL=1.D-12
C
C   CHECK IF THIS IS THE FIRST CALL ... HAMING (LIKE ALL PREDICTOR-
C   CORRECTORS) NEEDS 'PREVIOUS' VALUES
C
      IF(NXT) 190,10,200
C
C   IT IS A PICARD INTERATION (SLOW AND EXPENSIVE) TO STEP BACKWARDS
C   IN TIME THREE STEPS TO GET THE 4 PREVIOUS POINTS.  A SUCCESSFUL
C   STARTUP RETURNS NXT=1, AND TIME HAS NOT BEEN INCREMENTED.  IF
C   STARTUP FAILS, NXT WILL BE RETURNED AS ZERO.
C
 10   XO=VBAR
      HH=H/TWO
      CALL RHS(1)
      DO 40 L=2,4
         VBAR=VBAR+HH
           DO 20 I=1,N
              Y(I,L)=Y(I,L-1)+HH*F(I,L-1)
 20        CONTINUE
         CALL RHS(L)
```

```
          VBAR=VBAR+HH
            DO 30 I=1,N
               Y(I,L)=Y(I,L-1)+H*F(I,L)
 30         CONTINUE
          CALL RHS(L)
 40   CONTINUE
      JSW=-10
 50   ISW=1
      DO 120 I=1,N
         HH=Y(I,1)+H*(9.D0*F(I,1)+19.D0*F(I,2)-5.D0*F(I,3)
     1        +F(I,4))/24.D0
      IF(DABS(HH-Y(I,2)).LT.TOL) GOTO 70
      ISW=0
 70   Y(I,2)=HH
      HH=Y(I,1)+H*(F(I,1)+FOUR*F(I,2)+F(I,3))/THREE
      IF(DABS(HH-Y(I,3)).LT.TOL) GOTO 90
      ISW=0
 90   Y(I,3)=HH
      HH=Y(I,1)+H*(THREE*F(I,1)+9.D0*F(I,2)+9.D0*F(I,3)
     1   +THREE*F(I,4))/8.D0
      IF(DABS(HH-Y(I,4)).LT.TOL) GOTO 110
      ISW=0
 110  Y(I,4)=HH
 120  CONTINUE
      VBAR=XO
      DO 130 L=2,4
         VBAR=VBAR+H
         CALL RHS(L)
 130  CONTINUE
      IF(ISW) 140,140,150
 140  JSW=JSW+1
      IF(JSW) 50,280,280
 150  VBAR=XO
      ISW=1
      JSW=1
      DO 160 I=1,N
         ERR(I)=ZERO
 160  CONTINUE
      NXT=1
      GOTO 280
C
C  A CALL TO HAMING WITH NXT=-NXT, AFTER A SUCCESSFUL STARTUP,
C  WILL TURN OFF THE SECOND EVALUATION OF THE EQUATIONS OF MOTION
C  FOLLOWING THE CORRECTOR STEP.  IN SYSTEMS WHERE THE EQUATIONS OF
C  MOTION ARE VERY EXPENSIVE, THIS CAN HALVE YOUR RUN TIME.
C
 190  JSW=2
      NXT=IABS(NXT)
C
C  THIS IS THE PREDICTOR-CORRECTOR ALGORITHM ... FIRST THE INDICES
C  ARE PERMUTED
C
```

```
 200  VBAR=VBAR+H
      NP1=MOD(NXT,4)+1
      GOTO (210,230),ISW
 210  GOTO (270,270,270,220),NXT
 220  ISW=2
 230  NM2=MOD(NP1,4)+1
      NM1=MOD(NM2,4)+1
      NPO=MOD(NM1,4)+1
C
C   ... THEN THE PREDICTOR PART IS RUN TO FIND AN EXTRAPOLATED VALUE
C   OF THE STATE VECTOR AT THE NEW TIME ...
C
      DO 240 I=1,N
         F(I,NM2)=Y(I,NP1)+FOUR*H*(TWO*F(I,NPO)-F(I,NM1)
     1              +TWO*F(I,NM2))/THREE
         Y(I,NP1)=F(I,NM2)-0.925619835D0*ERR(I)
 240  CONTINUE
C
C   THE EQUATIONS OF MOTION ARE EVALUATED AT THE EXTRAPOLATED VALUE
C   OF THE STATE VECTOR ...
C
      CALL RHS(NP1)
C
C   AND THE CORRECTOR ALGORITHM IS USED TO ADD THIS NEW INFORMATION
C   AND OBTAIN A BETTER VALUE OF THE NEW STATE VECTOR ...
C
      DO 250 I=1,N
         Y(I,NP1)=(9.D0*Y(I,NPO)-Y(I,NM2)+THREE*H*(F(I,NP1)
     1              +TWO*F(I,NPO)-F(I,NM1)))/8.D0
         ERR(I)=F(I,NM2)-Y(I,NP1)
         Y(I,NP1)=Y(I,NP1)+0.0743801653D0*ERR(I)
 250  CONTINUE
      GOTO (260,270),JSW
C
C   FINALLY, THE EQUATIONS OF MOTION ARE RE-EVALUATED AT THE BETTER
C   VALUE OF THE STATE VECTOR ... THIS CAN BE SUPPRESSED.
C
 260  CALL RHS(NP1)
 270  NXT=NP1
 280  RETURN
      END
C
C
C     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C     *                       SUBROUTINE INTERP3                        *
C     *                                                                 *
C     *   The following subroutine interpolates between four data      *
C     *            points using a third order polynomial.               *
C     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
      SUBROUTINE INTERP3 (XBAR,X,D,P)
C
```

```
      IMPLICIT DOUBLE PRECISION (A-H,L-Z)
      DIMENSION X(4),P1(3),D(4)
C
      DO 300 K=1,3
        DO 300 J=(K+1),4
          J1=4-J+(K+1)
          D(J1)=(D(J1)-D(J1-1))/(X(J1)-X(J1-K))
  300     CONTINUE
C
      Z=XBAR-X(1)
      P1(1)=D(1)+D(2)*Z
      DO 310 J=2,3
        Z=Z*(XBAR-X(J))
        P1(J)=P1(J-1)+D(J+1)*Z
  310   CONTINUE
C
      P=P1(3)
C
      RETURN
      END
C
C
C
C
C     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C     *                     SUBROUTINE INTERPO                      *
C     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
C     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C     *  The following subroutine utilizes the Newton formula       *
C     *  to interpolate/extrapolate from the given data sets        *
C     *  utilizing a 4th degree interpolating polynomial.           *
C     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
C
      SUBROUTINE INTERPO(EBAR,DA,DE,ID,ICNT)
C
      IMPLICIT DOUBLE PRECISION (A-H,L-Z)
      DIMENSION E(100),LAMDA(100),DELA(100),DELE(100)
C
C
C     * OPEN DATA FILES *
C     *******************
C
C     * Constant ra Data
C
      IF ((ID.EQ.0).AND.(ICNT.EQ.0)) THEN
        OPEN (UNIT=15, STATUS = 'OLD',FILE = 'ALAM.DAT')
        OPEN (UNIT=16, STATUS = 'OLD',FILE = 'ADELA.DAT')
        OPEN (UNIT=17, STATUS = 'OLD',FILE = 'ADELE.DAT')
        IDATA=99
        ENDIF
```

```
C
C
C     * Constant rp Data
C
      IF ((ID.EQ.1).AND.(ICNT.EQ.0)) THEN
        OPEN (UNIT=15, STATUS = 'OLD',FILE = 'PLAM.DAT')
        OPEN (UNIT=16, STATUS = 'OLD',FILE = 'PDELA.DAT')
        OPEN (UNIT=17, STATUS = 'OLD',FILE = 'PDELE.DAT')
        IDATA=98
        ENDIF
C
C
      IF (ICNT.EQ.0) THEN
C
        REWIND (15)
        REWIND (16)
        REWIND (17)
C
C
C     * INPUT INITIAL DATA *
C     **********************
C
        DO 5, I=1,IDATA
          READ (15,*) E(I),LAMDA(I)
          READ (16,*) E(I),DELA(I)
          READ (17,*) E(I),DELE(I)
    5     CONTINUE
C
        IDEG=4
C
        CLOSE (15)
        CLOSE (16)
        CLOSE (17)
C
        ENDIF
C
C
C     * EVALUATE FUNCTION AT SPECIFIED VALUES OF ECCENTRICITY *
C     *********************************************************
C
C     * Check location of data point *
C
      DO 30 I=1,IDATA
        DIFF=EBAR-E(I)
        IF (ABS(DIFF).LE.1.E-15) THEN
          LAM=LAMDA(I)
          DA=DELA(I)
          DE=DELE(I)
          GO TO 40
          ENDIF
C
        IF ((I.EQ.1).AND.(DIFF.LT.0.)) THEN
```

