This dissertation has been microfilmed exactly as received

69-12,152

KERN, Edward Adolph, 1934-

MINIMUM-FUEL THRUST-LIMITED TRANSFER TRAJECTORIES BETWEEN COPLANAR ELLIPTIC ORBITS.

The University of Michigan, Ph.D., 1968
Engineering, aeronautical

University Microfilms, Inc., Ann Arbor, Michigan

MINIMUM-FUEL THRUST-LIMITED TRANSFER TRAJECTORIES BETWEEN COPLANAR ELLIPTIC ORBITS

by

Edward Adolph Kern

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy in
The University of Michigan
1968

Doctoral Committee:

Professor Donald T. Greenwood, Chairman
Professor Harm Buning
Professor Elmer G. Gilbert
Professor Robert M. Howe
Assistant Professor William F. Powers
Doctor Lynn E. Wolaver, United States Air Force

Abstract

MINIMUM-FUEL THRUST-LIMITED
TRANSFER TRAJECTORIES BETWEEN COPLANAR
ELLIPTIC ORBITS

By Edward Adolph Kern

Chairman: Donald T. Greenwood

A method is developed for the efficient computation of minimum-fuel transfer trajectories between coplanar elliptic orbits with a thrust-limited variable-mass rocket moving in a central gravitation force field. Each orbit is defined through the eccentricity, semi-latus rectum, and argument of pericenter. Transfer time is left open. The minimum-fuel trajectory is assumed to consist of two thrusting phases separated by a coasting phase.

Computation of the minimum-fuel transfer trajectory is accomplished by a direct integration of the rocket equations of motion and the associated adjoint equations which are developed from variational calculus techniques. This direct approach is made possible by a unique transformation of the adjoint equations into a set of equations which provide a much better understanding of the general behavior of minimum-fuel transfer trajectories. Reasonable estimates of the thrusting interval durations, initial thrust angle, and initial true anomaly are derived from the corresponding minimum-fuel, two-impulse transfer trajectory. The initial value of a newly discovered variable is used in conjunction with the initial thrust angle to obtain the thrusting interval durations specified

by the two-impulse transfer. Small adjustments in the initial true anomaly are made to obtain the desired pericenter argument of the final orbit. Final convergence is achieved through the use of sensitivity coefficients which relate small changes in the initial values of the thrust angle, true anomaly and the newly discovered variable to small changes in the eccentricity, semi-latus rectum, and pericenter argument of the final orbit.

Computation of the minimum-fuel transfer trajectory, including the corresponding two-impulse transfer, is accomplished with one program on an IBM 7094 digital computer employing primarily single-precision arithmetic. Rapid convergence is obtained over a broad class of transfer trajectories and rocket thrust levels.

ACKNOWLEDGEMENTS

I wish to thank Professor Donald T. Greenwood for his inspiration and guidance in the preparation of this dissertation.

The timely comments and suggestions made by the committee are greatly appreciated.

The motivation and encouragement provided through informal discussions with Dr. James H. Burghart, my associate at the Air Force Institute of Technology, will not be easily forgotten.

I am also indebted to the Air Force ASD digital computer center at Wright-Patterson Air Force Base for a generous amount of computer time and also for the open shop policy which made it possible for me to do my own programming and analysis.

Finally, I wish to thank my wife, Eunice, who typed this dissertation and who was patient and understanding throughout the long hours of its preparation.

Edward Adolph Kern

TABLE OF CONTENTS

		Page
ACKNOWLEDGEMENTS		ii
TABLE OF CONTENTS		iii
LIST OF TABLES		vi
LIST OF FIGURES		vii
LIST OF APPENDICES		viii
LIST OF SYMBOLS		ix
LIST OF SUBSCRIPTS		xii
CHAPTER 1.	INTRODUCTION	1
CHAPTER 2.	THE NECESSARY CONDITIONS FOR OPTIMALITY	4
2.1	Optimal Control Theory-Necessary Conditions	4
2.2	Units of Length, Acceleration, and Mass	6
2.3	The Rocket Equations of Motion in h, u, v, w Coordinates	6
2.4	The Hamiltonian, Adjoint Equations, and the Optimal Thrust Angle	10
2.5	Boundary Conditions on the Adjoint Variables	12
2.6	Determination of the Thrust-Intensity Control Variable and Definition of the Switching Function	15
2.7	Differential Equations for the Primer and the Thrust-Orientation Angle	17
2.8	Differential Equation for the Switching Function	18
2.9	Differential Equations for Semi-Latus Rectum, Eccentricity and the Polar Angle θ	19
2.10	Introduction of the Variable ν and the Development of the Differential Equation for ν	20
2.11	Transformation to the z, A, B, Coordinates	21
2.12	Summary of the Necessary Conditions for Minimum-Fuel Trajectories	22

Page

CHAPTER 3. DETERMINATION OF THE INITIAL-APPROXIMATE TRANSFER TRAJECTORY 27

3.1 Selection of the Unknown Initial Conditions - An Outline of the Problem 27
3.2 Development of the Properties of the Initial-Approximate Transfer Trajectory from the Minimum-Fuel Two-Impulse Transfer Trajectory 29
3.3 Selection of Proper Initial Values for Ψ and ν to Achieve the Desired First Thrusting Interval 34
3.4 Selection of Proper Initial Values for Ψ and ν to Achieve the Desired Second Thrusting Interval 42
3.4.1 Selection of the Initial Value of ν for Forward-Rearward or Rearward-Forward Thrusting Sequences 44
3.4.2 Selection of the Initial Value of ν for Forward-Forward or Rearward-Rearward Thrusting Sequences 52

CHAPTER 4. CONVERGENCE TO THE DESIRED MINIMUM-FUEL FINITE-THRUST TRANSFER TRAJECTORY 57

4.1 Definition of the Final Condition Errors 58
4.2 Adjustment of the Initial True Anomaly to Reduce the Argument of Pericenter Error 58
4.3 Definition and Computation of the Sensitivity Matrix which Relates Small Changes in Initial Conditions to Changes in the Final Conditions 59
4.4 Simultaneous Adjustment of the Initial Values of f, Ψ, and ν and Convergence to the Desired Finite-Thrust Transfer Trajectory 62

CHAPTER 5. COMPUTATIONAL TECHNIQUES 64

5.1 Transformation to the z, A, B Coordinates for Computation of the Coast Phase Trajectory 64
5.2 Computation of the Thrusting Phases 75
5.3 Backward Integration of the System Equations 76
5.4 Determination of the Initial Value of ν 77

CHAPTER 6. COMPUTER RESULTS AND LIMITATIONS OF THE CONVERGENCE METHOD 80

6.1 Summarization of the Computer Results 80
6.2 Thrust Angle Behavior Along the Minimum-Fuel Transfer Trajectory 85
6.3 Computer Time Requirements 87
6.4 Convergence Limitations Imposed by Orbital Eccentricities and Rocket Thrust Levels 89

Page

CHAPTER 7. CONCLUSIONS AND RECOMMENDATIONS 92

7.1 Summary of the Results 92
7.2 Recommendations for Further Study 93

APPENDICES 95

BIBLIOGRAPHY 157

LIST OF TABLES

Table		Page
6.1.	Summary of Minimum-Fuel Transfer Trajectories - Forward Computation	81
6.2.	Summary of Minimum-Fuel Transfer Trajectories - Backward Computation	82
6.3.	Typical Error Behavior During Convergence to a Desired Transfer Trajectory	84
6.4.	Summary of IBM-7094 computer time requirements for typical transfer trajectories	88

LIST OF FIGURES

Figure		Page
2-1	Definition of the Thrust-Orientation Angle and the Variables that Define the Rocket Motion	8
2-2	General Time Behavior of the Switching Function	16
3-1	Desired Switching Function Behavior for the First Thrusting Interval	35
3-2	Desired Switching Function Behavior Along the Initial-Approximate Transfer Trajectory	43
3-3	Typical Behavior of Ψ, ν, β, $(\Psi - \beta)$, and the Switching Function over an Entire Transfer Trajectory for the Case of a Forward-Rearward Thrusting Sequence	46
3-4	Effect of $\nu(t_o)$ upon the Initiation Time of the Second Thrusting Interval (Forward-Rearward Thrusting Sequence).	49
3-5	Effect of $\nu(t_o)$ upon the Duration of the Second Thrusting Interval (Forward-Rearward Thrusting Sequence)	50
5-1	Error in the Duration of the Second Thrusting Interval as a Function of the Initial Value of ν (Forward-Rearward Thrusting Sequence)	78
6-1	Typical Thrust Angle Time Histories Along the Minimum-Fuel Transfer Trajectory	86

LIST OF APPENDICES

Appendix		Page
A	DIGITAL COMPUTER PROGRAM FOR THE COMPUTATION OF THE MINIMUM-FUEL TWO-IMPULSE TRANSFER TRAJECTORY	95
B	DIGITAL COMPUTER PROGRAM FOR THE COMPUTATION OF THE MINIMUM-FUEL FINITE-THRUST TRANSFER TRAJECTORY . . .	117

LIST OF SYMBOLS

A - $z\ e\ \cos\varphi$

a - ratio of maximum rocket thrust to initial rocket weight

B - $z\ e\ \sin\varphi$

c - rocket effective exhaust velocity

E - eccentric anomaly

e - orbital eccentricity

f - true anomaly

$[G]$ - sensitivity matrix relating small changes in the initial and final conditions

H - hamiltonian

H_o - a portion of the hamiltonian

H_1 - a portion of the hamiltonian

h - reciprocal of the radial distance from the center of attraction

J - performance index

k - angular momentum per unit mass

L - hamiltonian associated with set of variables (t, z, A, B, w; θ)

m - instantaneous rocket mass

p - semi-latus rectum

q - orbital energy per unit mass

r - distance from center of force to the rocket

s - switching function (s positive implies thrust is on)

t - time

t_r - time at which ν passes through zero

u_r - radial velocity

u_θ - transverse velocity

u - $\frac{u_r}{r}$

V - velocity magnitude on two-impulse trajectory

v - $\frac{u_\theta}{r}$

w - reciprocal of the instantaneous mass

z - reciprocal of the per-unit-mass angular momentum

α - semi-major axis

β - $\tan^{-1}\left(\frac{\lambda_\varphi}{\lambda_\nu}\right)$

ε_e - error in the final (forward computation) or initial (backward computation) eccentricity

ε_p - error in the final (forward computation) or initial (backward computation) semi-latus rectum

ε_φ - error in the final (forward computation) or initial (backward computation) argument of pericenter

θ - $f + \varphi$

λ_A - adjoint variable associated with A

λ_B - adjoint variable associated with B

λ_h - adjoint variable associated with h

λ_r - adjoint variable associated with r

λ_u - adjoint variable associated with u

λ_{u_r} - adjoint variable associated with u_r

λ_{u_θ} - adjoint variable associated with u_θ

λ - $\sqrt{\lambda_{u_\theta}^2 + \lambda_{u_r}^2}$

λ_v - adjoint variable associated with v

λ_t - adjoint variable associated with t

λ_w - adjoint variable associated with w

λ_z - adjoint variable associated with z

λ_φ - adjoint variable associated with φ

λ_θ - adjoint variable associated with θ

$$\Lambda_u - \lambda_u - \frac{\lambda_\varphi (v^2 - h^3)v}{h^6 - v^2(2h^3 - u^2 - v^2)}$$

$$\Lambda_v - \lambda_v + \frac{\lambda_\varphi (v^2 + h^3)u}{h^6 - v^2(2h^3 - u^2 - v^2)}$$

$$\Lambda - \sqrt{\Lambda_u^2 + \Lambda_v^2}$$

$$\Lambda_{uh} - \frac{\partial \Lambda_u}{\partial h}$$

$$\Lambda_{uu} - \frac{\partial \Lambda_u}{\partial u}$$

$$\Lambda_{uv} - \frac{\partial \Lambda_u}{\partial v}$$

$$\Lambda_{vh} - \frac{\partial \Lambda_v}{\partial h}$$

$$\Lambda_{vu} - \frac{\partial \Lambda_v}{\partial u}$$

$$\Lambda_{vv} - \frac{\partial \Lambda_v}{\partial v}$$

$$\nu - h\lambda_h - \frac{3\lambda_\varphi \sin f}{e}$$

ξ - thrust intensity control variable ($0 \leq \xi \leq 1$)

$$\rho - \sqrt{\nu^2 + \lambda_\varphi^2}$$

φ - argument of pericenter

Ψ - angle of thrust orientation above local horizontal

LIST OF SUBSCRIPTS

$_o$ - refers to condition at the initial time

$_1$ - refers to condition at the end of first thrusting interval

$_2$ - refers to condition at beginning of second thrusting interval

$_f$ - refers to condition at the final time

$_I$ - refers to condition on the first impulse of the two-impulse transfer trajectory

$_{II}$ - refers to condition on the second impulse of the two-impulse transfer trajectory

$_d$ - refers to a desired initial or final value

$_a$ - refers to an actual initial or final value obtained by computation

CHAPTER 1

INTRODUCTION

The practical problem of determining minimum-fuel transfer trajectories between satellite orbits has received a considerable amount of attention within the last decade. This is due primarily to the expediencies of maximizing satellite or space vehicle payloads. The problem can be stated as follows:

> Given two elliptic orbits, find the trajectory which will transfer a space vehicle from the first orbit to the second orbit in such a manner that the final mass of the vehicle is maximized.

A method for computing two-impulse minimum-fuel transfer trajectories between non-coplanar orbits has been developed by Lee[4] and McCue and Bender[3] using ordinary calculus procedures. The transfer trajectory is generated by a sequence of two thrusting phases, where the thrust levels are assumed to be of infinite magnitude. Kern and Burghart[11] recently have made the computation of minimum-fuel two-impulse transfers more reliable and efficient by utilizing a Fletcher-Powell[8], conjugate-gradient minimization algorithm in conjunction with the equations developed by McCue and Bender[3].

For the more realistic problem of thrust-limited minimum-fuel trajectories, most of the progress to date has resulted only in gaining a better understanding of some of the properties of these transfer trajectories[1,5,6,10], but very little has been done in the development of computational techniques. Whereas the impulsive transfer problem

could be handled with techniques from ordinary calculus, the finite-thrust transfer problem must be considered from the standpoint of variational calculus. To find the minimum-fuel transfer trajectory it is necessary, therefore, to solve a system of non-linear differential equations with mixed boundary conditions. Several contributions towards gaining a better appreciation and understanding of the computational complexities associated with problems of this type were made by McCue[2], and Handelsman[15], who considered the related problem of computing a thrust-limited time-fixed minimum-fuel trajectory between two points with given position and velocity, where the position and velocity vectors were assumed to lie in the same plane. McCue's treatment of the problem was more realistic since he allowed for the possibility of coast periods at the beginning and end of the transfer trajectory while Handelsman did not.

This study considers minimum-fuel thrust-limited time-open transfer trajectories between coplanar elliptic orbits. The problem can be stated in the following manner:

> Given two coplanar elliptic orbits in a central-body gravitational force field with arbitrary eccentricities, semi-latus rectums, and arguments of pericenter. Find the trajectory which transfers a variable mass thrust-limited rocket from the first orbit to the second orbit with a minimum expenditure of fuel. The minimum-fuel trajectory is assumed to consist of a sequence of two thrusting phases separated by a coasting phase. Time to accomplish the transfer is left open.

As mentioned earlier, the finite-thrust problem must be considered within the framework of variational calculus. Applying variational principles results in a set of differential equations, the two-body equations of motion and their associated adjoint equations, which form the necessary conditions for the minimum-fuel trajectory.

The difficulty lies in integrating these equations to satisfy the boundary conditions since some of the boundary conditions are given at the initial time and the remaining at the final time. The more popular computation techniques such as the gradient methods, quasilinearization, etc., are not well suited to solve this problem because of the discontinuous nature of the rocket thrust.

The approach taken in this study is to transform the conventional adjoint variables into a set of variables which provide more insight into the characteristics of the transfer trajectory. Differential equations are developed for the transformed variables, and these equations are integrated directly along with the rocket equations of motion to find the minimum-fuel transfer trajectory. The known two-impulse transfer trajectory is used to assist in the choice of the unknown initial conditions. A systematic approach is employed to force the set of differential equations to satisfy the desired boundary conditions.

A major assumption is that the minimum-fuel transfer trajectory consists of two thrusting phases separated by a coasting phase. This assumption appears valid over a large class of transfers since Marchal[13] indicates that minimum-fuel one-impulse transfers are very rare (the two orbits must be very nearly tangent) and, excluding transfers to infinity and back, that minimum-fuel three-impulse transfers can exist only if the sum of the eccentricities of the initial and final orbits is greater than 1.712 .

CHAPTER 2

THE NECESSARY CONDITIONS FOR OPTIMALITY

In this chapter the basic rocket equations of motion along with the associated adjoint equations and boundary conditions are developed and transformed. The resulting set of equations, which are believed to be original with this study, will prove to be of great value in the computation of minimum-fuel trajectories.

2.1 Optimal Control Theory - Necessary Conditions

This section summarizes some of the more important concepts from optimal control theory. The results presented here are completely general and will be applied to the minimum-fuel problem in subsequent sections of this chapter.

Define:

$\bar{x} = (x_1, x_2, \ldots, x_n)$ the vector of state variables

$\bar{\zeta} = (\zeta_1, \zeta_2, \ldots, \zeta_m)$ the vector of controls

t = independent variable $(t_o \leq t \leq t_f)$

Let the components of the state vector satisfy the system of equations

$$\frac{dx_i}{dt} = g_i(x_1, x_2, \ldots, x_n, \zeta_1, \zeta_2, \ldots \zeta_m, t) \qquad i = 1,2,\ldots,n$$

with boundary conditions

$$U_y = U_y[x_1(t_o), \ldots, x_n(t_o), t_o; \; x_1(t_f), \ldots, x_n(t_f), t_f] = 0$$

$$y = 1,2,\ldots,q$$

The control vector, $\bar{\zeta}$, is to be chosen such that a function C of the initial and final conditions is minimized, where

$$C = C[x_1(t_o),\ldots,x_n(t_o),t_o;\ x_1(t_f),\ldots,x_n(t_f),t_f]$$

A set of necessary conditions for the control which minimizes C can be derived from the variational formulation

$$\delta[J + \int_{t_o}^{t_f} (\sum_{i=1}^{n} \lambda_i dx_i - Hdt)] = 0 \qquad (2\text{-}1)$$

where "δ" implies the first variation operator, and

$\bar{\lambda} = (\lambda_1,\lambda_2,\ldots,\lambda_n)$ the vector of adjoint variables

$H = \sum_{i=1}^{n} \lambda_i g_i$ the system hamiltonian

$J = \sum_{y=1}^{q} \eta_y U_y + C$ the performance index

η_y = constant Lagrange multipliers

The vanishing of the first variation results in the following differential equations and transversality conditions:

$$\frac{dx_i}{dt} = \frac{\partial H}{\partial \lambda_i} \qquad (i = 1,2,\ldots,n) \qquad (2\text{-}2)$$

$$\frac{d\lambda_i}{dt} = -\frac{\partial H}{\partial x_i} \qquad (i = 1,2,\ldots,n) \qquad (2\text{-}3)$$

$$\frac{dH}{dt} = \frac{\partial H}{\partial t} \qquad (2\text{-}4)$$

$$\lambda_i(t_o) = \frac{\partial J}{\partial x_i(t_o)} \qquad (i = 1,2,\ldots,n) \qquad (2\text{-}5)$$

$$\lambda_i(t_f) = -\frac{\partial J}{\partial x_i(t_f)} \qquad (i = 1,2,\ldots,n) \qquad (2\text{-}6)$$

$$H(t_o) = - \frac{\partial J}{\partial t_o} \tag{2-7}$$

$$H(t_f) = \frac{\partial J}{\partial t_f} \tag{2-8}$$

In addition, the controls should be chosen to maximize the hamiltonian. Hence, for the case of continuous controls

$$\frac{\partial H}{\partial \zeta_i} = 0 \qquad (\zeta_i \text{ continuous}) \tag{2-9}$$

2.2 Units of Length, Acceleration, and Mass

In order to obtain better scaling of the problem variables and parameters and to make the results more readily applicable to motion about any central-body attracting force, the following set of units is employed throughout the study:

unit length = r^* = convenient distance from attracting center

unit acceleration = g^* = acceleration of gravity at distance r^*

unit mass = initial mass of the rocket vehicle

As a result of this choice of units

$\sqrt{r^*/g^*}$ = unit of time (in one time unit a satellite in circular orbit of radius r^* would traverse through a central angle of one radian).

$\sqrt{r^* g^*}$ = unit of velocity (the orbital velocity of a satelite in a circular orbit of radius r^* would equal one velocity unit).

$g^* r^{*2}$ = gravitational parameter of the central body = 1

2.3 The Rocket Equations of Motion in h, u, v, w, Coordinates

The equations for a rocket in coplanar motion about its central attracting body are

$$\frac{dr}{dt} = u_r \tag{2-10}$$

$$\frac{du_r}{dt} = \frac{u_\theta^2}{r} - \frac{1}{r^2} + \frac{a\xi}{m} \sin\Psi \tag{2-11}$$

$$\frac{du_\theta}{dt} = -\frac{u_r u_\theta}{r} + \frac{a\xi}{m} \cos\Psi \tag{2-12}$$

$$\frac{d\varphi}{dt} = \frac{u_\theta}{r} - \frac{df}{dt} \tag{2-13}$$

$$\frac{dm}{dt} = -\frac{a\xi}{c} \tag{2-14}$$

where:

r = radial distance from central force

u_r = radial velocity

u_θ = transverse velocity

φ = argument of pericenter

f = true anomaly

a = ratio of maximum rocket thrust to initial rocket weight

ξ = thrust intensity variable ($0 \leq \xi \leq 1$)

m = rocket mass

c = rocket effective exhaust velocity

Ψ = thrust-orientation angle above local horizontal

See Fig. 2-1 .

From the fundamental two-body orbit relationships

$$u_r = \frac{e \sin f}{\sqrt{p}}$$

$$r = \frac{p}{1 + e \cos f}$$

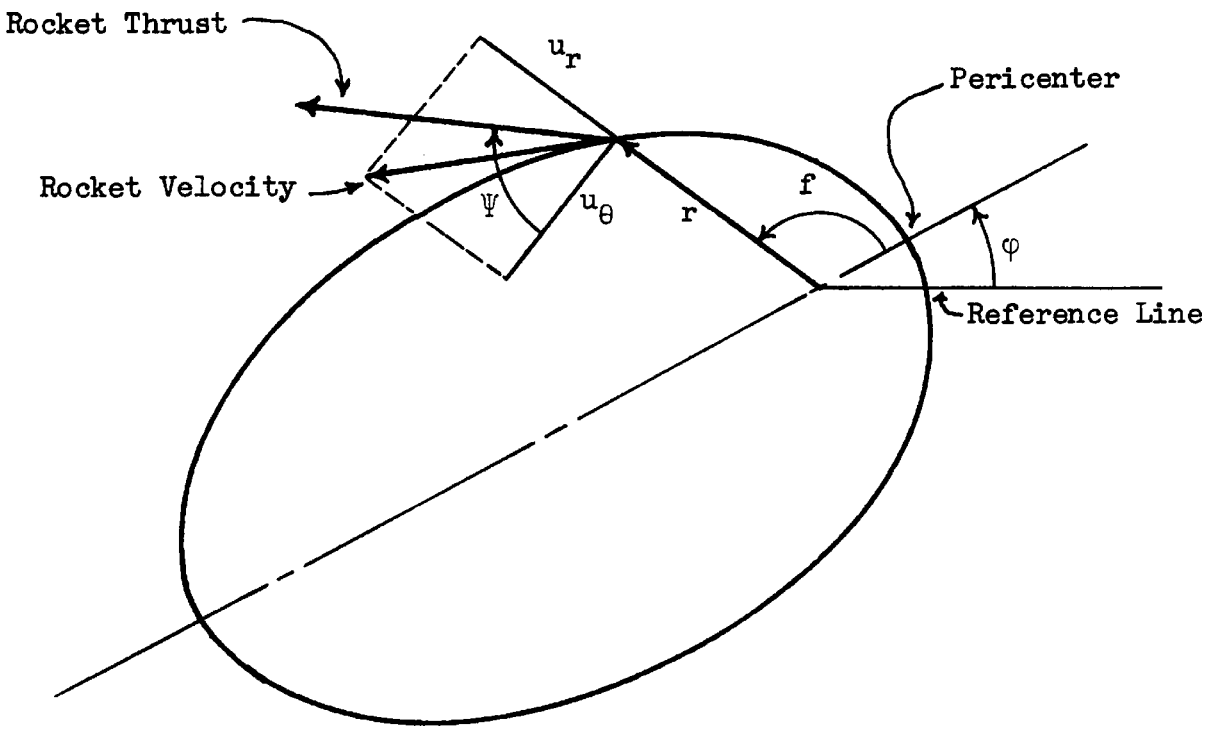


Fig. 2-1: Definition of the thrust-orientation angle and the variables that define the rocket motion.

where:

e = eccentricity

p = semi-latus rectum

It follows that

$$e \sin f = u_r\sqrt{p} \tag{2-15}$$

$$e \cos f = \frac{p}{r} - 1 \tag{2-16}$$

Dividing Eq. (2-16) into Eq. (2-15) and making use of the two-body orbit identity

$$p = r^2u_\theta^2$$

we have

$$\tan f = \frac{ru_r u_\theta}{ru_\theta^2 - 1}$$

or

$$f = \tan^{-1}\left[\frac{ru_r u_\theta}{ru_\theta^2 - 1}\right] \tag{2-16}$$

Differentiating the above equation, making use of Eqs. (2-10) - (2-12), and substituting the resultant derivative into Eq. (2-13) gives, after some simplification

$$\frac{d\varphi}{dt} = \frac{a\xi[(r^2u_\theta^2 + r)u_r\cos\Psi - (r^2u_\theta^2 - r)u_\theta\sin\Psi]}{m[1 - r^2u_\theta^2(2/r - u_r^2 - u_\theta^2)]} \tag{2-17}$$

Under a transformation to the new coordinates h, u, v, and w, defined by

$$h = \frac{1}{r} \tag{2-18}$$

$$u = \frac{u_r}{r} \tag{2-19}$$

$$v = \frac{u_\theta}{r} \tag{2-20}$$

$$w = \frac{1}{m} \tag{2-21}$$

Eqs. (2-10), (2-11), (2-12), (2-17) and (2-14) become

$$\frac{dh}{dt} = - uh \tag{2-22}$$

$$\frac{du}{dt} = v^2 - u^2 - h^3 + a\xi\, wh\sin\Psi \tag{2-23}$$

$$\frac{dv}{dt} = - 2uv + a\xi\, wh\cos\Psi \tag{2-24}$$

$$\frac{d\varphi}{dt} = \frac{a\xi wh[(v^2 + h^3)u\cos\Psi - (v^2 - h^3)v\sin\Psi]}{h^6 - v^2(2h^3 - u^2 - v^2)} \tag{2-25}$$

$$\frac{dw}{dt} = \frac{a\xi w^2}{c} \tag{2-26}$$

2.4 The Hamiltonian, Adjoint Equations, and the Optimal Thrust-Orientation Angle

It is now necessary to design a control such that the transfer will be accomplished with a minimum expenditure of fuel. This optimal control must indicate the magnitude and orientation of the thrust vector, the duration of the thrusting phases, and the point on the initial orbit at which the transfer is to be initiated.

As indicated in Section 2.1, the optimal or minimum-fuel control must maximize the hamiltonian. The hamiltonian (see Section 2.1) for the system defined by Eqs. (2-22) - (2-26) is

$$H = -uh\lambda_h + (v^2-u^2-h^3)\lambda_u - 2uv\lambda_v + a\xi w\left[h(\Lambda_v\cos\Psi + \Lambda_u\sin\Psi) + \frac{w\lambda_w}{c}\right] \quad (2\text{-}27)$$

where

λ_h = adjoint variable associated with h

λ_u = adjoint variable associated with u

λ_v = adjoint variable associated with v

λ_φ = adjoint variable associated with φ

λ_w = adjoint variable associated with w

and, for convenience

$$\Lambda_v = \lambda_v + \frac{\lambda_\varphi(v^2 + h^3)u}{h^6 - v^2(2h^3 - u^2 - v^2)} \quad (2\text{-}28)$$

$$\Lambda_u = \lambda_u - \frac{\lambda_\varphi(v^2 - h^3)v}{h^6 - v^2(2h^3 - u^2 - v^2)} \quad (2\text{-}29)$$

In order to maximize the hamiltonian, the optimal thrust-orientation angle should be chosen such that

$$\sin\Psi = \frac{\Lambda_u}{\Lambda} \quad (2\text{-}30)$$

$$\cos\Psi = \frac{\Lambda_v}{\Lambda} \tag{2-31}$$

where

$$\Lambda = \sqrt{\Lambda_u^2 + \Lambda_v^2} \tag{2-32}$$

In future discussions, we shall refer to the variable Λ defined by Eq. (2-32) as the primer. It can be shown that $h\Lambda$ is equal to the primer defined by Lawden[5]. The hamiltonian, given by Eq. (2-27) and maximized with respect to the thrust-orientation angle, becomes

$$H = -uh\lambda_h + (v^2-u^2-h^3)\lambda_u - 2uv\lambda_v + a\xi w(h\Lambda + \frac{w\lambda_w}{c}) \tag{2-33}$$

The following system of adjoint equations is derived by applying Eq. (2-3) to the above hamiltonian.

$$\frac{d\lambda_h}{dt} = \lambda_h u + 3h^2\lambda_u - a\xi w\Lambda - \frac{a\xi wh}{\Lambda}(\Lambda_u\Lambda_{uh} + \Lambda_v\Lambda_{vh}) \tag{2-34}$$

$$\frac{d\lambda_u}{dt} = \lambda_h h + 2u\lambda_u + 2\lambda_v v - \frac{a\xi wh}{\Lambda}(\Lambda_u\Lambda_{uu} + \Lambda_v\Lambda_{vu}) \tag{2-35}$$

$$\frac{d\lambda_v}{dt} = -2\lambda_u v + 2\lambda_v u - \frac{a\xi wh}{\Lambda}(\Lambda_u\Lambda_{uv} + \Lambda_v\Lambda_{vv}) \tag{2-36}$$

$$\frac{d\lambda_\varphi}{dt} = 0 \tag{2-37}$$

$$\frac{d\lambda_w}{dt} = -\frac{2a\xi w\lambda_w}{c} - a\xi h\Lambda \tag{2-38}$$

where:

$$\Lambda_{uh} = \lambda_\varphi \frac{6h^5v^3 + 3h^2v^3u^2 - 3h^8v - 3h^2v^5}{[h^6 - v^2(2h^3 - u^2 - v^2)]^2} \tag{2-39}$$

$$\Lambda_{uu} = \lambda_\varphi \frac{2uv^5 - 2uv^3h^3}{[h^6 - v^2(2h^3 - u^2 - v^2)]^2} \tag{2-40}$$

$$\Lambda_{uv} = \lambda_{\varphi} \frac{h^9 + v^6 - h^6v^2 - h^3v^4 - v^4u^2 - h^3v^2u^2}{[h^6 - v^2(2h^3 - u^2 - v^2)]^2} \tag{2-41}$$

$$\Lambda_{vh} = \lambda_{\varphi} \frac{9h^2uv^4 - 3h^8u - 6h^5uv^2 + 3h^2u^3v^2}{[h^6 - v^2(2h^3 - u^2 - v^2)]^2} \tag{2-42}$$

$$\Lambda_{vu} = \lambda_{\varphi} \frac{h^9 - h^6v^2 - v^4h^3 - u^2v^2h^3 - u^2v^4 + v^6}{[h^6 - v^2(2h^3 - u^2 - v^2)]^2} \tag{2-43}$$

$$\Lambda_{vv} = \lambda_{\varphi} \frac{6uvh^6 - 2v^5u - 2vu^3h^3 - 4v^3h^3u}{[h^6 - v^2(2h^3 - u^2 - v^2)]^2} \tag{2-44}$$

2.5 Boundary Conditions on the Adjoint Variables

The boundary conditions on the state variables can be defined by specifying the energy, angular momentum and argument of pericenter for both the initial and final orbits.

With

$q(t)$ = energy per unit mass

$k(t)$ = angular momentum per unit mass

$\Delta\varphi$ = desired change in the argument of pericenter between the initial and final orbits

the constraints on the state variables at the initial and final times t_o and t_f are

$$\frac{1}{2h^2(t_o)} [u^2(t_o) + v^2(t_o)] - h(t_o) - q(t_o) = 0 = c_1$$

$$\frac{v(t_o)}{h^2(t_o)} - k(t_o) = 0 = c_2$$

$$w(t_o) - 1 = 0 = c_3$$

$$\frac{1}{2h^2(t_f)} [u^2(t_f) + v^2(t_f)] - h(t_f) - q(t_f) = 0 = c_4$$

$$\frac{v(t_f)}{h^2(t_f)} - k(t_f) = 0 = c_5$$

$$\varphi(t_f) - \varphi(t_o) - \Delta\varphi = 0 = c_6$$

Constraints c_1, c_2, c_4 and c_5 are a consequence of the given energy and angular momentum levels of the initial and final orbits. The initial mass of the rocket is equal to unity, as seen from constraint c_3, while the change in pericenter argument $\Delta\varphi$ between the initial and final orbits is taken into account by constraint c_6 .

In addition to satisfying the above six constraints, the optimal control must maximize the final mass or minimize the reciprocal of the final mass

$$w(t_f) = \text{minimum}$$

Hence, the J function defined in Section 2.1, is

$$J = w(t_f) + \sum_{i=1}^{6} \eta_i c_i$$

or

$$J = w(t_f) + \eta_1\left[\frac{1}{2h^2(t_o)}[u^2(t_o) + v^2(t_o)] - h(t_o) - q(t_o)\right]$$

$$+ \eta_2\left[\frac{v(t_o)}{h^2(t_o)} - k(t_o)\right] + \eta_3[w(t_o) - 1]$$

$$+ \eta_4\left[\frac{1}{2h^2(t_f)}[u^2(t_f) + v^2(t_f)] - h(t_f) - q(t_f)\right]$$

$$+ \eta_5\left[\frac{v(t_f)}{h^2(t_f)} - k(t_f)\right] + \eta_6[\varphi(t_f) - \varphi(t_o) - \Delta\varphi]$$

The following boundary conditions result directly from Eqs. (2-5) - (2-8) and the above expression for J .

$$\lambda_u(t_o) = \frac{\eta_1 u(t_o)}{h^2(t_o)} \tag{2-45}$$

$$\lambda_v(t_o) = \frac{\eta_1 v(t_o)}{h^2(t_o)} + \frac{\eta_2}{h^2(t_o)} \tag{2-46}$$

$$\lambda_h(t_o) = -\frac{2\eta_2 v(t_o)}{h^3(t_o)} - \frac{\eta_1}{h^3(t_o)}[u^2(t_o) + v^2(t_o)] - \eta_1 \tag{2-47}$$

$$\lambda_\varphi(t_o) = -\eta_6 \tag{2-48}$$

$$\lambda_w(t_o) = \eta_3 \tag{2-49}$$

$$H(t_o) = 0 \tag{2-50}$$

$$\lambda_u(t_f) = -\frac{\eta_4 u(t_f)}{h^2(t_f)} \tag{2-51}$$

$$\lambda_v(t_f) = -\frac{\eta_4 v(t_f)}{h^2(t_f)} - \frac{\eta_5}{h^2(t_f)} \tag{2-52}$$

$$\lambda_h(t_f) = \frac{2\eta_5 v(t_f)}{h^3(t_f)} + \frac{\eta_4}{h^3(t_f)}[u^2(t_f) + v^2(t_f)] + \eta_4 \tag{2-53}$$

$$\lambda_\varphi(t_f) = -\eta_6 \tag{2-54}$$

$$\lambda_w(t_f) = -1 \tag{2-55}$$

$$H(t_f) = 0 \tag{2-56}$$

It is convenient to partition the hamiltonian of Eq. (2-33) as follows.

$$H = H_o + a\xi w H_1 \tag{2-57}$$

where:

$$H_o = -uh\lambda_h + (v^2 - u^2 - h^3)\lambda_u - 2uv\lambda_v \tag{2-58}$$

$$H_1 = h\Lambda + \frac{w\lambda_w}{c} \tag{2-59}$$

Evaluating H_o at the initial and final times through the use of Eqs. (2-45) - (2-47) and (2-51) - (2-53) respectively gives

$$H_o(t_o) = 0 \tag{2-60}$$

$$H_o(t_f) = 0 \tag{2-61}$$

Since time t does not appear explicitely in the hamiltonian, it follows directly from Eq. (2-4) that

$$\frac{dH}{dt} = 0$$

or

$$H(t) = \text{constant} \tag{2-62}$$

But, from Eqs. (2-50) and (2-56) we have that H must vanish at the initial and final points of the trajectory. Therefore, in accordance with Eq. (2-62), the hamiltonian must equal zero over the entire transfer trajectory

$$H(t) = 0 \qquad t_o \leq t \leq t_f \tag{2-63}$$

2.6 Determination of the Thrust-Intensity Control Variable and Definition of the Switching Function

The thrust-intensity control variable ξ can assume values in the range

$$0 \leq \xi \leq 1$$

and must be chosen to maximize the hamiltonian. From Eq. (2-57) it follows that ξ must be chosen according to the rule

$$\xi = 1 \qquad H_1 > 0$$

and

$$\xi = 0 \qquad H_1 < 0$$

In other words, if H_1 is positive, the rocket should be thrusting at maximum intensity while for H_1 negative, the rocket should be in a non-thrusting or coasting phase. If H_1 is zero over any finite time

interval, ξ could take on any value in its permissible range without affecting the hamiltonian. However, for the case of time-open transfer trajectories, minimum-fuel thrusting arcs for which the thrust-intensity control variable takes on intermediate values in the range

$$0 < \xi < 1$$

have been shown by Robbins[9] and Kopp and Moyer[14] to be non-existent.