```
          IMIN=1
          IMAX=IMIN+IDEG
          GO TO 35
          ENDIF
C
        IF ((I.EQ.IDATA).AND.(DIFF.GT.0.)) THEN
          IMAX=IDATA
          IMIN=IMAX-IDEG
          GO TO 35
          ENDIF
C
        ILEFT=I
        IF (DIFF.LT.0.) GO TO 31
C
   30   CONTINUE
C
C
   31 IDEGP1=IDEG+1
      IDEGL=IDEGP1/2
      IMIN=ILEFT+1-IDEGL
      IMAX=IMIN+IDEG
C
      IF (IMIN.LT.1) THEN
        IMIN=1
        IMAX=IMIN+IDEG
        ENDIF
C
      IF (IMAX.GT.IDATA) THEN
        IMAX=IDATA
        IMIN=IMAX-IDEG
        ENDIF
C
C
C     * Evaluate Polynomial *
C
   35 CALL INTERP(IDEG,IMIN,IMAX,EBAR,E,LAMDA,DELA,DELE,LAM,DA,DE)
C
   40 CONTINUE
      RETURN
      END
C
C
C
C
C     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C     *                   SUBROUTINE INTERP                          *
C     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
C     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C     *     The following algorithm evaluates an interpolating       *
C     *                  nth degree polynomial                       *
C     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
```

```
C
      SUBROUTINE INTERP(IDEG,IMIN,IMAX,EBAR,E,LAMDA,DELA,DELE,LAM,DA,DE)
C
      IMPLICIT DOUBLE PRECISION (A-H,L-Z)
      DIMENSION E(100),LAMDA(100),DELA(100),DELE(100)
      DIMENSION X(100),D(100),P(100)
C
      IDEGP1=IDEG+1
C
      DO 200 I=1,3
C
        DO 150 J=IMIN,IMAX
          K=J-IMIN+1
          IF (I.EQ.1) D(K)=LAMDA(J)
          IF (I.EQ.2) D(K)=DELA(J)
          IF (I.EQ.3) D(K)=DELE(J)
          X(K)=E(J)
  150     CONTINUE
C
        DO 155 K=1,IDEG
          DO 155 J=(K+1),IDEGP1
            J1=IDEGP1-J+(K+1)
            D(J1)=(D(J1)-D(J1-1))/(X(J1)-X(J1-K))
  155       CONTINUE
C
        Z=EBAR-X(1)
        P(1)=D(1)+D(2)*Z
C
        DO 160 J=2,IDEG
          Z=Z*(EBAR-X(J))
          P(J)=P(J-1)+D(J+1)*Z
  160     CONTINUE
C
        IF (I.EQ.1) LAM=P(IDEG)
        IF (I.EQ.2) DA=P(IDEG)
        IF (I.EQ.3) DE=P(IDEG)
C
  200   CONTINUE
C
      RETURN
      END
```

Appendix C-2: Program TRANSALT

```
C     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C     *                                                               *
C     *                        Program TRANSALT                       *
C     *                                                               *
C     *                    Captain Gregory Beeker                     *
C     *               Air Force Institute of Technology               *
C     *                          July 1988                            *
C     *                                                               *
C     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
C     The following program was written to solve the long timescale
C     problem to determine the total transfer time of a spacecraft
C     traveling between two orbits.  The program takes a transfer
C     vehicle from an initial orbit to a final orbit though many
C     revolutions.  A combination of the fast timescale changes in the
C     orbital elements for constant distance to perigee (rp) and apogee
C     (ra) over two revolutions is implemented.  Subroutine Haming
C     is incorporated to solve the differential equations for the
C     nondimensional changes in eccentricity and semimajor axis
C     (dabar/dVbar and de/dVbar).  In addition, Program INTERPO, which
C     provided the final solutions to the fast timescale problem, is
C     also incorporated as a subroutine to provide the values of delta
C     a* and delta e* as functions of eccentricity.
C
C
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DOUBLE PRECISION ISP,MU,MANOMO,MANOM,MDOTBAR,MDOT,MO
      DIMENSION X(4),VAR1(4),VAR2(4),VAR3(4),VAR4(4)
      COMMON /HAM/ VBAR,Y(42,4),F(42,4),ERR(42),N,H,MODE
      COMMON /RH/ RABAR(4),RPBAR(4),ICASE,ICNT,DIFF
C
C
C     * INITIALIZATION OF PARAMETERS FOR HAMING *
C     *******************************************
C
      N=2
      MODE=0
C
C
C
C     * INITIAL DATA ENTRY *
C     **********************
C
C     * Primary Body Parameters
C
      WRITE (*,*) 'Do you wish to change the value of the'
```

```
      WRITE (*,*) 'gravitational parameter and radius of the'
      WRITE (*,*) 'primary body (currently set to those of'
      WRITE (*,*) 'the Earth)?   (0 - yes; 1 - no)'
      READ (*,*) IP
C
      IF (IP.EQ.0) THEN
        WRITE (*,*) 'Enter the new values of mu (km3/sec2) and r (km).'
        READ (*,*) MU,R
        ENDIF
      IF (IP.EQ.1) THEN
        MU=3.986012D5
        R=6378.145D0
        ENDIF
C
C     * Orbit Data
C
      WRITE (*,*) 'Specify the type of program run (0 or 1).'
      WRITE (*,*)
      WRITE (*,*) '         0 - Single Transfer'
      WRITE (*,*) '         1 - Multiple Transfers (Includes plot'
      WRITE (*,*) '             data file of af/a0 vs delta-V total)'
      READ (*,*) ITYP
C
      WRITE (*,*) 'Enter the eccentricity and semimajor axis (km)'
      WRITE (*,*) 'of the initial orbit.'
      READ (*,*) E0,A0
C
    2 IF (A0.LE.R) THEN
        WRITE(*,*) 'Initial orbit intersects the surface of the'
        WRITE(*,*) 'primary body.  Please select a new value.'
        READ (*,*) A0
        GOTO 2
        ENDIF
C
      WRITE (*,*) 'Enter the eccentricity and semimajor axis (km)'
      WRITE (*,*) 'of the final orbit.'
      READ (*,*) EF,AF
C
    3 IF (AF.LE.R) THEN
        WRITE(*,*) 'Final orbit intersects the surface of the'
        WRITE(*,*) 'primary body.  Please select a new value.'
        READ (*,*) AF
        GOTO 3
        ENDIF
C
      IF (ITYP.EQ.1) THEN
        WRITE (*,*) 'Specify the number of transfers between'
        WRITE (*,*) 'the initial and final orbits.'
        READ (*,*) NTRANS
        ENDIF
C
C     * Vehicle Parameters
```

```
C
      M0=2000
      MDOT=4.0D-4
      MDOTBAR=MDOT/M0
      ISP=5000
C
      WRITE (*,*) 'Do you wish to change the value of the vehicle mass'
      WRITE (*,*) 'and mass flow rate parameters?  (0 - yes; 1 - no)'
      WRITE (*,*)
      WRITE (*,*) 'Current Settings:'
      WRITE (*,11) M0
      WRITE (*,12) MDOT
      WRITE (*,13) MDOTBAR
      READ (*,*) IM
C
      IF (IM.EQ.0) THEN
        WRITE (*,*) 'Enter the new values of the mass flow rate (kg/s)'
        WRITE (*,*) 'and initial vehicle mass (kg).'
        READ (*,*) MDOT,M0
        MDOTBAR=MDOT/M0
        WRITE (*,13) MDOTBAR
        ENDIF
C
      WRITE (*,*) 'Do you wish to change the value of propulsion'
      WRITE (*,14) ISP
      WRITE (*,*) '(0 - yes; 1 - no)'
      READ(*,*)IISP
C
      IF (IISP.EQ.0) THEN
        WRITE (*,*) 'Enter the new value of Isp.'
        READ (*,*) ISP
        ENDIF
C
C     * Integration Data
C
      WRITE (*,*) 'Enter the step size of the independent'
      WRITE (*,*) 'variable, V-bar.'
      READ (*,*) H
      WRITE (*,*) 'Enter the maximum number of iterations to be'
      WRITE (*,*) 'performed by Haming during each transfer.'
      READ (*,*) IMAX
C
C     * Output Parameters
C
      IF (ITYP.EQ.1) THEN
        IOUT=2
        OPEN (UNIT = 10, STATUS = 'NEW', FILE = 'EDVALT')
        OPEN (UNIT = 11, STATUS = 'NEW', FILE = 'SDVALT')
        OPEN (UNIT = 12, STATUS = 'NEW', FILE = 'HDVALT')
        GO TO 8
        ENDIF
C
```

```
      WRITE (*,*) 'Enter output code.'
      WRITE (*,*) '(0 for screen, 1 for file and screen)'
      READ (*,*) IOUT
C
    8 IF (IOUT.GE.1) THEN
        WRITE (*,*) 'Specify the name of the data output file.'
        READ (*,'(A)') TRANSOUT
        OPEN (UNIT = 9, STATUS = 'NEW', FILE = TRANSOUT)
        ENDIF
C
      IPNT=0
      IF (IOUT.LE.1) THEN
        WRITE (*,*) 'Enter the number of steps between each'
        WRITE (*,*) 'data printout.'
        READ (*,*) IPNT
        ENDIF
C
C
C
C     * INITIALIZE  ORBIT PARAMETERS (NONDIMENSIONAL) *
C     *************************************************
C
C     * State Vector
C
      Y2F=AF/A0
      Y2FCK=Y2F
C
      IF (ITYP.EQ.1) THEN
        DELAF=(AF-A0)/NTRANS
        Y2F=1.D0
        ENDIF
C
   10 IF (ITYP.EQ.1) Y2F=Y2F+DELAF/A0
C
      Y(1,1)=E0
      Y(2,1)=A0/A0
C
C
C     * Initial Mean Anomaly
C
      MANOM0=DSQRT(MU/A0**3)
C
C
C     * Initialize ra* & rp*
C
      RABAR(1)=Y(2,1)*(1.D0+E0)
      RABARI=RABAR(1)
      RABARF=Y2F*(1.D0+EF)
      RPBAR(1)=Y(2,1)*(1.D0-E0)
      RPBARI=RPBAR(1)
      RPBARF=Y2F*(1.D0-EF)
C
```

```
C
C      * Identify Orbit Type *
C      ***********************
C
C      * CASE 11 - Increasing a and e
C      * CASE 12 - Increasing a and Decreasing e
C      * CASE 21 - Decreasing a and Increasing e
C      * CASE 22 - Decreasing a and e
C
       ICASE=11
       IF (A0.GT.AF) ICASE=ICASE+10
C
C
C      * PRINT INITIAL DATA *
C      **********************
C
       IF (IOUT.EQ.1) THEN
         WRITE (9,*) '  Orbit Data'
         WRITE (9,*) '  **********'
         WRITE (9,*)
         WRITE (9,15) E0,A0,Y(2,1)
         WRITE (9,16) EF,AF,Y2FCK
         WRITE (9,*)
         WRITE (9,*)
         WRITE (9,*) '  Primary Body Data'
         WRITE (9,*) '  *****************'
         WRITE (9,*)'                                         3    2'
         WRITE (9,17) MU,R
         WRITE (9,*)
         WRITE (9,*)
         WRITE (9,*) '  Transfer Vehicle Data'
         WRITE (9,*) '  *********************'
         WRITE (9,*)
         WRITE (9,18) M0,ISP
         WRITE (9,19) MDOT,MDOTBAR
         WRITE (9,*)
         WRITE (9,*)
         ENDIF
C
       IF (ITYP.EQ.1) THEN
         WRITE (*,*) '  *********************'
         WRITE (*,20) Y2F
         WRITE (*,*) '  *********************'
         WRITE (9,*) '  *********************'
         WRITE (9,20) Y2F
         WRITE (9,*) '  *********************'
         ENDIF
C
       WRITE (*,*)
       WRITE (*,21)
       WRITE (*,*)
C
```

```
      IF (IOUT.GE.1) THEN
        WRITE (9,*)
        WRITE (9,21)
        WRITE (9,*)
        ENDIF
C
   11 FORMAT (3X,'Vehicle Initial Mass:  ',F9.2,' kg')
   12 FORMAT (3X,'Propellant Mass Flow Rate:  ',E13.7,' kg/sec')
   13 FORMAT (3X,'Nondimensional Propellant Mass Flow Rate:  ',E13.7,
     +        '/sec')
   14 FORMAT (1X,'system Isp currently set at ',F6.1,' sec ?')
   15 FORMAT (3X,'Initial Orbit Eccentricity:  ',F6.3,6X,'Initial ',
     +        'Orbit Semimajor Axis:  ',F10.2,' km  (',E13.7,')')
   16 FORMAT (3X,'Final Orbit Eccentricity:    ',F6.3,6X,'Final ',
     +        'Orbit Semimajor Axis:    ',F10.2,' km  (',E13.7,')')
   17 FORMAT (3X,'Gravitational Parameter:  ',E13.6,' km /sec ',6X,
     +        'Radius:  ',F9.3,' km')
   18 FORMAT (3X,'Total Initial Mass:  ',F9.2,' kg',6X,'Specific ',
     +        'Impulse:  ',F9.2,' sec')
   19 FORMAT (3X,'Propellant Mass Flow Rate:  ',E13.7,'kg/sec (',
     +        E13.7,' /sec)')
   20 FORMAT (3X,'AF/A0 = ',E13.7)
   21 FORMAT (10X,'V-bar',21X,'e'20X,'a-bar'18X,'ra-bar',17X,'rp-bar')
C
C
C     * BEGIN ITERATIONS OF STATE VECTOR *
C     ************************************
C
      VBAR=0.D0
   25 NXT=0
      ICNT=0
      CALL HAMING(NXT)
      IF (NXT.EQ.0) THEN
         WRITE(*,*) 'HAMING DID NOT INITIALIZE'
         STOP
         ENDIF
C
      WRITE (*,130) VBAR,Y(1,1),Y(2,1),RABAR(1),RPBAR(1)
      IF (IOUT.GE.1) WRITE (9,130) VBAR,Y(1,1),Y(2,1),
     +              RABAR(1),RPBAR(1)
C
   50 DO 100 I=1,IMAX
C
        CALL HAMING(NXT)
C
        IF (ICASE.EQ.11) DIFF=RABARF-RABAR(NXT)
        IF (ICASE.EQ.21) DIFF=RPBAR(NXT)-RPBARF
C
        IF (DABS(DIFF).LT.1.D-13) GO TO 80
        IF (DIFF.GT.0.D0) GOTO 90
C
C       WRITE (*,*)
```

```
C       WRITE (*,130) VBAR,Y(1,NXT),Y(2,NXT),RABAR(NXT),RPBAR(NXT)
C       IF (IOUT.EQ.1) THEN
C         WRITE (9,*)
C         WRITE (9,130) VBAR,Y(1,NXT),Y(2,NXT),RABAR(NXT),RPBAR(NXT)
C         ENDIF
C
        CALL HAMING(NXT)
C
C       WRITE (*,*)
C       WRITE (*,130) VBAR,Y(1,NXT),Y(2,NXT),RABAR(NXT),RPBAR(NXT)
C       WRITE (*,*)
C       IF (IOUT.EQ.1) THEN
C         WRITE (9,*)
C         WRITE (9,130) VBAR,Y(1,NXT),Y(2,NXT),RABAR(NXT),RPBAR(NXT)
C         WRITE (9,*)
C         ENDIF
C
        NXT1=NXT
        VBAR1=VBAR
C
        DO 55 J=1,4
          J1=5-J
          IF (ICASE.EQ.11) X(J1)=RABAR(NXT1)
          IF (ICASE.EQ.21) X(J1)=RPBAR(NXT1)
          VAR1(J1)=VBAR1
          VAR2(J1)=Y(1,NXT1)
          VAR3(J1)=Y(2,NXT1)
          IF (ICASE.EQ.11) VAR4(J1)=RPBAR(NXT1)
          IF (ICASE.EQ.21) VAR4(J1)=RABAR(NXT1)
          VBAR1=VBAR1-H
          NXT1=NXT1-1
          IF (NXT1.EQ.0) NXT1=4
   55     CONTINUE
C
        IF (ICASE.EQ.11) XBAR=RABARF
        IF (ICASE.EQ.21) XBAR=RPBARF
        CALL INTERP3 (XBAR,X,VAR1,P)
        VBAR=P
        CALL INTERP3 (XBAR,X,VAR2,P)
        Y(1,NXT)=P
        CALL INTERP3 (XBAR,X,VAR3,P)
        Y(2,NXT)=P
        CALL INTERP3 (XBAR,X,VAR4,P)
        IF (ICASE.EQ.11) RPBAR(NXT)=P
        IF (ICASE.EQ.21) RABAR(NXT)=P
C
        IF (ICASE.EQ.11) RABAR(NXT)=RABARF
        IF (ICASE.EQ.21) RPBAR(NXT)=RPBARF
C
   80   WRITE (*,130) VBAR,Y(1,NXT),Y(2,NXT),RABAR(NXT),RPBAR(NXT)
        WRITE (*,*)
C
```