Because of the manner in which H_1 controls the rocket thrusting, a switching function s can be defined as

$$s = \frac{H_1}{\Lambda} \tag{2-64}$$

Since Λ is by definition a positive quantity, we conclude that for s positive the rocket thrust should be at maximum intensity ($\xi = 1$), and for s negative the rocket thrust should be zero ($\xi = 0$). See Fig. 2-2 .

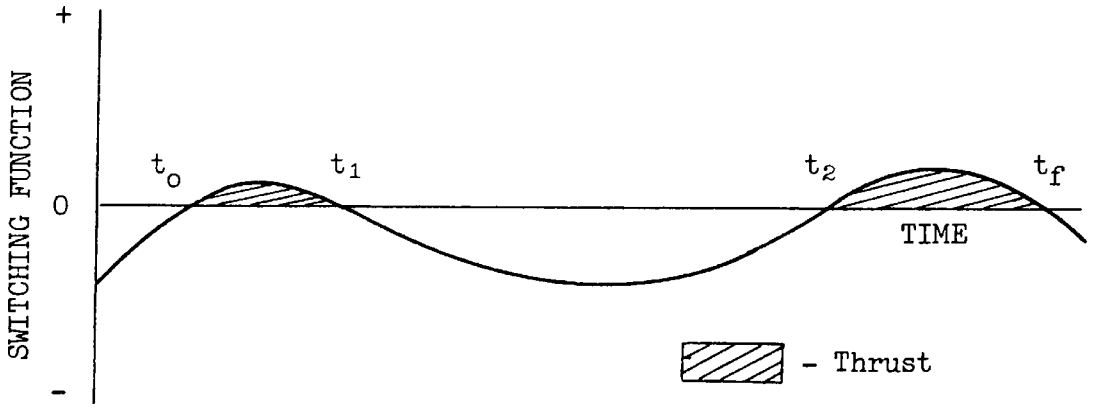


Fig. 2-2: General time behavior of the switching function

2.7 Differential Equations for the Primer and the Thrust-Orientation Angle

The generation of minimum-fuel trajectories is simplified considerably through the direct integration of differential equations for the primer Λ and the thrust-orientation angle Ψ. These two equations will replace two of the original adjoint equations. From Eqs. (2-30) and (2-31) we have

$$\Lambda \sin\Psi = \Lambda_u$$

$$\Lambda \cos\Psi = \Lambda_v$$

Differentiating both sides of these equations gives

$$\frac{d\Lambda}{dt} \sin\Psi + \Lambda \cos\Psi \frac{d\Psi}{dt} = \frac{d\Lambda_u}{dt} \tag{2-65}$$

$$\frac{d\Lambda}{dt} \cos\Psi - \Lambda \sin\Psi \frac{d\Psi}{dt} = \frac{d\Lambda_v}{dt} \tag{2-66}$$

The right-hand sides of the above two equations can be evaluated by differentiating Eqs. (2-28) and (2-29) and then applying Eqs. (2-22) - (2-26), (2-34) - (2-38), and the fundamental two-body orbital relationships

$$v = \sqrt{p}\, h^2 \tag{2-67}$$

$$h = \frac{1 + e \cos f}{p} \tag{2-68}$$

$$u = \frac{he \sin f}{\sqrt{p}} \tag{2-69}$$

where:

e = eccentricity

f = true anomaly

p = semi-latus rectum

Eqs. (2-65) and (2-66) become, after a considerable amount of algebraic manipulation,

$$\frac{d\Lambda}{dt}\sin\Psi + \Lambda\frac{d\Psi}{dt}\cos\Psi = h\lambda_h + 2u\Lambda_u + 2v\Lambda_v - \frac{3\lambda_\varphi \sin f}{e} \tag{2-70}$$

$$\frac{d\Lambda}{dt}\cos\Psi - \Lambda\frac{d\Psi}{dt}\sin\Psi = 2u\Lambda_v - 2v\Lambda_u - \lambda_\varphi \tag{2-71}$$

Multiplying Eq. (2-70) by $\sin\Psi$ and Eq. (2-71) by $\cos\Psi$, adding the resulting equations and solving for $\frac{d\Lambda}{dt}$ gives

$$\frac{d\Lambda}{dt} = 2u\Lambda + (h\lambda_h - \frac{3\lambda_\varphi \sin f}{e})\sin\Psi - \lambda_\varphi\cos\Psi \tag{2-72}$$

Likewise, multiplying Eqs. (2-70) and (2-71) by $\cos\Psi$ and $(-\sin\Psi)$ respectively, adding the resulting equations and solving for $\frac{d\Psi}{dt}$ yields

$$\frac{d\Psi}{dt} = 2v + \frac{1}{\Lambda}[(h\lambda_h - \frac{3\lambda_\varphi \sin f}{e})\cos\Psi + \lambda_\varphi\sin\Psi] \tag{2-73}$$

2.8 Differential Equation for the Switching Function

Direct integration of the differential equation for the switching function s will lead to further computational simplifications. Rearranging Eq. (2-64) and differentiating both sides of the resulting equation yields

$$\Lambda\frac{ds}{dt} + s\frac{d\Lambda}{dt} = \frac{dH_1}{dt} \tag{2-74}$$

The following expression for $\frac{ds}{dt}$ results from the above equation by differentiating Eq. (2-59) and then applying Eqs. (2-22), (2-26), (2-38) (2-64), and (2-72).

$$\frac{ds}{dt} = \frac{s - h}{\Lambda}[\lambda_\varphi \cos\Psi - (h\lambda_h - \frac{3\lambda_\varphi}{e} \sin f)\sin\Psi] + uh - s(2u + \frac{a\xi w}{c}) \quad (2\text{-}75)$$

2.9 Differential Equations for Semi-Latus Rectum, Eccentricity and the Polar Angle θ

Solving Eq. (2-67) for the semi-latus rectum gives

$$p = \frac{v^2}{h^4} \quad (2\text{-}76)$$

Differentiating this equation, and simplifying the resulting derivative with the aid of Eqs. (2-22), (2-24), (2-67), and (2-68), we find

$$\frac{dp}{dt} = \frac{2a\xi w\, p^{1.5} \cos\Psi}{1 + e \cos f} \quad (2\text{-}77)$$

The eccentricity for an elliptic orbit in terms of semi-latus rectum and semi-major axis is

$$e = \sqrt{1 - \frac{p}{a}} \quad (2\text{-}78)$$

where

a = semi-major axis of the elliptic orbit

The semi-major axis can be expressed in terms of h, u, and v through application of the orbital energy equation[7] to give

$$a = \frac{h^2}{2h^3 - u^2 - v^2} \quad (2\text{-}79)$$

Eliminating a from Eq. (2-78) by means of Eq. (2-79), differentiating Eq. (2-78) and simplifying the resulting equation by means of Eqs. (2-22), (2-23), (2-24), (2-67), (2-68), and (2-77) results in the differential equation for eccentricity.

$$\frac{de}{dt} = a\xi w \sqrt{p}[\sin f \sin\Psi + \frac{2\cos f + e(1 + \cos^2 f)}{1 + e \cos f} \cos\Psi] \quad (2\text{-}80)$$

Define the angle θ as

$$\theta = f + \varpi \tag{2-81}$$

Then it follows directly from Eqs. (2-13) and (2-20) that

$$\frac{d\theta}{dt} = v \tag{2-82}$$

2.10 Introduction of the Variable ν and the Development of the Differential Equation for ν

The discovery of the variable ν has been a most important factor in making possible the satisfactory solution of the minimum-fuel transfer problem. This variable, which is a combination of state variables and adjoint variables, is defined as

$$\nu = h\lambda_h - \frac{3\lambda_\varphi \sin f}{e} \tag{2-83}$$

The desire to simplify Eqs. (2-72), (2-73), and (2-75) for the primer, the thrust-orientation angle and the switching function, respectively, led to this particular form for the variable ν. Substituting ν into these equations results in the following simplified set:

$$\frac{d\Lambda}{dt} = 2u\Lambda + \nu \sin\Psi - \lambda_\varpi \cos\Psi \tag{2-84}$$

$$\frac{d\Psi}{dt} = 2v + \frac{1}{\Lambda}(\nu \cos\Psi + \lambda_\varphi \sin\Psi) \tag{2-85}$$

$$\frac{ds}{dt} = \frac{s - h}{\Lambda}(\lambda_\varphi \cos\Psi - \nu\sin\Psi) + uh - s\left(2u + \frac{a\xi w}{c}\right) \tag{2-86}$$

A differential equation for ν can be derived by differentiating Eq. (2-83) and then applying (2-22), (2-34), (2-39), (2-42), (2-67), (2-68), (2-69), (2-80), (2-13), (2-20) and (2-25). This leads to the

following remarkably simple result

$$\frac{d\nu}{dt} = \Lambda\, h(3h^2 \sin\Psi - a\xi w) \tag{2-87}$$

As is evident from Eqs. (2-84) - (2-86), the variable ν influences considerably the behavior of the primer, the thrust-orientation angle, and the switching function. This fact along with equation (2-87), which makes it easy to predict the behavior of ν, provides a great deal of insight into the selection of an initial value for ν and how to vary this initial choice of ν to obtain the desired transfer trajectory. This will be discussed in detail in Chapter 3.

2.11 Transformation to the z, A, B Coordinates

The instantaneous orbital elements, p, e, and φ, of the transfer trajectory can be determined by direct integration of Eqs. (2-77), (2-80) and (2-25), respectively. However, with the aid of Eqs. (2-67)- (2-69) we have, from Eq. (2-25)

$$\frac{d\varphi}{dt} = \frac{a\xi w\sqrt{h}}{e}\left[\frac{2 + e\cos f}{1 + e\cos f}\sin f\cos\Psi - \cos f\sin\Psi\right] \tag{2-88}$$

It is obvious that the above equation has a singularity at e = 0 and thus would not be convenient to integrate for low values of e. The difficulties associated with this singularity can be overcome by defining, as suggested by DeVeubeke[1,10], a new set of orbital elements z, A, and B as follows

$$z = \frac{h^2}{v} \tag{2-89}$$

$$A = z\, e\cos\varphi \tag{2-90}$$

$$B = z\, e\sin\varphi \tag{2-91}$$

From Eq. (2-81)

$$\varphi = \theta - f \tag{2-92}$$

Substituting Eq. (2-92) into Eqs. (2-90) and (2-91) and expanding $\cos(\theta - f)$ and $\sin(\theta - f)$ gives

$$A = z\, e(\cos\theta \cos f + \sin\theta \sin f) \tag{2-93}$$

$$B = z\, e(\sin\theta \cos f - \sin f \cos\theta) \tag{2-94}$$

From Eqs. (2-67) - (2-69) and (2-89) the above two equations become

$$A = \frac{u}{h} \sin\theta + \left(\frac{v}{h} - z\right)\cos\theta \tag{2-95}$$

$$B = -\frac{u}{h} \cos\theta + \left(\frac{v}{h} - z\right)\sin\theta \tag{2-96}$$

Differentiating Eqs. (2-89), (2-95) and (2-96) and making use of Eqs. (2-22) - (2-24) and (2-89) gives

$$\frac{dz}{dt} = -\frac{a\xi w z^2}{h} \cos\Psi \tag{2-97}$$

$$\frac{dA}{dt} = a\xi w\left(\cos\theta \cos\Psi + \sin\theta \sin\Psi + \frac{z^2}{h} \cos\Psi \cos\theta\right) \tag{2-98}$$

$$\frac{dB}{dt} = a\xi w\left(\sin\theta \cos\Psi - \cos\theta \sin\Psi + \frac{z^2}{h} \cos\Psi \sin\theta\right) \tag{2-99}$$

It is obvious by inspection that these equations can be easily integrated at very low eccentricities without loss of accuracy.

2.12 Summary of the Necessary Conditions for Minimum-Fuel Trajectories

The differential equations (2-97), (2-98), (2-99), (2-82), (2-26), (2-84), (2-85), (2-86), (2-87), and (2-37) are the basic set of equations which must be integrated to find minimum-fuel trajectories. This set is repeated and renumbered to facilitate future discussions.

$$\frac{dz}{dt} = -\frac{a\xi w z^2}{h} \cos\Psi \tag{2-100}$$

$$\frac{dA}{dt} = a\xi w\left(\cos\theta \cos\Psi + \sin\theta \sin\Psi + \frac{z^2}{h} \cos\Psi \cos\theta\right) \tag{2-101}$$

$$\frac{dB}{dt} = a\xi w\left(\sin\theta \cos\Psi - \cos\theta \sin\Psi + \frac{z^2}{h} \cos\Psi \sin\theta\right) \tag{2-102}$$

$$\frac{d\theta}{dt} = v \tag{2-103}$$

$$\frac{dw}{dt} = \frac{a\xi w^2}{c} \tag{2-104}$$

$$\frac{d\Lambda}{dt} = 2u\Lambda + v \sin\Psi - \lambda_\varphi \cos\Psi \tag{2-105}$$

$$\frac{d\Psi}{dt} = 2v + \frac{1}{\Lambda}(v \cos\Psi + \lambda_\varphi \sin\Psi) \tag{2-106}$$

$$\frac{dv}{dt} = \Lambda h(3h^2 \sin\Psi - a\xi w) \tag{2-107}$$

$$\frac{ds}{dt} = \frac{s - h}{\Lambda}(\lambda_\varphi \cos\Psi - v \sin\Psi) + uh - s(2u + \frac{a\xi w}{c}) \tag{2-108}$$

$$\frac{d\lambda_\varphi}{dt} = 0 \tag{2-109}$$

The original orbital elements, p, e, and φ, can be computed from z, A, and B by means of the following identities which are easily derived from Eqs. (2-89) - (2-91) and (2-67):

$$\varphi = \tan^{-1}(\frac{B}{A}) \tag{2-110}$$

$$e = \frac{A}{z \cos\varphi} \tag{2-111}$$

$$p = \frac{1}{z^2} \tag{2-112}$$

The variables h, u, and v which must be evaluated in order to integrate Eqs. (2-100) - (2-109) are from Eqs. (2-67) - (2-69) and (2-89)

$$h = (1 + e \cos f)z^2 \tag{2-113}$$

$$u = h\ e\ z \sin f \tag{2-114}$$

$$v = \frac{h^2}{z} \tag{2-115}$$

where the true anomaly f can be computed from Eq. (2-81) as

$$f = \theta - \varphi \tag{2-116}$$

In addition the thrust-intensity control variable ξ is established by the switching function s as follows:

$$\xi = 1 \qquad s > 0 \tag{2-117}$$

$$\xi = 0 \qquad s < 0 \tag{2-118}$$

Let the sets (p_o, e_o, φ_o) and (p_f, e_f, φ_f) be the parameter, eccentricity, and argument of pericenter of the given initial and final orbits respectively. The reference line from which all angular measurements are made can be defined to pass through pericenter of the initial orbit so that

$$\varphi(t_o) = 0 \tag{2-119}$$

The initial mass of the rocket will be equal to unity since the basic unit of mass is taken as the initial mass of the rocket. In all optimal control problems of this type the initial condition on one of the adjoint variables can be selected with a certain degree of freedom since the theory permits the scaling of the adjoint variables to any desired positive constant. Therefore, without any loss of generality, the primer Λ is given the initial value of unity.

Using the definition of the switching function given in Eq. (2-64), we have for the hamiltonian of Eq. (2-57)

$$H = H_o + a\xi w \Lambda s \tag{2-120}$$

It has been proven earlier in this chapter with Eqs. (2-60), (2-61), and (2-63) that H_o must vanish at the initial and final times, and H must remain zero throughout the entire transfer. Therefore, it follows from Eqs. (2-117), (2-118) and (2-120) that the switching function must be less than or equal to zero at the initial and final

points of the transfer trajectory. With s negative, the rocket is coasting, and hence the orbital elements are remaining fixed. We shall define the initial time t_o as the time at which the rocket begins moving out of the initial orbit and the final time t_f as the time at which the rocket attains its final orbit. Hence, the switching function must equal zero at the initial and final times defined in this manner.

In light of the above discussion and with the aid of Eqs.(2-89), (2-90), (2-91) and (2-115), we conclude that Eqs. (2-100) - (2-109) must satisfy the following boundary conditions

$$z(t_o) = \frac{1}{\sqrt{p_o}} \qquad z(t_f) = \frac{1}{\sqrt{p_f}} \tag{2-121}$$

$$A(t_o) = z(t_o)e_o\cos\varphi_o \qquad A(t_f) = z(t_f)e_f\cos\varphi_f \tag{2-122}$$

$$B(t_o) = z(t_o)e_o\sin\varphi_o \qquad B(t_f) = z(t_f)e_f\sin\varphi_f \tag{2-123}$$

$$w(t_o) = 1 \tag{2-124}$$

$$\Lambda(t_o) = 1 \tag{2-125}$$

$$s(t_o) = 0 \qquad s(t_f) = 0 \tag{2-126}$$

$$H_o(t_o) = 0 \qquad H_o(t_f) = 0 \tag{2-127}$$

The initial values of the angle θ, the thrust angle Ψ, the variable ν, and the adjoint variable λ_φ are unknown. These initial conditions must be chosen, however, so that the required final conditions are satisfied at the variable final time t_f. Only three of the missing initial conditions can be chosen independently. The fourth missing initial condition must be chosen to satisfy the requirement that H_o vanish at the initial time.

At the initial time

$$\theta(t_o) = f(t_o) \tag{2-128}$$

This result follows directly from Eqs. (2-81) and (2-119). Therefore, determining the initial value of the true anomaly f is equivalent to finding the initial value of the angle θ .

CHAPTER 3

DETERMINATION OF THE INITIAL-APPROXIMATE TRANSFER TRAJECTORY

In this chapter a method for selecting the unknown initial conditions for the set of differential Eqs. (2-100) - (2-109) is developed such that when these equations are integrated, the resulting transfer trajectory has final conditions which are reasonably close to the desired final conditions. This trajectory is defined as the initial-approximate transfer trajectory. In Chapter 4, the initial-approximate transfer trajectory will be used as a starting trajectory for computing the final transfer trajectory which more closely satisfies the desired boundary conditions.

3.1 Selection of the Unknown Initial Conditions - An Outline of the Problem

The minimum-fuel trajectory must satisfy the differential Eqs. (2-100) - (2-109) and the boundary conditions (2-121) - (2-127). To find a trajectory which satisfies this system of equations is a relatively difficult task since at the initial time, the values of the true anomaly f, the thrust angle Ψ, the variable ν and the adjoint variable λ_φ are unknown. The basic problem is to find that correct combination of the initial conditions for f, Ψ, ν and λ_φ such that when the system of equations (2-100) - (2-109) is integrated, the boundary conditions at the final time are satisfied. The final time

in this case is not fixed but can be defined as the time at which all the conditions at the end of the trajectory are satisfied.

As mentioned in Chapter 2, only three of the four unknown initial conditions are independent. With three of the unknown initial conditions selected, the fourth condition is fixed through the requirement that H_o vanish at the initial time. It has proved convenient in this study to set independently the initial conditions on f, Ψ and ν and then solve for λ_φ in the equation

$$H_o(t_o) = 0$$

The expression for H_o given by Eq. (2-58) can be transformed through application of Eqs. (2-28) - (2-31), (2-83), and (2-67) - (2-69) to give

$$H_o = \frac{e(1 + e\cos f)\sin f}{p^{1.5}}\left[\frac{\Lambda(1 + e\cos f)}{p^{1.5}}\left[\sin\Psi\,\frac{\cos f + e\cos 2f}{\sin f} - 2\cos\Psi(1 + e\cos f)\right] - \nu + \frac{\lambda_\varphi(1 + e\cos f)}{e\sin f}\right] \tag{3-1}$$

Setting H_o equal to zero and solving for λ_φ yields

$$\lambda_\varphi = \left[\frac{e\,\nu\sin f}{1 + e\cos f} - \frac{e\,\Lambda\sin f}{p^{1.5}}\left[\sin\Psi\,\frac{\cos f + e\cos 2f}{\sin f} - 2\cos\Psi(1 + e\cos f)\right]\right]_{t=t_o} \tag{3-2}$$

Therefore, given the eccentricity e and the semi-latus rectum p of the initial orbit and having selected the initial values for f, Ψ, and ν, the value of λ_φ can be computed from Eq. (3-2). From Eq. (2-109) it is obvious that λ_φ remains constant throughout the entire transfer trajectory.

Having reduced the number of unknown initial conditions from four to three, we are still faced with a formidable problem in selecting these three initial conditions to obtain the desired transfer trajectory. The first step in solving this problem is to consider the equivalent minimum-fuel, two-impulse transfer trajectory.

3.2 Development of the Properties of the Initial-Approximate Transfer Trajectory from the Minimum-Fuel Two-Impulse Transfer Trajectory

Using the optimum two-impulse solution as the primary input, we desire to construct a finite-thrust transfer trajectory which, hopefully, will serve as a good first approximation to the desired final transfer trajectory. A discussion of the method for computing optimal two-impulse transfers along with the associated computer program is presented in Appendix A. As mentioned in Chapter 1, the method is based primarily on the work done by Lee[4], McCue and Bender[3], and Kern and Burghart[11]. The inputs to the program are the eccentricities and semi-latus rectums of the initial and final orbits and the change in the argument of pericenter. The program then computes the minimum-fuel two-impulse transfer and returns as output the following information concerning this transfer.

f_I = true anomaly at the beginning of the first impulse

f_{II} = true anomaly at the beginning of the second impulse

ΔV_I = magnitude of the impulsive velocity change vector for the first impulse

ΔV_{II} = magnitude of the impulsive velocity change vector for the second impulse

e_t = eccentricity of the transfer trajectory

p_t = semi-latus rectum of the transfer trajectory

Ψ_I = thrust-orientation angle for the first impulse

Ψ_{II} = thrust-orientation angle for the second impulse

$\Delta\varphi_I$ = change in the argument of pericenter on the first impulse

$\Delta\varphi_{II}$ = change in the argument of pericenter on the second impulse

Reasonable estimates for the minimum-fuel finite-thrust transfer trajectory can be computed from the above properties of the two-impulse trajectory. In the discussion which follows, these estimated quantities will be designated with a primed notation.

For the finite-thrust transfer trajectory, in addition to the initial and final times t_o and t_f, we define the switching times t_1 and t_2 in the following manner:

t_1 = time at which the first thrusting interval is terminated

t_2 = time at which the second thrusting interval is initiated

As a first approximation, the durations of the first and second thrusting intervals for the finite-thrust trajectory are chosen so that the integrals of rocket acceleration for the finite-thrust case are equal to the respective magnitudes of the velocity change vectors for the impulsive case. Integrating the basic finite-thrust rocket equations

$$m \frac{dv}{dt} = - c \frac{dm}{dt} \tag{3-3}$$

$$\frac{dm}{dt} = - \frac{a}{c} \tag{3-4}$$

over the two thrusting phases (t_o, t_1) and (t_2, t_f), eliminating the

mass between the resulting equations and solving for the time intervals, we have the following estimates for the durations of the thrusting intervals

$$\text{estimated duration of the first thrust interval} = \frac{\varrho}{a}\left(1 - e^{-\frac{\Delta V_I}{c}}\right) \tag{3-5}$$

$$\text{estimated duration of the second thrust interval} = \frac{\varrho}{a} e^{-\frac{\Delta V_I}{c}}\left(1 - e^{-\frac{\Delta V_{II}}{c}}\right) \tag{3-6}$$

Reasonable estimates for the true anomalies $f(t_o)$, $f(t_1)$, $f(t_2)$ and $f(t_f)$ of the finite-thrust trajectory at the initial time t_o and the switching times t_1, t_2, and t_f respectively are computed from the two-impulse transfer trajectory. Let these estimates be denoted by $f'(t_o)$, $f'(t_1)$, $f'(t_2)$ and $f'(t_f)$. The true anomaly $f'(t_o)$ is used as an initial condition on the finite-thrust trajectory. In some cases it will prove advantageous from a convergence standpoint to compute transfer trajectories in the backward direction. Under these conditions the true anomaly $f'(t_f)$ will be needed to start the computation. The angles $f'(t_1)$ and $f'(t_2)$, which are estimates of the initial and final true anomalies for the coast trajectory, are necessary in the actual computation of the coast trajectory and are also used to obtain an initial estimate for the time duration of the coast trajectory.

Initial estimates for the true anomalies $f(t_o)$, $f(t_1)$, $f(t_2)$ and $f(t_f)$ are computed from the corresponding two-impulse trajectory by conjecturing that on each thrusting phase the average true anomaly for the impulsive thrust should be equal to the true anomaly at the

midpoint of the finite-thrust interval. For the impulsive case the two average true anomalies are

$$\text{average true anomaly on the first impulse} = f_I - \frac{\Delta\varphi_I}{2} \tag{3-7}$$

$$\text{average true anomaly on the second impulse} = f_{II} - \frac{\Delta\varphi_{II}}{2} \tag{3-8}$$

Let v_I and v_{II} be approximations for the average transverse velocities on the first and second thrusting intervals respectively of the finite-thrust trajectory. The transverse velocity v, from Eqs. (2-67) and (2-68), is

$$v = \frac{(1 + e \cos f)^2}{p^{1.5}} \tag{3-9}$$

Substituting average eccentricities, semi-latus rectums, and the average true anomalies of Eqs. (3-7) and (3-8) into Eq. (3-9) we have

$$v_I = \frac{\left[1 + \frac{e_o + e_t}{2} \cos\left(f_I - \frac{\Delta\varphi_I}{2}\right)\right]^2}{\left(\frac{p_o + p_t}{2}\right)^{1.5}} \tag{3-10}$$

$$v_{II} = \frac{\left[1 + \frac{e_t + e_f}{2} \cos\left(f_{II} - \frac{\Delta\varphi_{II}}{2}\right)\right]^2}{\left(\frac{p_t + p_f}{2}\right)^{1.5}} \tag{3-11}$$

The true anomalies at the midpoints of the thrusting phases of the finite-thrust trajectory will then be approximately

$$\text{true anomaly at the midpoint of the first finite-thrust phase} = f'(t_o) + \frac{v_I(t_1 - t_o)}{2} - \frac{\Delta\varphi_I}{2} \tag{3-12}$$

$$\begin{matrix}\text{true anomaly at the} \\ \text{midpoint of the second} \\ \text{finite-thrust phase}\end{matrix} = f'(t_2) + \frac{v_{II}(t_f - t_2)}{2} - \frac{\Delta\varphi_{II}}{2} \qquad (3\text{-}13)$$

Equating the right hand sides of Eqs. (3-7) and (3-12) and solving for $f'(t_o)$ gives

$$f'(t_o) = f_I - \frac{v_I(t_1 - t_o)}{2} \qquad (3\text{-}14)$$

Likewise, equating the right hand sides of Eqs. (3-8) and (3-13) and solving for $f'(t_2)$, we have

$$f'(t_2) = f_{II} - \frac{v_{II}(t_f - t_2)}{2} \qquad (3\text{-}15)$$

The changes in the true anomalies on the first and second finite-thrust phases will be approximately $v_I(t_1 - t_o) - \Delta\varphi_I$, and $v_{II}(t_f - t_2) - \Delta\varphi_{II}$, respectively. Therefore, from Eqs. (3-14) and (3-15), the true anomalies $f'(t_1)$ and $f'(t_f)$ at the ends of the two finite-thrusting phases will be as follows:

$$f'(t_1) = f_I - \frac{v_I(t_1 - t_o)}{2} + v_I(t_1 - t_o) - \Delta\varphi_I \qquad (3\text{-}16)$$

$$f'(t_f) = f_{II} - \frac{v_{II}(t_f - t_2)}{2} + v_{II}(t_f - t_2) - \Delta\varphi_{II} \qquad (3\text{-}17)$$

or

$$f'(t_1) = f_I + \frac{v_I(t_1 - t_o)}{2} - \Delta\varphi_I \qquad (3\text{-}18)$$

$$f'(t_f) = f_{II} + \frac{v_{II}(t_f - t_2)}{2} - \Delta\varphi_{II} \qquad (3\text{-}19)$$

A reasonable first approximation for the duration of the coast interval can be computed by means of Kepler's Eq.[7], using the

estimated true anomalies $f'(t_1)$ and $f'(t_2)$. Let

$$t_2' - t_1' = \text{estimated duration of the coast trajectory}$$

Then

$$t_2' - t_1' = \left[\frac{p_t}{1 - e_t^2}\right]^{1.5} [E'(t_2) - E'(t_1) - e_t(\sin E'(t_2) - \sin E'(t_1))] \quad (3\text{-}20)$$

where the eccentric anomalies $E'(t_2)$ and $E'(t_1)$ are computed as follows

$$E'(t_1) = 2\tan^{-1}\left[\sqrt{\frac{1 - e_t}{1 + e_t}} \tan \frac{f'(t_1)}{2}\right] \quad (3\text{-}21)$$

$$E'(t_2) = 2\tan^{-1}\left[\sqrt{\frac{1 - e_t}{1 + e_t}} \tan \frac{f'(t_2)}{2}\right] \quad (3\text{-}22)$$

3.3 Selection of Proper Initial Values for Ψ and ν to Achieve the Desired First Thrusting Interval

The properties of the initial-approximate transfer trajectory were derived in Section 3.2 using as a basis the minimum-fuel two-impulse transfer. This analysis established the initial true anomaly f and the durations of the two thrusting phases for the finite-thrust trajectory.

It is now desired to find initial values for Ψ and ν, such that integration of Eqs. (2-100) - (2-109) will result in the initial-approximate transfer trajectory. In this section we shall consider selection of an initial set of Ψ and ν values such that the desired first thrusting interval is attained, or, in other words, the switching function s exhibits the behavior shown in Fig. 3-1 . This will be accomplished by choosing the initial value of ν from some reasonable range and then finding the initial value of Ψ to achieve the desired switching. It is therefore necessary to establish a reasonable range of initial values for ν .

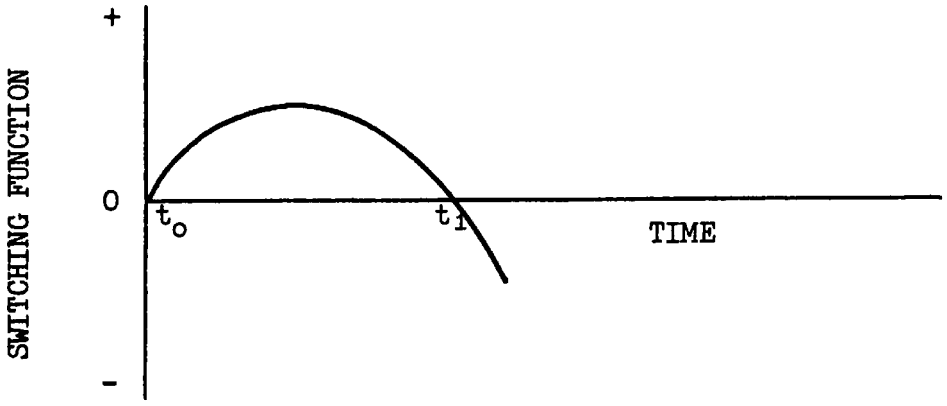


Fig. 3-1: Desired switching function behavior for the first thrusting interval.

The initial thrust angle Ψ_I available from the two-impulse transfer will be used as the first estimate of the starting value for Ψ in the finite-thrust case. Culp[17] has proved that Ψ_I will be near either 0^o or 180^o. For the transfer trajectories considered in this study, Ψ_I was usually within $\pm 5^o$ of 0^o or 180^o.

With the initial value of Ψ restricted as outlined above, it is now possible to determine a range of possible initial values for ν. To gain a better understanding of the switching function action, we write Eqs. (2-106), (2-108) and (2-105) in the following manner

$$\frac{d\Psi}{dt} = 2\nu + \frac{\rho}{\Lambda}\cos(\Psi - \beta) \tag{3-23}$$

$$\frac{ds}{dt} = \frac{h - s}{\Lambda}\rho\sin(\Psi - \beta) + hu - s\left(2u + \frac{a\xi w}{c}\right) \tag{3-24}$$

$$\frac{d\Lambda}{dt} = 2u\Lambda + \rho\sin(\Psi - \beta) \tag{3-25}$$

where

$$\rho = \sqrt{\lambda_\varphi^2 + \nu^2} \tag{3-26}$$

$$\beta = \tan^{-1}\left[\frac{\lambda_\varphi}{\nu}\right] \tag{3-27}$$

With the initial value of Ψ restricted to a small range near either 0° or 180°, the term containing $\sin\Psi$ in Eq. (3-2) can be safely neglected. Making use of Eqs. (2-68), (2-125), and (3-9), we have from Eq. (3-2) the following approximate expression for λ_{φ}.

$$\lambda_{\varphi} \cong \left[\frac{h\ e\ \sin f}{\sqrt{p}}\left(\frac{\nu}{v} + 2\cos\Psi\right)\right]_{t=t_o} \tag{3-28}$$

A reasonable range of starting values for ν can now be established from the insight provided by Eqs. (3-23) - (3-28). It is obvious from Fig. 3-1 that $\frac{ds}{dt}$ must be positive at time t_o and must become negative at some time within the interval (t_o, t_1). Since the optimum thrust interval must be relatively small, this reversal of $\frac{ds}{dt}$ from positive to negative must take place within a relatively short time interval.

A closer examination of the right hand side of Eq. (3-24) reveals under what conditions the switching function will perform in the desired manner. Obviously, the third term on the right hand side of Eq. (3-24) will not be important in forming the desired switching function characteristics since this term will be zero at the beginning and end of the thrusting interval. Therefore, proper switching must be provided by the combined actions of the terms $\frac{h - s}{\Lambda}\rho\sin(\Psi - \beta)$ and hu of Eq. (3-24). For convenience define

$$F(t) = \frac{h - s}{\Lambda}\rho\sin(\Psi - \beta) + hu \tag{3-29}$$

If switching is to take place, it is obvious that at times t_o and t_1 the following conditions must hold

$$F(t_o) > 0 \tag{3-30}$$

and

$$F(t_1) < 0 \tag{3-31}$$

Since the time interval (t_o, t_1) is relatively small, proper switching will be most logically accomplished if both $F(t_o)$ and $F(t_1)$ take on relatively small values. The term hu will ordinarily be relatively small since u is the non-dimensional radial velocity. Therefore, the term $\frac{h - s}{\Lambda} \rho \sin(\Psi - \beta)$ must also be "small", where "small" implies the same order of magnitude as the term hu. It follows that either $\rho(t_o)$ or $\sin(\Psi - \beta)_{t=t_o}$ must be "small". From Eqs. (3-26) and (3-28) it appears reasonable to conclude that $\rho(t_o)$ can be made "small" by choosing a $v(t_o)$ with a "small" absolute value. However, in the next section it will be shown that the second thrusting interval cannot be attained if the absolute value of $v(t_o)$ is "small". Therefore, the term $\frac{h - s}{\Lambda} \rho \sin(\Psi - \beta)_{t=t_o}$ must be made "small" by making $\sin(\Psi - \beta)_{t=t_o}$ "small". This requires that either

$$(\Psi - \beta)_{t=t_o} \cong 0^o \tag{3-32}$$

or

$$(\Psi - \beta)_{t=t_o} \cong 180^o \tag{3-33}$$

If $(\Psi - \beta)_{t=t_o}$ is near 0^o, then from Eq. (3-23) $\frac{d\Psi}{dt}$ will be positive, since ρ, Λ, and v are all, by definition, positive quantities. Therefore, the angle Ψ will be increasing and the term $\frac{h - s}{\Lambda} \rho \sin(\Psi - \beta)$ will likewise be increasing. This results in higher positive values for $\frac{ds}{dt}$ making it impossible for the switching function to pass through zero at time t_1 .