```
        IF (IOUT.GE.1) THEN
          WRITE (9,130) VBAR,Y(1,NXT),Y(2,NXT),RABAR(NXT),RPBAR(NXT)
          WRITE (9,*)
          ENDIF
C
        GO TO 150
C
C
   90   IF (IPNT.EQ.0) GO TO 100
        IF ((I/IPNT*IPNT).EQ.I) THEN
          WRITE (*,130) VBAR,Y(1,NXT),Y(2,NXT),RABAR(NXT),RPBAR(NXT)
          IF (IOUT.EQ.1) WRITE (9,130) VBAR,Y(1,NXT),Y(2,NXT),
     +                      RABAR(NXT),RPBAR(NXT)
          ENDIF
C
C
  100   CONTINUE
C
C
      WRITE (*,*) 'Maximum number of iterations reached.'
      WRITE (*,*) 'Program Terminated'
      GOTO 170
C
C
  130 FORMAT (3X,E20.13,3X,E20.13,3X,E20.13,3X,E20.13,3X,E20.13)
  135 FORMAT (3X,'de/dVbar = ',E20.13)
C
  150 CONTINUE
C
C
C     * TRANSFER TOTAL DELTA-V AND TIME CALCULATION *
C     ***********************************************
C
      DELV=VBAR*A0*MANOMO
      DELVM=DELV*1000.
      TOF=1./MDOTBAR*(1.-DEXP(-DELV/(9.81D-3*ISP)))/3600./24.
      TOFY=TOF/365.25
C
C
C     * Comparison Data *
C      * * * * * * * * *
C
C     * Spiral Transfer
C
      VCS1=DSQRT(MU/A0)
      VCS2=DSQRT(MU/A0/Y2F)
C
      DELVSP=VCS1-VCS2
      DELVSM=DELVSP*1000.
      VSPBAR=DELVSP/A0/MANOMO
      TOFSP=1./MDOTBAR*(1.-DEXP(-DELVSP/(9.81D-3*ISP)))/3600./24.
      TOFSPY=TOFSP/365.25
```

```
C
C
C     * Hohmann Transfer
C
      V1=DSQRT(2*MU*(1./A0-1./(A0+A0*Y2F)))
      V2=DSQRT(2*MU*(1./A0/Y2F-1./(A0+A0*Y2F)))
      DELVHM=(V1-VCS1)+(VCS2-V2)
      DELVHMM=DELVHM*1000.
      VHMBAR=DELVHM/A0/MANOMO
C
C
      WRITE (*,*)
      WRITE (*,160) DELV,DELVM,VBAR
      WRITE (*,165) TOF,TOFY
      WRITE (*,*)
      WRITE (*,*)
      WRITE (*,*) '  Comparison Transfer Data'
      WRITE (*,*) '  ************************'
      WRITE (*,*)
      WRITE (*,*) '  Spiral Transfer'
      WRITE (*,*)
      WRITE (*,160) DELVSP,DELVSM,VSPBAR
      WRITE (*,165) TOFSP,TOFSPY
      WRITE (*,*)
      WRITE (*,*) '  Hohmann Transfer'
      WRITE (*,*)
      WRITE (*,160) DELVHM,DELVHMM,VHMBAR
      WRITE (*,*)
      WRITE (*,*)
      WRITE (*,*)
C
      IF (IOUT.GE.1) THEN
        WRITE (9,*)
        WRITE (9,160) DELV,DELVM,VBAR
        WRITE (9,165) TOF,TOFY
        WRITE (9,*)
        WRITE (9,*)
        WRITE (9,*) '  Comparison Transfer Data'
        WRITE (9,*) '  ************************'
        WRITE (9,*)
        WRITE (9,*) '  Spiral Transfer'
        WRITE (9,*)
        WRITE (9,160) DELVSP,DELVSM,VSPBAR
        WRITE (9,165) TOFSP,TOFSPY
        WRITE (9,*)
        WRITE (9,*) '  Hohmann Transfer'
        WRITE (9,*)
        WRITE (9,160) DELVHM,DELVHMM,VHMBAR
        WRITE (9,*)
        WRITE (9,*)
        WRITE (9,*)
        ENDIF
```

```
C
      IF (ITYP.EQ.1) THEN
        WRITE (10,*) Y2F,VBAR
        WRITE (11,*) Y2F,VSPBAR
        WRITE (12,*) Y2F,VHMBAR
        ENDIF
C
  160 FORMAT (3X,'Total Transfer Delta-V:  ',F8.4,' km/s  (',F8.2,
     +         ' m/s)',6X,'Nondimensional Delta-V:  ',F8.4,' /sec')
  165 FORMAT (3X,'Total Transfer Time:  ',F9.3,' days   (',
     +         F7.3,' yr)')
C
C
      DIFFYF=Y2FCK-Y2F
      IF (DABS(DIFFYF).GT.1.E-13) GOTO 10
C
C
  170 CONTINUE
C
C
      IF (IOUT.GE.1) CLOSE (9)
      IF (ITYP.EQ.1) THEN
        CLOSE (10)
        CLOSE (11)
        CLOSE (12)
        ENDIF
C
C
      STOP
      END
C
C
C
C     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C     *                         SUBROUTINE RHS                        *
C     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
      SUBROUTINE RHS(NXT)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      COMMON /HAM/ VBAR,Y(42,4),F(42,4),ERR(42),N,H,MODE
      COMMON /RH/ RABAR(4),RPBAR(4),ICASE,ICNT,DIFF
C
C
      CALL INTERPOA(Y(1,NXT),DELAA,DELEA,ICNT)
      CALL INTERPOP(Y(1,NXT),DELAP,DELEP,ICNT)
      ICNT=ICNT+1
C
C     * Define Derivatives
C
      PI=DBLE(ACOS(-1.0))
C
      IF (ICASE.LT.20) THEN
```

```
        DELE=DELEP
        DELA=DELAP
        F(1,NXT)=Y(2,NXT)**.5*(DELEP-DELEA)/(4.*PI)
        F(2,NXT)=Y(2,NXT)**1.5*(DELAP-DELAA)/(4.*PI)
        ENDIF
C
      IF (ICASE.GT.20) THEN
        DELE=DELEA
        DELA=DELAA
        F(1,NXT)=Y(2,NXT)**.5*(DELEA-DELEP)/(4.*PI)
        F(2,NXT)=Y(2,NXT)**1.5*(DELAA-DALAP)/(4.*PI)
        ENDIF
C
      IF (F(1,NXT).LT.0.D0) THEN
        F(1,NXT)=Y(2,NXT)**.5*DELE/(2.*PI)
        F(2,NXT)=Y(2,NXT)**1.5*DELA/(2.*PI)
        WRITE (*,50) Y(1,NXT),Y(2,NXT)
        ENDIF
C
C     * Define Changes in ra-bar and rp-bar
C
      RABAR(NXT)=Y(2,NXT)*(1.+Y(1,NXT))
      RPBAR(NXT)=Y(2,NXT)*(1.-Y(1,NXT))
C
   50 FORMAT (3X,'Single Orbit Performed at e = ',E13.7,' and abar = ',
     +        E13.7)
      RETURN
      END
C
C
      SUBROUTINE HAMING(NXT)
C VERSION OF 11/20/1987
C PURPOSE
C   HAMING IS AN ORDINARY DIFFERENTIAL EQUATIONS INTEGRATOR
C   IT IS A FOURTH ORDER PREDICTOR-CORRECTOR ALGORITHM WHICH
C   MEANS THAT IT CARRIES THE LAST FOUR VALUES OF THE STATE
C   VECTOR, AND EXTRAPOLATES THESE VALUES TO OBTAIN A PREDICTED
C   NEXT VALUE (THE PREDICTION STEP) AND EVALUATES THE EQUATIONS
C   OF MOTION AT THE PREDICTED POINT, AND THEN CORRECTS THE
C   EXTRAPOLATED POINT USING A HIGHER ORDER POLYNOMIAL (THE
C   CORRECTION STEP).
C INPUT
C   NXT --  IN THE CALL SPECIFIES WHICH OF THE FOUR VALUES OF
C           THE STATE VECTOR IS THE CURRENT ONE.  NXT IS UPDATED
C           BY HAMING AUTOMATICALLY, BUT MUST BE SET TO ZERO ON
C           THE FIRST CALL.
C CALL ROUTINES
C   RHS(NXT)
C REFERENCES
C   WILLIAM WEISEL
C PROGRAMMER
C   RODNEY D. BAIN
```