On the other hand, if the initial value of $(\Psi - \beta)$ is chosen to be near 180^o, it is reasonable to expect that proper switching can be

accomplished since now, from Eq. (3-23), the initial value of $\frac{d\Psi}{dt}$ can be made either positive or negative depending upon the initial value of ρ .

From the above discussion it can be concluded that if proper switching is to take place, the initial value of the angle $(\Psi - \beta)$ must be near 180°. That is

$$(\Psi - \beta)_{t=t_o} \cong 180^\circ \tag{3-34}$$

Since $\Psi(t_o)$ is either near 0° or 180° it follows from the above equation that $\nu(t_o)$ must be chosen so that the angle $\beta(t_o)$ differs from the initial thrust angle $\Psi(t_o)$ by approximately 180°. Therefore, as a consequence of Eq. (3-27), $\nu(t_o)$ must be negative for $\Psi(t_o)$ near 0° and positive for $\Psi(t_o)$ near 180°.

The above arguments have led to conclusions regarding the sign of $\nu(t_o)$. Further refinements on the choice of the absolute magnitude of $\nu(t_o)$, can be established through the following reasoning. Since $\beta(t_o)$ should be near either 180° or 0°, it follows from Eq. (3-27) that $|\nu(t_o)|$ should be much larger than $|\lambda_\varphi|$. That is

$$|\nu(t_o)| \gg |\lambda_\varphi| \tag{3-35}$$

If, for example

$$\nu(t_o) = - \nu(t_o)$$

and

$$\cos\Psi = 1$$

then we can apply Eqs. (3-28) and (2-67) to give

$$\frac{\lambda_\varphi}{\nu(t_o)} \cong - \frac{e \sin f}{ph}$$

It is obvious, therefore, from the above analysis that $|\nu(t_o)|$ should

certainly not be less than $v(t_o)$ if inequality (3-35) is to be valid. This then establishes a somewhat arbitrary lower limit on $|v(t_o)|$. The maximum value of $|v(t_o)|$ can be established from the requirement that $\frac{ds}{dt}$ must undergo a reversal from positive to negative on the first thrusting interval. From inequality (3-35) and Eq. (3-26) an approximate initial value for ρ will be

$$\rho(t_o) \cong |v(t_o)| \tag{3-36}$$

Since $(\Psi - \beta)_{t=t_o}$ is near 180^o and the initial value of Λ, from Eq. (2-125), is equal to unity, we have approximately from Eqs. (3-23) and (3-36)

$$\left.\frac{d\Psi}{dt}\right]_{t=t_o} \cong 2v(t_o) - |v(t_o)| \tag{3-37}$$

From Eq. (3-37) it is obvious that as $|v(t_o)|$ becomes larger than $2v(t_o)$, $\left.\frac{d\Psi}{dt}\right]_{t=t_o}$ will become increasingly negative. Therefore, the term $\frac{h - s}{\Lambda}\rho\sin(\Psi - \beta)$ will become more positive since $(\Psi - \beta)$ will be decreasing from an initial value near 180^o. This will make impossible a reversal of $\frac{ds}{dt}$ from positive to negative on the first thrusting interval, and hence the desired switching will not be attained. Therefore an approximate upper limit on $|v(t_o)|$ will be

$$|v(t_o)|_{max} \cong 2v(t_o) \tag{3-38}$$

The above conclusions concerning reasonable initial values for v are in summary

$$v(t_o)]_{\Psi(t_o)\cong 0^o} < 0 \tag{3-39}$$

$$v(t_o)]_{\Psi(t_o)\cong 180^o} > 0 \tag{3-40}$$

and

$$v(t_o) < |v(t_o)| \leq 2v(t_o) \tag{3-41}$$

Further refinements on the choice of $v(t_o)$ will be made in the next section when establishing the desired second thrusting interval.

With $v(t_o)$ established in the range provided by inequalities (3-39) - (3-41), it is now necessary to develop a method for adjusting the initial thrust angle $\Psi(t_o)$ such that the switching function exhibits the behavior shown in Fig. 3-1. The following method for finding $\Psi(t_o)$ is based upon the second derivative of the switching function. Differentiating Eq. (2-108), substituting Eqs. (2-22) and (2-125) and setting $s = 0$, gives for the second derivative of the switching function evaluated at $t = t_o$

$$\left.\frac{d^2s}{dt^2}\right]_{t_o} = \left[\left[(\lambda_\varphi \cos\Psi - v \sin\Psi)\left(\frac{ds}{dt} + uh + h\frac{d\Lambda}{dt}\right)\right.\right.$$

$$+ h\left(\frac{dv}{dt}\sin\Psi + v\frac{d\Psi}{dt}\cos\Psi + \lambda_\varphi \frac{d\Psi}{dt}\sin\Psi\right)$$

$$\left.\left. + h\frac{du}{dt} - u^2h - 2u\frac{ds}{dt} - \frac{a\xi w}{c}\frac{ds}{dt}\right]\right]_{t=t_o} \tag{3-42}$$

This second derivative can be evaluated with the aid of Eqs. (2-23), (2-105) - (2-108), and (2-67) - (2-69).

The computation of the initial thrust angle is based upon principle that the duration of the first thrusting interval is proportional to the ratio of first and second derivatives of the switching function evaluated at time t_o. In other words

$$(t_1 - t_o) = - K \left[\frac{\frac{ds}{dt}}{\frac{d^2s}{dt^2}} \right]_{t=t_o} \tag{3-43}$$

If $\frac{d^2s}{dt^2}$ were constant over the time interval $t_o - t_1$, then the proportionality factor K in the above equation would be equal to two.

The initial thrust angle $\Psi(t_o)$ is now computed in the following manner. To facilitate the discussion we re-write Eq. (3-43) in the form

$$\left[\frac{\frac{ds}{dt}}{\frac{d^2s}{dt^2}} \right]_{t=t_o} = - \frac{t_1 - t_o}{K} \tag{3-44}$$

To solve this equation, the initial true anomaly, $f(t_o)$, is computed from Eq. (3-14), and a reasonable starting value for $\nu(t_o)$ can be selected from the range defined by inequalities (3-39) - (3-41). Using K = 2 as a starting value, we solve Eq. (3-44) for $\Psi(t_o)$ by an iterative type approach. The Eqs. (2-100) - (2-109) are then integrated to determine the actual duration of the first thrusting interval. Readjustment of the proportionality factor K will then be necessary in order to make the actual thrusting interval equal to the desired thrusting interval. If we define

$t_{1_{desired}}$ = desired time for termination of first thrusting phase

$t_{1_{actual}}$ = termination time of first thrusting interval established by integration

then a new value of K is computed according to the following rule

$$K_{new} = \left(1 - \frac{t_{1_{desired}} - t_{1_{actual}}}{t_{1_{desired}} - t_o}\right) K_{old} \qquad (3\text{-}45)$$

This new value of K is now substituted into Eq. (3-44), and Eq. (3-44) is once again solved iteratively for $\Psi(t_o)$. The process is repeated until $t_{1_{actual}}$ becomes within a certain tolerance of $t_{1_{desired}}$.

This method of determining $\Psi(t_o)$ has proven very reliable and efficient. In the actual computer program, double-precision arithmetic is used for the iterative solution of Eq. (3-44), but the integration of Eqs. (2-100) - (2-109) is performed in single-precision arithmetic. For most minimum-fuel problems considered, the difference between $t_{1_{desired}}$ and $t_{1_{actual}}$ can be made less 5×10^{-7} time units in three or four iterations. The computer subprograms PSISET, Y7SET, and DURATN, described in Appendix B, are used to set $\Psi(t_o)$ in the manner illustrated above. The subprograms Y7SET and DURATN are used for the iterative solution of Eq. (3-44).

3.4 Selection of Proper Initial Values for Ψ and ν to Achieve the Desired Second Thrusting Interval

The durations of the thrusting intervals along the initial-approximate transfer trajectory have been established from the minimum-fuel two-impulse transfer. With the initial value of the true anomaly f also derived from the two-impulse transfer, we are considering a method for finding the initial values for the thrust orientation angle Ψ and the variable ν such that the desired durations of the two thrusting intervals are attained. In Section 3.3, after determining the range of possible $\nu(t_o)$ values, a method was developed for finding $\Psi(t_o)$ such that the desired duration of the first thrusting interval

was attained. In this section we shall develop a method for determining $\nu(t_o)$ such that the desired duration of the second thrusting interval is also achieved. The switching function for the complete transfer trajectory will be as shown in Fig. 3-2 .

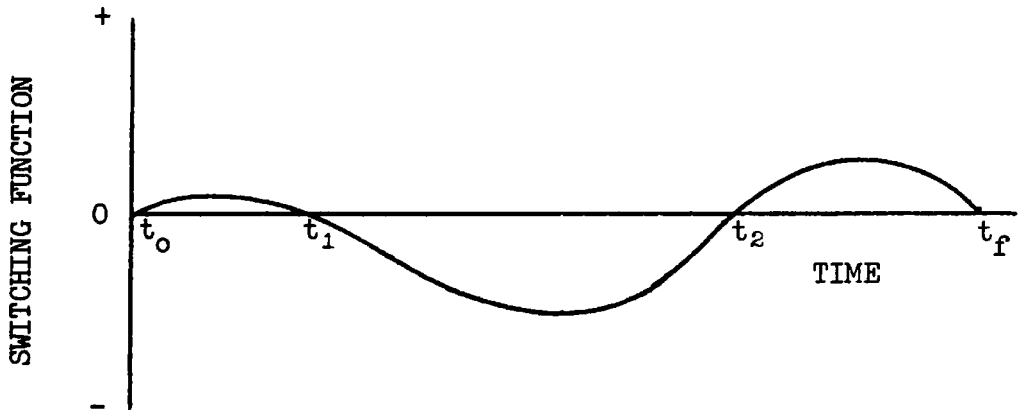


Fig. 3-2: Desired switching function behavior along the initial-approximate transfer trajectory.

In computing the initial-approximate transfer trajectory, only the duration $(t_2 - t_f)$ of the second thrusting interval is attained while the time t_2 at which this thrusting interval is initiated is left open. Eqs. (3-23), (3-24), (3-25), (2-107), (3-26), and (3-27) provide a considerable amount of insight as to how $\nu(t_o)$ should be chosen in order to satisfy the requirements of the second thrusting phase. These equations are repeated for convenience

$$\frac{d\Psi}{dt} = 2\nu + \frac{\rho}{\Lambda}\cos(\Psi - \beta) \tag{3-46}$$

$$\frac{ds}{dt} = \frac{h - s}{\Lambda}\rho\sin(\Psi - \beta) + hu - s\left(2u + \frac{a\xi w}{c}\right) \tag{3-47}$$

$$\frac{d\Lambda}{dt} = 2u\Lambda + \rho\sin(\Psi - \beta) \tag{3-48}$$

$$\frac{d\nu}{dt} = \Lambda\, h(3h^2 \sin\Psi - a\xi w) \tag{3-49}$$

$$\rho = \sqrt{\lambda_\varphi^2 + \nu^2} \tag{3-50}$$

$$\beta = \tan^{-1}\left[\frac{\lambda_\varphi}{\nu}\right] \tag{3-51}$$

Minimum-fuel transfer trajectories can be classified according to the type of thrusting which takes place on the first and second thrusting intervals. Let forward and rearward thrustings be defined as thrustings along which the thrust angle Ψ is near 0° and 180° respectively. Then each transfer trajectory will fall into one of the following four classes:

1. Forward-rearward
2. Rearward-forward
3. Forward-forward
4. Rearward-rearward

The corresponding minimum-fuel two-impulse transfer trajectory will indicate the particular thrusting sequence to be employed.

Each of the above four thrusting sequences can be realized through proper selection of $\nu(t_o)$. However, the switching mechanics for the first two sequences and the last two sequences are quite notably different. Therefore, the selection of $\nu(t_o)$ for classes one and two will be discussed separately from classes three and four.

3.4.1 Selection of the Initial Value of ν for Forward-Rearward or Rearward-Forward Thrusting Sequences

For discussion purposes we shall consider first the case of a forward-rearward transfer trajectory. Hence $\Psi(t_o)$ is near 0°, $\beta(t_o)$ is very close to 180° and from inequalities (3-39) - (3-41) we have for $\nu(t_o)$

$$-2v(t_o) \leq \nu(t_o) < -v(t_o) \tag{3-52}$$

For these two types of thrusting sequences the thrust vector must rotate through approximately 180^o along the transfer trajectory. If, for example, $\nu(t_o)$ is only slightly greater than $-2v(t_o)$, then, from Eqs. (3-46) and (3-50), $\frac{d\Psi}{dt}$ will be slightly positive and Ψ will increase slowly from an initial value near zero. With Ψ in this range, it is evident from Eq. (3-49) that ν will increase towards zero very slowly. Hence, ρ will remain relatively large (Eq. 3-50) and, consequently, $\frac{d\Psi}{dt}$ and also Ψ will remain small (Eq. 3-46). Therefore, if $\nu(t_o)$ is nearly equal to $-2v(t_o)$, the inherent "stability" of Eqs. (3-46) and (3-49) will prevent the thrust angle from increasing significantly along the transfer trajectory. Thus, in order to produce a 180^o rotation of the thrust vector it is necessary that $\nu(t_o)$ be chosen appreciably greater than $-2v(t_o)$. This choice of $\nu(t_o)$ will then generate a relatively large positive $\frac{d\Psi}{dt}$ at the start of the transfer trajectory.

Suppose for example that the initial value of $\nu(t_o)$ is chosen equal to $-1.5v(t_o)$. The initial thrust angle $\Psi(t_o)$ is then determined, as outlined in Section 3.3, to obtain the correct first thrusting interval. Let the time behavior of Ψ, ν, β, $(\Psi - \beta)$, and s for this example case be as shown in Fig. 3-3. In addition to the times t_o, t_1, t_2, and t_f, which define the three phases of the transfer trajectory, we define the time t_r as that time at which the variable ν passes through zero.

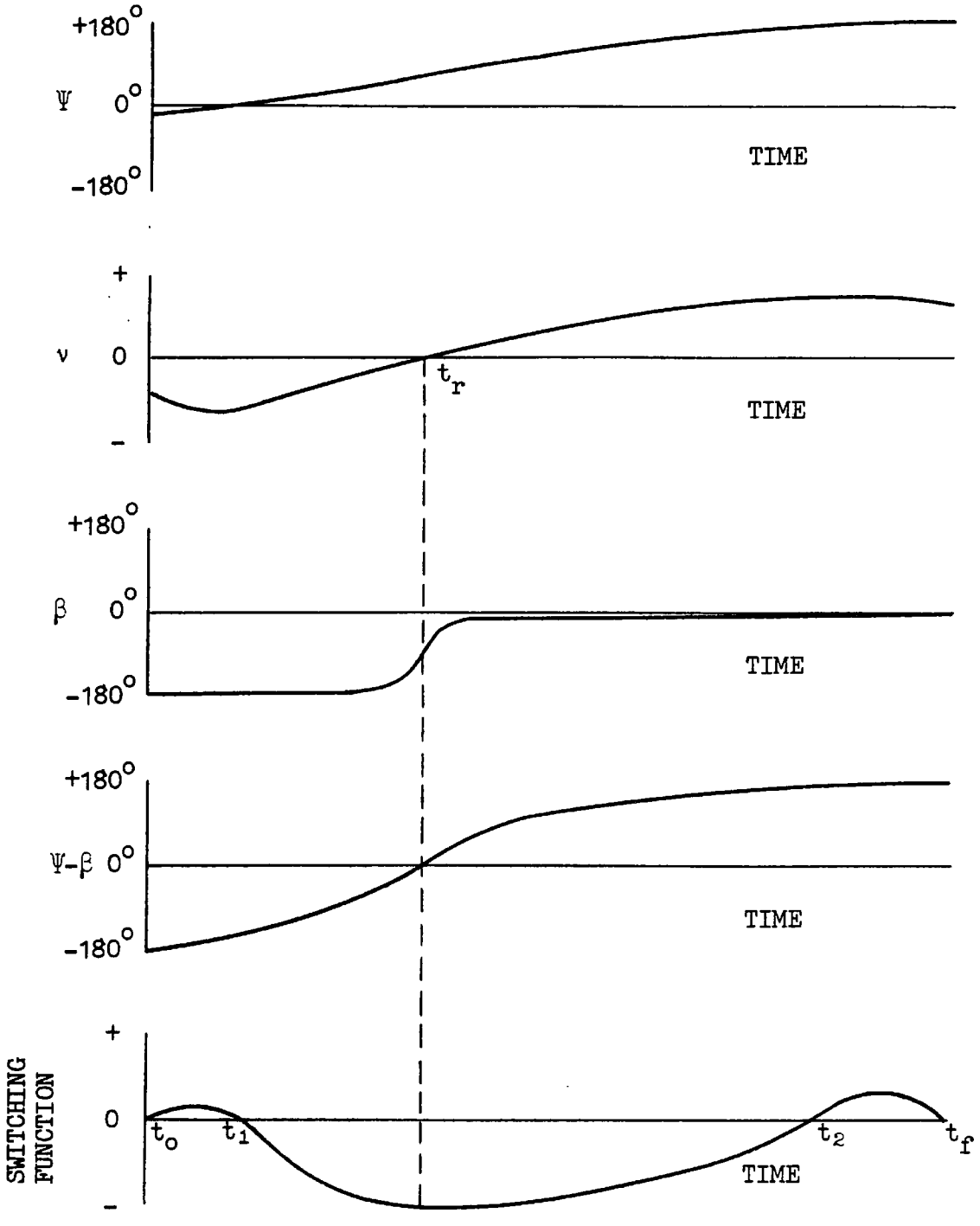


Fig. 3-3: Typical behavior of Ψ, ν, β, $\Psi - \beta$, and the switching function over an entire transfer trajectory for the case of a forward-rearward thrusting sequence.

We consider first the coupled behavior of ν and Ψ over the coast interval (t_1, t_2). For the coast phase $\xi = 0$, and Eq. (3-49) becomes

$$\left.\frac{d\nu}{dt}\right]_{coast} = 3h^3\Lambda \sin\Psi \tag{3-53}$$

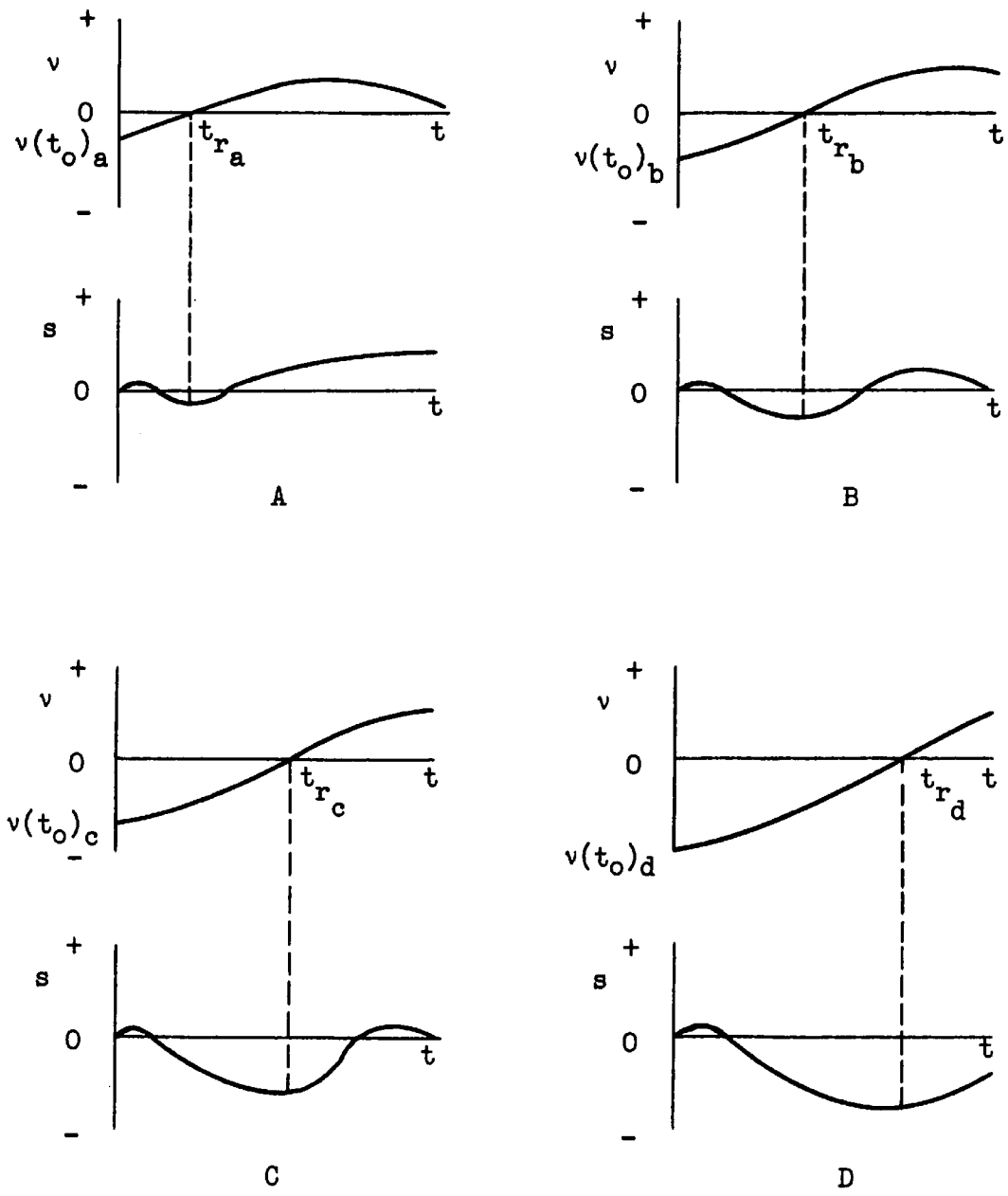
Since $\nu(t_o)$ was chosen so that initially $\frac{d\Psi}{dt}$ would be positive, Ψ must be increasing at the beginning of the transfer trajectory. As $\sin\Psi$ becomes more positive, ν must increase, as is obvious from Eq. (3-53). This increase in ν from negative values towards zero causes a decrease in ρ . Hence, $\rho\cos(\Psi - \beta)$ takes on smaller negative values, and $\frac{d\Psi}{dt}$ becomes more positive. The net result is an increase in Ψ and hence, a corresponding increase in ν, as is evident from Eq. (3-53). With this choice of $\nu(t_o)$ it follows from Eq. (3-28) that $|\lambda_\varphi|$ will ordinarily be much less than $|\nu(t_o)|$. Therefore, as a consequence of Eq. (3-51), β will remain near -180° until ν approaches zero at which time β will undergo a very rapid change from approximately 180° to 0°. The actual manner in which β changes depends upon the sign of λ_φ, but this is not critical to the discussion. Because of this rapid change in β at the time t_r, the angle $(\Psi - \beta)$ will change by almost 180°. Thus, $\sin(\Psi - \beta)$ reverses from negative to positive in a very short time span. This creates a reversal in $\frac{ds}{dt}$ from negative to positive as is evident from Eq. (3-48). At time t_2 the switching function again passes through zero, and the second thrusting phase begins. As $(\Psi - \beta)$ increases through 180°, $\frac{ds}{dt}$ again becomes negative, and the switching function returns to zero at time t_f.

It is of critical importance to recognize from the above discussion that the reversal in $\frac{ds}{dt}$ along the coast phase takes place very close to the time t_r at which ν passes through zero. Since t_r is primarily dependent on the initial value of ν, the time at which the switching function starts to increase on the coast phase can be controlled by the initial value of ν . As $\nu(t_o)$ is made more negative, t_r will increase.

Fig. 3-4 shows the effects on the switching function of different choices for $\nu(t_o)$. In each of the cases shown it is assumed that $\Psi(t_o)$ is adjusted so that the duration of the first thrusting interval remains fixed.

As is evident from Fig. 3-4, the larger values of $|\nu(t_o)|$ result in larger values for t_r, which in turn cause t_2 to be larger.

It is not immediately obvious that larger values of $|\nu(t_o)|$, in addition to delaying the second thrust interval, actually shorten this thrust interval. Although Ψ continues to increase throughout the transfer trajectory, $\sin(\Psi - \beta)$ remains negative until the reversal time t_r at which time β undergoes the 180° shift. A delay in this reversal time causes $\sin(\Psi - \beta)$ to remain negative over a larger percentage of the transfer trajectory, and therefore, the switching function becomes more negative. The time interval during which $\sin(\Psi - \beta)$ is positive is now reduced since $\sin(\Psi - \beta)$ increases through 180°. This reduction in the second thrusting interval caused by an increase in $|\nu(t_o)|$ is illustrated in Fig. 3-5 . In case B of Fig. 3-5 the larger $|\nu(t_o)|$ causes a larger t_r. This results in $\sin(\Psi - \beta)$ being negative over a larger proportion of the transfer trajectory, and consequently, the switching function tends to be more negative. The time t_p is shown in Fig. 3-5 as the time at which



$$\left|v(t_o)_a\right| < \left|v(t_o)_b\right| < \left|v(t_o)_c\right| < \left|v(t_o)_d\right|$$

Fig. 3-4: Effect of $v(t_o)$ upon the initiation time of the second thrusting interval (forward-rearward thrusting sequence).

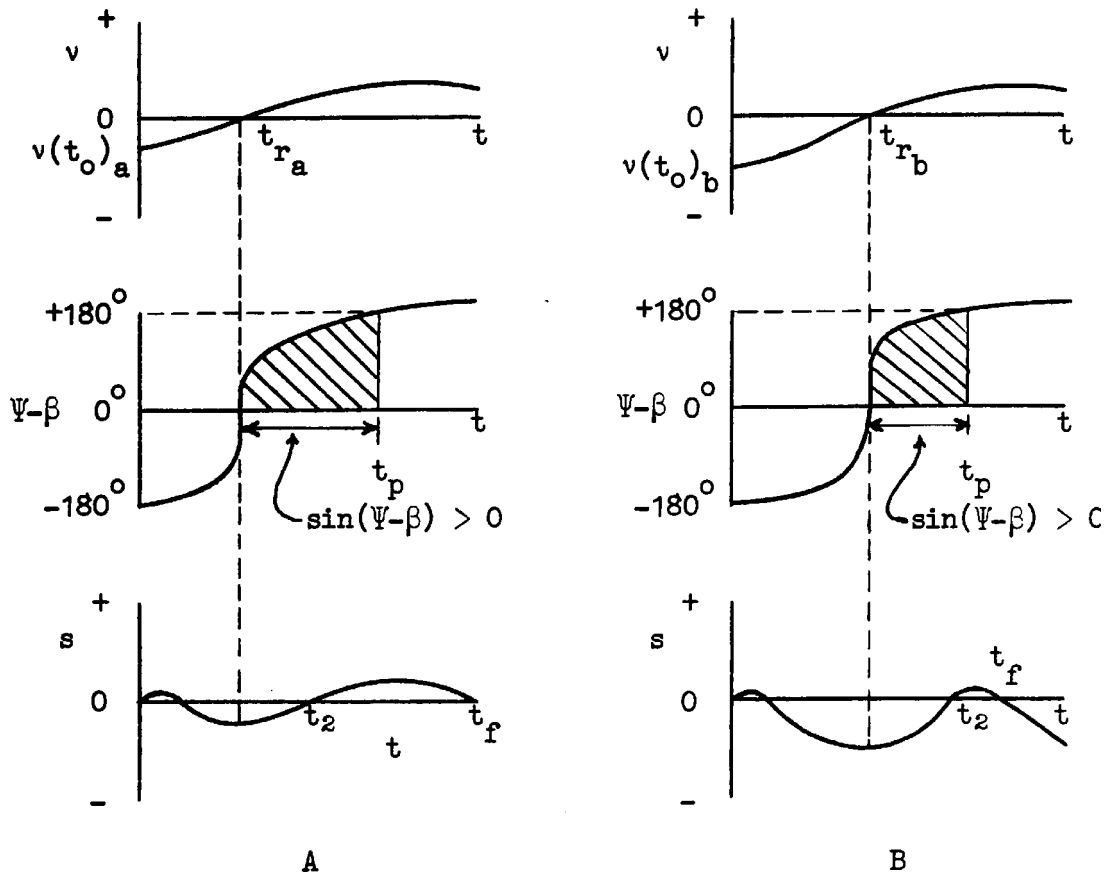


Fig. 3-5: Effect of $v(t_o)$ upon the length of the second thrusting interval (forward-rearward thrusting sequence).

$\sin(\Psi - \beta)$ again becomes negative. Because of the steady increase in Ψ, this time does not vary to any large extent under reasonable changes in $v(t_o)$.

This discussion has centered about the forward-rearward thrusting sequence. Similar conclusions are applicable for the rearward-forward case. In this case $v(t_o)$ is initially positive and decreases to zero at the time t_r. Increasing $|v(t_o)|$ will again cause t_r to increase, resulting in a reduction in the second thrusting interval.

The preceding analysis has shown for the forward-rearward and rearward-forward thrusting sequences that increasing $|\nu(t_o)|$ will decrease the duration of the second thrusting interval. Therefore, an iterative type approach can be employed to determine the value of ν which will yield the desired second thrusting interval duration. The method is as follows:

1. Select an initial, reasonable $\nu(t_o)$. For example

   $\nu(t_o) = 1.5\nu(t_o)$ for rearward-forward thrusting

   $\nu(t_o) = -1.5\nu(t_o)$ for forward-rearward thrusting

2. Compute $\Psi(t_o)$ by the method of Section 3.3 such that the correct first thrusting interval is attained.

3. Integrate Eqs. (2-100) - (2-109) to some time $t_f + \Delta t$ sufficient for the transfer to be accomplished. The time t_f can be easily obtained from the two-impulse transfer.

4. Compute the second thrust interval. If this interval is too small, decrease $|\nu(t_o)|$. If too large, increase $|\nu(t_o)|$.

5. Repeat, using the improved value for $\nu(t_o)$, until the actual second thrusting interval duration as determined from the integration is within a certain tolerance of the desired second thrusting interval duration.

The details of the actual computation of $\nu(t_o)$ are presented under subroutine NUSET in Appendix B. Using the above approach with single-precision arithmetic, it is possible in most cases to obtain

the actual second thrust interval duration within a tolerance of 5×10^{-5} of the desired second thrust interval duration.

3.4.2 Selection of the Initial Value of ν for Forward-Forward or Rearward-Rearward Thrusting Sequences

For transfer trajectories of this type the thrust angles on the two thrusting intervals are approximately equal. The problem now is to find a reasonable starting value for $\nu(t_o)$ such that this type of thrust angle behavior can be realized and that the switching function performs in the desired manner. It is obvious that the thrust vector must either rotate very little or else rotate by approximately 360° along the transfer trajectory.

A brief consideration of the switching mechanics defined by Eqs. (3-46) - (3-51) shows that the thrust vector is very unlikely to incur a rotation of 360° along the transfer trajectory. Large initial $\frac{d\Psi}{dt}$ values can be generated by employing relatively low $|\nu(t_o)|$ values. However, as discussed in Section 3.4.1 , this would not result in a large change in Ψ along the transfer trajectory since $\nu(t_o)$ would quickly pass through zero and the second thrusting interval would be initiated prematurely, with the angle Ψ most likely changing less than 180°. Relatively high values of $|\nu(t_o)|$ naturally will lead to low rates of change in Ψ, thus ruling out any possibility of a 360° rotation of the thrust vector in this region.

The thrust vector appears most likely therefore to experience only a small net rotation for the types of transfers considered here. It was pointed out in Section 3.4.1 that with $|\nu(t_o)|$ nearly equal to $2\nu(t_o)$ the thrust angle would not increase significantly along the transfer trajectory because of the stability afforded by the coupled behavior of Eqs. (3-46) and (3-49).

The above reasoning combined with computational experience has led to the conclusion that minimum-fuel transfer trajectories with forward-forward or rearward-rearward thrusting sequences must be generated with $|\nu(t_o)|$ very nearly equal to $2\nu(t_o)$. The sign of $\nu(t_o)$ is still determined from inequalities (3-39) and (3-40). This particular choice of $\nu(t_o)$ causes only a small net change in Ψ and β on the entire transfer trajectory. In the forward-rearward and rearward-forward transfer trajectories, switching is accomplished with both the angles Ψ and β changing by approximately 180^o. Hence the switching function is dominated by the term $\frac{h - s}{\Lambda} \rho \sin(\Psi - \beta)$ appearing on the right-hand side of Eq. (3-47). For the case of forward-forward or rearward-rearward transfers the switching function is influenced primarily by the first two terms on the right-hand side of Eq. (3-47), namely $\frac{h - s}{\Lambda} \rho \sin(\Psi - \beta)$ and hu. Successful switching is dependent upon maintaining a rather delicate balance between these two terms.

The actual switching mechanism on transfer trajectories of this type is rather complex. However, for the purposes of computing the minimum-fuel transfer trajectory it is only necessary to develop a method for establishing the desired first and second thrusting intervals. The first thrusting interval can be determined by the method of Section 3.3 . To establish the second thrusting interval we need to determine how changes in $\nu(t_o)$ will affect the duration of the second thrusting interval. Transfer trajectories of this type appear to fall into a rather consistent pattern. The following observations derived from computational experience and a careful analysis of Eqs. (3-46) - (3-51) have made possible the development of a method for computing $\nu(t_o)$.

The minimum-fuel two-impulse transfer indicates that the thrusting intervals for transfers of this type most normally occur within $\pm 60^{\circ}$ of apocenter and pericenter. Forward-forward transfer trajectories normally have the initial and final thrust intervals near pericenter and apocenter respectively, while just the opposite thrusting sequence takes place on rearward-rearward transfer trajectories.

For the switching near pericenter, where the transverse velocity v is relatively high, the thrust angle Ψ is increasing while for the switching near apocenter, where v is relatively low, Ψ is slowly decreasing. This is consistent with Eq. (3-46). Therefore, the pericenter switching is accomplished by a reversal from positive to negative of the term $\frac{h - s}{\Lambda} \rho \sin(\Psi - \beta)$, and apocenter switching is accomplished by a small increase in $\frac{h - s}{\Lambda} \rho \sin(\Psi - \beta)$ followed by a decrease in the term hu caused by a rapidly changing eccentricity.

Since $\frac{h - s}{\Lambda} \rho \sin(\Psi - \beta)$ reverses from positive to negative near pericenter, this term must be negative along the coast trajectory for true anomalies less than 180° and positive for true anomalies greater than 180°. In other words, along the coast trajectory the terms $\frac{h - s}{\Lambda} \rho \sin(\Psi - \beta)$ and hu will always be of opposite sign. Since these two terms tend to cancel one another along the coast trajectory, $\frac{ds}{dt}$ will remain relatively small.

The relative magnitudes of the terms $\frac{h - s}{\Lambda} \rho \sin(\Psi - \beta)$ and hu along the coast trajectory will establish the timing and duration of the second thrusting phase. As the sum of these two terms is made more negative, the second thrusting interval becomes shortened and delayed. From a consideration of Eqs. (3-50) and (3-46) we can conclude that decreasing $|v(t_o)|$ will decrease $\rho(t_o)$ and hence the initial value of $\frac{d\Psi}{dt}$ will increase making Ψ larger along the coast

trajectory. The net result will be a decrease in $\frac{h - s}{\Lambda} \rho \sin(\Psi - \beta)$ along the coast trajectory since the angle $(\Psi - \beta)$ is near 180°. Therefore $\frac{ds}{dt}$ will be more negative on the coast trajectory, and the second thrusting interval will be delayed and shortened. Likewise, if $|\nu(t_o)|$ is increased, the second thrusting interval will be earlier and of longer duration.

The ability of $\nu(t_o)$ to influence the duration of the second thrusting interval in the manner discussed above has led to the following method for computing $\nu(t_o)$ on a forward-forward or rearward-rearward type transfer:

1. As a starting value for $\nu(t_o)$ let

   $\nu(t_o) = -2\nu(t_o)$ for forward-forward thrusting

   $\nu(t_o) = 2\nu(t_o)$ for rearward-rearward thrusting

2. Compute $\Psi(t_o)$ by the method of Section 3.3 to obtain the correct first thrust interval.

3. Integrate Eqs. (2-100) - (2-109) to some time $t_f + \Delta t_f$ sufficient for the transfer to be accomplished.

4. Compute the second thrust interval. If this interval is too small, increase $|\nu(t_o)|$, if too large, decrease $|\nu(t_o)|$.

5. Repeat, using the improved value for $\nu(t_o)$, until the actual second thrust interval duration, as determined from the integration, is within a certain tolerance of the desired second thrust interval duration.

The initial value of $\nu(t_o)$ is computed in the digital program by subroutine NUSET, described in Appendix B. As mentioned previously this same subroutine computes $\nu(t_o)$ for the case of forward-rearward or rearward-forward transfer trajectories.

CHAPTER 4

CONVERGENCE TO THE DESIRED MINIMUM-FUEL FINITE-THRUST TRANSFER TRAJECTORY

In Chapter 3 a technique was developed for computing a transfer trajectory which was defined as the initial-approximate transfer trajectory. In establishing this trajectory, the durations of the two thrusting phases and the initial value of the true anomaly f were derived from the two-impulse transfer trajectory. Initial values of the thrust angle Ψ and the variable ν were then determined such that the thrusting intervals were of the desired duration. The initial thrust orientation angle from the two-impulse transfer was used as a starting value when computing the initial-approximate transfer trajectory. With few exceptions the final values of semi-latus rectum, eccentricity, and argument of pericenter for the initial-approximate transfer trajectory were reasonably close to the desired final values.

In this chapter the initial values of f, Ψ, and ν from the initial-approximate transfer trajectory will be adjusted so that the resulting finite-thrust transfer trajectory will have final values which more closely match the desired final values. This will be accomplished by first changing only the initial true anomaly $f(t_o)$ and then changing simultaneously the initial values of f, Ψ, and ν .

4.1 Definition of the Final Condition Errors

The initial-approximate transfer trajectory has final values of semi-latus rectum p eccentricity e and argument of pericenter φ which differ by a modest amount from the desired final values. In most cases, however, these differences between the actual and desired final values are too large to be ignored.

We define the errors in matching the final boundary conditions in the following manner. Let $p(t_f)_d$, $e(t_f)_d$, and $\varphi(t_f)_d$ be the desired final values of p, e, and φ respectively. In addition, define $p(t_f)_a$, $e(t_f)_a$, and $\varphi(t_f)_a$ as the actual final values of p, e, and φ respectively which result from direct integration of Eqs. (2-100) - (2-109). Then the errors ε_p, ε_e, ε_φ are defined in the following manner

$$\varepsilon_p = p(t_f)_d - p(t_f)_a \tag{4-1}$$

$$\varepsilon_e = e(t_f)_d - e(t_f)_a \tag{4-2}$$

$$\varepsilon_\varphi = \varphi(t_f)_d - \varphi(t_f)_a \tag{4-3}$$

with ε_φ being expressed in radians.

4.2 Adjustment of the Initial True Anomaly to Reduce the Argument of Pericenter Error

To start the convergence procedure, the initial value of the true anomaly is computed from the two-impulse transfer trajectory by means of Eq. (3-14). Initial values of Ψ and ν are then determined in order to achieve the desired thrusting durations. The resulting transfer trajectory is defined as the initial-approximate transfer trajectory. Adjustments are now made on the initial true anomaly $f(t_o)$ to reduce the argument of pericenter error ε_φ . This is accomplished by

recording ε_φ and then changing $f(t_o)$ slightly. With this new value of $f(t_o)$ we readjust the initial values of Ψ and ν in order to achieve the desired thrusting durations. ε_φ is recorded for this new transfer trajectory. The above procedure yields two pairs of $f(t_o)$ and ε_φ values. Subsequent values of $f(t_o)$ are chosen to reduce ε_φ to zero by linearly interpolating or extrapolating on the last two pairs of initial true anomalies and argument of pericenter errors. For example if $f(t_o)_n$ and $\varepsilon_{\varphi n}$ are the initial true anomaly and error in argument of pericenter respectively for the n^{th} iteration and $f(t_o)_{n+1}$ and $\varepsilon_{\varphi n+1}$ are defined in a similar manner for the $(n + 1)^{th}$ iteration, then the initial true anomaly for $(n + 2)^{th}$ iteration will be

$$f(t_o)_{n+2} = f(t_o)_{n+1} - \frac{\varepsilon_{\varphi_{n+1}}}{\text{SLOPE}} \tag{4-4}$$

where

$$\text{SLOPE} = \frac{\varepsilon_{\varphi_{n+1}} - \varepsilon_{\varphi_n}}{f(t_o)_{n+1} - f(t_o)_n} \tag{4-5}$$

This procedure is continued until the magnitude of $|\varepsilon_\varphi|$ becomes less than 2×10^{-8} radians.

4.3 Definition and Computation of the Sensitivity Matrix which Relates Small Changes in Initial Conditions to Changes in the Final Conditions

In this study the goal is to determine precisely three unknown initial conditions, $f(t_o)$, $\Psi(t_o)$, and $\nu(t_o)$, such that three prespecified terminal conditions, $p(t_f)$, $e(t_f)$, and $\varphi(t_f)$, are satisfied. Changing any of the initial conditions will result in some corresponding change in each of the three terminal conditions. A sensitivity matrix defined along some reference trajectory can be computed

by varying one at a time $f(t_o)$, $\Psi(t_o)$, and $\nu(t_o)$ and observing the corresponding changes produced in the terminal conditions $p(t_f)$, $e(t_f)$, and $\varphi(t_f)$. With $\Delta f(t_o)$, $\Delta\Psi(t_o)$, and $\Delta\nu(t_o)$ the changes in the initial conditions and $\Delta p(t_f)$, $\Delta e(t_f)$, and $\Delta\varphi(t_f)$ the resulting changes in the terminal conditions, we define a sensitivity matrix [G] as follows

$$[G] = \begin{bmatrix} \frac{\Delta p(t_f)}{\Delta f(t_o)} & \frac{\Delta p(t_f)}{\Delta \Psi(t_o)} & \frac{\Delta p(t_f)}{\Delta \nu(t_o)} \\ \frac{\Delta e(t_f)}{\Delta f(t_o)} & \frac{\Delta e(t_f)}{\Delta \Psi(t_o)} & \frac{\Delta e(t_f)}{\Delta \nu(t_o)} \\ \frac{\Delta \varphi(t_f)}{\Delta f(t_o)} & \frac{\Delta \varphi(t_f)}{\Delta \Psi(t_o)} & \frac{\Delta \varphi(t_f)}{\Delta \nu(t_o)} \end{bmatrix} \qquad (4\text{-}6)$$

Since $p(t_f)$, $e(t_f)$ and $\varphi(t_f)$ are independent and $f(t_o)$, $\Psi(t_o)$ and $\nu(t_o)$ are also independent, the matrix [G] must be non-singular. If the right-hand sides of Eqs. (2-100) - (2-109) were continuous, then the individual elements of the matrix [G] can be proven (ref. 19) to be continuous functions of the initial conditions $f(t_o)$, $\Psi(t_o)$ and $\nu(t_o)$. Continuity can also be shown for the actual case where Eqs. (2-100) - (2-109) are discontinuous because the switching function s is "well-behaved" over the entire trajectory. To prove this it would be necessary to consider the three trajectory phases separately and then show continuity at the switching points.

The matrix [G] can be used to relate small changes in the initial conditions to the final condition errors through the following equation

$$\begin{bmatrix} \Delta f(t_o) \\ \Delta \Psi(t_o) \\ \Delta \nu(t_o) \end{bmatrix} = [G]^{-1} \begin{bmatrix} \varepsilon_p \\ \varepsilon_e \\ \varepsilon_\varphi \end{bmatrix} \tag{4-7}$$

With ε_p, ε_e, ε_φ representing the errors in the final conditions, a new reference trajectory can be generated by changing the initial conditions in accordance with Eq. (4-7). The sensitivity matrix [G] corresponding to this new reference trajectory can be computed by again perturbing individually $f(t_o)$, $\Psi(t_o)$, and $\nu(t_o)$ and observing the resulting changes in $p(t_f)$, $e(t_f)$, and $\varphi(t_f)$. However, it is only necessary to compute the matrix [G] by this approach at the start of the convergence procedure. A method exists (Ref. 18) whereby subsequent computations of the matrix [G] can be made directly from the new reference trajectory without requiring small pertubations of the initial conditions followed by integration of the system equations. Let $[G]_n$, ε_{p_n}, ε_{e_n}, and ε_{φ_n} represent the sensitivity matrix and the final condition errors, respectively, for the n^{th} reference trajectory. Then the initial condition changes, $\Delta f(t_o)_{n+1}$, $\Delta \Psi(t_o)_{n+1}$, and $\Delta \nu(t_o)_{n+1}$, which are used to compute the $(n + 1)^{th}$ reference trajectory, are, according to Eq. (4-7), as follows:

$$\begin{bmatrix} \Delta f(t_o)_{n+1} \\ \Delta \Psi(t_o)_{n+1} \\ \Delta \nu(t_o)_{n+1} \end{bmatrix} = [G_n]^{-1} \begin{bmatrix} \varepsilon_{p_n} \\ \varepsilon_{e_n} \\ \varepsilon_{\varphi_n} \end{bmatrix} \tag{4-8}$$

These initial condition changes can then be employed to compute the $(n + 1)^{th}$ reference trajectory along with the corresponding final

condition errors $\varepsilon_{p_{n+1}}$, $\varepsilon_{e_{n+1}}$, and $\varepsilon_{\varphi_{n+1}}$. The sensitivity matrix $[G_{n+1}]$ for the $(n+1)^{th}$ reference trajectory is then computed as follows.

$$[G_{n+1}] = [G_n] - \frac{\begin{bmatrix} \varepsilon_{p_{n+1}} \Delta f(t_o)_{n+1} & \varepsilon_{p_{n+1}} \Delta \Psi(t_o)_{n+1} & \varepsilon_{p_{n+1}} \Delta \nu(t_o)_{n+1} \\ \varepsilon_{e_{n+1}} \Delta f(t_o)_{n+1} & \varepsilon_{e_{n+1}} \Delta \Psi(t_o)_{n+1} & \varepsilon_{e_{n+1}} \Delta \nu(t_o)_{n+1} \\ \varepsilon_{\varphi_{n+1}} \Delta f(t_o)_{n+1} & \varepsilon_{\varphi_{n+1}} \Delta \Psi(t_o)_{n+1} & \varepsilon_{\varphi_{n+1}} \Delta \nu(t_o)_{n+1} \end{bmatrix}}{\Delta f(t_o)^2_{n+1} + \Delta \Psi(t_o)^2_{n+1} + \Delta \nu(t_o)^2_{n+1}} \quad (4\text{-}9)$$

4.4 Simultaneous Adjustment of the Initial Values of f, Ψ, and ν and Convergence to the Desired Finite-Thrust Transfer Trajectory

In Section 4.2, starting with the initial-approximate transfer trajectory developed in Chapter 3, we adjusted the initial true anomaly until the magnitude of the argument of pericenter error for the resulting transfer trajectory was reduced below a certain prescribed value. This transfer trajectory found in Section 4.2 is now used as the initial reference trajectory for the convergence method discussed in Section 4.3 . The sensitivity matrix [G] as defined by Eq. (4-6), is computed for this reference trajectory by varying the initial conditions $f(t_o)$, $\Psi(t_o)$, and $\nu(t_o)$ one at a time and then computing the resultant changes in the final conditions $p(t_f)$, $e(t_f)$, and $\varphi(t_f)$. A new reference trajectory is established by changing the initial conditions in accordance with Eq. (4-7). The matrix [G] for this new reference trajectory is computed by means of Eq. (4-9).

The initial conditions are changed in the above manner until the final condition errors, ε_p, ε_e, and ε_φ all have absolute magnitudes which are less than 10^{-4} . This final set of initial conditions, $f(t_o)$, $\Psi(t_o)$, and $\nu(t_o)$, defines the desired minimum-fuel finite-thrust transfer trajectory.

CHAPTER 5

COMPUTATIONAL TECHNIQUES

In the process of determining the minimum-fuel transfer trajectory it is necessary to compute transfer trajectories for a relatively large number of different starting conditions. In order that the computation be efficient, it is essential that both the total number of transfer trajectories computed and the amount of required computation for each transfer trajectory be kept within reasonable bounds. The techniques which make possible the efficient computation of minimum-fuel transfer trajectories are described in this chapter.

5.1 Transformation to the z, A, B Coordinates for Computation of the Coast Phase Trajectory

The basic transfer trajectory consists of two thrusting phases separated by a coasting phase. Computation is performed in separate stages on each of these three phases. A fixed step-size fourth-order Runge-Kutta integration algorithm is employed to compute the two thrusting phases of the transfer trajectory. It proves very wasteful, however, to integrate along the coast trajectory. This section will describe a method for transferring the system variables from the cutoff condition at the end of the first thrusting phase to the relight condition at the beginning of the second thrusting phase without integrating along the coast trajectory.

Computation along the coast trajectory is made possible by means of a canonical transformation suggested by DeVeubeke[1,10] of the system state variables, adjoint variables, and the independent variable t. The independent variable in this transformed set is the polar angle θ and the new set of state variables are z, A, B, w, and t, where:

$$z = \frac{1}{ru_\theta} \tag{5-1}$$

$$A = z\ e\ \cos\varphi \tag{5-2}$$

$$B = z\ e\ \sin\varphi \tag{5-3}$$

Since z is the reciprocal of the angular momentum, z, as well as A and B, will remain constant along the coast trajectory.

The variables z, A, and B have been utilized previously in Chapter 2 to improve integration accuracy for low eccentricity orbits. Using the identities of Eqs. (2-18) - (2-20), and (5-1) in Eq. (2-95) and (2-96) gives the following expressions for A and B:

$$A = u_r \sin\theta + (u_\theta - \frac{1}{ru_\theta})\cos\theta \tag{5-4}$$

$$B = -u_r \cos\theta + (u_\theta - \frac{1}{ru_\theta})\sin\theta \tag{5-5}$$

To develop the canonical transformation from the set (h, u, v, φ, w; t) to the set (t, z, A, B, w; θ), it proves convenient to first transform to the set (r, u_r, u_θ, θ, w; t). The canonical transformation from (h, u, v, φ, w; t) to (r, u_r, u_θ, θ, w; t) is governed by the following equation (Ref. 1).

$$\lambda_h dh + \lambda_u du + \lambda_v dv + \lambda_\varphi d\varphi + \lambda_w dw - H dt = \lambda_r dr + \lambda_{u_r} du_r + \lambda_{u_\theta} du_\theta$$

$$+ \lambda_\theta d\theta + \lambda_w dw - H dt \tag{5-6}$$

Taking the differentials of Eqs. (2-18) - (2-20) and (2-81) gives

$$dh = -\frac{dr}{r^2} \tag{5-7}$$

$$du = \frac{du_r}{r} - \frac{u_r}{r^2} dr \tag{5-8}$$

$$dv = \frac{du_\theta}{r} - \frac{u_\theta}{r^2} dr \tag{5-9}$$

$$d\varphi = d\theta - df \tag{5-10}$$

The differential df can be expressed in terms of r, u_r, u_θ and the corresponding differentials dr, du_r, and du_θ in the following manner. From Eqs. (2-18) - (2-20) and (2-67) - (2-69) it is easily shown that

$$e \sin f = r u_\theta u_r \tag{5-11}$$

and

$$e \cos f = r u_\theta^2 - 1 \tag{5-12}$$

Division of Eq. (5-11) by Eq. (5-12) gives

$$\tan f = \frac{r u_\theta u_r}{r u_\theta^2 - 1} \tag{5-13}$$

or

$$f = \tan^{-1}\left[\frac{r u_\theta u_r}{r u_\theta^2 - 1}\right] \tag{5-14}$$

Taking the differential of Eq. (5-14) and making use of Eqs. (2-18) - (2-20) and (2-67) - (2-69), we have

$$df = \frac{1}{h^6 e^2} \left[- h^4 uv \, dr + hv(v^2 - h^3) du_r - hu(v^2 + h^3) du_\theta \right] \qquad (5\text{-}15)$$

From Eqs. (5-10) and (5-15) it follows that

$$d\varphi = d\theta + \frac{1}{h^6 e^2} \left[h^4 u \, v \, dr - hv(v^2 - h^3) du_r + hu(v^2 + h^3) du_\theta \right] \qquad (5\text{-}16)$$

Substituting Eqs. (5-7) - (5-9) and (5-16) into Eq. (5-6), applying Eqs. (2-18) - (2-21), and equating the coefficients of like differentials, we have

$$\lambda_r = - h^2 \lambda_h - uh\lambda_u - vh\lambda_v + \frac{h^4 uv}{h^6 e^2} \lambda_\varphi \qquad (5\text{-}17)$$

$$\lambda_{u_r} = h\lambda_u - \frac{hv(v^2 - h^3)}{h^6 e^2} \lambda_\varphi \qquad (5\text{-}18)$$

$$\lambda_{u_\theta} = h\lambda_v + \frac{hu(v^2 + h^3)}{h^6 e^2} \lambda_\varphi \qquad (5\text{-}19)$$

$$\lambda_\theta = \lambda_\varphi \qquad (5\text{-}20)$$

The corresponding inverse transformation can be obtained by solving Eq. (5-17) - (5-19) for λ_h, λ_u, and λ_v, to give

$$\lambda_h = - \frac{\lambda_r}{h^2} - \frac{u}{h^2} \lambda_{u_r} - \frac{v}{h^2} \lambda_{u_\theta} + \frac{3h^2 uv}{h^6 e^2} \lambda_\varphi \qquad (5\text{-}21)$$

$$\lambda_u = \frac{\lambda_{u_r}}{h} + \frac{v(v^2 - h^3)}{h^6 e^2} \lambda_\varphi \qquad (5\text{-}22)$$

$$\lambda_v = \frac{\lambda_{u_\theta}}{h} - \frac{u(v^2 + h^3)}{h^6 e^2} \lambda_\varphi \qquad (5\text{-}23)$$

Solving Eqs. (5-1), (5-4), and (5-5) for r, u_r, and u_θ, gives

$$r = \frac{1}{z(z + A\cos\theta + B\sin\theta)} \tag{5-24}$$

$$u_r = A\sin\theta - B\cos\theta \tag{5-25}$$

$$u_\theta = z + A\cos\theta + B\sin\theta \tag{5-26}$$

Let L be the hamiltonian corresponding to the set (t, z, A, B, w; θ). A canonical transformation between the corresponding adjoint variables of the sets (r, u_r, u_θ, θ, w; t) and (t, z, A, B, w; θ) is obtained by substituting the differentials of Eqs. (5-1), (5-4), and (5-5) into the equation

$$\lambda_r dr + \lambda_{u_r} du_r + \lambda_{u_\theta} du_\theta + \lambda_\theta d\theta + \lambda_w dw - Hdt = \lambda_z dz + \lambda_A dA$$

$$+ \lambda_B dB + \lambda_t dt + \lambda_w dw - Ld\theta \tag{5-27}$$

and equating the coefficients of like differentials. This gives, with the aid of Eq. (5-1)

$$\lambda_r = z^2 u_\theta(\lambda_A\cos\theta + \lambda_B\sin\theta - \lambda_z) \tag{5-28}$$

$$\lambda_{u_r} = \lambda_A\sin\theta - \lambda_B\cos\theta \tag{5-29}$$

$$\lambda_{u_\theta} = -\frac{z\lambda_z}{u_\theta} + (1 + \frac{z}{u_\theta})(\lambda_A\cos\theta + \lambda_B\sin\theta) \tag{5-30}$$

$$\lambda_\theta = A\lambda_B - B\lambda_A - L \tag{5-31}$$

$$H = -\lambda_t \tag{5-32}$$

The corresponding inverse relationships can be found by solving Eqs. (5-28) - (5-30) for λ_A, λ_B, and λ_z. This gives

$$\lambda_A = -\frac{\lambda_r}{zu_\theta^2}\cos\theta + \lambda_{u_r}\sin\theta + \lambda_{u_\theta}\cos\theta \tag{5-33}$$

$$\lambda_B = -\frac{\lambda_r}{zu_\theta^2}\sin\theta - \lambda_{u_r}\cos\theta + \lambda_{u_\theta}\sin\theta \tag{5-34}$$

$$\lambda_z = -\frac{1}{zu_\theta}\left(\frac{1}{z} + \frac{1}{u_\theta}\right)\lambda_r + \lambda_{u_\theta} \tag{5-35}$$

Starting from Eq. (2-58) and making use of Eqs. (5-21) - (5-23), (2-67) - (2-69), (2-18) - (2-20), and (5-20), it can be shown that

$$H_o = \lambda_r u_r + \frac{\lambda_\theta u_\theta}{r} + \lambda_{u_r}\left(\frac{u_\theta^2}{r} - \frac{1}{r^2}\right) - \lambda_{u_\theta}\frac{u_r u_\theta}{r} \tag{5-36}$$

Substitution of Eqs. (5-24) - (5-26), and (5-28) - (5-31) into Eq. (5-36) gives

$$H_o = -z(z + A\cos\theta + B\sin\theta)^2 L \tag{5-37}$$

The following expression for the new hamiltonian L results directly by solving Eq. (5-37) for L, and using Eq. (5-32) and (2-57) to eliminate H_o

$$L = \frac{\lambda_t + a\xi w H_1}{z(z + A\cos\theta + B\sin\theta)^2} \tag{5-38}$$

Since L is the new hamiltonian and θ is the independent variable, the adjoint variables λ_z, λ_A, and λ_B must satisfy

$$\frac{d\lambda_z}{d\theta} = -\frac{\partial L}{\partial z} \tag{5-39}$$

$$\frac{d\lambda_A}{d\theta} = -\frac{\partial L}{\partial A} \tag{5-40}$$

$$\frac{d\lambda_B}{d\theta} = -\frac{\partial L}{\partial B} \tag{5-41}$$

For the time open case which we are considering, $\lambda_t = 0$. Therefore, from Eq. (5-38) it is obvious that the new hamiltonian L must remain identically zero over the coast trajectory ($\xi = 0$). In addition to A, B, and z being constant over the coast trajectory, it follows, therefore, from Eqs. (5-39) - (5-41) that λ_z, λ_A, and λ_B will also remain constant on the coast trajectory.

With the aid of Eq. (2-32) we have from Eq. (2-59)

$$H_1 = \sqrt{h^2\Lambda_u^2 + h^2\Lambda_v^2} + \frac{w\lambda_w}{c} \tag{5-42}$$

Elimination of λ_v and λ_u from Eqs. (2-28) and (2-29) by means of Eqs. (5-22) and (5-23) gives

$$\Lambda_v = \frac{\lambda_{u_\theta}}{h} \tag{5-43}$$

$$\Lambda_u = \frac{\lambda_{u_r}}{h} \tag{5-44}$$

Therefore,from Eq. (5-42)

$$H_1 = \sqrt{\lambda_{u_\theta}^2 + \lambda_{u_r}^2} + \frac{w\lambda_w}{c} \tag{5-45}$$

Let

$$\lambda = \sqrt{\lambda_{u_\theta}^2 + \lambda_{u_r}^2} \tag{5-46}$$

Then

$$H_1 = \lambda + \frac{w\lambda_w}{c} \tag{5-47}$$

The partial derivative of H_1 with respect to the polar angle θ along the coast trajectory is, from the last two equations

$$\left.\frac{\partial H_1}{\partial \theta}\right]_{coast} = \frac{1}{\lambda}\left[\frac{\partial \lambda_{u_\theta}}{\partial \theta} + \frac{\partial \lambda_{u_r}}{\partial \theta}\right]_{coast} \tag{5-48}$$

where, from Eqs. (5-29), (5-30), and (5-26)

$$\left.\frac{\partial \lambda_{u_r}}{\partial \theta}\right]_{coast} = \lambda_A \cos\theta + \lambda_B \sin\theta \tag{5-49}$$

$$\left.\frac{\partial \lambda_{u_\theta}}{\partial \theta}\right]_{coast} = \frac{z}{u_\theta^2}\left[\lambda_z - \lambda_A \cos\theta - \lambda_B \sin\theta\right]\frac{\partial u_\theta}{\partial \theta}$$

$$- \left[1 + \frac{z}{u_\theta}\right]\left[\lambda_A \sin\theta - \lambda_B \cos\theta\right] \tag{5-50}$$

$$\left.\frac{\partial u_\theta}{\partial \theta}\right]_{coast} = -A \sin\theta + B \cos\theta \tag{5-51}$$

The equations developed in this section make it possible to transfer from cutoff of the first thrusting phase to relight of the second thrusting phase without resorting to integration along the coast trajectory. At the cutoff and relight conditions

$$\left.H_1\right]_{\substack{\text{cutoff} \\ \text{or} \\ \text{relight}}} = 0 \tag{5-52}$$

while over the remainder of the coast trajectory

$$\left.H_1\right]_{coast} < 0 \tag{5-53}$$

Since H_1 must vanish at relight, a transfer to the relight condition can be easily accomplished by finding that polar angle θ for which H_1 will be zero. Making use of this principle, we formulate the following algorithm for transferring from cutoff to relight along the coast trajectory. Let p, e, φ, and w represent the semi-latus rectum, eccentricity, argument of pericenter, and reciprocal of mass respectively along the coast trajectory. For parameters which change along the coast trajectory, the subscripts "1" and "2" refer to cutoff and relight conditions respectively.

For discussion purposes the transfer trajectory is assumed to be computed in the forward direction. Only slight modifications are necessary for the case of backward trajectory computation.

1. At the cutoff condition defined by p, e, φ, f_1, w, Λ_1, Ψ_1, and ν_1 compute:

   (a) eccentric anomaly, E_1, where

   $$E_1 = 2 \tan^{-1} \left[\sqrt{\frac{1 - e}{1 + e}} \tan \frac{f_1}{2} \right]$$

   (b) h_1, u_1, v_1, from Eqs. (2-67) - (2-69)

   (c) λ_w from Eq. (2-59) and the fact that H_1 at cutoff is zero

   $$\lambda_w = - \frac{c h_1 \Lambda_1}{w}$$

   (d) λ_{h_1} from Eq. (2-83)

   $$\lambda_{h_1} = \frac{1}{h_1} \left[\nu_1 + \frac{3\lambda_\varphi \sin f_1}{e} \right]$$

   (e) Λ_{u_1} and Λ_{v_1} from Eqs. (2-30) and (2-31)

   (f) λ_{v_1} and λ_{u_1} from Eqs. (2-28) and (2-29)

(g) λ_{r_1}, $\lambda_{u_{r_1}}$, and $\lambda_{u_{\theta_1}}$ from Eqs. (5-17) - (5-19)

(h) r_1, and u_{θ_1} from Eqs. (2-18) and (2-20)

(i) θ_1 from Eq. (2-81)

2. Compute the coast trajectory constants z, A, B and the corresponding constant adjoints λ_z, λ_A, and λ_B from Eqs. (5-1) - (5-3) and (5-33) - (5-35) with $\theta = \theta_1$.

3. Estimate the relight polar angle, θ_2, from the minimum-fuel, two-impulse transfer trajectory. With the notation defined in Section 3.2 a good estimate of this angle is, from Eq. (2-81)

$$\theta_2\Big]_{\text{estimated}} = f'(t_2) + \Delta\varphi_I$$

where the estimated true anomaly $f'(t_2)$ is available from Eq. (3-15)

4. At the estimated relight condition compute u_θ, λ_{u_r}, λ_{u_θ}, λ, H_1 and $\frac{\partial H_1}{\partial \theta}$ using Eqs. (5-26), (5-29), (5-30), (5-46), (5-47), and (5-48) - (5-51).

5. If $|H_1|$ is greater than some small number, for example if $|H_1| > 2 \times 10^{-7}$, then a new relight polar angle is determined according to

$$\theta_{2_{new}} = \theta_{2_{old}} - \frac{H_1}{\frac{\partial H_1}{\partial \theta}}$$

and a new H_1 and $\frac{\partial H_1}{\partial \theta}$ are computed as outlined in step four.

6. After several iterations on steps four and five, $|H_1|$ will become very small ($|H_1| < 2 \times 10^{-7}$), thus establishing the correct polar angle θ_2 for the relight condition. A transformation back to the original system variables at the relight condition is accomplished in the following manner. Compute:

   (a) f_2, h_2, u_2, v_2, u_{θ_2}, λ_{r_2}, $\lambda_{u_{r_2}}$, $\lambda_{u_{\theta_2}}$, and λ_{h_2} from Eqs. (2-81), (2-68), (2-69), (2-67), (5-26), (5-28), (5-29), (5-30) and (5-21) respectively.

   (b) Λ_{v_2}, Λ_{u_2}, and Λ_2 from Eqs. (5-43), (5-44) and (2-32) respectively.

   (c) Ψ_2 from Eqs. (2-30) and (2-31)

$$\Psi_2 = \tan^{-1}\left[\frac{\Lambda_{u_2}}{\Lambda_{v_2}}\right]$$

   (d) ν_2 from Eq. (2-83)

   (e) Eccentric anomaly E_2 and relight time t_2, where

$$E_2 = 2\tan^{-1}\left[\sqrt{\frac{1-e}{1+e}}\tan\frac{f_2}{2}\right]$$

$$t_2 = t_1 + \left[\frac{p}{1-e^2}\right]^{1.5}[E_2 - E_1 - e(\sin E_2 - \sin E_1)]$$

The method outlined above has proven to be very accurate and efficient. Because of the excellent estimation of the relight polar angle θ_2 which is available from the two-impulse trajectory, only two or three iterations on the polar angle are ordinarily required to find the desired relight polar angle.

5.2 Computation of the Thrusting Phases

The two thrusting phases are computed using a fixed step-size, fourth-order Runge-Kutta integration algorithm with single-precision arithmetic. With proper choice of step-size, application of the fixed step-size integration routine in favor of an integration routine employing automatic step-size control reduces the required computation time by a factor of about three. The system hamiltonian, which must remain zero over the entire transfer trajectory, provides a convenient measure of integration accuracy. Integration step-size can, therefore, be chosen so that the hamiltonian remains very small. An experimental investigation has led to the following formulas based on the rocket thrust levels for computing the maximum integration step-size on the two thrusting phases

$$\Delta t_{max} = \frac{.01}{a} \qquad (a > 1)$$

$$\Delta t_{max} = .01 + \frac{4}{90}(1 - a) \qquad (.1 < a \leq 1)$$

$$\Delta t_{max} = \frac{.005}{a} \qquad (.05 < a \leq .1)$$

$$\Delta t_{max} = .1 \qquad (a \leq .05)$$

This particular choice of maximum step-size results in the magnitude of the hamiltonian seldom becoming larger than 5×10^{-7}.

The thrusting phases must be terminated at precisely the instant at which the switching function s passes through zero. This is accomplished by allowing the integration to proceed until the switching function reverses sign and becomes negative. The values of s and $\frac{ds}{dt}$ are computed at this time, and the integration routine is given a new step-size

$$\Delta t = -\frac{s}{\frac{ds}{dt}}$$

This process is repeated until the magnitude of s becomes less than 10^{-7}.

5.3 Backward Integration of the System Equations

For certain classes of transfer trajectories the resulting final values of semi-latus rectum, eccentricity, and argument of pericenter become very sensitive to small changes in the problem initial conditions. This is particularly true when the second thrusting interval is very small. For this case it becomes practically impossible to find the set of initial conditions which will allow the switching function to provide proper switching on the second thrusting interval. Very small changes in the initial conditions on the order of 5×10^{-8} result in either too much thrusting time or else no thrusting time for the second thrusting interval. This difficulty is overcome by computing these transfer trajectories in the reverse sense, starting at the desired terminal conditions and integrating backwards in order to meet the desired initial conditions. For the trajectories in this study the normal computation mode is considered to be the forward direction. If, however, the second thrusting interval requires a smaller velocity change magnitude than the first thrusting interval, the trajectory is computed in a backward direction. In other more isolated instances a reversal in the direction of the trajectory computation can also be used to good advantage. Convergence becomes particularly difficult to achieve when the eccentricity of the final orbit is less than .025. For this

case, convergence is again more readily attained through backward computation of the trajectory.

For backward integration the errors in matching the initial parameters of the desired orbit are now used as the criteria for termination of the convergence method. Let $p(t_o)_d$, $e(t_o)_d$, and $\varphi(t_o)_d$ be the desired initial values and $p(t_o)_a$, $e(t_o)_a$, and $\varphi(t_o)_a$ the actual initial values for p, e, and φ respectively. Then, for backward trajectory computation, the errors ε_p, ε_e, ε_φ are defined as follows

$$\varepsilon_p = p(t_o)_d - p(t_o)_a \tag{5-54}$$

$$\varepsilon_e = e(t_o)_d - e(t_o)_a \tag{5-55}$$

$$\varepsilon_\varphi = \varphi(t_o)_d - \varphi(t_o)_a \tag{5-56}$$

with ε_φ expressed in radians.

5.4 Determination of the Initial Value of ν

The initial value of the variable ν is determined so that the desired duration of the second thrusting interval is attained. For most transfer trajectories, the duration of the second thrusting interval is very sensitive to the choice of the initial value of ν. In order to limit computer time, a considerable amount of computational logic is required in establishing the desired initial value of ν. The general behavior of the error in the duration of the second thrusting interval as a function of the initial value of $|\nu|$ is shown in Figure 5-1 for the case of a forward-rearward transfer trajectory. The maximum error in Fig. 5-1 is a consequence of the initial $|\nu|$ being too large. This causes the switching function s to remain negative in the manner of Fig. 3-4D resulting in a

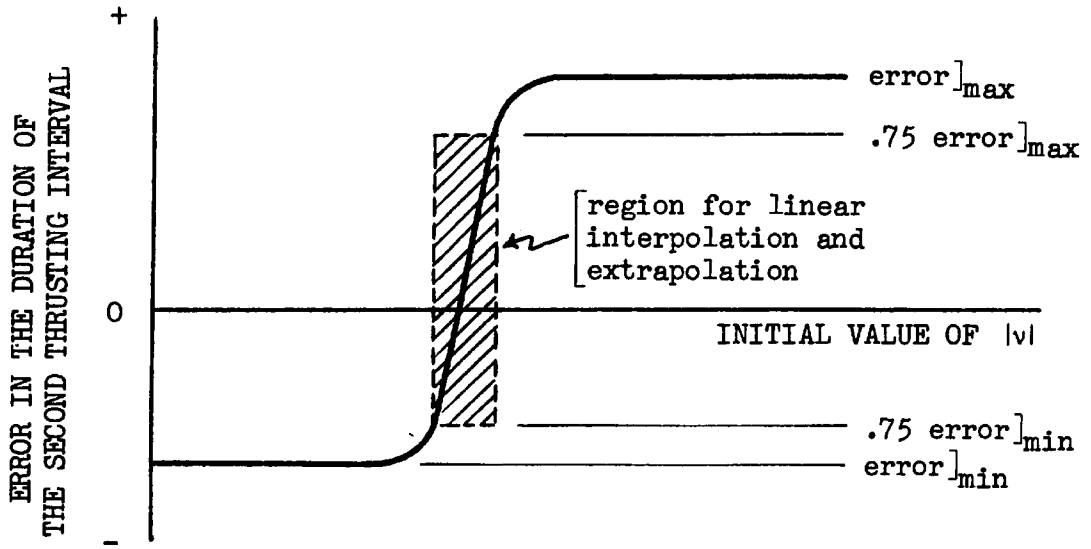


Fig. 5-1: Error in the duration of the second thrusting interval as a function of the initial value of ν (forward-rearward thrusting sequence).

complete absence of the second thrusting interval. On the other hand, the minimum error is caused by the initial value of $|\nu|$ being too small. This results in a failure of the switching function to return to zero on the second thrusting interval and therefore, the thrusting phase is not terminated in the normal manner. This behavior is shown in Fig. 3-4A .

The basic computational problem is to find the initial value of ν which reduces the duration error to a small value without requiring an unreasonable number of trajectory computations. Therefore, in the early stages of the computation, large changes in the initial value of ν are programmed in order to establish quickly the minimum and maximum error bounds. Once these two error bounds have been established, linear interpolation between these bounds is employed, with the restriction that the interpolation always be conducted between values of positive and negative error. This procedure is followed until two initial values of ν can be found for which the errors lie within a region with upper and lower bounds of

.75 x (maximum error) and .75 x (minimum error) respectively. From this point on in the computation, linear interpolation or extrapolation is conducted between the two most recent pairs of initial ν and error values.

The initial value of ν chosen to start the computation is discussed in Section 3.4 . With this starting value, from ten to fifteen trajectory computations are required to find that initial value of ν which will result in a thrust interval duration error of less than 10^{-4}.