```
C PROGRAM MODIFICATIONS
C   NONE
C COMMENTS
C   TOL       -- IS HAMING'S START UP TOLERANCE ... SET TO REASONABLE VALUE
C                AS NECESSARY
C   THE COMMON BLOCK CONTAINS:
C     VBAR      -- IS THE INDEPENDENT VARIABLE (OFTEN TIME)
C     Y(42,4) -- IS THE STATE VECTOR, 4 COPIES OF IT, WITH NXT POINTING
C                POINTING TO THE CURRENT ONE, THE LIMIT OF 42 EQUATIONS
C                OF MOTION CAN BE CHANGED.
C     F(42,4) -- ARE THE EQUATIONS OF MOTION EVALUATED AT THE SAME TIMES
C                AS THE STATE VECTOR Y ... IT IS THE JOB OF SUBROUTINE
C                RHS TO CALCULATE THESE.
C     ERR(42) -- IS AN ESTIMATE OF THE ONE-STEP INTEGRATION ERROR
C     N       -- IS THE NUMBER OF ODES ... LIMIT IS 42 UNLESS YOU CHANGE
C                THE COMMON BLOCK
C     H       -- IS THE TIMESTEP ... ONE CALL TO HAMING INCREMENTS X BY H
C     MODE    -- IS ZERO FOR EOM ONLY, 1 FOR EOM AND EOV
C   THE USER MUST SUPPLY A MAIN PROGRAM, AND THE SUBROUTINE RHS(NXT) WHICH
C   EVALUATES THE EQUATIONS OF MOTION.
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      COMMON /HAM/ VBAR,Y(42,4),F(42,4),ERR(42),N,H,MODE
      DATA ZERO,ONE,TWO,THREE,FOUR/0.D0,1.D0,2.D0,3.D0,4.D0/
      TOL=1.D-12
C
C   CHECK IF THIS IS THE FIRST CALL ... HAMING (LIKE ALL PREDICTOR-
C   CORRECTORS) NEEDS 'PREVIOUS' VALUES
C
      IF(NXT) 190,10,200
C
C   IT IS A PICARD INTERATION (SLOW AND EXPENSIVE) TO STEP BACKWARDS
C   IN TIME THREE STEPS TO GET THE 4 PREVIOUS POINTS.  A SUCCESSFUL
C   STARTUP RETURNS NXT=1, AND TIME HAS NOT BEEN INCREMENTED.  IF
C   STARTUP FAILS, NXT WILL BE RETURNED AS ZERO.
C
 10   XO=VBAR
      HH=H/TWO
      CALL RHS(1)
      DO 40 L=2,4
         VBAR=VBAR+HH
           DO 20 I=1,N
              Y(I,L)=Y(I,L-1)+HH*F(I,L-1)
 20        CONTINUE
         CALL RHS(L)
         VBAR=VBAR+HH
           DO 30 I=1,N
              Y(I,L)=Y(I,L-1)+H*F(I,L)
 30        CONTINUE
         CALL RHS(L)
 40   CONTINUE
      JSW=-10
```

```
 50   ISW=1
      DO 120 I=1,N
         HH=Y(I,1)+H*(9.D0*F(I,1)+19.D0*F(I,2)-5.D0*F(I,3)
     1      +F(I,4))/24.D0
      IF(DABS(HH-Y(I,2)).LT.TOL) GOTO 70
      ISW=0
 70   Y(I,2)=HH
      HH=Y(I,1)+H*(F(I,1)+FOUR*F(I,2)+F(I,3))/THREE
      IF(DABS(HH-Y(I,3)).LT.TOL) GOTO 90
      ISW=0
 90   Y(I,3)=HH
      HH=Y(I,1)+H*(THREE*F(I,1)+9.D0*F(I,2)+9.D0*F(I,3)
     1   +THREE*F(I,4))/8.D0
      IF(DABS(HH-Y(I,4)).LT.TOL) GOTO 110
      ISW=0
 110  Y(I,4)=HH
 120  CONTINUE
      VBAR=XO
      DO 130 L=2,4
         VBAR=VBAR+H
         CALL RHS(L)
 130  CONTINUE
      IF(ISW) 140,140,150
 140  JSW=JSW+1
      IF(JSW) 50,280,280
 150  VBAR=XO
      ISW=1
      JSW=1
      DO 160 I=1,N
         ERR(I)=ZERO
 160  CONTINUE
      NXT=1
      GOTO 280
C
C   A CALL TO HAMING WITH NXT=-NXT, AFTER A SUCCESSFUL STARTUP,
C   WILL TURN OFF THE SECOND EVALUATION OF THE EQUATIONS OF MOTION
C   FOLLOWING THE CORRECTOR STEP.  IN SYSTEMS WHERE THE EQUATIONS OF
C   MOTION ARE VERY EXPENSIVE, THIS CAN HALVE YOUR RUN TIME.
C
 190  JSW=2
      NXT=IABS(NXT)
C
C   THIS IS THE PREDICTOR-CORRECTOR ALGORITHM ... FIRST THE INDICES
C   ARE PERMUTED
C
 200  VBAR=VBAR+H
      NP1=MOD(NXT,4)+1
      GOTO (210,230),ISW
 210  GOTO (270,270,270,220),NXT
 220  ISW=2
 230  NM2=MOD(NP1,4)+1
      NM1=MOD(NM2,4)+1
```

```
      NPO=MOD(NM1,4)+1
C
C   ... THEN THE PREDICTOR PART IS RUN TO FIND AN EXTRAPOLATED VALUE
C   OF THE STATE VECTOR AT THE NEW TIME ...
C
      DO 240 I=1,N
         F(I,NM2)=Y(I,NP1)+FOUR*H*(TWO*F(I,NPO)-F(I,NM1)
     1            +TWO*F(I,NM2))/THREE
         Y(I,NP1)=F(I,NM2)-0.925619835D0*ERR(I)
 240  CONTINUE
C
C   THE EQUATIONS OF MOTION ARE EVALUATED AT THE EXTRAPOLATED VALUE
C   OF THE STATE VECTOR ...
C
      CALL RHS(NP1)
C
C   AND THE CORRECTOR ALGORITHM IS USED TO ADD THIS NEW INFORMATION
C   AND OBTAIN A BETTER VALUE OF THE NEW STATE VECTOR ...
C
      DO 250 I=1,N
         Y(I,NP1)=(9.D0*Y(I,NPO)-Y(I,NM2)+THREE*H*(F(I,NP1)
     1            +TWO*F(I,NPO)-F(I,NM1)))/8.D0
         ERR(I)=F(I,NM2)-Y(I,NP1)
         Y(I,NP1)=Y(I,NP1)+0.0743801653D0*ERR(I)
 250  CONTINUE
      GOTO (260,270),JSW
C
C   FINALLY, THE EQUATIONS OF MOTION ARE RE-EVALUATED AT THE BETTER
C   VALUE OF THE STATE VECTOR ... THIS CAN BE SUPPRESSED.
C
 260  CALL RHS(NP1)
 270  NXT=NP1
 280  RETURN
      END
C
C
C     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C     *                       SUBROUTINE INTERP3                        *
C     *                                                                 *
C     *   The following subroutine interpolates between four data      *
C     *             points using a third order polynomial.              *
C     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
      SUBROUTINE INTERP3 (XBAR,X,D,P)
C
      IMPLICIT DOUBLE PRECISION (A-H,L-Z)
      DIMENSION X(4),P1(3),D(4)
C
      DO 300 K=1,3
        DO 300 J=(K+1),4
          J1=4-J+(K+1)
          D(J1)=(D(J1)-D(J1-1))/(X(J1)-X(J1-K))
```

```
  300     CONTINUE
C
      Z=XBAR-X(1)
      P1(1)=D(1)+D(2)*Z
      DO 310 J=2,3
        Z=Z*(XBAR-X(J))
        P1(J)=P1(J-1)+D(J+1)*Z
  310   CONTINUE
C
      P=P1(3)
C
      RETURN
      END
C
C
C
C
C     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C     *                    SUBROUTINE INTERPO                       *
C     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
C     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C     *  The following subroutine utilizes the Newton formula       *
C     *  to interpolate/extrapolate from the given data sets        *
C     *  utilizing an 4th degree interpolating polynomial.          *
C     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
C
      SUBROUTINE INTERPOA(EBAR,DA,DE,ICNT)
C
      IMPLICIT DOUBLE PRECISION (A-H,L-Z)
      DIMENSION E(100),LAMDA(100),DELA(100),DELE(100)
C
C
C     * OPEN DATA FILES *
C     *******************
C
C     * Constant ra Data
C
      IF (ICNT.EQ.0) THEN
        OPEN (UNIT=15, STATUS = 'OLD',FILE = 'ALAM.DAT')
        OPEN (UNIT=16, STATUS = 'OLD',FILE = 'ADELA.DAT')
        OPEN (UNIT=17, STATUS = 'OLD',FILE = 'ADELE.DAT')
        IDATA=99
        ENDIF
C
C
      IF (ICNT.EQ.0) THEN
C
        REWIND (15)
        REWIND (16)
        REWIND (17)
```