After the proper initial value of ν has been determined, the initial true anomaly is changed to reduce the error in the argument of pericenter. With a new true anomaly, the duration of the second thrusting interval will change, and the initial value of ν must be recomputed to compensate for this true anomaly change. Approximately three to four trajectory computations are required to establish the initial value of ν this time because of the improved starting value.

CHAPTER 6

COMPUTER RESULTS AND LIMITATIONS OF THE CONVERGENCE METHOD

In this chapter the results of the digital computation are summarized, and the limitations of the convergence method are discussed.

6.1 Summarization of the Computer Results

In order to define the limits of applicability of the convergence method and to eliminate any serious deficiencies in the actual computer program, a large number of different transfer trajectories are considered. The digital computer results for a number of the trajectories are summarized in Tables 6.1 and 6.2 . The trajectories in Table 6.1 have been computed by forward integration and the trajectories in Table 6.2 by backward integration. The first sixteen entries in Tables 6.1 and 6.2 refer to minimum-fuel transfer trajectories with forward-rearward or rearward-forward thrusting sequences, and the last four entries in each of the tables refer to trajectories with forward-forward or rearward-rearward thrusting sequences. An effective rocket exhaust velocity c of .5 is used for all the transfer trajectories appearing in the tables. If the basic unit of length is taken as the earth's radius, this effective exhaust velocity is equivalent to a specific impulse of about 400 seconds. The subscripts "d" and "a" appearing in the tables refer to desired and actual values respectively.

		INITIAL CONDITIONS					FINAL CONDITIONS						CHARACTERISTIC VELOCITY	
Run	a	$e(t_o)$	$p(t_o)$	$f(t_o)$ (deg)	$\Psi(t_o)$ (deg)	$\nu(t_o)$	$e(t_f)_d$	$e(t_f)_a$	$p(t_f)_d$	$p(t_f)_a$	$\Delta\phi_d$ (deg)	$\Delta\phi_a$ (deg)	Finite Thrust	Impulse
1	.1	.6	1.5	168.5454	2.176428	- .125025	.2	.199999	1.0	1.000008	90.	89.9977	.2903170	.2740106
2	1.0	.1	1.5	162.5289	.038986	- .702203	.6	.599986	1.0	1.000030	160.	160.0029	.2775387	.2775262
3	.2	.1	1.5	157.1693	- 2.228114	- .706152	.6	.599976	1.0	1.000058	160.	159.9999	.2778188	.2775262
4	.4	.6	1.5	177.3557	.593005	- .133366	.1	.099997	1.0	1.000004	150.	149.9987	.2981130	.2967320
5	.4	.6	1.5	174.1632	1.428030	- .133814	.1	.099998	1.0	1.000007	90.	89.9961	.2688114	.2673853
6	.2	.05	1.5	89.4731	- 2.057451	- .848176	.6	.599976	1.0	1.000061	90.	89.9998	.2449814	.2447462
7	.4	10^{-5}	1.25	56.9605	- 1.415735	-1.097715	.4	.399955	1.0	1.000075	60.	59.9999	.1789320	.1788747
8	.4	10^{-8}	1.25	56.9603	- 1.415751	-1.097714	.4	.400036	1.0	.999940	60.	59.9999	.1789746	.1788747
9	.4	.7	1.0	171.3739	3.021204	- .165891	.2	.200045	1.5	1.499996	90.	90.0018	.2812813	.2812177
10	.4	.6	1.5	170.3668	2.652045	- .155213	.6	.599982	1.0	1.000046	150.	149.9998	.4158884	.4157940
11	.4	.8	1.0	174.0371	2.956251	- .074908	.2	.199989	1.5	1.499981	90.	90.0040	.3049516	.3048220
12	.4	.7	1.0	175.8765	1.171365	- .161451	.2	.199979	1.5	1.500043	150.	149.9961	.3310217	.3309618
13	.4	.7	1.5	176.3898	1.127843	- .077609	.2	.200020	1.0	.999974	150.	150.0016	.3635435	.3622068
14	.4	.2	1.5	44.8033	1.472143	-1.090884	.6	.600001	1.0	1.000004	30.	29.9976	.1836232	.1835938
15	.4	.2	1.5	217.9759	- 4.155855	- .615945	.6	.600035	1.0	.999909	-120.	-120.0019	.2857107	.2856211
16	.4	.6	1.5	188.2096	- 2.298954	- .138888	.2	.199986	1.0	1.000018	-120.	-119.9966	.3071478	.2960499
17	.4	.2	1.25	218.4235	175.9881	1.046928	.2	.200096	1.0	1.000073	30.	30.0039	.1005244	.1005622
18	.4	.2	2.0	178.2190	179.9811	.450176	.2	.199934	1.0	.999943	0.	- .000167	.2602451	.2601412
19	.4	.4	2.0	179.1514	180.0153	.256722	.4	.399924	1.0	.999942	0.	.000001	.2175787	.2173890
20	.4	.05	2.0	177.3094	179.9003	.626378	.05	.049902	1.0	.999905	0.	.000186	.2803388	.2802910

Table 6.1: Summary of minimum-fuel transfer trajectories - forward computation.

		FINAL CONDITIONS					INITIAL CONDITIONS						CHARACTERISTIC VELOCITY	
Run	a	$e(t_f)$	$p(t_f)$	$f(t_f)$ (deg)	$\Psi(t_f)$ (deg)	$v(t_f)$	$e(t_o)_d$	$e(t_o)_a$	$p(t_o)_d$	$p(t_o)_a$	$\Delta\phi_d$ (deg)	$\Delta\phi_a$ (deg)	Finite Thrust	Impulse
21	1.	.2	1.5	206.4490	178.2682	.560773	.2	.200004	1.25	1.249959	120.	120.0056	.1424656	.1424484
22	.05	.2	1.5	211.4922	180.9997	.560189	.2	.199994	1.25	1.250010	120.	119.9989	.1462795	.1424484
23	.4	.6	1.0	182.0061	179.7901	.253952	.05	.050004	1.0	1.000007	30.	29.9954	.2567177	.2558796
24	1.	.2	1.5	233.3768	176.7988	.639880	.2	.199999	1.25	1.249999	60.	59.9979	.0844863	.0844839
25	.025	.2	1.5	234.6045	178.9339	.606502	.2	.200000	1.25	1.250002	60.	59.9993	.0928740	.0844839
26	.4	.7	1.0	188.3546	176.9096	.165336	.2	.199977	1.5	1.500019	90.	89.9965	.2813049	.2812176
27	.4	.7	1.0	183.7815	178.7515	.161113	.2	.200032	1.5	1.499934	150.	150.0056	.3310844	.3309616
28	.4	.6	1.0	184.9792	178.9585	.265466	.2	.200047	1.0	.999938	150.	150.0054	.3421063	.3413652
29	.4	.5	1.25	191.1983	177.8446	.317090	.5	.500002	1.25	1.249992	150.	150.0004	.3594310	.3593506
30	.4	.2	1.25	195.0845	179.9245	.725268	.2	.199993	1.25	1.250012	150.	149.9994	.1634735	.1634314
31	.4	.6	1.50	182.5922	179.3777	.133503	.1	.099969	1.0	1.000035	150.	149.9955	.2981991	.2967313
32	.4	.8	1.0	185.8476	177.0205	.074770	.2	.200020	1.5	1.499975	90.	90.0043	.3050198	.3048221
33	.4	.7	1.5	183.5382	178.8466	.077668	.2	.199993	1.0	1.000007	150.	150.0032	.3637593	.3622072
34	.4	.2	1.5	186.0433	179.6074	.524812	.03	.030001	1.25	1.249997	120.	120.0038	.0920852	.0920252
35	.4	.6	1.0	181.2356	180.2586	.286676	.6	.599987	1.0	1.000032	180.	179.9598	.4650439	.4648906
36	.4	.2	1.5	141.0179	183.7374	.618743	.6	.600021	1.0	.999947	-120.	-120.0005	.285818	.2857213
37	.4	.4	2.0	180.6265	-.011710	-.256768	.4	.400014	1.0	.999999	0.	+ .000043	.2176104	.2174009
38	.4	.2	2.0	181.2416	.012173	-.450202	.2	.200065	1.0	1.000054	0.	.000067	.2603650	.2601414
39	.4	.4	1.25	198.9113	-5.02905	-.604370	.2	.200010	1.0	1.000003	- 30.	-30.0020	.1250836	.1250455
40	.4	.05	2.0	181.8235	.066822	-.626398	.05	.050027	1.0	1.000026	0.	.000004	.2807767	.2805122

Table 6.2: Summary of minimum-fuel transfer trajectories - backward computation.

The characteristic velocities for the finite-thrust and two-impulse transfers are given in the last two columns of Tables 6.1 and 6.2. Characteristic velocity is defined as follows

$$[\text{Characteristic Velocity}]_{\substack{\text{two-}\\\text{impulse}}} = |\text{Impulse 1}| + |\text{Impulse 2}|$$

$$[\text{Characteristic Velocity}]_{\substack{\text{finite-}\\\text{thrust}}} = c \ln[w(t_f)] \qquad (6\text{-}1)$$

Because of the gradient method employed to compute the two-impulse transfer, the resulting two-impulse characteristic velocities appearing in the tables are only accurate to about the first four digits. As a consequence, in run seventeen the finite-thrust characteristic velocity appears lower than the two-impulse characteristic velocity. This is in contradiction to the theory which indicates that the characteristic velocity should decrease as the rocket thrust level increases. The effect of the rocket thrust magnitude on the characteristic velocity is illustrated by the pairs of runs 2 and 3, 21 and 22, and 24 and 25.

A transfer trajectory is defined to be convergent if the absolute magnitudes of the errors ε_p, ε_e, and ε_φ as defined in Eqs. (4-1) - (4-3) for forward computations or in Eqs. (5-54) - (5-56) for rearward computations are each less than 10^{-4} . Computation is terminated upon the attainment of convergence. All the trajectories summarized in Tables 6.1 and 6.2 are convergent by the above definition.

The individual errors ε_p, ε_e, and ε_φ for run 34 of Table 6.2 are given in Table 6.3 as a function of the iteration number. A relatively large number of iterations are required to compute the transfer trajectory of Table 6.3 . This particular transfer trajectory is

ITERATION	ERROR		
	$\varepsilon_p \times 10^3$	$\varepsilon_e \times 10^3$	$\varepsilon_\varphi \times 10^3$
0	1.17368	-22.87911	329.3026
1	1.12581	-25.11474	349.50271
2	- .16358	1.88112	- 45.97852
3	.00699	- 1.03564	29.97921
4	- .03710	.04818	1.08311
5	- .15672	.09923	2.11155
6	.01580	.00327	- .88245
7	- .04964	.02363	1.04833
8	- .01092	.00512	.23766
9	- .00547	.00264	.12189
10	- .02010	.00963	.41148
11	- .01302	.00654	.29120
12	- .00454	.00227	.09516

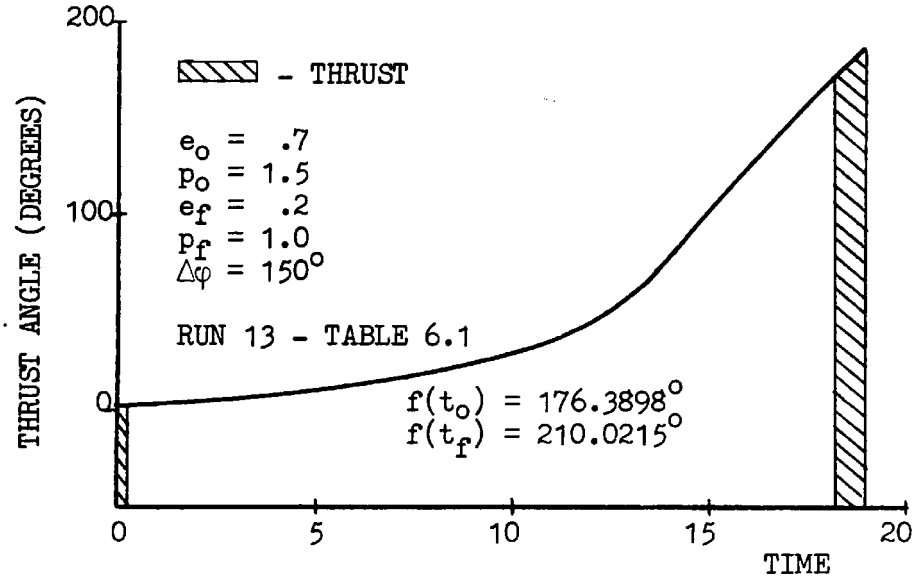
$e_o = .03$, $p_o = 1.25$, $e_f = .2$, $p_f = 1.5$, $\varphi_f - \varphi_o = 120^o$

Table 6.3: Typical error behavior during convergence to a desired transfer trajectory [ε_p, ε_e, and ε_φ, defined in Eqs. (5-59) - (5-61)].

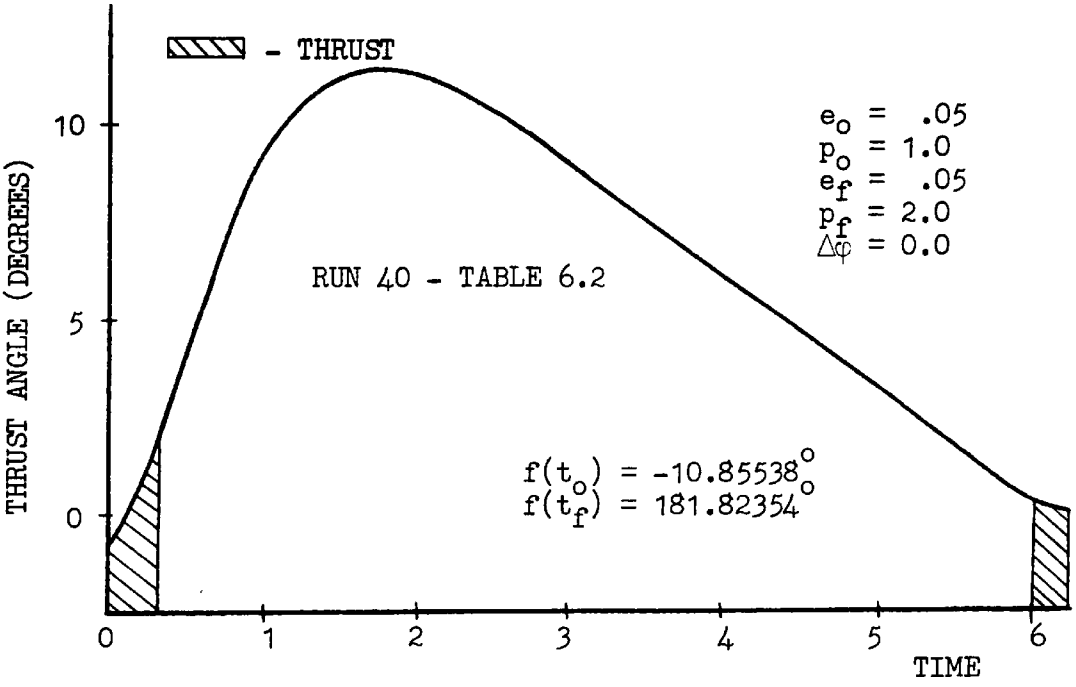
computed in a backward sense, starting at the desired final conditions and attempting to match the desired initial conditions. The low initial eccentricity, $e_o = .03$, results in rapid changes in the argument of pericenter as the initial orbit is approached. This causes the argument of pericenter error to be relatively large, thus slowing down the convergence. The errors corresponding to iteration zero are the errors for the initial-approximate transfer trajectory. On iterations 1 - 4 the initial true anomaly $f(t_o)$ is adjusted to reduce the argument of pericenter error ε_φ. Iteration one obtains the initial sensitivity relating changes in $f(t_o)$ to changes in ε_φ. The true anomaly $f(t_o)$ is increased arbitrarily by .01, and $\Psi(t_o)$ and $\nu(t_o)$ are recomputed to attain the required thrusting interval durations. At the end of iteration four, $|\varepsilon_\varphi|$ is less than the preestablished maximum value of 2×10^{-3}, and the simultaneous adjustment of $f(t_o)$, $\Psi(t_o)$, and $\nu(t_o)$ by means of the sensitivity matrix [G] is initiated. Computation is terminated at the end of iteration 12 with $|\varepsilon_p|$, $|\varepsilon_e|$, and $|\varepsilon_\varphi|$ all less than 10^{-4}. Sixteen seconds of IBM 7094 computer time are required for this particular trajectory computation.

6.2 Thrust Angle Behavior Along the Minimum-Fuel Transfer Trajectory

The thrust angle behavior for runs 13 and 40 of Tables 6.1 and 6.2 are shown in Fig. 6-1 . Also included are the initial and final true anomalies. The forward-rearward transfer, Fig. 6-1A, is typified by the thrust angle changing by approximately 180° along the transfer trajectory, while in the case of the forward-forward thrusting sequence, Fig. 6-1B, the thrust angle experiences only a very small net change over the entire transfer trajectory. Fig. 6-1B shows the thrust angle



A. FORWARD-REARWARD THRUSTING



B. FORWARD-FORWARD THRUSTING

Fig. 6-1: Typical thrust angle time histories along the minimum-fuel transfer trajectory.

behavior for the transfer between almost circular orbits ($e_o = e_f = .05$). The thrust angle behavior and the initial and final true anomalies bear close resemblances to the minimum-fuel two-impulse transfer between circular orbits, better known as the Hohmann transfer.

6.3 Computer Time Requirements

All the trajectories considered in this study were computed on an IBM 7094 digital computer. Computer time requirements ranged from a low of ten seconds to a high of about fifty seconds, with the higher computer times being associated with transfer trajectories which are very sensitive to small changes in the initial conditions. The above times are for the finite-thrust transfer trajectory computation only. Approximately twenty additional seconds of computer time is required for the computation of the corresponding minimum-fuel two-impulse transfer trajectory.

The computation time is dependent upon the various tolerances which must be met at each stage of the convergence program before the computation is allowed to proceed to the next stage. For example, the first and second thrusting intervals must be within 10^{-6} time units and 5×10^{-4} time units respectively of the desired values when computing the initial-approximate transfer trajectory, and the final argument of pericenter must be within 2×10^{-3} radians of the desired value when making the adjustment of $f(t_o)$. The above tolerances were established from computational experience with a large number of transfer trajectories. No attempt was made to tailor these tolerances to each specific transfer trajectory although it is conceivable that some computer time could be saved by doing so. A detailed computer time breakdown for six of the trajectories summarized

RUN ‡	COMPUTATION TIME (SECONDS)			
	Initial-Approximate Transfer Trajectory	Adjustment of $f(t_o)$ to reduce ε_φ	Convergence to final trajectory with the sensitivity Matrix [G]	Total
13	4	5 (2)*	2 (5)*	11
20	8	10 (2)*	3 (11)*	21
22	7	6 (2)*	2 (3)*	15
32	9	14 (2)*	4 (11)*	27
34	4	9 (4)*	3 (9)*	16
40	7	36 (5)*	2 (5)*	45

*Integers enclosed by parenthesis indicate the number of iterations required.

‡Run numbers are consistent with Tables 6.1 and 6.2 .

Table 6.4: Summary of IBM-7094 computer time requirements for typical transfer trajectories.

in Tables 6.1 and 6.2 is shown in Table 6.4. As is obvious from the table, most of the computer time is consumed in establishing the initial-approximate transfer trajectory and adjusting $f(t_o)$ to reduce ε_φ . It has been found from experience that not much can be gained by relaxing the tolerances in the thrust interval determinations since larger variations in the first thrusting interval make it difficult to set the second thrusting interval, and larger variations in both the first and second thrusting intervals result in convergence difficulties when adjusting $f(t_o)$. In certain isolated instances it is possible that some computation time could be saved by relaxing the required tolerance on ε_φ when performing the adjustment of $f(t_o)$.

This would place more of the convergence responsibility on the matrix [G] method. The danger here is that the trajectory resulting from the $f(t_o)$ adjustment will not be close enough to the desired trajectory to make it possible for the matrix [G] method to converge. Some of these convergence difficulties could conceivably be overcome by reducing the step-size when employing the matrix [G] convergence method.

6.4 Convergence Limitations Imposed by Orbital Eccentricities and Rocket Thrust Levels

The regions of applicability of the convergence method with respect to orbital eccentricities and rocket thrust levels cannot be precisely defined because of the many possible combinations of initial and final orbits. From computational experience, however, it is possible to define some approximate boundaries inside of which convergence is usually attainable. Minimum-fuel transfer trajectories have been successfully computed over the following ranges of initial and final eccentricities e and thrust to initial weight ratios a .

$$10^{-8} \leq e \leq .8$$

$$.025 \leq a \leq 1.0$$

Integration of the variables z, A, and B, defined in Chapter 2, in place of e, p, and φ makes convergence possible for very low eccentricities. In those instances, however, when the trajectory is being computed in the direction of decreasing eccentricity, convergence becomes difficult to achieve for eccentricities less than approximately .025 . This difficulty is caused by the very rapid change in argument

of pericenter at low eccentricities. In certain cases convergence can be achieved by reversing the computation direction.

Convergence difficulties with orbital eccentricities in excess of .8 are due primarily to the large velocity changes and the longer transfer times required to effect transfers of this type. Consequently, the thrusting and coasting intervals become longer, and the switching function becomes more sensitive to small changes in the initial conditions.

The convergence method fails for thrust-to-weight ratios less than about .025 primarily because the initial-approximate transfer trajectory derived from the two-impulse transfer is not sufficiently close to the desired final transfer trajectory to serve as a reasonable starting trajectory. As the thrust levels are reduced, larger characteristic velocities are required to accomplish the required transfer. However, the thrusting intervals for the initial-approximate transfer trajectory remain fixed since they are computed in such a manner that the characteristic velocities on each thrusting interval are equal to the characteristic velocities on the corresponding thrust intervals of the equivalent two-impulse transfer. Therefore, as the thrust-to-weight ratio is reduced, the fixed characteristic velocity of the initial-approximate transfer trajectory becomes too small to allow the initial-approximate transfer trajectory to be sufficiently close to the desired final transfer trajectory. At this point the method proposed in this study to find the minimum-fuel transfer trajectory fails to converge.

The convergence method fails at high thrust-to-weight ratios primarily because of the shortened thrusting interval durations. In general the initial value of Ψ is used to set the duration of the

first thrusting interval, and the initial value of ν establishes the duration of the second thrusting interval. As these thrusting intervals become smaller, the duration of the first thrusting interval can still be set accurately, but the duration of the second thrusting interval becomes extremely sensitive to the choice of initial ν. Very small changes in $\nu(t_o)$ produce relatively large percentage changes in the duration of the second thrusting interval. This makes convergence by means of the sensitivity matrix [G] more difficult to attain. In addition, more preliminary trajectory computations are required to find a value of $\nu(t_o)$ which will yield the desired duration for the second thrusting interval.

Convergence is more difficult to attain for trajectories with forward-forward or rearward-rearward thrusting sequences. This is primarily due to the delicate balance which must be maintained between the terms $\frac{h - s}{\Lambda} \rho \sin(\Psi - \beta)$ and hu of Eq. (3-47) in order to obtain the desired switching function characteristics. Both of these terms remain very small and are of opposite sign over most of the transfer trajectory. This results in very small changes in the switching function over the entire transfer trajectory. For example, the switching function for run thirty-nine of Table 6.2 has a maximum amplitude of about 7×10^{-8} on the first thrusting interval. The timing and duration of the second thrusting interval depends primarily upon the integrated sum of the terms hu and $\frac{h - s}{\Lambda} \rho \sin(\Psi - \beta)$ along the coast trajectory. Changes in $\nu(t_o)$ affect primarily the second of the above two terms. Since both of these terms are relatively small, a small change in $\frac{h - s}{\Lambda} \rho \sin(\Psi - \beta)$ when integrated over the coast trajectory can have a large relative effect upon the switching function. This causes the timing and duration of the second thrusting interval to be very sensitive to changes in $\nu(t_o)$.

CHAPTER 7

CONCLUSIONS AND RECOMMENDATIONS

In this chapter the important results of the study are summarized, and recommendations for further study are made.

7.1 Summary of the Results

An efficient method for computing time-open minimum-fuel finite-thrust transfer trajectories between two given coplanar elliptic orbits has been developed. Each orbit is defined by an eccentricity, semi-latus rectum and argument of pericenter. The minimum-fuel transfer trajectory is assumed to consist of two thrusting phases separated by a coasting phase.

Computation of the minimum-fuel transfer trajectory is accomplished by a direct integration of the rocket equations of motion and the associated adjoint equations which are developed from variational calculus principles. This direct approach is made possible by a unique transformation of the adjoint equations into a set of equations which provide a much better understanding of the general behavior of minimum-fuel trajectories. The discovery of a new variable and the use of this variable in conjunction with the thrust angle to obtain precise control of the thrusting interval durations has been a very important factor in the success of the method.

A large number of different transfer trajectories have been computed using a 7094 digital computer with predominantly

single-precision arithmetic. The transfer trajectories are grouped into four main classes according to the direction of the thrust vector on each of the two thrusting intervals. The computation of forward-rearward and rearward-forward transfer trajectories require approximately ten to twenty-five seconds of computer time while forward-forward and rearward-rearward transfer trajectories require approximately twenty to fifty seconds of computer time. Convergence has been demonstrated over the following ranges of eccentricities e and rocket thrust-to-weight ratios a

$$10^{-8} \leq e \leq .8$$

$$.025 \leq a \leq 1.0$$

7.2 Recommendations for Further Study

Investigations should be made into extending the limits of applicability of the method to eccentricities beyond .8 and to rocket thrust-to-weight ratios beyond 1. With the present program, convergence beyond these limits becomes difficult to achieve mainly because of the increased sensitivity of the switching function to small changes in the initial conditions. Therefore, it is believed that the convergence method could easily be extended to higher eccentricities and higher thrust-to-weight ratios by the use of double-precision arithmetic throughout the entire computation.

Transfer trajectories with forward-forward or rearward-rearward thrusting sequences are very sensitive to small changes in the initial conditions because of the low switching function magnitudes associated with trajectories of this type. The use of double-precision arithmetic should also alleviate this difficulty and make possible the

computation of a larger class of transfer trajectories with forward-forward and rearward-rearward thrusting sequences.

As discussed in Section 6.4, the basic convergence method proposed in this study fails at low rocket thrust-to-weight ratios because the thrusting intervals, which are fixed by the two-impulse transfer, become inadequate to generate an initial-approximate trajectory which is sufficiently close to the desired transfer trajectory. In order to extend convergence to lower thrust-to-weight ratios, the thrusting interval durations for the initial-approximate transfer trajectory should be increased as some function of the thrust-to-weight ratio. It is believed that the results published by Robbins[6] could be used directly to compute improved thrusting interval durations and thereby extend the convergence to lower thrust-to-weight ratios.

A very challenging undertaking would be the extension of this study to include the case of non-coplanar transfers. The two-impulse solution is applicable to non-coplanar transfers, but a major reworking of the finite-thrust equations would be necessary. The inclusion of an additional state variable and adjoint variable may pose very serious difficulties in solving the resulting two-point boundary value problem.

APPENDIX A

DIGITAL COMPUTER PROGRAM FOR THE COMPUTATION OF THE MINIMUM-FUEL TWO-IMPULSE TRANSFER TRAJECTORY

In this appendix the Fortran IV digital computer program for computing minimum-fuel two-impulse transfer trajectories is presented. The inputs to the program are:

EO = eccentricity of the initial orbit

PO = semi-latus rectum of the initial orbit

EF = eccentricity of the final orbit

PF = semi-latus rectum of the final orbit

WMBR = change in the argument of pericenter from the initial to final orbit

The computation is initiated by selecting ten equally spaced points, with respect to true anomaly, on both the initial and final orbits. A method developed by Lee[4] is then employed to compute the semi-latus rectums of the minimum-fuel transfer trajectories between each of the one hundred possible combinations of initial and final points. The trajectories corresponding to the ten lowest total impulses are saved, and further minimization with respect to the semi-latus rectum and the initial and final true anomalies is accomplished on these ten trajectories by means of a Fletcher-Powell conjugate-gradient algorithm[8,11]. In this computation use is made of the gradient developed by McCue and Bender[3].

The output of the program consists of the following information concerning the minimum-fuel two-impulse transfer trajectory

ET = eccentricity of the transfer trajectory

PT = semi-latus rectum of the transfer trajectory

PHI1R = initial true anomaly

PHI2R = final true anomaly + WMBR

DLV1 = magnitude of the first impulse

DLV2 = magnitude of the second impulse

DN1R = argument of pericenter of transfer trajectory

PSIA1 = thrust orientation angle on the first impulse

PSIA2 = thrust orientation angle on the second impulse

Following are brief discriptions of the main program and the subroutines utilized in this computation. The complete Fortran IV program is also included

IPMAIN - The main program for computing the minimum-fuel two-impulse transfer trajectory.

IMPLSE - The subroutine which computes the minimum-fuel two-impulse transfer trajectory between two fixed points in accordance with Lee[4].

FUNCT - The subroutine which computes the gradient of the total impulse in accordance with McCue and Bender[3].

GRADNT - The subroutine which supplies the logic for the Fletcher-Powell[8] conjugate-gradient convergence method.

SMIN - The subroutine which provides for a minimization of the impulse function along a direction derived from the Fletcher-Powell[8] method.

CUBIC - The subroutine which performs cubic interpolation to assist subroutine SMIN.

SMULR - The subroutine which solves the eighth-order polynomial to assist subroutine IMPLSE.

```
C                              IPMAIN
C
C
C       THE MAIN PROGRAM FOR COMPUTING ONLY THE TWO-IMPULSE
C       MINIMUM FUEL TRANSFER TRAJECTORY
C
C
C       PRIMARY INPUTS
C            E0 = ECCENTRICITY OF THE INITIAL ORBIT
C            P0 = SEMI-LATUS RECTUM OF THE INITIAL ORBIT
C            EF = ECCENTRICITY OF THE FINAL ORBIT
C            PF = SEMI-LATUS RECTUM OF THE FINAL ORBIT
C            WMBR = CHANGE IN THE ARGUMENT OF PERICENTER FROM THE FIRST
C                   ORBIT TO THE SECOND ORBIT
C            MAXSAV = THE NUMBER OF FIXED POINT TRAJECTORIES WHICH
C                     ARE USED AS STARTING POINTS FOR THE
C                     FLETCHER-POWELL METHOD
C            ITMAX = MAXIMUM ALLOWABLE NUMBER OF FLETCHER-POWELL
C                    ITERATIONS
C            GRADMN = MINIMUM CUTOFF VALUE OF THE GRADIENT IN THE
C                     FLETCHER-POWELL METHOD (GRADMN = .0001)
C
C
$IBFTC IPMAIN  XR7,M94,DECK
       COMMON/CMN5/S(10,10),MAXSAV
       COMMON/CMN6/GUNA,TUBX,TUBY,TUBZ,GUNBX,GUNBY,GUNBZ,GUNB,TRANS,
      1        EAX,EAY,EAZ,EBX,EBY,EBZ,SLRA,SLRB,WMA,WMB,DLTS,SO(3),
      2 AINC,EA,EB,WAX,WAY,WAZ,WBX,WBY,WBZ ,PIE,SMU,ITMAX,GRADMN,
      3SMLVX,SMLVY,SMLVZ,ZLITT,TUAX,TUAY,TUAZ,TRAMG,GUNAX,GUNAY,GUNAZ
      4,PULSE,TSLR,TAPH,TBPH,TRSTAN,TRSTB,ETRAN,PERARG,TAPHDG,TBPHDG
       NAMELIST/IMPULZ/MAXSAV,ITMAX,GRADMN
       READ(5,IMPULZ)
   10  READ(5,1050)NR
       READ(5,1000)E0,P0,EF,PF,WMBR
   20  WRITE(6,1011)NR
       WRITE(6,1003)
       WRITE(6,1001)E0,P0,EF,PF,WMBR
       WRITE(6,1005)
       WRITE(6,1014)MAXSAV,ITMAX,GRADMN
       SLRA=P0
       SLRB=PF
       EA=E0
       EB=EF
       WMB=WMBR
       CALL IMPLSE
       DLVTOT=S(1,1)
       DLV1=S(2,1)
       DLV2=S(3,1)
       PT=S(4,1)
       ET=S(5,1)
       PHI1R=S(6,1)
       PHI2R=S(7,1)
       DN1R=S(8,1)
       PSIA1=S(9,1)
       PSIA2=S(10,1)
       WRITE(6,1010)
       WRITE(6,1012)(S(I,1),I=1,10)
```

```
      PUNCH 1050, NR
      PUNCH 1000, E0,P0,ET,PT,EF,PF
      PUNCH 1000, PHI1R,PHI2R,DLV1,DLV2
      PUNCH 1000, DN1R,WMBR
      PUNCH 1000, PSIA1,PSIA2
      GO TO 10
 1000 FORMAT(6F12.7)
 1001 FORMAT(1X,9F14.8)
 1002 FORMAT(1X,9I14)
 1003 FORMAT(1H0,8X,2HE0,12X,2HP0,12X,2HEF,12X,2HPF,11X,4HWMBR)
 1005 FORMAT(1H0,6X,6HMAXSAV,9X,5HITMAX,8X,6HGRADMN)
 1009 FORMAT(1H0,6X,6HMAXSAV,8X,5HITMAX,9X,6HICNTMX)
 1010 FORMAT(5X,6HDLVTOT,9X,4HDLV1,9X,4HDLV2,11X,2HPT,11X,2HET,8X,
     1 5HPHI1R,8X,5HPHI2R,9X,4HDN1R,8X,5HPSIA1,8X,5HPSIA2)
 1011 FORMAT(1H1,13HRUN NUMBER B-I3)
 1012 FORMAT(10F13.8)
 1014 FORMAT(1X,2I14,F14.8)
 1019 FORMAT(1H0,4X,7HIWRT(1),5X,7HIWRT(2),5X,7HIWRT(3),5X,7HIWRT(4))
 1050 FORMAT(6I12)
      END
$IMPULZ  GRADMN=.0001,   MAXSAV=10,   ITMAX=10  $
```

```
C                          SUBROUTINE IMPLSE
C
C
C       THE SUBROUTINE WHICH COMPUTES THE MINIMUM FUEL TWO-IMPULSE
C       TRANSFER TRAJECTORY BETWEEN TWO FIXED POINTS IN
C       ACCORDANCE WITH LEE (REF. 4)
C
C
$IBFTC IMPS     XR7,M94,DECK
       SUBROUTINE IMPLSE
       COMMON/CMN5/S(10,10),MAXSAV
       COMMON/CMN6/GUNA,TUBX,TUBY,TUBZ,GUNBX,GUNBY,GUNBZ,GUNB,TRANS,
      1        EAX,EAY,EAZ,EBX,EBY,EBZ,SLRA,SLRB,WMA,WMB,DLTS,SO(3),
      2 AINC,EA,EB,WAX,WAY,WAZ,WBX,WBY,WBZ ,PIE,SMU,ITMAX,GRADMN,
      3SMLVX,SMLVY,SMLVZ,ZLITT,TUAX,TUAY,TUAZ,TRAMG,GUNAX,GUNAY,GUNAZ
      4,PULSE,TSLR,TAPH,TBPH,TRSTAN,TRSTB,ETRAN,PERARG,TAPHDG,TBPHDG
       DIMENSION A(9),B(8),C(8),PAR(5,300),TEMP(4),ANDEP(12),ANAR(13)
       AINC=0.
       WMA=0.
       WRITE(6,1001)
       WRITE(6,1002)AINC,WMA,GRADMN,(SO(I),I=1,3)
       WRITE(6,1003)
       WRITE(6,1004)MAXSAV,ITMAX
       WRITE(6,1005)
       WRITE(6,1002)SLRA,SLRB,EA,EB,WMB
        PIE=3.1415926/2.
        AINC=AINC*PIE/90.
       SMU=1.
       NG=0
        WMA=WMA*PIE/90.
        WMB=WMB*PIE/90.
        EAX=EA*COS(WMA)
        EAY=EA*SIN(WMA)
        EAY=EAY*COS(AINC)
        EAZ=EA*SIN(WMA)*SIN(AINC)
        EBX=EB*COS(WMB)
        EBY=EB*SIN(WMB)
        EBZ=0.
       WAX=0.0
        WAY=-SIN(AINC)
        WAZ=COS(AINC)
        WBX=0.
         WBY=0.
        WBZ=1.0
       DO 660 K=1,10
        ANDEP(K)=36.*FLOAT(K)-26.
       DO 660 J=1,10
        ANAR(J)=36.*FLOAT(J)
        BPH=ANDEP(K)
        APH=ANAR(J)
        AXLE=ABS(APH-BPH)
        AXEL=ABS(AXLE-180.)
        IF(AXEL-1.)271,271,272
271     APH=APH+5.
272     CONTINUE
        IF(AXLE-.1)273,273,274
273     APH=APH+1.
```

```
274     CONTINUE
 259    SAP=APH-BPH
        APH=APH*PIE/90.
        BPH=BPH*PIE/90.
        ANGA=APH-WMA
        ANGB=BPH-WMB
        UAX=COS(APH)
        UAY=SIN(APH)*COS(AINC)
        UAZ=SIN(APH)*SIN(AINC)
        UBX=COS(BPH)
        UBY=SIN(BPH)
        UBZ=0.
        RAMG=SLRA/(1.+EA*COS(ANGA))
        RBMG=SLRB/(1.+EB*COS(ANGB))
        RAX=UAX*RAMG
        RAY=UAY*RAMG
        RAZ=RAMG*UAZ
        RBX=UBX*RBMG
        RBY=UBY*RBMG
        RBZ=UBZ*RBMG
 339   RDIFF=(RBX-RAX)**2+(RBY-RAY)**2+(RBZ-RAZ)**2
       XCR=RAY*RBZ-RAZ*RBY
       YCR=-RAX*RBZ+RAZ*RBX
       ZCR=RAX*RBY-RAY*RBX
        XCR=XCR**2
        YCR=YCR**2
        ZCR=ZCR**2
        CRPD=XCR+YCR+ZCR
        ZA=SMU*RDIFF/CRPD
       SUMX=EAX+UAX
       SUMY=EAY+UAY
       SUMZ=EAZ+UAZ
       WCSX=WAY*SUMZ-WAZ*SUMY
       WCSY=-WAX*SUMZ+WAZ*SUMX
       WCSZ=WAX*SUMY-WAY*SUMX
       BX=(RBX-RAX)*WCSX
       BY=(RBY-RAY)*WCSY
       BZ=(RBZ-RAZ)*WCSZ
        ZB=-SMU*(BX+BY+BZ)/(SQRT(SLRA)*SQRT(CRPD))
       DU=UAX*UBX+UAY*UBY+UAZ*UBZ
        THET=(1.-DU)/SQRT(1.-(DU**2))
        U=WAY*EAZ-WAZ*EAY
        V=-WAX*EAZ+WAZ*EAX
        W=WAX*EAY-WAY*EAX
        PO=U*UAX+V*UAY+W*UAZ
        ZC=SMU*THET*PO/SQRT(SLRA)
         D=-SMU*THET*THET
       XB=EBX+UBX
       YB=EBY+UBY
        TB=EBZ+UBZ
        XX=WBY*TB-WBZ*YB
         YY=-WBX*TB+WBZ*XB
       ZZ=WBX*YB-WBY*XB
       TX=XX*(RBX-RAX)
       TY=YY*(RBY-RAY)
       TZ=ZZ*(RBZ-RAX)
       TMAG=TX+TY+TZ
```

```
      E=SMU*TMAG/(SQRT(SLRB)*SQRT(CRPD))
       E=-E
      AL=WBY*EBZ-WBZ*EBY
      BET=-WBX*EBZ+WBZ*EBX
      GAM=WBX*EBY-WBY*EBX
      RED=AL*UBX+BET*UBY+GAM*UBZ
       F=-SMU*THET*RED/SQRT(SLRB)
       WCS=WCSX**2+WCSY**2+WCSZ**2
      AA=SMU*WCS/SLRA
      DOT=UAX*(RBX-RAX)+UAY*(RBY-RAY)+UAZ*(RBZ-RAZ)
        BB=2.0*SMU*THET*DOT/SQRT(CRPD)
      G=AA+BB
       SAL=XX**2+YY**2+ZZ**2
       AC=SMU*SAL/SLRB
       TOM=UBX*(RBX-RAX)+UBY*(RBY-RAY)+UBZ*(RBZ-RAZ)
       DC=2.0*SMU*THET*TOM/SQRT(CRPD)
        RH=AC-DC
       PHA=ZA*ZA*(G-RH)+ZA*(E*E-ZB*ZB)
       PHBA=ZA*ZA*(4.*F-4.*ZC)+ZA*(2.*E*G-2.*ZB*RH)
        PHBB=2.*(E*E*ZB-E*ZB*ZB)
      PHB=PHBA+PHBB
       PHCA=ZA*(8.*ZB*F-8.*E*ZC+2.*E*F-2.*ZB*ZC)
       PHCB=E*E*G-RH*ZB*ZB
       PHC=PHCA+PHCB
       PHDA=ZA*(4.*ZB*D-4.*E*D+2.*F*G-2.*ZC*RH)
       PHDB=4.*ZB*E*F-2.*ZC*E*E-4.*ZB*ZC*E+2.*F*ZB*ZB
      PHD=PHDA+PHDB
       PHEA=D*(2.*ZA*G-2.*RH*ZA-E*E+ZB*ZB)
       PHEB=ZA*(F*F-ZC*ZC)-2.*ZB*ZC*RH+2.*G*E*F
      PHE=PHEA+PHEB
       PHFA=D*(4.*F*ZA-4.*ZA*ZC+2.*E*G-2.*ZB*RH)
       PHFB=4.*F*ZB*ZC-4.*ZC*E*F+2.*ZB*F*F-2.*E*ZC*ZC
      PHF=PHFA+PHFB
       PHGA=D*(8.*ZB*F-8.*E*ZC+2.*ZB*ZC-2.*E*F)
       PHGB=F*F*G-ZC*ZC*RH
      PHG=PHGA+PHGB
       PHHA=D*D*(4.*ZB-4.*E)+D*(2.*F*G-2.*RH*ZC)
       PHHB=2.*F*ZC*ZC-2.*ZC*F*F
      PHH=PHHA+PHHB
       PHI=D*D*(G-RH)+D*(ZC*ZC-F*F)
         USSA=SQRT(SLRA)*SQRT(CRPD)
       M=8
       A(1)=PHA
      A(2)=PHB
      A(3)=PHC
       A(4)=PHD
       A(5)=PHE
       A(6)=PHF
       A(7)=PHG
       A(8)=PHH
       A(9)=PHI
       CALL SMULR(A,M,B,C)
      DO 660 I=1,8
       TIRE=ABS(C(I))
       WHEEL=TIRE-.001
      IF(WHEEL)630,630,660
630 SIGN=1.
```

```
      IF(B(I).LT.0.)SIGN=-1.
      TRANS=SIGN
      SLR=B(I)*B(I)
      DFDP=ZA+SIGN*(ZB/SQRT(SLR)+ZC/SLR**1.5)+D/SLR**2
      DGDP=ZA+SIGN*(E/SQRT(SLR)+F/SLR**1.5)+D/SLR**2
      IF(DFDP/DGDP)650,660,660
  650 RP=SQRT(SLR)
      FOP=ZA*SLR+SIGN*(ZB*2.*RP-2.*ZC/RP)+G-D/SLR
      GOP=ZA*SLR+SIGN*(2.*E*RP-2.*F/RP)+RH-D/SLR
       SIGMA=-DFDP/DGDP
       TAU=SQRT(FOP)/SQRT(GOP)
       SHOT=SQRT(FOP)+SQRT(GOP)
      NG=NG+1
      PAR(1,NG)=APH*90./PIE
      PAR(2,NG)=BPH*90./PIE
      PAR(3,NG)=SLR
      PAR(4,NG)=SHOT
      PAR(5,NG)=TRANS
  660 CONTINUE
      LIMIT=MAXSAV+1
      DO 690 J=2,LIMIT
      DO 690 I=J,NG
      DO 670 M=1,5
  670 TEMP(M)=PAR(M,J-1)
      IF(TEMP(4).LT.PAR(4,I))GO TO 690
      DO 680 M=1,5
      PAR(M,J-1)=PAR(M,I)
  680 PAR(M,I)=TEMP(M)
  690 CONTINUE
      WRITE(6,1007)
      WRITE(6,1008)((PAR(M,J),M=1,5),J=1,MAXSAV)
      DO 710 J=1,MAXSAV
      DO 700 K=1,2
  700 PAR(K,J)=PAR(K,J)*PIE/90.
      TRANS=PAR(5,J)
      CALL GRADNT(PAR(1,J),PAR(2,J),PAR(3,J))
      S(1,J)=PULSE
      S(2,J)=GUNA
      S(3,J)=GUNB
      S(4,J)=TSLR
      S(5,J)=ETRAN
      S(6,J)=TAPHDG
      S(7,J)=TBPHDG
      S(8,J)=PERARG
      S(9,J)=TRSTAN
      S(10,J)=TRSTB
  710 CONTINUE
      DO 740 I=2,MAXSAV
      IF(S(1,1).LT.S(1,I))GO TO 740
      DO 730 M=1,10
  730 S(M,1)=S(M,I)
  740 CONTINUE
      RETURN
 1000 FORMAT(6F12.7)
 1001 FORMAT(1H4,5X,4HAINC,10X,3HWMA,10X,6HGRADMN,8X,5HSO(1),9X,5HSO(2),
     19X,5HSO(3))
 1002 FORMAT(1H ,8F14.7)
```

```
1003 FORMAT(1HO,4X,6HMAXSAV,9X,5HITMAX)
1004 FORMAT(1H ,2I14)
1005 FORMAT(1HO,5X,4HSLRA,10X,4HSLRB,11X,2HEA,11X,2HEB,11X,3HWMB)
1006 FORMAT(2I12)
1007 FORMAT(1HO,5X,6HPHI(1),10X,6HPHI(2),5X,17HSEMI-LATUS RECTUM,4X,7HI
    1MPULSE,10X,5HTRANS)
1008 FORMAT(1H ,5F16.8)
     END
```

```
C                          SUBROUTINE FUNCT
C
C
C       THE SUBROUTINE WHICH COMPUTES THE GRADIENT OF THE TOTAL
C       IMPULSE IN ACCORDANCE WITH MC CUE AND BENDER (REF. 3)
C
C
$IBFTC FNCT     XR7,M94,DECK
       SUBROUTINE FUNCT(PAM,PLS2,GRD)
       DIMENSION PAM(3),GRD(3)
       COMMON/CMN6/GUNA,TUBX,TUBY,TUBZ,GUNBX,GUNBY,GUNBZ,GUNB,TRANS,
      1        EAX,EAY,EAZ,EBX,EBY,EBZ,SLRA,SLRB,WMA,WMB,DLTS,SO(3),
      2 AINC,EA,EB,WAX,WAY,WAZ,WBX,WBY,WBZ ,PIE,SMU,ITMAX,GRADMN,
      3SMLVX,SMLVY,SMLVZ,ZLITT,TUAX,TUAY,TUAZ,TRAMG,GUNAX,GUNAY,GUNAZ
      4,PULSE,TSLR,TAPH,TBPH,TRSTAN,TRSTB,ETRAN,PERARG
       TAPH=PAM(1)
       TBPH=PAM(2)
       TSLR=PAM(3)
       TUAX=COS(TAPH)
        TUAY=SIN(TAPH)*COS(AINC)
        TUAZ=SIN(TAPH)*SIN(AINC)
       TUBX=COS(TBPH)
       TUBY=SIN(TBPH)
        TUBZ=0.
       TANGA=TAPH-WMA
        TANGB=TBPH-WMB
        TRAMG=SLRA/(1.+EA*COS(TANGA))
        TRBMG=SLRB/(1.+EB*COS(TANGB))
        TRAX=TRAMG*TUAX
        TRAY=TRAMG*TUAY
        TRAZ=TRAMG*TUAZ
        TRBX=TRBMG*TUBX
        TRBY=TRBMG*TUBY
        TRBZ=TRBMG*TUBZ
       XANG=COS(TAPH)*COS(TBPH)+SIN(TAPH)*SIN(TBPH)*COS(AINC)
       ANG=ARCOS(XANG)
        COTD=COS(ANG)/SIN(ANG)
        CSCD=1./SIN(ANG)
       DENOM=SQRT(1.-(COS(TAPH)*COS(TBPH)+SIN(TAPH)*SIN(TBPH)*COS(
      1 AINC))**2)
       DTHTA=-(-SIN(TAPH)*COS(TBPH)+COS(TAPH)*SIN(TBPH)*COS(AINC))/
      1 DENOM
        DTHTB=-(-COS(TAPH)*SIN(TBPH)+SIN(TAPH)*COS(TBPH)*COS(AINC))/
      1 DENOM
        DCSCA=-CSCD*COTD*DTHTA
        DCSCB=-CSCD*COTD*DTHTB
        DCOTA=-CSCD*CSCD*DTHTA
       DCOTB=-CSCD*CSCD*DTHTB
       DUAXA=-SIN(TAPH)
        DUAYA=COS(TAPH)*COS(AINC)
        DUAZA=COS(TAPH)*SIN(AINC)
        DUBXB=-SIN(TBPH)
       DUBYB=COS(TBPH)
        DUBZB=0.
       DRADP=(TRAMG*TRAMG*EA*SIN(TANGA))/SLRA
        DRBDP=(TRBMG*TRBMG*EB*SIN(TANGB))/SLRB
        TSUMX=EAX+TUAX
```

```
      TSUMY=EAY+TUAY
       TSUMZ=EAZ+TUAZ
       VAX=WAY*TSUMZ-WAZ*TSUMY
       VAY=WAZ*TSUMX-WAX*TSUMZ
       VAZ=WAX*TSUMY-WAY*TSUMX
       VAX=VAX*SQRT(SMU/SLRA)
      VAY=VAY*SQRT(SMU/SLRA)
       VAZ=VAZ*SQRT(SMU/SLRA)
      SUMBX=EBX+TUBX
       SUMBY=EBY+TUBY
      SUMBZ=EBZ+TUBZ
       VBX=WBY*SUMBZ-WBZ*SUMBY
      VBY=WBZ*SUMBX-WBX*SUMBZ
       VBZ=WBX*SUMBY-WBY*SUMBX
       VBX=VBX*SQRT(SMU/SLRB)
       VBY=VBY*SQRT(SMU/SLRB)
      VBZ=VBZ*SQRT(SMU/SLRB)
      ANGH=ANG/2.
       ZLITT=SQRT(SMU/TSLR)*TAN(ANGH)
      TRCRX=(TRAY*TRBZ-TRAZ*TRBY)**2
       TRCRY=(TRAZ*TRBX-TRAX*TRBZ)**2
       TRCRZ=(TRAX*TRBY-TRAY*TRBX)**2
      TRCR=SQRT(TRCRX+TRCRY+TRCRZ)
      SMLVX=SQRT(SMU*TSLR)*(TRBX-TRAX)/TRCR
      SMLVY=SQRT(SMU*TSLR)*(TRBY-TRAY)/TRCR
       SMLVZ=SQRT(SMU*TSLR)*(TRBZ-TRAZ)/TRCR
       GUNAX=TRANS*(SMLVX*ZLITT*TUAX)-VAX
       GUNAY=TRANS*(SMLVY*ZLITT*TUAY)-VAY
       GUNAZ=TRANS*(SMLVZ+ZLITT*TUAZ)-VAZ
      GUNA=SQRT(GUNAX**2+GUNAY**2+GUNAZ**2)
       GUNBX=VBX-TRANS*(SMLVX-ZLITT*TUBX)
       GUNBY=VBY-TRANS*(SMLVY-ZLITT*TUBY)
       GUNBZ=VBZ-TRANS*(SMLVZ-ZLITT*TUBZ)
       GUNB=SQRT(GUNBX**2+GUNBY**2+GUNBZ**2)
       DVAAX=-SQRT(SMU/SLRA)*COS(TAPH)
       DVAAY=-SQRT(SMU/SLRA)*SIN(TAPH)*COS(AINC)
       DVAAZ=-SQRT(SMU/SLRA)*SIN(TAPH)*SIN(AINC)
       DVBBX=-SQRT(SMU/SLRB)*COS(TBPH)
       DVBBY=-SQRT(SMU/SLRB)*SIN(TBPH)
      DVBBZ=0.
       DVTAX=SQRT(SMU/TSLR)*((TSLR/TRAMG)*TUBX*(DCSCA-CSCD*DRADP/
     1 TRAMG)+(1.-TSLR/TRBMG)*(TUAX*DCSCA+CSCD*DUAXA)-TUAX*DCOTA
     2 -COTD*DUAXA)
      DVTAY=SQRT(SMU/TSLR)*((TSLR/TRAMG)*TUBY*(DCSCA-CSCD*
     1DRADP/TRAMG)+(1.-TSLR/TRBMG)*(TUAY*DCSCA+CSCD*DUAYA)-
     2TUAY*DCOTA-COTD*DUAYA)
      DVTAZ=SQRT(SMU/TSLR)*((TSLR/TRAMG)*TUBZ*(DCSCA-CSCD*DRADP/
     1 TRAMG)+(1.-TSLR/TRBMG)*(TUAZ*DCSCA+CSCD*DUAZA)-TUAZ*
     2 DCOTA-COTD*DUAZA)
       DVTBX=SQRT(SMU/TSLR)*((TSLR/TRAMG)*(TUBX*DCSCB+CSCD*DUBXB)
     1 +TUAX*((1.-(TSLR/TRBMG))*DCSCB+TSLR*CSCD*DRBDP/(TRBMG*
     2 TRBMG)-DCOTB))
       DVTBY=SQRT(SMU/TSLR)*((TSLR/TRAMG)*(TUBY*DCSCB+CSCD*DUBYB
     1 )+TUAY*((1.-(TSLR/TRBMG))*DCSCB+TSLR*CSCD*DRBDP/
     2 (TRBMG*TRBMG)-DCOTB))
       DVTBZ=SQRT(SMU/TSLR)*((TSLR/TRAMG)*(TUBZ*DCSCB+CSCD*
     1DUBZB)+TUAZ*((1.-(TSLR/TRBMG))*DCSCB+TSLR*CSCD*DRBDP/
```

```
2(TRBMG*TRBMG)-DCOTB))
  PVTAX=SQRT(SMU/TSLR)*(-(TSLR/TRBMG)*(TUAX*DCSCA+CSCD*
1DUAXA)+TUBX*(-(1.-(TSLR/TRAMG))*DCSCA-TSLR*CSCD*
2 DRADP/(TRAMG*TRAMG)+DCOTA))
 PVTAY=SQRT(SMU/TSLR)*(-(TSLR/TRBMG)*(TUAY*DCSCA+CSCD*
1DUAYA)+TUBY*(-(1.-(TSLR/TRAMG))*DCSCA-TSLR*CSCD*
2 DRADP/(TRAMG*TRAMG)+DCOTA))
  PVTAZ=SQRT(SMU/TSLR)*(-(TSLR/TRBMG)*(TUAZ*DCSCA+CSCD*
1DUAZA)+TUBZ*(-(1.-(TSLR/TRAMG))*DCSCA-TSLR*CSCD*
2 DRADP/(TRAMG*TRAMG)+DCOTA))
  PVTBX=SQRT(SMU/TSLR)*((TSLR/TRBMG)*TUAX*(-DCSCB+CSCD*
1 DRBDP/TRBMG)-(1.-TSLR/TRAMG)*(TUBX*DCSCB+CSCD*DUBXB)
2 +TUBX*DCOTB+COTD*DUBXB)
  PVTBY=SQRT(SMU/TSLR)*((TSLR/TRBMG)*TUAY*(-DCSCB+CSCD*
1 DRBDP/TRBMG)-(1.-TSLR/TRAMG)*(TUBY*DCSCB+CSCD*DUBYB)+TUBY*
2 DCOTB+COTD*DUBYB)
  PVTBZ=SQRT(SMU/TSLR)*((TSLR/TRBMG)*TUAZ*(-DCSCB+CSCD*
1 DRBDP/TRBMG)-(1.-TSLR/TRAMG)*(TUBZ*DCSCB+CSCD*DUBZB)+
2 TUBZ*DCOTB+COTD*DUBZB)
  RUNX=-TRANS*PVTAX
  RUNY=-TRANS*PVTAY
  RUNZ=-TRANS*PVTAZ
  FALLX=TRANS*DVTBX
  FALLY=TRANS*DVTBY
  FALLZ=TRANS*DVTBZ
  RHITX=-TRANS*PVTBX+DVBBX
  RHITY=-TRANS*PVTBY+DVBBY
  RHITZ=-TRANS*PVTBZ+DVBBZ
  SWIM=DVTAX*TRANS-DVAAX
  SWAM=DVTAY*TRANS-DVAAY
  SWUM=DVTAZ*TRANS-DVAAZ
  XXXXX=(SWIM*GUNAX+SWAM*GUNAY+SWUM*GUNAZ)/(GUNA)+(RUNX*GUNBX+
1 RUNY*GUNBY+RUNZ*GUNBZ)/(GUNB)
  YYYYY=(FALLX*GUNAX+FALLY*GUNAY+FALLZ*GUNAZ)/(GUNA)+(
1 RHITX*GUNBX+RHITY*GUNBY+RHITZ*GUNBZ)/(GUNB)
  DIDPO=(GUNAX*(SMLVX-ZLITT*TUAX)+GUNAY*(SMLVY-ZLITT*TUAY)+
1GUNAZ*(SMLVZ-ZLITT*TUAZ))/GUNA
  DIDPT=(GUNBX*(SMLVX+ZLITT*TUBX)+GUNBY*(SMLVY+ZLITT*TUBY)+
1GUNBZ*(SMLVZ+ZLITT*TUBZ))/GUNB
  ZZZZ=TRANS*(DIDPO-DIDPT)/(2.*TSLR)
 PLS2=GUNA+GUNB
 GRD(1)=XXXXX
 GRD(2)=YYYYY
 GRD(3)=ZZZZ
 RETURN
 END
```

```
C                          SUBROUTINE GRADNT
C
C
C       THE SUBROUTINE WHICH SUPPLIES THE LOGIC FOR THE FLETCHER-
C       POWELL CONJUGATE GRADIENT CONVERGENCE METHOD
C
C
$IBFTC GRAD     XR7,M94,DECK
       SUBROUTINE GRADNT(TAPH,TBPH,TSLR)
       COMMON/CMN6/GUNA,TUBX,TUBY,TUBZ,GUNBX,GUNBY,GUNBZ,GUNB,TRANS,
      1        EAX,EAY,EAZ,EBX,EBY,EBZ,SLRA,SLRB,WMA,WMB,DLTS,SO(3),
      2 AINC,EA,EB,WAX,WAY,WAZ,WBX,WBY,WBZ ,PIE,SMU,ITMAX,GRADMN,
      3SMLVX,SMLVY,SMLVZ,ZLITT,TUAX,TUAY,TUAZ,TRAMG,GUNAX,GUNAY,GUNAZ
      4,PULSE,TSLR,TAPH,TBPH,TRSTAN,TRSTB,ETRAN,PERARG,TAPHDG,TBPHDG
       DIMENSION VA(3),G(3),PRM(3),H(3,3),G1(3),SIG(3),Y(3),A(3,3),HY(3),
      1 YH(3),B(3,3)
       WRITE(6,300)
       MM=0
       PRM(1)=TAPH
       PRM(2)=TBPH
       PRM(3)=TSLR
       IDENTH=0
       CALL FUNCT(PRM,PULSE,G)
       WRITE(6,301)MM,STEP,PULSE,(PRM(I),I=1,3),(G(I),I=1,3)
       N=3
    5  DO 10 I=1,N
       DO 10 J=1,N
   10  H(I,J)=0.
       DO 20 I=1,N
   20  H(I,I)=1.
   40  CALL SMIN(STEP,PRM,H,$250)
       IF(STEP.LT..00000005)GO TO 130
       DO 50 I=1,N
       G1(I)=G(I)
   50  SIG(I)=0.
       DO 70 I=1,N
       DO 60 J=1,N
   60  SIG(I)=SIG(I)-STEP*H(I,J)*G(J)
   70  PRM(I)=PRM(I)+SIG(I)
       MM=1+MM
       PLSM=PULSE
       CALL FUNCT(PRM,PULSE,G)
       WRITE(6,301)MM,STEP,PULSE,(PRM(I),I=1,3),(G(I),I=1,3)
       IF((PLSM-PULSE).LT.1.E-07)GO TO 200
       IF(MM.GT.ITMAX)GO TO 200
       JUMP=0
       DO 75 I=1,N
   75  IF(ABS(G(I)).GT.GRADMN)JUMP=5
       IF(JUMP.EQ.0)GO TO 200
       DO 80 I=1,N
   80  Y(I)=G(I)-G1(I)
       SIGDTY=0.
       DO 85 I=1,N
   85  SIGDTY=SIGDTY+SIG(I)*Y(I)
       DO 90 I=1,N
       DO 90 J=1,N
   90  A(I,J)=SIG(I)*SIG(J)/SIGDTY
```

```
      DO 100 I=1,N
      HY(I)=0.
  100 YH(I)=0.
      DO 110 I=1,N
      DO 110 J=1,N
      HY(I)=H(I,J)*Y(J)+HY(I)
  110 YH(I)=H(J,I)*Y(J)+YH(I)
      YDTHY=0.
      DO 115 I=1,N
  115 YDTHY=YDTHY+Y(I)*HY(I)
      DO 120 I=1,N
      DO 120 J=1,N
      B(I,J)=-HY(I)*YH(J)/YDTHY
  120 H(I,J)=H(I,J)+A(I,J)+B(I,J)
      GO TO 40
  130 IDENTH=IDENTH+1
      IF(IDENTH.GT.2)GO TO 260
      WRITE(6,1004)
      GO TO 5
  260 WRITE(6,1005)
      GO TO 200
  250 WRITE(6,1003)
      GO TO 130
  200 TAPH=PRM(1)*90./PIE
      TBPH=PRM(2)*90./PIE
      TAPHDG=TAPH
      TBPHDG=TBPH
      TSLR=PRM(3)
       VA(1)=TRANS*(SMLVX+ZLITT*TUAX)
       VA(2)=TRANS*(SMLVY+ZLITT*TUAY)
       VA(3)=TRANS*(SMLVZ+ZLITT*TUAZ)
       SPEED=VA(1)**2+VA(2)**2+VA(3)**2
       BIGAX=1./(2./TRAMG-SPEED/SMU)
       ETRAN=SQRT(1.-TSLR/BIGAX)
      ALPO=ARCOS((GUNAX*TUAX+GUNAY*TUAY+GUNAZ*TUAZ)/GUNA)
      IF(TSLR.LT.SLRA)ALPO=-ALPO
      TRSTAN=(PIE-ALPO)*90./PIE
      ZETA=ARCOS((GUNBX*TUBX+GUNBY*TUBY+GUNBZ*TUBZ)/GUNB)
      IF(TSLR.GT.SLRB)ZETA=-ZETA
      TRSTB=(PIE-ZETA)*90./PIE
      PHASE=0.
      IF(TSLR/TRAMG-1..LT.0.)PHASE=2.*PIE
      FO=(ATAN(SQRT(TSLR/SMU)*(TUAX*VA(1)+TUAY*VA(2)+TUAZ*VA(3))/(TSLR/
     1 TRAMG-1.))+PHASE)*90./PIE
      PERARG=TAPH-FO
      WRITE(6,1001)
      WRITE(6,1002)PULSE,TSLR,TAPH,TBPH,TRSTAN,TRSTB,ETRAN,MM
      WRITE(6,1006)
      WRITE(6,1007)GUNA,GUNB,PERARG,TRANS
       RETURN
  300 FORMAT(1HO,4X,2HMM,9X,4HSTEP,9X,7HIMPULSE,7X,6HPRM(1),8X,6HPRM(2),
     18X,6HPRM(3),9X,4HG(1),10X,4HG(2),10X,4HG(3))
  301 FORMAT(1H ,I3,7X,E14.7,4F14.7,3E14.7)
 1001 FORMAT(1HO,4X,7HIMPULSE,8X,10HP-TRANSFER,8X,6HPHI(1),10X,6HPHI(2),
     17X,14HTHRUST ANGLE-I,4X,15HTHRUST ANGLE-II,3X,10HE-TRANSFER,3X,
     2 10HITERATIONS)
 1002 FORMAT(1H ,7F16.8,I10)
```

```
1003 FORMAT(1H0,47HSUBROUTINE SMIN IS UNABLE TO FIND THE NEXT STEP)
1004 FORMAT(1H ,22HMATRIX H REINITIALIZED)
1005 FORMAT(1H ,66HAFTER 2 ATTEMPTS TO REDUCE IMPULSE BY REINITIALIZING
    1 M, I AM TIRED)
1006 FORMAT(1H ,4X,9HIMPULSE-I,6X,10HIMPULSE-II,5X,17HTRNSFR PRG ARGMNT
    1,4X,5HTRANS)
1007 FORMAT(1H ,7F16.8)
       END
```

```
C                              SUBROUTINE SMIN
C
C
C       THE SUBROUTINE WHICH PROVIDES FOR A MINIMIZATION OF THE
C       IMPULSE FUNCTION ALONG A DIRECTION DERIVED FROM THE
C       FLETCHER-POWELL METHOD
C
C
$IBFTC SMMM     XR7,M94,DECK
       SUBROUTINE SMIN(C,XX,H,*)
C       SINGLE-DIMENSIONAL MINIMIZATION
       DIMENSION G1(10),G2(10),G3(10),X1(10),X2(10),X3(10),XX(3),X(10),
      1 GRX(10),DD(10),H(3,3)
       REAL MARG
       INTEGER COUNT
       JAG=0
       JIG=0
       COUNT=1
       JILL=0
       SIZE=0.0
       SLOPE1=0.0
       SLOPE2=0.0
       SLOPE3=0.0
       C=0.0
       N=3
       CSTEP=0.0
       DO 5 J=1,N
 5     X1(J)=XX(J)
       CALL FUNCT(X1,VAL1,GRX)
       DO 6 J=1,N
    6  DD(J)=0.
       DO 8 J=1,N
       DO 7 I=1,N
    7  DD(J)=DD(J)-H(J,I)*GRX(I)
       G1(J)=GRX(J)
    8  SLOPE1=SLOPE1+G1(J)*DD(J)
 10    IF (COUNT-1) 16,11,16
 11    C=0.0
       DO 13 J=1,N
       SIZE=ABS(DD(J))
       IF (C-SIZE) 12,13,13
 12    C=SIZE
 13    CONTINUE
       IF (C-2.0) 15,15,14
 14    C=4.0/C
       GO TO 16
 15    C=2.0
 16    IF (COUNT-2) 18,18,17
 17    C=2.0*C
 18    DO 19 J=1,N
 19    X2(J)=X1(J)+C*DD(J)
       JIG=JIG+1
       IF(JIG.GT.20)GO TO 37
       IF(X2(3)-0.)191,191,192
  191  C=.9*C
       GO TO 18
  192  CALL FUNCT(X2,VAL2,G2)
```

```
      MARG=VAL1+0.33*ABS(VAL1)
      IF (VAL2-MARG) 21,21,20
20    C=0.2*C
      GO TO 18
21    SLOPE2=0.0
      JIG=0
      DELC=C
      DO 22 J=1,N
22    SLOPE2=SLOPE2+G2(J)*DD(J)
      COUNT=COUNT+1
      IF (VAL2-VAL1) 23,24,24
23    IF (SLOPE2-0.0) 25,25,24
  24  CALL CUBIC(VAL1,VAL2,SLOPE1,SLOPE2,C,DELC,$37)
      JAG=JAG+1
      IF(JAG.GT.20)GO TO 37
      GO TO 27
25    VAL1=VAL2
      SLOPE1=SLOPE2
      DO 26 J=1,N
26    X1(J)=X2(J)
      CSTEP=CSTEP+C
      JILL=1+JILL
      IF(JILL.GT.30)GO TO 37
      GO TO 10
27    DO 28 J=1,N
28    X3(J)=X1(J)+C*DD(J)
      CALL FUNCT(X3,VAL3,G3)
      SLOPE3=0.0
      DO 31 J=1,N
31    SLOPE3=SLOPE3+G3(J)*DD(J)
      IF (VAL3-VAL1) 29,29,34
29    IF (VAL3-VAL2) 36,36,30
30    IF (SLOPE3-0.0) 32,34,34
32    VAL1=VAL3
      SLOPE1=SLOPE3
      DO 33 J=1,N
33    X1(J)=X3(J)
      CSTEP=CSTEP+C
      DELC=DELC-C
      GO TO 24
34    VAL2=VAL3
      SLOPE2=SLOPE3
      DO 35 J=1,N
35    X2(J)=X3(J)
      DELC=C
      GO TO 24
36    C=CSTEP+C
      RETURN
  37  RETURN 1
      END
```

```
C                          SUBROUTINE CUBIC
C
C
C       THE SUBROUTINE WHICH PERFORMS CUBIC INTERPOLATION TO ASSIST
C       SUBROUTINE SMIN
C
C
$IBFTC CUBI     XR7,M94
       SUBROUTINE CUBIC(VALL1,VALL2,SLOP1,SLOP2,CC,DELLC,*)
C       FITS A CUBIC POLYNOMIAL TO A ONE-DIMENSIONAL FUNCTION WITH POINT
C       1 AT ORIGIN, POINT 2 AT A DISTANCE DELLC AWAY
 505   FORMAT(1H0,25HBEGIN CUBIC INTERPOLATION)
 510   FORMAT(1H ,4E15.8)
       Z=(3.0*(VALL1-VALL2)/DELLC)+SLOP1+SLOP2
       TEST=Z*Z-SLOP1*SLOP2
       IF(TEST.LT.0.)GO TO 10
       Q=SQRT(TEST)
       DIS=(SLOP2+Q-Z)/(SLOP2-SLOP1+Q+Q)
       CC=DELLC*(1.0-DIS)
       RETURN
    10 WRITE(6,20)
       RETURN 1
    20 FORMAT(1H ,44HCUBIC FORCED TO TAKE SQRT OF NEGATIVE NUMBER)
       END
```

```
C                          SUBROUTINE SMULR
C
C
C       THE SUBROUTINE WHICH SOLVES THE EIGHTH-ORDER POLYNOMIAL TO
C       ASSIST SUBROUTINE IMPLSE
C
C
$IBFTC SMUSG   DECK,XR7,M94
      SUBROUTINE SMULR (COE,N1,ROOTR,ROOTI)
C
C
C*********************************************************************
C
C     POLYNOMIAL ROOT FINDER SUBROUTINE ....
C
C     ITERATIVE METHOD FOR POLYNOMIAL EQUATIONS ....
C
C     THIS METHOD APPROXIMATES THE FUNCTION F(Z) BY A QUADRATIC
C   WHICH MAY ,IN GENERAL, HAVE COMPLEX COEFFICIENTS AND DOES NOT
C   REQUIRE A KNOWLEDGE OF THE DERIVATIVE OF F(Z) THOUGH
C   THE FUNCTION F(Z) MUST BE EVALUATED ONCE PER ITERATION ....
C
C     THIS SUBROUTINE FINDS REAL AND COMPLEX ROOTS OF A POLYNOMIAL
C   WITH REAL COEFFICIENTS ....
C
C
C     USE OF MULLER SUBROUTINE ....
C   1.  CALL MULER  (COE,N1,ROOTR,ROOTI) ....
C   2.  COE IS THE TAG OF THE ARRAY OF COEFFICIENTS.
C       THE COEFFICIENTS MUST BE ORDERED FROM HIGHEST DEGREE
C       TO LOWEST DEGREE .
C   3.  N1 IS DEGREE OF THE POLYNOMIAL .
C   4.  ROOTR IS THE TAG OF THE ARRAY WHERE THE REAL PARTS
C       OF THE COMPLEX ROOTS ARE STORED .
C   5.  ROOTI IS THE TAG OF THE ARRAY WHERE THE IMAGINARY
C       PARTS OF THE COMPLEX ROOTS ARE STORED ....
C
C     ALL ARITHMETIC IS IN THE COMPLEX MODE ....
C     THEREFORE UNDER-FLOW IS NORMAL FOR REAL ROOTS ....
C
C     MULTIPLE ROOTS DECREASES ACCURACY OF THIS SUBROUTINE .
C     WHEN MULTIPLICITY IS FOUR THE ACCURACY DECREASES TO
C     ABOUT TWO PLACES ....
C
C
C
C
C*********************************************************************
C
C
C
C
      DIMENSION COE(1),ROOTR(1),ROOTI(1)
C
      N2=N1+1
      N4=0
      I=N1+1
```

```
19      IF(COE(I))9,7,9
7       N4=N4+1
        ROOTR(N4)=0.0
        ROOTI(N4)=0.0
        I=I-1
        IF(N4-N1)19,37,19
9       CONTINUE
C
10      AXR=0.8
        AXI=0.0
        L=1
        N3=1
        ALP1R=AXR
        ALP1I=AXI
        M=1
        GO TO 99
C
11      BET1R=TEMR
        BET1I=TEMI
        AXR=0.85
        ALP2R=AXR
        ALP2I=AXI
        M=2
        GO TO 99
C
12      BET2R=TEMR
        BET2I=TEMI
        AXR=0.9
        ALP3R=AXR
        ALP3I=AXI
        M=3
        GO TO 99
C
13      BET3R=TEMR
        BET3I=TEMI
14      TE1=ALP1R-ALP3R
        TE2=ALP1I-ALP3I
        TE5=ALP3R-ALP2R
        TE6=ALP3I-ALP2I
        TEM=TE5*TE5+TE6*TE6
        TE3=(TE1*TE5+TE2*TE6)/TEM
        TE4=(TE2*TE5-TE1*TE6)/TEM
        TE7=TE3+1.0
        TE9=TE3*TE3-TE4*TE4
        TE10=2.0  *TE3*TE4
        DE15=TE7*BET3R-TE4*BET3I
        DE16=TE7*BET3I+TE4*BET3R
        TE11=TE3*BET2R-TE4*BET2I+BET1R-DE15
        TE12=TE3*BET2I+TE4*BET2R+BET1I-DE16
        TE7=TE9-1.0
        TE1=TE9*BET2R-TE10*BET2I
        TE2=TE9*BET2I+TE10*BET2R
        TE13=TE1-BET1R-TE7*BET3R+TE10*BET3I
        TE14=TE2-BET1I-TE7*BET3I-TE10*BET3R
        TE15=DE15*TE3-DE16*TE4
        TE16=DE15*TE4+DE16*TE3
        TE1=TE13*TE13-TE14*TE14-4.0  *(TE11*TE15-TE12*TE16)
```

```
      TE2=2.0  *TE13*TE14-4.0  *(TE12*TE15+TE11*TE16)
      TEM= SQRT(TE1*TE1+TE2*TE2)
      IF(TE1)113,113,112
113   TE4= SQRT(0.5  *(TEM-TE1))
      TE3=0.5  *TE2/TE4
      GO TO 111
C
112   TE3= SQRT(0.5  *(TEM+TE1))
      IF(TE2)110,200,200
110   TE3=-TE3
200   TE4=0.5  *TE2/TE3
111   TE7=TE13+TE3
      TE8=TE14+TE4
      TE9=TE13-TE3
      TE10=TE14-TE4
      TE1=2.0  *TE15
      TE2=2.0  *TE16
      IF(TE7*TE7+TE8*TE8-TE9*TE9-TE10*TE10)204,204,205
204   TE7=TE9
      TE8=TE10
205   TEM=TE7*TE7+TE8*TE8
      TE3=(TE1*TE7+TE2*TE8)/TEM
      TE4=(TE2*TE7-TE1*TE8)/TEM
      AXR=ALP3R+TE3*TE5-TE4*TE6
      AXI=ALP3I+TE3*TE6+TE4*TE5
      ALP4R=AXR
      ALP4I=AXI
      M=4
      GO TO 99
C
15    N6=1
C*********************************************************************
38    IF( ABS(HELL)+ ABS(BELL)-1.0E-20)18,18,16
16    TE7= ABS(ALP3R-AXR)+ ABS(ALP3I-AXI)
      IF(TE7/( ABS(AXR)+ ABS(AXI))-1.0E-7)18,18,17
C*********************************************************************
17    N3=N3+1
      ALP1R=ALP2R
      ALP1I=ALP2I
      ALP2R=ALP3R
      ALP2I=ALP3I
      ALP3R=ALP4R
      ALP3I=ALP4I
      BET1R=BET2R
      BET1I=BET2I
      BET2R=BET3R
      BET2I=BET3I
      BET3R=TEMR
      BET3I=TEMI
      IF(N3-100)14,18,18
18    N4=N4+1
      ROOTR(N4)=ALP4R
      ROOTI(N4)=ALP4I
      N3=0
41    IF(N4-N1)30,37,37
37    RETURN
C*********************************************************************
```

```
30      IF( ABS(ROOTI(N4))-1.0E-5)10,10,31
31      GO TO (32,10),L
32      AXR=ALP1R
        AXI=-ALP1I
        ALP1I=-ALP1I
        M=5
        GO TO 99
33      BET1R=TEMR
        BET1I=TEMI
        AXR=ALP2R
        AXI=-ALP2I
        ALP2I=-ALP2I
        M=6
        GO TO 99
C
34      BET2R=TEMR
        BET2I=TEMI
        AXR=ALP3R
        AXI=-ALP3I
        ALP3I=-ALP3I
        L=2
        M=3
99      TEMR=COE(1)
        TEMI=0.0
        DO 100 I=1,N1
        TE1=TEMR*AXR-TEMI*AXI
        TEMI=TEMI*AXR+TEMR*AXI
100     TEMR=TE1+COE(I+1)
        HELL=TEMR
        BELL=TEMI
42      IF(N4)102,103,102
102     DO 101 I=1,N4
        TEM1=AXR-ROOTR(I)
        TEM2=AXI-ROOTI(I)
        TE1=TEM1*TEM1+TEM2*TEM2
        TE2=(TEMR*TEM1+TEMI*TEM2)/TE1
        TEMI=(TEMI*TEM1-TEMR*TEM2)/TE1
101     TEMR=TE2
103     GO TO (11,12,13,15,33,34),M
        END
```