```
C
C
C     * INPUT INITIAL DATA *
C     **********************
C
        DO 5, I=1,IDATA
          READ (15,*) E(I),LAMDA(I)
          READ (16,*) E(I),DELA(I)
          READ (17,*) E(I),DELE(I)
    5     CONTINUE
C
        IDEG=4
C
        CLOSE (15)
        CLOSE (16)
        CLOSE (17)
C
        ENDIF
C
C
C     * EVALUATE FUNCTION AT SPECIFIED VALUES OF ECCENTRICITY *
C     *********************************************************
C
C     * Check location of data point *
C
      DO 30 I=1,IDATA
        DIFF=EBAR-E(I)
        IF (ABS(DIFF).LE.1.E-15) THEN
          LAM=LAMDA(I)
          DA=DELA(I)
          DE=DELE(I)
          GO TO 40
          ENDIF
C
        IF ((I.EQ.1).AND.(DIFF.LT.0.)) THEN
          IMIN=1
          IMAX=IMIN+IDEG
          GO TO 35
          ENDIF
C
        IF ((I.EQ.IDATA).AND.(DIFF.GT.0.)) THEN
          IMAX=IDATA
          IMIN=IMAX-IDEG
          GO TO 35
          ENDIF
C
        ILEFT=I
        IF (DIFF.LT.0.) GO TO 31
C
   30   CONTINUE
C
C
```

```
   31 IDEGP1=IDEG+1
      IDEGL=IDEGP1/2
      IMIN=ILEFT+1-IDEGL
      IMAX=IMIN+IDEG
C
      IF (IMIN.LT.1) THEN
        IMIN=1
        IMAX=IMIN+IDEG
        ENDIF
C
      IF (IMAX.GT.IDATA) THEN
        IMAX=IDATA
        IMIN=IMAX-IDEG
        ENDIF
C
C
C     * Evaluate Polynomial *
C
   35 CALL INTERP(IDEG,IMIN,IMAX,EBAR,E,LAMDA,DELA,DELE,LAM,DA,DE)
C
   40 CONTINUE
      RETURN
      END
C
C
C
      SUBROUTINE INTERPOP(EBAR,DA,DE,ICNT)
C
      IMPLICIT DOUBLE PRECISION (A-H,L-Z)
      DIMENSION E(100),LAMDA(100),DELA(100),DELE(100)
C
C
C     * OPEN DATA FILES *
C     *******************
C
C     * Constant rp Data
C
      IF (ICNT.EQ.0) THEN
        OPEN (UNIT=18, STATUS = 'OLD',FILE = 'PLAM.DAT')
        OPEN (UNIT=19, STATUS = 'OLD',FILE = 'PDELA.DAT')
        OPEN (UNIT=20, STATUS = 'OLD',FILE = 'PDELE.DAT')
        IDATA=98
        ENDIF
C
C
      IF (ICNT.EQ.0) THEN
C
        REWIND (18)
        REWIND (19)
        REWIND (20)
C
C
```

```
C     * INPUT INITIAL DATA *
C     **********************
C
        DO 5, I=1,IDATA
          READ (18,*) E(I),LAMDA(I)
          READ (19,*) E(I),DELA(I)
          READ (20,*) E(I),DELE(I)
    5     CONTINUE
C
        IDEG=4
C
        CLOSE (18)
        CLOSE (19)
        CLOSE (20)
C
        ENDIF
C
C
C     * EVALUATE FUNCTION AT SPECIFIED VALUES OF ECCENTRICITY *
C     *********************************************************
C
C     * Check location of data point *
C
      DO 30 I=1,IDATA
        DIFF=EBAR-E(I)
        IF (ABS(DIFF).LE.1.E-15) THEN
          LAM=LAMDA(I)
          DA=DELA(I)
          DE=DELE(I)
          GO TO 40
          ENDIF
C
        IF ((I.EQ.1).AND.(DIFF.LT.0.)) THEN
          IMIN=1
          IMAX=IMIN+IDEG
          GO TO 35
          ENDIF
C
        IF ((I.EQ.IDATA).AND.(DIFF.GT.0.)) THEN
          IMAX=IDATA
          IMIN=IMAX-IDEG
          GO TO 35
          ENDIF
C
        ILEFT=I
        IF (DIFF.LT.0.) GO TO 31
C
   30   CONTINUE
C
C
   31 IDEGP1=IDEG+1
      IDEGL=IDEGP1/2
```

```
      IMIN=ILEFT+1-IDEGL
      IMAX=IMIN+IDEG
C
      IF (IMIN.LT.1) THEN
        IMIN=1
        IMAX=IMIN+IDEG
        ENDIF
C
      IF (IMAX.GT.IDATA) THEN
        IMAX=IDATA
        IMIN=IMAX-IDEG
        ENDIF
C
C
C     * Evaluate Polynomial *
C
   35 CALL INTERP(IDEG,IMIN,IMAX,EBAR,E,LAMDA,DELA,DELE,LAM,DA,DE)
C
   40 CONTINUE
      RETURN
      END
C
C
C
C     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C     *                    SUBROUTINE INTERP                          *
C     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
C     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C     *     The following algorithm evaluates an interpolating       *
C     *                   nth degree polynomial                       *
C     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
      SUBROUTINE INTERP(IDEG,IMIN,IMAX,EBAR,E,LAMDA,DELA,DELE,LAM,DA,DE)
C
      IMPLICIT DOUBLE PRECISION (A-H,L-Z)
      DIMENSION E(100),LAMDA(100),DELA(100),DELE(100)
      DIMENSION X(100),D(100),P(100)
C
      IDEGP1=IDEG+1
C
      DO 200 I=1,3
C
        DO 150 J=IMIN,IMAX
          K=J-IMIN+1
          IF (I.EQ.1) D(K)=LAMDA(J)
          IF (I.EQ.2) D(K)=DELA(J)
          IF (I.EQ.3) D(K)=DELE(J)
          X(K)=E(J)
  150     CONTINUE
C
        DO 155 K=1,IDEG
```

```
         DO 155 J=(K+1),IDEGP1
           J1=IDEGP1-J+(K+1)
           D(J1)=(D(J1)-D(J1-1))/(X(J1)-X(J1-K))
  155      CONTINUE
C
       Z=EBAR-X(1)
       P(1)=D(1)+D(2)*Z
C
       DO 160 J=2,IDEG
         Z=Z*(EBAR-X(J))
         P(J)=P(J-1)+D(J+1)*Z
  160    CONTINUE
C
       IF (I.EQ.1) LAM=P(IDEG)
       IF (I.EQ.2) DA=P(IDEG)
       IF (I.EQ.3) DE=P(IDEG)
C
  200  CONTINUE
C
      RETURN
      END
```

Appendix D-1: Program TRANSMUL Data - LEO to GEO Transfer

Orbit Data

Initial Orbit Eccentricity: 0.000 Initial Orbit Semimajor Axis: 6678.14 km (0.1000000E+01)
Final Orbit Eccentricity: 0.000 Final Orbit Semimajor Axis: 42241.15 km (0.6325281E+01)

Primary Body Data

Gravitational Parameter: 398,601 km^3/sec^2 Radius: 6378.145 km

Transfer Vehicle Data

Total Initial Mass: 5000.00 kg Specific Impulse: 863.00 sec
Propellant Mass Flow Rate: 0.3977100E-03kg/sec (0.7954200E-07 /sec)

Rev	Time (sec)	NU (Rad)	V-bar	e	a-bar	ra-bar	rp-bar
1	0.0000E+00	0.000	0.0000E+00	0.0000E+00	0.1000E+01	0.1000E+01	0.1000E+01
50	0.2723E+06	2.196	0.2400E-01	0.2568E-01	0.1026E+01	0.1052E+01	0.1000E+01
100	0.5607E+06	3.591	0.5000E-01	0.5348E-01	0.1056E+01	0.1113E+01	0.1000E+01
150	0.8637E+06	3.084	0.7800E-01	0.8333E-01	0.1090E+01	0.1181E+01	0.1000E+01
201	0.1190E+07	0.554	0.1090E+00	0.1162E+00	0.1131E+01	0.1263E+01	0.1000E+01
251	0.1527E+07	1.316	0.1420E+00	0.1511E+00	0.1178E+01	0.1356E+01	0.1000E+01
301	0.1894E+07	0.507	0.1790E+00	0.1899E+00	0.1234E+01	0.1469E+01	0.1000E+01
349	0.2277E+07	6.258	0.2190E+00	0.2315E+00	0.1301E+01	0.1602E+01	0.1000E+01
400	0.2655E+07	4.517	0.2600E+00	0.2736E+00	0.1376E+01	0.1753E+01	0.1000E+01
450	0.3054E+07	4.556	0.3050E+00	0.3190E+00	0.1468E+01	0.1937E+01	0.1000E+01
501	0.3420E+07	0.235	0.3480E+00	0.3617E+00	0.1566E+01	0.2133E+01	0.1000E+01
550	0.3828E+07	0.535	0.3980E+00	0.4102E+00	0.1695E+01	0.2391E+01	0.1000E+01
600	0.4218E+07	2.889	0.4480E+00	0.4574E+00	0.1843E+01	0.2686E+01	0.1000E+01
650	0.4613E+07	0.827	0.5010E+00	0.5058E+00	0.2023E+01	0.3047E+01	0.1000E+01
700	0.4995E+07	1.249	0.5550E+00	0.5533E+00	0.2238E+01	0.3477E+01	0.1000E+01
750	0.5432E+07	1.126	0.6200E+00	0.6076E+00	0.2550E+01	0.4100E+01	0.1000E+01
775	0.5669E+07	3.036	0.6570E+00	0.6375E+00	0.2758E+01	0.4517E+01	0.1000E+01
800	0.5892E+07	6.011	0.6930E+00	0.6653E+00	0.2988E+01	0.4977E+01	0.1000E+01
825	0.6102E+07	3.337	0.7280E+00	0.6914E+00	0.3241E+01	0.5482E+01	0.1000E+01
849	0.6368E+07	5.017	0.7740E+00	0.7243E+00	0.3627E+01	0.6254E+01	0.1000E+01
850	0.6389E+07	4.715	0.8085E+00	0.7269E+00	0.3662E+01	0.6325E+01	0.1000E+01
855	0.6552E+07	1.431	0.8376E+00	0.6710E+00	0.3785E+01	0.6325E+01	0.1245E+01
860	0.6765E+07	1.006	0.8772E+00	0.5884E+00	0.3982E+01	0.6325E+01	0.1638E+01
865	0.6992E+07	0.781	0.9208E+00	0.4893E+00	0.4247E+01	0.6325E+01	0.2168E+01
870	0.7250E+07	3.835	0.9728E+00	0.3624E+00	0.4642E+01	0.6325E+01	0.2959E+01
875	0.7526E+07	0.281	0.1031E+01	0.2119E+00	0.5219E+01	0.6325E+01	0.4112E+01
879	0.7878E+07	4.915	0.1110E+01	-0.3053E-10	0.6325E+01	0.6325E+01	0.6325E+01

Total Transfer Delta-V: 8.5780 km/s (8577.98 m/s)
Total Transfer Time: 92.682 days (0.25 yr)

Comparison Transfer Data

Spiral Transfer

Total Transfer Delta-V: 4.6539 km/s (4653.90 m/s)
Total Transfer Time: 61.534 days (0.17 yr)

Hohmann Transfer

Total Transfer Delta-V: 3.8937 km/s (3893.75 m/s)

Appendix D-2: Program TRANSALT Data - LEO to GEO Transfer

Orbit Data

Initial Orbit Eccentricity: 0.000 Initial Orbit Semimajor Axis: 6678.14 km (0.1000000E+01)
Final Orbit Eccentricity: 0.000 Final Orbit Semimajor Axis: 42241.15 km (0.6325281E+01)

Primary Body Data

Gravitational Parameter: 398,601 km^3/sec^2 Radius: 6378.145 km

Transfer Vehicle Data

Total Initial Mass: 5000.00 kg Specific Impulse: 863.00 sec
Propellant Mass Flow Rate: 0.3977100E-03 kg/sec (0.7954200E-07 /sec)

V-bar	e	a-bar	ra-bar	rp-bar
0.0000000000E+00	0.0000000000E+00	0.1000000000E+01	0.1000000000E+01	0.1000000000E+01
0.5000000000E-01	0.1302531479E-08	0.1055770406E+01	0.1055770408E+01	0.1055770405E+01
0.1000000000E+00	0.2603426238E-08	0.1116339822E+01	0.1116339825E+01	0.1116339820E+01
0.1500000000E+00	0.3902640116E-08	0.1182275068E+01	0.1182275073E+01	0.1182275064E+01
0.2000000000E+00	0.5200126439E-08	0.1254229190E+01	0.1254229196E+01	0.1254229183E+01
0.2500000000E+00	0.6495835801E-08	0.1332957690E+01	0.1332957699E+01	0.1332957681E+01
0.3000000000E+00	0.7789715809E-08	0.1419338442E+01	0.1419338453E+01	0.1419338431E+01
0.3500000000E+00	0.9081710809E-08	`.1514396269E+01	0.1514396283E+01	0.1514396255E+01
0.4000000000E+00	0.1037176156E-07	0.1619333479E+01	0.1619333496E+01	0.1619333462E+01
0.4500000000E+00	0.1165980488E-07	0.1735568073E+01	0.1735568093E+01	0.1735568052E+01
0.5000000000E+00	0.1294577325E-07	0.1864781897E+01	0.1864781921E+01	0.1864781873E+01
0.5500000000E+00	0.1422959429E-07	0.2008981829E+01	0.2008981858E+01	0.2008981801E+01
0.6000000000E+00	0.1551119029E-07	0.2170578175E+01	0.2170578209E+01	0.2170578141E+01
0.6500000000E+00	0.1679047753E-07	0.2352486046E+01	0.2352486085E+01	0.2352486006E+01
0.7000000000E+00	0.1806736559E-07	0.2558257756E+01	0.2558257802E+01	0.2558257709E+01
0.7500000000E+00	0.1934175646E-07	0.2792257572E+01	0.2792257626E+01	0.2792257518E+01
0.8000000000E+00	0.2061354358E-07	0.3059895048E+01	0.3059895111E+01	0.3059894985E+01
0.8500000000E+00	0.2188261064E-07	0.3367940493E+01	0.3367940566E+01	0.3367940419E+01
0.9000000000E+00	0.2314883016E-07	0.3724957392E+01	0.3724957478E+01	0.3724957306E+01
0.9500000000E+00	0.2441206177E-07	0.4141904050E+01	0.4141904151E+01	0.4141903949E+01
0.1000000000E+01	0.2567215017E-07	0.4632984627E+01	0.4632984746E+01	0.4632984509E+01
0.1050000000E+01	0.2692892257E-07	0.5216875095E+01	0.5216875236E+01	0.5216874955E+01
0.1100000000E+01	0.2818218550E-07	0.5918525489E+01	0.5918525656E+01	0.5918525322E+01
0.1125094975E+01	0.2880980737E-07	0.6325281014E+01	0.6325281197E+01	0.6325280832E+01

Total Transfer Delta-V: 8.6922 km/s (8692.22 m/s) Nondimensional Delta-V: 1.1251 /sec
Total Transfer Time: 93.390 days (0.256 yr)

<u>Comparison Transfer Data</u>

Spiral Transfer

Total Transfer Delta-V: 4.6539 km/s (4653.90 m/s) Nondimensional Delta-V: 0.6024 /sec
Total Transfer Time: 61.534 days (0.168 yr)

Hohmann Transfer

Total Transfer Delta-V: 3.8937 km/s (3893.75 m/s) Nondimensional Delta-V: 0.5040 /sec

Appendix D-3: Program TRANSMUL Data - Transfer to Molniya Orbit

Orbit Data

Initial Orbit Eccentricity: 0.000 Initial Orbit Semimajor Axis: 7184.76 km (0.1000000E+01)
Final Orbit Eccentricity: 0.730 Final Orbit Semimajor Axis: 26610.23 km (0.3703705E+01)

Primary Body Data

Gravitational Parameter: 398,601 km^3/sec^2 Radius: 6378.145 km

Transfer Vehicle Data

Total Initial Mass: 5000.00 kg Specific Impulse: 863.00 sec
Propellant Mass Flow Rate: 0.3977100E-03kg/sec (0.7954200E-07 /sec)

Rev	Time (sec)	NU (Rad)	V-bar	e	a-bar	ra-bar	rp-bar
1	0.0000E+00	0.000	0.0000E+00	0.0000E+00	0.1000E+01	0.1000E+01	0.1000E+01
50	0.3059E+06	3.138	0.2800E-01	0.2996E-01	0.1030E+01	0.1061E+01	0.1000E+01
101	0.6359E+06	2.159	0.5900E-01	0.6308E-01	0.1067E+01	0.1134E+01	0.1000E+01
150	0.9672E+06	0.332	0.9100E-01	0.9716E-01	0.1107E+01	0.1215E+01	0.1000E+01
200	0.1338E+07	1.554	0.1280E+00	0.1363E+00	0.1157E+01	0.1315E+01	0.1000E+01
250	0.1746E+07	0.156	0.1700E+00	0.1805E+00	0.1220E+01	0.1440E+01	0.1000E+01
300	0.2148E+07	1.396	0.2130E+00	0.2253E+00	0.1290E+01	0.1581E+01	0.1000E+01
325	0.2330E+07	2.174	0.2330E+00	0.2459E+00	0.1326E+01	0.1652E+01	0.1000E+01
350	0.2544E+07	3.604	0.2570E+00	0.2705E+00	0.1370E+01	0.1741E+01	0.1000E+01
376	0.2727E+07	0.468	0.2780E+00	0.2919E+00	0.1412E+01	0.1824E+01	0.1000E+01
400	0.2933E+07	5.557	0.3020E+00	0.3160E+00	0.1462E+01	0.1924E+01	0.1000E+01