APPENDIX B

DIGITAL COMPUTER PROGRAM FOR THE COMPUTATION OF THE MINIMUM-FUEL FINITE-THRUST TRANSFER TRAJECTORY

In this appendix, the complete Fortran IV digital computer program developed in this study to compute minimum-fuel finite-thrust transfer trajectories is presented. For convenience in computation, the state variables and the transformed adjoint variables are represented by the vector Y in the following manner:

$z = Y(1)$

$A = Y(2)$

$B = Y(3)$

$\theta = Y(4)$

$w = Y(5)$

$\Lambda = Y(6)$

$\Psi = Y(7)$

$\nu = Y(8)$

$s = Y(9)$

In addition, the adjoint variable λ_φ is represented by the symbol LN. The primary inputs to the program are:

EO = eccentricity of the initial orbit

PO = semi-latus rectum of the initial orbit

EF = eccentricity of the final orbit

PF = semi-latus rectum of the final orbit

AR = thrust to initial weight ratio of the rocket

CR = effective rocket exhaust velocity

The primary outputs of the program are:

1. The initial values of the true anomaly f, the thrust angle Ψ, and the variable ν.
2. The final values of eccentricity e, semi-latus rectum p, and argument of pericenter φ .
3. The total characteristic velocity of the transfer trajectory.

Following are brief descriptions of the main program and the subroutines including a complete listing of the Fortran IV program. Two main programs, IMPFTR and FTMAIN,are included. IMPFTR is the main program for the computation of both the two-impulse and finite-thrust transfer trajectories. The subprograms IMPLSE, FUNCT, GRADNT, SMIN, CUBIC, and SMULR from Appendix A must be used in conjunction with IMPFTR. FTMAIN is the main program which starts with the minimum-fuel two-impulse transfer trajectory and computes the corresponding minimum-fuel finite-thrust transfer trajectory.

IMPFTR - The main program for computing both the two-impulse and the finite-thrust minimum-fuel transfer trajectory.

FTMAIN - The main program for computing only the finite-thrust minimum-fuel transfer trajectory

FNTRST - The subroutine which establishes the initial-approximate transfer trajectory and calls the subroutines necessary for the execution of the convergence method.

PSISET - The subroutine which computes the initial value of the thrust-orientation angle so that the desired duration of the first thrusting interval is attained.

NUSET - The subroutine which computes the initial value of the variable ν so that the desired duration of the second thrusting interval is attained.

MIMIN - The subroutine which supplies the fixed step-size fourth-order Runge-Kutta integration algorithm.

DURATN - The subroutine which computes the first and second derivatives of the switching function. The derivatives are used in the iterative solution of Eq. (3-44). Double-precision arithmetic is employed.

Y7SET - The subroutine which supplies the logic for the iterative solution of Eq. (3-44). Double-precision arithmetic is employed throughout.

FNEW - The subroutine which computes the right hand sides of Eqs. (2-100) - (2-109).

WRTOUT - The subroutine which computes the hamiltonian and prints out the important characteristics of the finite-thrust minimum-fuel transfer trajectory as a function of time.

MINV - The subroutine which performs matrix inversion

F - The subroutine which computes the first derivative of the switching function.

COAST - The subroutine which performs the canonical transformation to eliminate integration along the coast interval.

PRMITR - The subroutine which controls the convergence method based on the sensitivity matrix [G] and computes the matrix [G] on the second and subsequent iterations.

PARHOM - The subroutine which performs the initial computation of the sensitivity matrix [G].