425	0.3134E+07	3.548	0.3260E+00	0.3400E+00	0.1515E+01	0.2030E+01	0.1000E+01
450	0.3340E+07	4.354	0.3510E+00	0.3646E+00	0.1574E+01	0.2148E+01	0.1000E+01
475	0.3533E+07	0.378	0.3750E+00	0.3880E+00	0.1634E+01	0.2268E+01	0.1000E+01
500	0.3768E+07	5.616	0.4050E+00	0.4169E+00	0.1715E+01	0.2430E+01	0.1000E+01
525	0.4005E+07	4.157	0.4360E+00	0.4462E+00	0.1805E+01	0.2611E+01	0.1000E+01
550	0.4257E+07	3.334	0.4700E+00	0.4777E+00	0.1914E+01	0.2829E+01	0.1000E+01
575	0.4474E+07	3.956	0.5000E+00	0.5049E+00	0.2019E+01	0.3039E+01	0.1000E+01
600	0.4692E+07	4.796	0.5310E+00	0.5324E+00	0.2138E+01	0.3277E+01	0.1000E+01
625	0.4971E+07	2.469	0.5720E+00	0.5678E+00	0.2314E+01	0.3628E+01	0.1000E+01
653	0.5247E+07	4.174	0.6140E+00	0.6029E+00	0.2518E+01	0.4037E+01	0.1000E+01
675	0.5469E+07	2.338	0.6490E+00	0.6311E+00	0.2711E+01	0.4422E+01	0.1000E+01
699	0.5726E+07	6.155	0.6910E+00	0.6638E+00	0.2974E+01	0.4949E+01	0.1000E+01
700	0.5732E+07	5.962	0.6920E+00	0.6646E+00	0.2981E+01	0.4963E+01	0.1000E+01
725	0.5986E+07	0.467	0.7350E+00	0.6966E+00	0.3296E+01	0.5592E+01	0.1000E+01
745	0.6254E+07	3.103	0.7822E+00	0.7300E+00	0.3703E+01	0.6407E+01	0.1000E+01

Total Transfer Delta-V: 5.8264 km/s (5826.39 m/s)
Total Transfer Time: 72.394 days (0.20 yr)

Bibliography

1. Alfano, Capt Salvatore. Low Thrust Orbit Transfer. MS thesis, AFIT/GA/AA/82D-2. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, December 1982.

2. Bate, Roger R. et al. Fundamentals of Astrodynamics. New York: Dover Publications, INC., 1971.

3. Bryson, Arthur E. and Ho, Yu-Chi. Applied Optimal Control. Washington DC: Hemisphere Publishing Corporation, 1975.

4. Byers, David C. Upper Stages Utilizing Electric Propulsion. NASA TM-81412. National Aeronautics and Space Administration, 1980.

5. Kreyszig, Erwin. Advanced Engineering Mathematics (Fifth Edition). New York: John Wiley & Sons, 1983.

6. National Aeronautics and Space Administration. Ion Propulsion for Spacecraft. Washington DC: U.S. Government Printing Office, 1977.

7. National Aeronautics and Space Administration, Lewis Research Center, Cleveland Ohio. Telephone interviews with Mr Mike Patterson, September thru November 1988.

8. Rocket Research Company, Redmond, Washington. Telephone interviews with Mr Joseph Cassady, Senior Development Engineer, October 1988.

9. Ruddy, John M. "A Novel Non-Geostationary Satellite Communications System," Institute of Electrical and Electronics Engineers 1981 International Communications Conference, Volume III. IEEE Publishing Services, 1981. Catalog No. 81CH1648-5

10. Smith, W. W. and Knowles, S. C. "Analysis of Electric Propulsion Concepts for Near-Term Mission Applications," 1985 Joint Army-Navy-NASA-Air Force Interagency Propulsion Meeting, Volume I. CPIA Publication 425, April 1985. Contract N00024-85-C-5301.

11. Sutton, George P. and Ross, Donald M. Rocket Propulsion Elements (Fourth Edition). New York: John Wiley & Sons, 1976.

12. United States Air Force Astronautics Lab, California. Telephone conversations with Capt Wayne Schmidt, AFAL/LSVE, April thru November 1988.

13. Vandergraft, James S. Introduction to Numerical Computations (Second Edition). New York: Academic Press, INC., 1983.

14. Wiesel, W. MECH 636, Advanced Astrodynamics class notes. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, October 1988.

Vita

Captain Gregory Leon Beeker, [redacted] He spent his childhood in London Kentucky, [redacted] He attended the University of Kentucky, Lexington Technical Institute where he received an Associates Degree in Architectural Technology in May 1980. After spending a third year at the University of Kentucky School of Engineering, he entered the United States Air Force Reserve Officer Training Corps (ROTC) and attended a six week training program at Eglin AFB, Florida during the summer of 1981. The following fall, he transferred to the Purdue University School of Aeronautics and Astronautics on a two year ROTC scholarship. In May 1983, he received the degree of Bachelor of Science in Aeronautical and Astronautical Engineering. Upon graduation, he was commissioned as a Second Lieutenant in the United States Air Force. In October 1983, he was assigned to the 6595 Shuttle Test Group, Vandenberg AFB, California, as an Astronaut Systems Engineer for the Vandenberg Space Transportation System (Space Shuttle) Program, where he served until entering the School of Engineering, Air Force Institute of Technology, in May 1987. Upon graduation from the Air Force Institute of

Technology, Captain Beeker will be assigned to the 6520 Test Group at Edwards AFB, California, as a Flight Test Engineer.

[redacted]

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION
UNCLASSIFIED

1b. RESTRICTIVE MARKINGS

2a. SECURITY CLASSIFICATION AUTHORITY

2b. DECLASSIFICATION / DOWNGRADING SCHEDULE

3. DISTRIBUTION/AVAILABILITY OF REPORT
Approved for public distribution;
distribution unlimited

4. PERFORMING ORGANIZATION REPORT NUMBER(S)
AFIT/GA/AA/88D-01

5. MONITORING ORGANIZATION REPORT NUMBER(S)

6a. NAME OF PERFORMING ORGANIZATION
School of Engineering

6b. OFFICE SYMBOL (If applicable)
AFIT/ENY

7a. NAME OF MONITORING ORGANIZATION

6c. ADDRESS (City, State, and ZIP Code)
Air Force Institute of Technology (AU)
Wright-Patterson AFB OH 45433-6583

7b. ADDRESS (City, State, and ZIP Code)

8a. NAME OF FUNDING / SPONSORING ORGANIZATION

8b. OFFICE SYMBOL (If applicable)

9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER

8c. ADDRESS (City, State, and ZIP Code)

10. SOURCE OF FUNDING NUMBERS

PROGRAM ELEMENT NO	PROJECT NO.	TASK NO	WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification)
Continuous, Low Thrust Coplanar Orbit Transfers with Varying Eccentricity

12. PERSONAL AUTHOR(S)
Gregory L. Beeker, B.S., Capt, USAF

13a. TYPE OF REPORT
MS Thesis

13b. TIME COVERED
FROM ________ TO ______

14. DATE OF REPORT (Year, Month, Day)
1988 December

15. PAGE COUNT
155

16. SUPPLEMENTARY NOTATION

17. COSATI CODES

FIELD	GROUP	SUB-GROUP
21	03	
22	03	

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
Continuous, Low Thrust; Orbit Transfers; Transfer Trajectories; Elliptical Orbit Trajectories; Electric Propulsion; Optimization

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

Thesis Advisor: William E. Wiesel, Jr
Professor of Astronautics
Department of Aeronautics and Astronautics

12 Jan 1989

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT
☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS

21. ABSTRACT SECURITY CLASSIFICATION
UNCLASSIFIED

22a. NAME OF RESPONSIBLE INDIVIDUAL
William E. Wiesel, Professor

22b. TELEPHONE (Include Area Code)
(513) 255-2362

22c. OFFICE SYMBOL
AFIT/ENY

The purpose of this study was to investigate the Δv requirements of a continuous, low thrust spacecraft performing coplanar orbit transfers that are constrained by either constant radius of apogee, perigee, or a combination of the two. The transfers were separated into two timescale problems. The fast timescale involved an optimization of the planar thrust control angle, α, to produce the maximum change in eccentricity or semimajor axis over a single revolution. The slow timescale applied the fast timescale results to complete the transfer through many revolutions about the primary body. The constrained radii control laws developed provide optimal circular-to-eccentric and eccentric-to-eccentric orbit transfers. However, when applied to circular-to-circular transfers, the resulting Δv is nearly twice that obtained using the most optimal continuous thrust control law ($\alpha = 0$), i.e. "spiral". Future recommended studies include the development of control laws to provide specified changes in apogee, perigee, and the argument of periapsis.