ZIPPO - The subroutine which controls the computation of a specific transfer trajectory.

```
C                                  IMPFTR
C
C
C       THE MAIN PROGRAM FOR COMPUTING BOTH THE TWO-IMPULSE AND THE
C       FINITE THRUST MINIMUM FUEL TRANSFER TRAJECTORY
C
C
C       PRIMARY INPUTS
C            AR = THRUST TO INITIAL WEIGHT RATIO
C            CR = EFFECTIVE ROCKET EXHAUST VELOCITY
C            E0 = ECCENTRICITY OF THE INITIAL ORBIT
C            P0 = SEMI-LATUS RECTUM OF THE INITIAL ORBIT
C            PF = SEMI-LATUS RECTUM OF THE FINAL ORBIT
C            EF = ECCENTRICITY OF THE FINAL ORBIT
C            WMBR = CHANGE IN THE ARGUMENT OF PERICENTER FROM THE FIRST
C                   ORBIT TO THE SECOND ORBIT
C            MAXSAV = THE NUMBER OF FIXED POINT TRAJECTORIES WHICH
C                     ARE USED AS STARTING POINTS FOR THE
C                     FLETCHER-POWELL METHOD
C            ITMAX = MAXIMUM ALLOWABLE NUMBER OF FLETCHER-POWELL
C                    ITERATIONS
C            GRADMN = MINIMUM CUTOFF VALUE OF THE GRADIENT IN THE
C                     FLETCHER-POWELL METHOD (GRADMN = .0001)
C
C
$IBFTC IMPFTR  XR7,M94,DECK
       REAL MAXDIF
       LOGICAL REGLAR,LBAK,NOCOAS
       DOUBLE PRECISION DURTOL
       COMMON/CMN1/AR,CR,PA,ES,THETA,CRT,DBLDOT,CPRO,PYE,P1,P2,PSI,DRN,D,
      1 Y50,Y60
       COMMON/CMN2/Y(9),P(9),XD(4),IWRT(4),TP(4),X,DX,N,TB,IP1,LN,IFRD
      1,DSTVAR(3,3)
       COMMON/CMN3/D2DIF,D1TOL1,D1TOL2,D2TOL,DL4,DL7,DL8,JUMP,JG,
      1 DL4II,DL8II,LBAK,TGTP,T1,T2,D1,D2,ABX8OV,X8OV,T2TOL1,T2TOL2,VTAN,
      2 D2RAT,DELCRT,DL4I,DL8III,THTLNR,JASH,JOSH,KOSH,JECK,JOCK
      3 ,JMPTHT,JMPCRT,CRTLNR,LNORM
       COMMON/CMN5/S(10,10),MAXSAV
       COMMON/CMN6/GUNA,TUBX,TUBY,TUBZ,GUNBX,GUNBY,GUNBZ,GUNB,TRANS,
      1        EAX,EAY,EAZ,EBX,EBY,EBZ,SLRA,SLRB,WMA,WMB,DLTS,SO(3),
      2 AINC,EA,EB,WAX,WAY,WAZ,WBX,WBY,WBZ ,PIE,SMU,ITMAX,GRADMN,
      3SMLVX,SMLVY,SMLVZ,ZLITT,TUAX,TUAY,TUAZ,TRAMG,GUNAX,GUNAY,GUNAZ
      4,PULSE,TSLR,TAPH,TBPH,TRSTAN,TRSTB,ETRAN,PERARG,TAPHDG,TBPHDG
       COMMON/CMN7/E0,P0,ET,PT,EF,PF,PHI1R,PHI2R,DLV1,DLV2,DN1R,WMBR,
      1 PSIA1,PSIA2,ASX8OV,MAXDIF,ICNTMX,D2TMN,DL8I,XDAR1,XDARP1,JMP2
      2 ,COASTN,IMPTRD,JMP1,ITPRM,KOGMAX,KOGMIN,REGLAR
      3 ,AX8VNR,DL81NR,DL82NR,DL83NR
       COMMON/CMN8/IDNWRT
       COMMON/CMN9/DURTOL
       COMMON/CMPH/KAK(3),PAR(3),HOM(3,3),EPS(3),C(3),DLT2,JPH
       COMMON/CMN10/ICNT,WB(3),ERRMIN,KEY1
       COMMON/CMN12/NOROT,PRAYER,NOCOAS,SIGMA9,JOEMAX,TST3MX
       COMMON/NAMCOM/  NFNTR,NPRM,NZIP,NMIN,NFNEW,NTHT,NNU,NCOAST,NPSI,
      1 NPRHM,NDRN,NY7SET,NWRT,NFFR,NEX1,NEX2,NEX3,NEX4,NEX5,NEX6,NEX7
       NAMELIST/NAMSUB/NFNTR,NPRM,NZIP,NMIN,NFNEW,NTHT,NNU,NCOAST,NPSI,
      1 NPRHM,NDRN,NY7SET,NWRT,NFFR,NEX1,NEX2,NEX3,NEX4,NEX5,NEX6,NEX7
       NAMELIST/Z/NR,AR,CR,E0,P0,ET,PT,EF,PF,PHI1R,PHI2R,DLV1,DLV2,DN1R,
```

```
     1 WMBR,LG
      NAMELIST/INPT/MAXDIF,IWRT,JMP1,ICNTMX,ASX8OV,DURTOL,D2RAT,D1TOL1,
     1 D1TOL2,D2TMN,DN4I,DN4II,DN8I,DN8II,DN8III,IDNWRT,JG,T2TOL1,
     2 T2TOL2,JMPTHT,JMPCRT,XDAR1,XDARP1,JMP2,THNLNR,COASTN,CRNLNR
     3 ,EPS,JPH,KEY1,INPTRD,ITPRM,KOGMAX,KOGMIN,REGLAR,IFRD,LBAK
     4 ,NOCOAS,SIGMA9,JOEMAX,TST3MX,AX8VNR,DN81NR,DN82NR,DN83NR
      NAMELIST/IMPULZ/MAXSAV,ITMAX,GRADMN
      READ(5,NAMSUB)
      READ(5,IMPULZ)
   5  READ(5,INPT)
  10  READ(5,1050)NR
      READ(5,1000)AR,CR
      READ(5,1000)EO,PO,EF,PF,WMBR
  20  WRITE(6,1011)NR
      WRITE(6,1004)
      WRITE(6,1001)AR,CR
      WRITE(6,1003)
      WRITE(6,1001)EO,PO,EF,PF,WMBR
      WRITE(6,1005)
      WRITE(6,1014)MAXSAV,ITMAX,GRADMN
      WRITE(6,1006)
      WRITE(6,1001)ASX8OV,MAXDIF,T2TOL1,T2TOL2,D2RAT,XDAR1,XDARP1,THNLNR
     1 ,COASTN
      WRITE(6,1026)
      WRITE(6,1001)CRNLNR
      WRITE(6,1007)
      WRITE(6,1008)DURTOL
      WRITE(6,1013)
      WRITE(6,1001)D1TOL1,D1TOL2,D2TMN,DN4I,DN4II,DN8I,DN8II,DN8III
      WRITE(6,1024)
      WRITE(6,1002)ICNTMX,IDNWRT,JMP1,JMPTHT,JMPCRT,JMP2,KEY1,JPH
      WRITE(6,1027)
      WRITE(6,1002)ITPRM,KOGMAX,KOGMIN,REGLAR,IFRD,LBAK
      WRITE(6,1025)
      WRITE(6,1001)(EPS(I),I=1,3)
      WRITE(6,1019)
      WRITE(6,1002)(IWRT(I),I=1,4)
      WRITE(6,1028)
      WRITE(6,1001)SIGMA9,TST3MX,AX8VNR,DN81NR,DN82NR,DN83NR
      WRITE(6,1029)
      WRITE(6,1002)NOCOAS,JOEMAX,JG
      SLRA=PO
      SLRB=PF
      EA=EO
      EB=EF
      WMB=WMBR
      CALL IMPLSE
      DLVTOT=S(1,1)
      DLV1=S(2,1)
      DLV2=S(3,1)
      PT=S(4,1)
      ET=S(5,1)
      PHI1R=S(6,1)
      PHI2R=S(7,1)
      DN1R=S(8,1)
      PSIA1=S(9,1)
      PSIA2=S(10,1)
```

```
      WRITE(6,1010)
      WRITE(6,1012)(S(I,1),I=1,10)
      PUNCH 1000, E0,P0,ET,PT,EF,PF
      PUNCH 1000, PHI1R,PHI2R,DLV1,DLV2
      PUNCH 1000, DN1R,WMBR
      PUNCH 1000, PSIA1,PSIA2
   30 DL4I=DN4I/AR
      DL4II=DN4II/AR
      DL8I=DN8I
      DL8II=DN8II/AR
      DL8III=DN8III/AR
      DL81NR=DN81NR/AR
      DL82NR=DN82NR/AR
      DL83NR=DN83NR/AR
      THTLNR=THNLNR/AR
      CRTLNR=CRNLNR/AR
      CALL FNTRST
      INPTRD=0
      READ(5,Z)
      IF(LG)5,10,20
1000 FORMAT(6F12.7)
1001 FORMAT(1X,9F14.8)
1002 FORMAT(1X,9I14)
1003 FORMAT(1H0,8X,2HE0,12X,2HP0,12X,2HEF,12X,2HPF,11X,4HWMBR)
1004 FORMAT(1H0,8X,2HAR,12X,2HCR)
1005 FORMAT(1H0,6X,6HMAXSAV,9X,5HITMAX,8X,6HGRADMN)
1006 FORMAT(1H0,6X,6HASX8OV,8X,6HMAXDIF,8X,6HT2TOL1,8X,6HT2TOL2,8X,
     1 5HD2RAT,8X,6HXDAR1 ,8X,6HXDARP1,8X,6HTHNLNR,8X,6HCOASTN)
1007 FORMAT(1H0,12X,6HDURTOL)
1008 FORMAT(1X,D25.16)
1009 FORMAT(1H0,6X,6HMAXSAV,8X,5HITMAX,9X,6HICNTMX)
1010 FORMAT(5X,6HDLVTOT,9X,4HDLV1,9X,4HDLV2,11X,2HPT,11X,2HET,8X,
     1 5HPHI1R,8X,5HPHI2R,9X,4HDN1R,8X,5HPSIA1,8X,5HPSIA2)
1011 FORMAT(1H1,13HRUN NUMBER B-I3)
1012 FORMAT(10F13.8)
1013 FORMAT(1H0,6X,6HD1TOL1,8X,6HD1TOL2,9X,5HD2TMN,10X,4HDN4I,9X,
     1 5HDN4II,10X,4HDN8I,9X,5HDN8II,8X,6HDN8III)
1014 FORMAT(1X,2I14,F14.8)
1019 FORMAT(1H0,4X,7HIWRT(1),5X,7HIWRT(2),5X,7HIWRT(3),5X,7HIWRT(4))
1024 FORMAT(1H0,6X,6HICNTMX,8X,6HIDNWRT,9X,4HJMP1,8X,6HJMPTHT,8X,
     1 6HJMPCRT,10X,4HJMP2,10X,4HKEY1,11X,3HJPH)
1025 FORMAT(1H0,6X,6HEPS(1),8X,6HEPS(2),8X,6HEPS(3))
1026 FORMAT(1H0,6X,6HCRNLNR)
1027 FORMAT(1H0,6X,5HITPRM,9X,6HKOGMAX,8X,6HKOGMIN,8X,6HREGLAR,10X,
     1 4HIFRD,10X,4HLBAK)
1028 FORMAT(1H0,6X,6HSIGMA9,8X,6HTST3MX,8X,6HAX8VNR,8X,6HDN81NR,8X,
     1 6HDN82NR,8X,6HDN83NR)
1029 FORMAT(1H0,6X,6HNOCOAS,8X,6HJOEMAX,8X,2HJG)
1050 FORMAT(6I12)
      END
$NAMSUB  NFNTR=0,   NPRM=0,NZIP=0,NMIN=0,NFNEW=0,NTHT=0,NNU=0,NCOAST=0,
NPSI=0,NPRHM=0,NDRN=0,NY7SET=0,NWRT=0,NFFR=0,NEX1=0,NEX2=0,NEX3=0,
NEX4=0,NEX5=0,NEX6=0,NEX7=0  $
$IMPULZ  GRADMN=.0001,   MAXSAV=10,   ITMAX=10  $
$INPT MAXDIF=.0001,   IWRT=1,5, 1,20,  JMP1=25,  ICNTMX=20, JMP2=15,
ASX8OV=1.50, DURTOL=.1D-07,  D2RAT=.75,      JMPTHT=4,   JMPCRT=5,
KEY1=1,   JPH=1,   INPTRD=0,   CRNLNR=-1.,   ITPRM=20,  KOGMAX=0,
```

```
THNLNR=.01,   COASTN=100,   EPS=.000001,    .000001,   .000001,
D1TOL1=.000001,  D1TOL2=.000001,   D2TMN=.00001,     XDAR1=.01,
DN4I=.00005,  DN4II=.0005,   DN8I=.1,   DN8II=.0005,   DN8III=.000005,
REGLAR=.TRUE.,   LBAK=.FALSE.,  IFRD=1 ,
KOGMIN=2,
NOCOAS=.FALSE., JOEMAX=5,   SIGMA9=.0000001,  TST3MX=.002,
AX8VNR=1.97,  DN81NR=-.002,    DN82NR=-.0002,  DN83NR=-.00002,
XDARP1=.05,      IDNWRT=1,   JG=1,    T2TOL1=.5,    T2TOL2=.00005   $
```

```
C                                  FTMAIN
C
C
C       THE MAIN PROGRAM FOR COMPUTING ONLY THE FINITE THRUST MINIMUM
C       FUEL TRANSFER TRAJECTORY
C
C
$IBFTC FTMAIN  XR7,M94,DECK
C       PRIMARY INPUTS
C            AR = THRUST TO INITIAL WEIGHT RATIO
C            CR = EFFECTIVE ROCKET EXHAUST VELOCITY
C            E0 = ECCENTRICITY OF THE INITIAL ORBIT
C            P0 = SEMI-LATUS RECTUM OF THE INITIAL ORBIT
C            ET = ECCENTRICITY OF THE TRANSFER TRAJECTORY
C            PT = SEMI-LATUS RECTUM OF THE TRANSFER TRAJECTORY
C            EF = ECCENTRICITY OF THE FINAL ORBIT
C            PF = SEMI-LATUS RECTUM OF THE FINAL ORBIT
C            WMBR = CHANGE IN THE ARGUMENT OF PERICENTER FROM THE FIRST
C                   ORBIT TO THE SECOND ORBIT
C            PHI1R = TRUE ANOMALY AT WHICH THE TWO IMPULSE TRANSFER
C                    IS INITIATED
C            PHI2R = (TRUE ANOMALY AT WHICH THE TWO IMPULSE TRANSFER
C                    IS COMPLETED) + WMBR
C            DLV1 = VELOCITY CHANGE ON THE FIRST THRUST INTERVAL
C                   OF THE TWO-IMPULSE TRANSFER TRAJECTORY
C            DLV2 = VELOCITY CHANGE ON THE SECOND THRUST INTERVAL
C                   OF THE TWO-IMPULSE TRANSFER TRAJECTORY
C            DN1R = CHANGE IN THE ARGUMENT OF PERICENTER ON THE
C                   FIRST THRUSTING PHASE OF THE TWO-IMPULSE TRANSFER
C            PSIA1 = INITIAL THRUST ANGLE ON THE TWO-IMPULSE TRAJECTORY
C            PSIA2 = FINAL THRUST ANGLE ON THE TWO-IMPULSE TRAJECTORY
C
C
      REAL MAXDIF
      LOGICAL REGLAR,LBAK,NOCOAS
      DOUBLE PRECISION DURTOL
      COMMON/CMN1/AR,CR,PA,EA,THETA,CRT,DBLDOT,CPRO,PIE,P1,P2,PSI,DRN,D,
     1 Y50,Y60
      COMMON/CMN2/Y(9),P(9),XD(4),IWRT(4),TP(4),X,DX,N,TB,IP1,LN,IFRD
     1,DSTVAR(3,3)
      COMMON/CMN3/D2DIF,D1TOL1,D1TOL2,D2TOL,DL4,DL7,DL8,JUMP,JG,
     1 DL4II,DL8II,LBAK,TGTP,T1,T2,D1,D2,ABX80V,X80V,T2TOL1,T2TOL2,VTAN,
     2 D2RAT,DELCRT,DL4I,DL8III,THTLNR,JASH,JOSH,KOSH,JECK,JOCK
     3 ,JMPTHT,JMPCRT,CRTLNR,LNORM
      COMMON/CMN7/E0,P0,ET,PT,EF,PF,PHI1R,PHI2R,DLV1,DLV2,DN1R,WMBR,
     1 PSIA1,PSIA2,ASX80V,MAXDIF,ICNTMX,D2TMN,DL8I,XDAR1,XDARP1,JMP2
     2 ,COASTN,IMPTRD,JMP1,ITPRM,KOGMAX,KOGMIN,REGLAR
     3 ,AX8VNR,DL81NR,DL82NR,DL83NR
      COMMON/CMN8/IDNWRT
      COMMON/CMN9/DURTOL
      COMMON/CMPH/KAK(3),PAR(3),HOM(3,3),EPS(3),C(3),DLT2,JPH
      COMMON/CMN10/ICNT,WB(3),ERRMIN,KEY1
      COMMON/CMN12/NOROT,PRAYER,NOCOAS,SIGMA9,JOEMAX,TST3MX
      COMMON/NAMCOM/  NFNTR,NPRM,NZIP,NMIN,NFNEW,NTHT,NNU,NCOAST,NPSI,
     1 NPRHM,NDRN,NY7SET,NWRT,NFFR,NEX1,NEX2,NEX3,NEX4,NEX5,NEX6,NEX7
      NAMELIST/NAMSUB/NFNTR,NPRM,NZIP,NMIN,NFNEW,NTHT,NNU,NCOAST,NPSI,
     1 NPRHM,NDRN,NY7SET,NWRT,NFFR,NEX1,NEX2,NEX3,NEX4,NEX5,NEX6,NEX7
```

```
      NAMELIST/Z/NR,AR,CR,EO,PO,ET,PT,EF,PF,PHI1R,PHI2R,DLV1,DLV2,DN1R,
     1 WMBR,LG
      NAMELIST/INPT/MAXDIF,IWRT,JMP1,ICNTMX,ASX8OV,DURTOL,D2RAT,D1TOL1,
     1 D1TOL2,D2TMN,DN4I,DN4II,DN8I,DN8II,DN8III,IDNWRT,JG,T2TOL1,
     2 T2TOL2,JMPTHT,JMPCRT,XDAR1,XDARP1,JMP2,THNLNR,COASTN,CRNLNR
     3 ,EPS,JPH,KEY1,INPTRD,ITPRM,KOGMAX,KOGMIN,REGLAR,IFRD,LBAK
     4 ,NOCOAS,SIGMA9,JOEMAX,TST3MX,AX8VNR,DN81NR,DN82NR,DN83NR
      READ(5,NAMSUB)
    5 READ(5,INPT)
   10 READ(5,1010)NR
      READ(5,1000)AR,CR
      READ(5,1000)EO,PO,ET,PT,EF,PF
      READ(5,1000)PHI1R,PHI2R,DLV1,DLV2
      READ(5,1000)DN1R,WMBR
      READ(5,1000)PSIA1,PSIA2
   20 WRITE(6,1011)NR
      WRITE(6,1002)
      WRITE(6,1001)AR,CR
      WRITE(6,1004)
      WRITE(6,1001)EO,PO,ET,PT,EF,PF
      WRITE(6,1005)
      WRITE(6,1001)PHI1R,PHI2R,DLV1,DLV2
      WRITE(6,1015)
      WRITE(6,1001)DN1R,WMBR
      WRITE(6,1014)
      WRITE(6,1001)PSIA1,PSIA2
      WRITE(6,1006)
      WRITE(6,1001)ASX8OV,MAXDIF,T2TOL1,T2TOL2,D2RAT,XDAR1,XDARP1,THNLNR
     1 ,COASTN
      WRITE(6,1026)
      WRITE(6,1001)CRNLNR
      WRITE(6,1007)
      WRITE(6,1008)DURTOL
      WRITE(6,1012)
      WRITE(6,1001)D1TOL1,D1TOL2,D2TMN,DN4I,DN4II,DN8I,DN8II,DN8III
      WRITE(6,1024)
      WRITE(6,1010)ICNTMX,IDNWRT,JMP1,JMPTHT,JMPCRT,JMP2,KEY1,JPH
      WRITE(6,1027)
      WRITE(6,1010)ITPRM,KOGMAX,KOGMIN,REGLAR,IFRD,LBAK
      WRITE(6,1025)
      WRITE(6,1001)(EPS(I),I=1,3)
      WRITE(6,1019)
      WRITE(6,1010)(IWRT(I),I=1,4)
      WRITE(6,1028)
      WRITE(6,1001)SIGMA9,TST3MX,AX8VNR,DN81NR,DN82NR,DN83NR
      WRITE(6,1029)
      WRITE(6,1010)NOCOAS,JOEMAX,JG
   30 DL4I=DN4I/AR
      DL4II=DN4II/AR
      DL8I=DN8I
      DL8II=DN8II/AR
      DL8III=DN8III/AR
      DL81NR=DN81NR/AR
      DL82NR=DN82NR/AR
      DL83NR=DN83NR/AR
      THTLNR=THNLNR/AR
      CRTLNR=CRNLNR/AR
```

```
      CALL FNTRST
      INPTRD=0
      READ(5,Z)
      IF(LG)5,10,20
 1000 FORMAT(6F12.8)
 1001 FORMAT(1H ,9F14.7)
 1002 FORMAT(1HO,8X,2HAR,12X,2HCR)
 1004 FORMAT(1HO,7X,2HE0,13X,2HP0,13X,2HET,13X,2HPT,13X,2HEF,13X,2HPF)
 1005 FORMAT(1HO,6X,4HPHI1,11X,4HPHI2,10X,4HDLV1,11X,4HDLV2)
 1006 FORMAT(1HO,6X,6HASX8OV,8X,6HMAXDIF,8X,6HT2TOL1,8X,6HT2TOL2,8X,
     1 5HD2RAT,8X,6HXDAR1 ,8X,6HXDARP1,8X,6HTHNLNR,8X,6HCOASTN)
 1007 FORMAT(1HO,12X,6HDURTOL)
 1008 FORMAT(1X,D25.16)
 1010 FORMAT(9I14)
 1011 FORMAT(1H1,13HRUN NUMBER B-I3)
 1012 FORMAT(1HO,6X,6HD1TOL1,8X,6HD1TOL2,9X,5HD2TMN,10X,4HDN4I,9X,
     1 5HDN4II,10X,4HDN8I,9X,5HDN8II,8X,6HDN8III)
 1014 FORMAT(1HO,7X,5HPSIA1,9X,5HPSIA2)
 1015 FORMAT(1HO,6X,3HDN1,12X,3HWMB)
 1019 FORMAT(1HO,6X,7HIWRT(1),7X,7HIWRT(2),7X,7HIWRT(3),7X,7HIWRT(4))
 1024 FORMAT(1HO,6X,6HICNTMX,8X,6HIDNWRT,9X,4HJMP1,8X,6HJMPTHT,8X,
     1 6HJMPCRT,10X,4HJMP2,10X,4HKEY1,11X,3HJPH)
 1025 FORMAT(1HO,6X,6HEPS(1),8X,6HEPS(2),8X,6HEPS(3))
 1026 FORMAT(1HO,6X,6HCRNLNR)
 1027 FORMAT(1HO,6X,5HITPRM,9X,6HKOGMAX,8X,6HKOGMIN,8X,6HREGLAR,10X,
     1 4HIFRD,10X,4HLBAK)
 1028 FORMAT(1HO,6X,6HSIGMA9,8X,6HTST3MX,8X,6HAX8VNR,8X,6HDN81NR,8X,
     1 6HDN82NR,8X,6HDN83NR)
 1029 FORMAT(1HO,6X,6HNOCOAS,8X,6HJOEMAX,8X,2HJG)
      END
$NAMSUB  NFNTR=0,   NPRM=0,NZIP=0,NMIN=0,NFNEW=0,NTHT=0,NNU=0,NCOAST=0,
NPSI=0,NPRHM=0,NDRN=0,NY7SET=0,NWRT=0,NFFR=0,NEX1=0,NEX2=0,NEX3=0,
NEX4=0,NEX5=0,NEX6=0,NEX7=0  $
$INPT MAXDIF=.0001,   IWRT=1,5, 1,20,  JMP1=25,  ICNTMX=20, JMP2=15,
ASX8OV=1.50, DURTOL=.1D-07,  D2RAT=.75,      JMPTHT=4,    JMPCRT=5,
KEY1=1,   JPH=1,   INPTRD=0,   CRNLNR=-1.,   ITPRM=20,  KOGMAX=0,
THNLNR=.01,   COASTN=100,   EPS=.000001,    .000001,    .000001,
D1TOL1=.000001,  D1TOL2=.000001,   D2TMN=.00001,     XDAR1=.01,
DN4I=.00005,  DN4II=.0005,   DN8I=.1,   DN8II=.0005,   DN8III=.000005,
REGLAR=.TRUE.,   LBAK=.FALSE.,  IFRD=1 ,
KOGMIN=2,
NOCOAS=.FALSE., JOEMAX=5,   SIGMA9=.0000001,  TST3MX=.002,
AX8VNR=1.97,  DN81NR=-.002,    DN82NR=-.0002,  DN83NR=-.00002,
XDARP1=.05,      IDNWRT=1,   JG=1,    T2TOL1=.5,    T2TOL2=.00005   $
```

```
C                          SUBROUTINE FNTRST
C
C
C       THE SUBROUTINE WHICH ESTABLISHES THE INITIAL FINITE THRUST
C       TRANSFER TRAJECTORY AND CALLS THE SUBROUTINES TO CONTROL THE
C       CONVERGENCE PROCEDURE
C
C
$IBFTC FNTR     XR7,M94,DECK
       SUBROUTINE FNTRST
       COMMON/CMN1/AR,CR,PA,EA,THETA,CRT,DBLDOT,CPRO,PIE,P1,P2,PSI,DRN,D,
      1 Y50,Y60
       COMMON/CMN2/Y(9),P(9),XD(4),IWRT(4),TP(4),X,DX,N,TB,IP1,LN,IFRD
      1,DSTVAR(3,3)
       COMMON/CMN3/D2DIF,D1TOL1,D1TOL2,D2TOL,DL4,DL7,DL8,JUMP,JG,
      1 DL4II,DL8II,LBAK,TGTP,T1,T2,D1,D2,ABX8OV,X8OV,T2TOL1,T2TOL2,VTAN,
      2 D2RAT,DELTHT,DL4I,DL8III,THTLNR,JASH,JOSH,KOSH,JECK,JOCK
      3 ,JMPTHT,JMPCRT,CRTLNR,LNORM
       COMMON/CMN7/EO,PO,ET,PT,EF,PF,PHI1R,PHI2R,DLV1,DLV2,DN1R,WMBR,
      1 PSIA1,PSIA2,ASX8OV,MAXDIF,ICNTMX,D2TMN,DL8I,XDAR1,XDARP1,JMP2
      2 ,COASTN,IMPTRD,JMP1,ITPRM,KOGMAX,KOGMIN,REGLAR
      3 ,AX8VNR,DL81NR,DL82NR,DL83NR
       COMMON/CMN8/IDNWRT
       COMMON/CMN9/DURTOL
       COMMON/CMPH/KAK(3),PAR(3),HOM(3,3),EPS(3),C(3),DLT2,JPH
       COMMON/CMN10/ICNT,WB(3),ERRMIN,KEY1
       COMMON/CMN11/POLAN2
       COMMON/CMN12/NOROT,PRAYER,NOCOAS,SIGMA9,JOEMAX,TST3MX
       COMMON/NAMCOM/  NFNTR,NPRM,NZIP,NMIN,NFNEW,NTHT,NNU,NCOAST,NPSI,
      1 NPRHM,NDRN,NY7SET,NWRT,NFFR,NEX1,NEX2,NEX3,NEX4,NEX5,NEX6,NEX7
       DIMENSION TEST(3),TDELT(3),DSTVIN(3,3)
       REAL MAXDIF,MISS(3),KAK,MIS3MN
       LOGICAL LBAK,TGTP,LNORM,REGLAR,NOROT,NOCOST,NOCOAS
       NOROT=ABS(ABS(PSIA1)-ABS(PSIA2)).LT.90.
       Y60=1.
       PIE=3.1415926/180.
       IF(.NOT.NOROT)GO TO 30
       ASX8OV=AX8VNR
       DL8I=DL81NR
       DL8II=DL82NR
       DL8III=DL83NR
   30  PHI1=PHI1R*PIE
       PHI2=PHI2R*PIE
       IF(DN1R.GT.360.)DN1R=DN1R-360.
       DN1=DN1R*PIE
       WMB=WMBR*PIE
       JOE=0
       MIS3MN=1.
C COMPUTE THETA
       DN2=WMB-DN1
       X5T2=EXP(DLV1/CR)
       T1=CR/AR*(1.-1./X5T2)
       X4A1=PHI1-DN1/2.
       EA1=(EO+ET)/2.
       PA1=(PO+PT)/2.
       VA1=(1.+EA1*COS(X4A1))**2/PA1**1.5
       DL41=VA1*T1-DN1
```

```
      TRANM1=X4A1+DL41/2.
      X4A2=PHI2-DN1-DN2/2.
      EA2=(ET+EF)/2.
      PA2=(PT+PF)/2.
      VA2=(1.+EA2*COS(X4A2))**2/PA2**1.5
      DLT2=CR/AR*(1./X5T2-1./(X5T2*EXP(DLV2/CR)))
      DL42=VA2*DLT2-DN2
      TRANM2=X4A2-DL42/2.
      E1 = 2.*ATAN(SQRT((1.-ET)/(1.+ET))*TAN(TRANM1/2.))
      E2 = 2.*ATAN(SQRT((1.-ET)/(1.+ET))*TAN(TRANM2/2.))
      IF(E1.LT.0.)E1=E1+2.*PIE*180.
      IF(E2.LT.0.)E2=E2+2.*PIE*180.
      IF(E1.GT.E2)E2=E2+2.*PIE*180.
      T2=T1+(PT/(1.-ET*ET))**1.5*(E2-E1-ET*(SIN(E2)-SIN(E1)))
      T3=T2+DLT2
      TB=T3+.5
      XD(2)=(T2-T1)/COASTN
      XD(4)=XD(2)
      IF(.NOT.REGLAR)GO TO 35
      IFRD=1
      IF(DLV1.GT.DLV2)IFRD=0
      LBAK=DLV1.GT.DLV2
   35 IF(IFRD.EQ.0)GO TO 40
      IF(ABS(PSIA1).GT.270.)PSIA1=SIGN(ABS(PSIA1)-360.,PSIA1)
      X8OV  =SIGN(ASX8OV,ABS(PSIA1)-90.)
      THETA=X4A1-DL41/2.
      WB(1)=PF
      WB(2)=EF
      WB(3)=WMB
      POLAN2=TRANM2+DN1
      P1=PSIA1*PIE
      P2=(PSIA1+.1)*PIE
      Y50=1.
      D1=T1
      D2=DLT2
      D=T1
      PA=P0
      EA=E0
      GO TO 50
   40 IF(ABS(PSIA2).GT.270.)PSIA2=SIGN(ABS(PSIA2)-360.,PSIA2)
      P1=PSIA2*PIE
      P2=(PSIA2+.1)*PIE
      X8OV  =SIGN(ASX8OV,ABS(PSIA2)-90.)
      THETA=X4A2+DL42/2.
      WB(1)=P0
      WB(2)=E0
      WB(3)=-WMB
      POLAN2=TRANM1-DN2
      D1=DLT2
      D2=T1
      D=DLT2
      T2=DLT2+(T2-T1)
      T1=DLT2
      T3=T2+D2
      Y50=EXP((DLV1+DLV2)/CR)
      PA=PF
      EA=EF
```

```
   50 THETDG=THETA/PIE
      JAR=-1
      XDMIN=XDAR1+(XDARP1-XDAR1)/(.1-1.)*(AR-1.)
      IF(AR.GT.1.)XDMIN=XDAR1/AR
      IF(AR.LT..1)XDMIN=AMAX1(.1*XDARP1/AR,XD(2))
      WRITE(6,1020)XDMIN
      GO TO 300
   60 WRITE(6,1018)
      WRITE(6,1001)(XD(I),I=1,4)
      WRITE(6,1009)
      WRITE(6,1022)T1,T2,T3,THETDG
C    COMPUTE U,VTAN,H,PR
      D2DIF=1.
      Y(1)=PA
      Y(2)=EA
      Y(4)=THETA
      C4=COS(Y(4))
      H=(1.+Y(2)*C4)/Y(1)
      PR=SQRT(Y(1))
      VTAN=H*H*PR
      ABX8OV=ABS(X8OV)
      CRT=X8OV*VTAN
      WRITE(6,1101)CRT,VTAN
      TGTP=SIN(Y(4)).LT.0.
      DL4=DL4I
      IF(TGTP)DL4=-DL4
      DL8=DL8I
      CPRO=2.
      ERRTMP=1.
      ERRMIN=1.
      JAW=0
      KOG=0
      ICNT=0
      CRTLNR=-1.
      JOSH=0
      JASH=0
      JOCK=0
      JECK=0
      LNORM=.TRUE.
      JUMP=JMP1
      D2TOL=10.*D2TMN
      JOGO=0
   90 CALL PSISET($940,$950)
      CALL NUSET($930)
      JUMP=JMP2
  200 ICNT=ICNT+1
      WRITE(6,1030)ICNT
  205 THETDG=THETA/PIE
      P2DG=P2/PIE
      Y3DG=Y(3)/PIE
      CRVL=CR*ALOG(Y(5))
      WRITE(6,1003)
      WRITE(6,1025)THETDG,P2DG,CRT,Y(1),Y(2),Y3DG,CRVL,TP(1),TP(2),TP(3)
      DO 210 I=1,3
  210 MISS(I)=WB(I)-Y(I)
      WRITE(6,1026)(MISS(I),I=1,3)
      DO 212 I=1,3
```

```
212 TEST(I)=ABS(MISS(I))
    ERROR=TEST(1)+TEST(2)+TEST(3)
    WRITE(6,1038)ERROR
    IF(ERROR.GT.ERRMIN)GO TO 215
    ERRMIN=ERROR
    Y4MIN=THETA
    Y7MIN=P2
    Y8MIN=CRT
215 IF(JOE.GT.JOEMAX)GO TO 220
    IF(ICNT.EQ.1.AND.TEST(3).GT.TST3MX)GO TO 216
    GO TO 220
216 IF(JAW)500,500,510
220 IF((TEST(1).LT.MAXDIF).AND.(TEST(2).LT.MAXDIF)
   1 .AND.(TEST(3).LT.MAXDIF))GO TO 930
    IF(ICNT.GT.ICNTMX)GO TO 900
    GO TO 700
300 KINT=IFIX(T1/XDMIN)
    IF(KINT.LT.4)GO TO 310
    XD(1)=T1/(FLOAT(KINT)+1.5)
    GO TO 320
310 XD(1)=T1/5.5
320 KINT=IFIX(D2/XDMIN)
    IF(KINT.LT.4)GO TO 330
    XD(3)=D2/(FLOAT(KINT)+1.5)
    GO TO 340
330 XD(3)=D2/5.5
340 GO TO 60
500 JAW=1
    THET1=THETA
    DLPHI1=MISS(3)
    THETA=THETA+.01
    GO TO 520
510 SLOPE=(MISS(3)-DLPHI1)/(THETA-THET1)
    JOE=JOE+1
    IF(ABS(MISS(3)).GT.MIS3MN)GO TO 515
    MIS3MN=ABS(MISS(3))
    THET1=THETA
515 THET1=THETA
    DLPHI1=MISS(3)
    THTCHA=-MISS(3)/SLOPE
    THTCHA=SIGN(AMIN1(.05/AR,ABS(THTCHA)),THTCHA)
    THETA=THETA+THTCHA
    THETDG=THETA/PIE
    WRITE(6,1039)THETDG
    IF(JOE.GT.JOEMAX)THETA=THTMIN
    JOGO=0
520 CALL PSISET($900,$960)
    CALL NUSET($900)
    GO TO 205
700 KAK(1)=Y4MIN
    KAK(2)=Y7MIN
    KAK(3)=Y8MIN
    ICNT=ICNTMX-ITPRM
    CALL PRMITR
    RETURN
800 WRITE(6,1023)
    RETURN
```

```
  900 WRITE(6,1031)ICNT,ITPRM
      GO TO 700
  930 NOCOST=NOCOAS
      NOCOAS=.TRUE.
      CALL ZIPPO(0,4,$800,$800)
      NOCOAS=NOCOST
  940 RETURN
  950 JOGO=JOGO+1
      IF(JOGO.GT.10)GO TO 940
      ABX8OV=ABX8OV-FLOAT(JOGO)*ABS(DL8I)
      X8OV=SIGN(ABX8OV,X8OV)
      CRT=X8OV*VTAN
      WRITE(6,1101)CRT,VTAN
      GO TO 90
  960 JOGO=JOGO+1
      IF(JOGO.GT.10)GO TO 930
      ABX8OV=ABX8OV-FLOAT(JOGO)*ABS(DL8I)
      X8OV=SIGN(ABX8OV,X8OV)
      CRT=X8OV*VTAN
      WRITE(6,1101)CRT,VTAN
      GO TO 520
 1001 FORMAT(1H ,9F14.7)
 1003 FORMAT(1X,12HTRUE ANOMALY,2X,12HTHRUST ANGLE,6X,3HCRT,4X,
     1 12HSMI-LAT RCTM,2X,12HECCENTRICITY,  2X,11HARG PERCNTR,2X,
     2 10HCHAR VLCTY,6X,5HTP(1),8X,5HTP(2),8X,5HTP(3))
 1009 FORMAT(1H0,6X,2HT1,13X,2HT2,13X,2HT3,13X,5HTHETA)
 1017 FORMAT(45X,17HD2TOL(260 MAIN) =F11.8)
 1018 FORMAT(1H0,7X,5HXD(1),9X,5HXD(2),9X,5HXD(3),9X,5HXD(4))
 1020 FORMAT(1X,7HXDMIN =F11.8)
 1021 FORMAT(1X,40HTIME INCREMENTS DECREASED BY FACTOR OF 2)
 1022 FORMAT(1H ,6F15.8)
 1023 FORMAT(1H0,19HSTOP AT 800 OF MAIN)
 1025 FORMAT(1X,10F13.8)
 1026 FORMAT(1X,15H(TEST(I),I=1,3),3F14.8)
 1027 FORMAT(5X,8HTDELT(1),6X,8HTDELT(2),6X,8HTDELT(3))
 1028 FORMAT(9X,2HT1,12X,2HT2,12X,2HT3)
 1029 FORMAT(1X,83HVARIATION OF THRUST ANGLE REQUIRES THAT TIME INCREMEN
     1TS BE REDUCED BY A FACTOR OF 2)
 1030 FORMAT(1X,18HITERATION NUMBER =I2)
 1031 FORMAT(1X,21HNO CONVERGENCE AFTER I2,16H ITERATIONS-TRY I2,
     1 35H ITERATIONS ON THE THREE PARAMETERS)
 1035 FORMAT(70X,3F20.8)
 1036 FORMAT(91X,17HTHE MATRIX DSTVAR)
 1037 FORMAT(91X,17HTHE MATRIX DSTVIN)
 1038 FORMAT(1X,7HERROR =F11.8)
 1039 FORMAT(1X,17HTHETA(515 MAIN) =F13.8)
 1101 FORMAT(1X,20HMAINB-(40-),CRT,VTAN,2F15.8)
      END
```

```
C                          SUBROUTINE PSISET
C
C
C       THE SUBROUTINE WHICH COMPUTES THE INITIAL VALUE OF THE THRUST
C       ORIENTATION ANGLE SO THAT THE DESIRED DURATION OF THE FIRST
C       THRUSTING INTERVAL IS ATTAINED
C
C
$IBFTC PSIS     XR7,M94,DECK
       SUBROUTINE PSISET(*,*)
       COMMON/CMN1/AR,CR,PA,EA,THETA,CRT,DBLDOT,CPRO,PIE,P1,P2,PSI,DRN,D,
      1 Y50,Y60
       COMMON/CMN2/Y(9),P(9),XD(4),IWRT(4),TP(4),X,DX,N,TB,IP1,LN,IFRD
      1,DSTVAR(3,3)
       COMMON/CMN3/D2DIF,D1TOL1,D1TOL2,D2TOL,DL4,DL7,DL8,JAMP,JG,
      1 DL4II,DL8II,LBAK,TGTP,T1,T2,D1,D2,ABX80V,X80V,T2TOL1,T2TOL2,VTAN,
      2 D2RAT,DELCRT,DL4I,DL8III,THTLST,JASH,JOSH,KOSH,JECK,JOCK
      3 ,JMPTHT,JMPCRT,CRTLNR
       LOGICAL LBAK,TGTP
       DIMENSION PZSI(2),DIFRD1(2)
       JUMP=JMPTHT
       JMPOUT=JUMP+JMPTHT
       IZ=1
       D1MN=1.
   40  CALL Y7SET($800)
   45  CALL ZIPPO(1,1,$500,$600)
       P2DG=P2/PIE
       WRITE(6,1002)P2DG
       D1DIF=D1-TP(1)
       IF(ABS(D1DIF).GT.D1MN)GO TO 50
       D1MN= ABS(D1DIF)
       D1DF=D1DIF
       P2MN=P2
   50  WRITE(6,1013)D1DIF
       IF((ABS(D1DIF).LT.D1TOL1).AND.(ABS(D2DIF).GT.D2RAT*D2))RETURN
       IF(ABS(D1DIF).LT.D1TOL2)RETURN
       IF(IZ.GT.1)GO TO 400
       PZSI(1)=P2
       DIFRD1(1)=D1DIF
   60  IZ=IZ+1
       IF(IZ.GT.JUMP)GO TO 900
       CPRO=(1.-D1DIF/D1)*CPRO
       GO TO 40
  400  IF(IZ.GT.2)GO TO 410
       PZSI(2)=P2
       DIFRD1(2)=D1DIF
       GO TO 60
  410  PZSI(1)=PZSI(2)
       DIFRD1(1)=DIFRD1(2)
       PZSI(2)=P2
       DIFRD1(2)=D1DIF
       GO TO 60
  500  IF(Y(9).GT.0.)GO TO 520
       WRITE(6,1000)
       RETURN 1
  520  WRITE(6,1001)
  600  RETURN 2
```

```
  800 IF(Y(9).GT.0.)GO TO 820
      WRITE(6,1005)
      RETURN 2
  820 WRITE(6,1006)
      RETURN 2
  900 SLOPE=(PZSI(2)-PZSI(1))/(DIFRD1(2)-DIFRD1(1))
      P2=PZSI(2)-DIFRD1(2)*SLOPE
      IF(IZ.GT.JMPOUT)GO TO 950
      GO TO 45
  950 IF(D1MN.LT.100.*D1TOL2)GO TO 960
      WRITE(6,1050)
      RETURN 1
  960 P2=P2MN
      P2DG=P2/PIE
      WRITE(6,1007)P2DG
      WRITE(6,1008)D1DF
      RETURN
 1000 FORMAT(1H0,21HSTOP AT 500 OF PSISET)
 1001 FORMAT(1H0,21HSTOP AT 420 OF PSISET)
 1002 FORMAT(1X,16HP2(40  PSISET) =F13.8)
 1005 FORMAT(1H0,21HSTOP AT 800 OF PSISET)
 1006 FORMAT(1H0,21HSTOP AT 820 OF PSISET)
 1007 FORMAT(1X,16HP2(960 PSISET) =F13.8)
 1008 FORMAT(1X,19HD1DIF(960 PSISET) =F11.8)
 1013 FORMAT(1X,19HD1DIF(40  PSISET) =F11.8)
 1050 FORMAT(1H0,21HSTOP AT 950 OF PSISET)
      END
```

```
C                          SUBROUTINE NUSFT
C
C
C       THE SUBROUTINE WHICH COMPUTES THE INITIAL VALUE OF THE
C       VARIABLE 'NU' SO THAT THE DESIRED DURATION OF THE SECOND
C       THRUSTING INTERVAL IS ATTAINED
C
C
$IBFTC NUST     XR7,M94,DECK
       SUBROUTINE NUSET(*)
       COMMON/CMN1/AR,CR,PA,EA,THETA,CRT,DBLDOT,CPRO,PIE,P1,P2,PSI,DRN,D,
      1 Y50,Y60
       COMMON/CMN2/Y(9),P(9),XD(4),IWRT(4),TP(4),X,DX,N,TB,IP1,LN,IFRD
      1,DSTVAR(3,3)
       COMMON/CMN3/D2DIF,D1TOL1,D1TOL2,D2TOL,DL4,DL7,DL8,JUMP,JG,
      1 DL4II,DL8II,LBAK,TGTP,T1,T2,D1,D2,ABX80V,X80V,T2TOL1,T2TOL2,VTAN,
      2 D2RAT,DELTHT,DL4I,DL8III,THTLNR,JASH,JOSH,KOSH,JECK,JOCK
      3 ,JMPTHT,JMPCRT,CRTLNR,LNORM
       COMMON/CMN12/NOROT,PRAYER,NOCOAS,SIGMA9,JOEMAX,TST3MX
       COMMON/NAMCOM/  NFNTR,NPRM,NZIP,NMIN,NFNEW,NTHT,NNU,NCOAST,NPSI,
      1 NPRHM,NDRN,NY7SET,NWRT,NFFR,NEX1,NEX2,NEX3,NEX4,NEX5,NEX6,NEX7
       DIMENSION CRTN(2),D2DIFN(2),CD(3)
       LOGICAL LNR,NOSTEP
       LOGICAL NOROT
    2  JILL=0
       IF(CRTLNR.LT.0.)GO TO 600
    5  IPO=0
       KEYLNR=0
       IPI=0
       IF(JASH.GT.0)GO TO 10
       ID=0
   10  IZ=1
       D2DFMN=1.
       JIMP=JUMP
       TMPCRT=CRT
       IF(JASH.LT.1)GO TO 20
       IF(JOSH.LT.2)GO TO 15
       CRT=CRT+SNSTVT*DELTHT
       ABX80V=ABS(CRT/VTAN)
       WRITE(6,1013)CRT
       DL8=DL8III
       JIMP=JMPCRT
   15  CALL PSISET($820)
   20  CALL ZIPPO(JG,4,$800,$800)
       IZ=IZ+1
       IF(IZ.GT.JIMP)GO TO 900
   40  D2A=TP(3)-TP(2)
       D2DIF=D2-D2A
       WRITE(6,1005)D2DIF
       IF( ABS(D2DIF).GT.D2DFMN)GO TO 45
       D2DFMN=ABS(D2DIF)
       D2DF=D2DIF
       P2MN=P2
       CRTMN=CRT
       THTMN=THETA
   45  JAG=0
       IF(LNRPOS.EQ.0.OR.LNRNEG.EQ.0)GO TO 55
```

```
      IF(ABS(D2DIF).LT.D2TOL)GO TO 300
      LNR=D2DIF.LT.DIFPOS*D2RAT
      IF(D2DIF.LT.0.)LNR=D2DIF.GT.DIFNEG*D2RAT
      IF(.NOT.LNR.AND.KEYLNR.EQ.1)GO TO 700
      IF(LNR)GO TO 400
   50 IF(LNR.AND.IPO.EQ.0.AND.JOCK.GT.0)GO TO 500
   55 IF(D2DIF.LT.0.)GO TO 200
C  DECREASE ABS(CRT)
      CRTHI=CRT
      D2HI =D2A
      IF(JILL.EQ.1)GO TO 100
      IM=0
      IF(LNRPOS.EQ.0)GO TO 620
   60 CRTHI=CRT
      D2HI=D2A
      IM=IM+1
      IF(IM.GT.JUMP)GO TO 900
      IF(IM.GT.1)DL8=2.*DL8
   62 ABX80V=ABX80V-DL8
      X80V=SIGN(ABX80V,X80V)
      CRT=X80V*VTAN
      WRITE(6,1001)CRT
      JOGO=0
   64 CALL PSISET($820,$920)
      CALL ZIPPO(JG,4,$800,$800)
      D2A=TP(3)-TP(2)
      D2DIF=D2-D2A
      WRITE(6,1006)D2DIF
      IF(ABS(D2DIF).GT.D2DFMN)GO TO 65
      D2DFMN=ABS(D2DIF)
      D2DF=D2DIF
      P2MN=P2
      CRTMN=CRT
      THTMN=THETA
   65 JAG=-1
      IF(LNRPOS.EQ.0.OR.LNRNEG.EQ.0)GO TO 70
      IF(ABS(D2DIF).LT.D2TOL)GO TO 300
      LNR=D2DIF.LT.DIFPOS*D2RAT
      IF(D2DIF.LT.0.)LNR=D2DIF.GT.DIFNEG*D2RAT
      IF(LNR)GO TO 400
   70 IF(D2DIF.LT.0.)GO TO 80
      GO TO 60
   80 CRTLO=CRT
      D2LO=D2A
      JILL =1
      IF(LNRNEG.EQ.0)GO TO 640
  100 CRT=CRTHI+(CRTLO-CRTHI)/(D2LO-D2HI)*(D2-D2HI)
      ABX80V=ABS(CRT)/VTAN
      WRITE(6,1002)CRT
      CALL PSISET($820,$820)
      GO TO 20
C  INCREASE ABS(CRT)
  200 IF(LNRNEG.EQ.0)GO TO 660
  210 CRTLO=CRT
      D2LO=D2A
      IF(JILL.EQ.1)GO TO 260
      IM=0
```

```
220 CRTLO=CRT
    D2LO=D2A
    IM=IM+1
    IF(IM.GT.JUMP)GO TO 900
    IF(IM.GT.1)DL8=2.*DL8
222 ABX8OV=ABX8OV+DL8
    X8OV=SIGN(ABX8OV,X8OV)
    CRT=X8OV*VTAN
    WRITE(6,1003)CRT
    JOGO=0
224 CALL PSISET($820,$940)
    CALL ZIPPO(JG,4,$800,$800)
    D2A=TP(3)-TP(2)
    D2DIF=D2-D2A
    WRITE(6,1007)D2DIF
    IF(ABS(D2DIF).GT.D2DFMN)GO TO 225
    D2DFMN=ABS(D2DIF)
    D2DF=D2DIF
    P2MN=P2
    CRTMN=CRT
    THTMN=THETA
225 JAG=1
    IF(LNRPOS.EQ.0.OR.LNRNEG.EQ.0)GO TO 230
    IF(ABS(D2DIF).LT.D2TOL)GO TO 300
    LNR=D2DIF.LT.DIFPOS*D2RAT
    IF(D2DIF.LT.0.)LNR=D2DIF.GT.DIFNEG*D2RAT
    IF(LNR)GO TO 400
230 IF(D2DIF.GT.0.)GO TO 240
    GO TO 220
240 CRTHI=CRT
    D2HI=D2A
    JILL =1
    IF(LNRPOS.EQ.0)GO TO 680
260 CRT=CRTLO+(CRTHI-CRTLO)/(D2HI-D2LO)*(D2-D2LO)
    ABX8OV=ABS(CRT)/VTAN
    WRITE(6,1004)CRT
    CALL PSISET($820)
    GO TO 20
300 DL8=DL8II
    IF(JASH.LT.1)RETURN
    DELCRT=CRT-TMPCRT
    SNSTVT =DELCRT/DELTHT
    WRITE(6,1010)SNSTVT
    RETURN
400 IPI=IPI+1
    IF(IPI.GE.3)GO TO 410
    D2DIFN(IPI)=D2DIF
    CRTN(IPI)=CRT
    IF(IPI.GE.2)GO TO 420
    IF(JAG)70,50,230
410 D2DIFN(1)=D2DIFN(2)
    CRTN(1)=CRTN(2)
    D2DIFN(2)=D2DIF
    CRTN(2)=CRT
    GO TO 440
420 IF(JOSH.GT.0)GO TO 440
    ID=ID+1
```

```
      IF(ID.GT.2)GO TO 440
      CD(ID)=(CRTN(2)-CRTN(1))/(D2DIFN(2)-D2DIFN(1))
      JOCK=1
      CRTOD2=0.
      DO 430 I=1,ID
  430 CRTOD2=CRTOD2+CD(I)/FLOAT(ID)
      WRITE(6,1015)CRTOD2
  440 IF(ABS(D2DIFN(1)).LT.ABS(D2DIFN(2)))GO TO 450
      TEMCRT=CRT
      CRT=CRT+(CRTN(2)-CRTN(1))/(D2DIFN(2)-D2DIFN(1))*(-D2DIFN(2))
      GO TO 460
  450 TEMCRT=CRTN(1)
      CRT=CRTN(1)+(CRTN(1)-CRTN(2))/(D2DIFN(1)-D2DIFN(2))*(-D2DIFN(1))
  460 DELCRT=CRT-TEMCRT
      FIXER=ABS(DL8II)
      DELCRT=SIGN(AMIN1(ABS(DELCRT),FIXER*VTAN*10.),DELCRT)
      CRT=TEMCRT+DELCRT
      ABX80V=ABS(CRT)/VTAN
      WRITE(6,1008)CRT
      KEYLNR=1
      CALL PSISET($820,$820)
      GO TO 20
  500 TEMCRT=CRT
      CRT=CRT-CRTOD2*D2DIF
      DELCRT=CRT-TEMCRT
      DELCRT=SIGN(AMIN1(ABS(DELCRT),DL8II*VTAN*10.),DELCRT)
      CRT=TEMCRT+DELCRT
      ABX80V=ABS(CRT/VTAN)
      WRITE(6,1014)CRT
      IPO=1
      GO TO 15
  600 CRTLNR=1.
      LNRPOS=0
      LNRNEG=0
      GO TO 5
  620 LNRPOS=1
      DIFPOS=D2DIF
      GO TO 60
  640 LNRNEG=1
      DIFNEG=D2DIF
      GO TO 100
  660 LNRNEG=1
      DIFNEG=D2DIF
      GO TO 210
  680 LNRPOS=1
      DIFPOS=D2DIF
      GO TO 260
  700 DELCRT=DELCRT/1.25
      CRT=TEMCRT+DELCRT
      WRITE(6,1016)CRT
      CALL PSISET($820,$820)
      GO TO 20
  800 WRITE(6,1000)
  820 RETURN 1
  900 IF(D2DFMN.LT.100.*D2TOL)GO TO 910
      WRITE(6,1050)
      RETURN 1
```

```
  910 P2=P2MN
      THETA=THTMN
      CRT=CRTMN
      WRITE(6,1011)CRT
      ABX80V=ABS(CRT)/VTAN
      WRITE(6,1012)D2DF
      CALL ZIPPO(JG,4,$800,$800)
      GO TO 300
  920 JOGO=JOGO+1
      ABX80V=ABX80V*DL8/2.**JOGO
      IF(JOGO.GT.5)GO TO 820
      X80V=SIGN(ABX80V,X80V)
      CRT=X80V*VTAN
      WRITE(6,1017)CRT
      GO TO 64
  940 JOGO=JOGO+1
      IF(JOGO.GT.5)GO TO 820
      ABX80V=ABX80V-DL8/2.**JOGO
      X80V=SIGN(ABX80V,X80V)
      CRT=X80V*VTAN
      WRITE(6,1018)CRT
      GO TO 224
 1000 FORMAT(45X,21HSTOP AT 800 OF CRTSET)
 1001 FORMAT(45X,17HCRT(60  CRTSET) =F11.8)
 1002 FORMAT(45X,17HCRT(100 CRTSET) =F11.8)
 1003 FORMAT(45X,17HCRT(220 CRTSET) =F11.8)
 1004 FORMAT(45X,17HCRT(260 CRTSET) =F11.8)
 1005 FORMAT(45X,19HD2DIF(40  CRTSET) =F11.8)
 1006 FORMAT(45X,19HD2DIF(60  CRTSET) =F11.8)
 1007 FORMAT(45X,19HD2DIF(220 CRTSET) =F11.8)
 1008 FORMAT(45X,20HCRT(440 OF CRTSET) =F11.8)
 1009 FORMAT(45X,16HCRT(20 CRTSET) =F11.8)
 1010 FORMAT(45X,20HSNSTVT(300 CRTSET) =F13.8)
 1011 FORMAT(45X,17HCRT(910 CRTSET) =F11.8)
 1012 FORMAT(45X,19HD2DIF(910 CRTSET) =F11.8)
 1013 FORMAT(45X,17HCRT(15  CRTSET) =F11.8)
 1014 FORMAT(45X,17HCRT(500 CRTSET) =F11.8)
 1015 FORMAT(45X,20HCRTOD2(430 CRTSET) =F11.8)
 1016 FORMAT(45X,17HCRT(700 CRTSET) =F11.8)
 1017 FORMAT(45X,17HCRT(920 CRTSET) =F11.8)
 1018 FORMAT(45X,17HCRT(940 CRTSET) =F11.8)
 1050 FORMAT(45X,21HSTOP AT 900 OF CRTSET)
      END
```

```
C                          SUBROUTINE MIMIN
C
C
C       THE SUBROUTINE WHICH SUPPLIES THE FIXED STEPSIZE FOURTH-ORDER
C       RUNGE-KUTTA INTEGRATION ALGORITHM
C
C
$IBFTC MIMGRL  XR7,M94,DECK
       SUBROUTINE MIMIN(SN)
       COMMON/CMN2/Y(9),P(9),XD(4),IWRT(4),TP(4),X,DX,N,TB,IP1,LM,LFRD
       DIMENSION XK(9,3),YO(9)
       XF=FRN(X+DX)
       DXMAX=XD(IP1)
       Y(1)=1./SQRT(Y(1))
       ZE=Y(1)*Y(2)
       PHI=Y(3)
       Y(4)=Y(4)+Y(3)
       Y(2)=ZE*COS(Y(3))
       Y(3)=ZE*SIN(Y(3))
  300  H=XF*SN
       XO=X
  305  H=SN*(FRN(XO+AMIN1(SN*H,DXMAX,SN*DX))-XO)
       DO 310 I=1,N
  310  YO(I)=Y(I)
       CALL FNEW
       X=X+.5*H
       DO 330 J=1,2
       DO 320 I=1,N
       XK(I,J)=H*P(I)
  320  Y(I)=YO(I)+.5*XK(I,J)
  330  CALL FNEW
       X=X+.5*H
       DO 340 I=1,N
       XK(I,3)=H*P(I)
  340  Y(I)=YO(I)+XK(I,3)
       CALL FNEW
       DO 350 I=1,N
  350  Y(I)=YO(I)+(XK(I,1)+2.*(XK(I,2)+XK(I,3))+H*P(I))/6.
       X=XO+H
       XO=X
       IF(SN)360,360,370
  360  IF(XO-XF)500,500,380
  370  IF(XF-XO)500,500,380
  380  IF(SN)390,390,400
  390  IF(XO+H-XF)410,410,305
  400  IF(XF-XO-H)410,410,305
  410  H=XF-XO
       GO TO 305
  500  Y(3)=ATAN2(Y(3),Y(2))
       Y(2)=Y(2)/(Y(1)*COS(Y(3)))
       Y(1)=1./Y(1)**2
       Y(4)=Y(4)-Y(3)
       RETURN
       END
        ENTRY   FRN
$IBMAP ROUN    DECK
FRN    TRA     1,4
       END
```

```
$IBFTC DRNDBL  XR7,M94
C                              SUBROUTINE DURATN
C
C
C       THE SUBROUTINE WHICH COMPUTES THE FIRST AND SECOND DERIVATIVES
C       OF THE SWITCHING FUNCTION.  THE DERIVATIVES ARE USED IN THE
C       ITERATIVE SOLUTION OF EQ. (3-44).  DOUBLE-PRECISION
C       ARITHMETIC IS EMPLOYED.
C
C
      SUBROUTINE DURATN(PSI,DRN)
      LOGICAL NOROT
      DOUBLE PRECISION Y(9),P(9),PSI,C4,C7,S4,C24,S7,H,PR,LN,A,B,U,
     1 VTAN,AD,AR,CR,UD,HD,DBLDOT,SGN,DRN,CPRO,TRANG
      COMMON/CMN1/ARS,CRS,PA,EA,THETA,CRT,DBLDTS,CPROS,PIES,P1,P2,
     1 PSIS,DRNS,D,Y50,Y60
      COMMON/CMN2/Q(9),Z(9),XD(4),IWRT(4),TP(4),X,DX,N,TB,IP1,LS,IFRD
     1,DSTVAR(3,3)
      COMMON/CMN8/IDNWRT
      AR=DBLE(ARS)
      CR=DBLE(CRS)
      CPRO=DBLE(CPROS)
      PIE=DBLE(PIES)
      Y(1)=DBLE(PA)
      Y(2)=DBLE(EA)
      Y(3)=0.
      Y(4)=DBLE(THETA)
      Y(5)=DBLE(Y50)
      Y(6)=DBLE(Y60)
      Y(7)=PSI
      Y(8)=DBLE(CRT)
    5 Y(9)=0.
      C4=DCOS(Y(4))
      C7=DCOS(Y(7))
      S4=DSIN(Y(4))
      S7=DSIN(Y(7))
      H=(1.+Y(2)*C4)/Y(1)
      PR=DSQRT(Y(1))
      C24=DCOS(2.*Y(4))
C  COMPUTE LN
      A=-Y(8)/H+Y(6)/PR*(S7*(C4+Y(2)*C24)/S4-2.*C7*(1.+Y(2)*C4))
      B=-(1.+Y(2)*C4)/(H*Y(2)*S4)
      LN = A/B
C   COMPUTE DERIVATIVES OF STATE VARIABLES
   18 U=Y(2)*S4*H/PR
      VTAN = H*M*PR
   20 AD=AR*Y(5)
      P(1)=2.*AD*PR*C7/H
      P(2)=AD*PR*(S4*S7+C7/(1.+Y(2)*C4)*(2.*C4+Y(2)*(1.+C4**2)))
      P(3)=AD*PR/Y(2)*(S4*C7/(1.+Y(2)*C4)*(2.+Y(2)*C4)-C4*S7)
      P(4)=VTAN-P(3)
      P(5)=AD*Y(5)/CR
      P(6)=2.*U*Y(6)+Y(8)*S7-LN*C7
      P(7)=2.*VTAN+(Y(8)*C7+LN*S7)/Y(6)
      P(8)=Y(6)*(3.*H**3*S7-AD*H)
      P(9)=(Y(9)-H)/Y(6)*(LN*C7-Y(8)*S7)+H*U-Y(9)*(2.*U+AD/CR)
C   COMPUTE DDT
```

```
      UD=VTAN**2-U**2-H**3+AR*H*S7
      HD=-U*H
      DBLDOT=(LN*C7-Y(8)*S7)*((P(9)-HD)/Y(6)-(Y(9)-H)*P(6)/Y(6)**2)
     1 -(Y(9)-H)/Y(6)*(P(8)*S7+Y(8)*P(7)*C7+LN*P(7)*S7)+UD*H+U*HD
     2 -P(9)*(2.*U+AD/CR)-Y(9)*(2.*UD+AD*P(5)/(Y(5)*CR))
      SGN=-1.
      IF(IFRD.EQ.1)SGN=1.
      DRN=-CPRO*P(9)/DBLDOT*SGN
      IF(IDNWRT.GT.0)RETURN
      TRANG=PSI/PIE
      WRITE(6,1100)P(9),DBLDOT,DRN,CPRO,TRANG
      RETURN
1100  FORMAT(1X,21HP9,P9DOT,DRN,CPRO,Y70,5D22.12)
      END
```

```
C                          SUBROUTINE Y7SET
C
C
C       THE SUBROUTINE WHICH SUPPLIES THE LOGIC FOR THE ITERATIVE
C       SOLUTION OF EQ. (3-44).  DOUBLE-PRECISION ARITHMETIC IS EMPLOYED
C
C
$IBFTC Y7SETB  XR7,M94,DECK
       SUBROUTINE Y7SET(*)
       DOUBLE PRECISION D,D1,D2,P1,P2,D1T,P1T,SLOPE,P2DLT,DURTOL
       COMMON/CMN1/AR,CR,PA,EA,THETA,CRT,DBLDOT,CPRO,PIE,P1S,P2S,PSIS,
      1 DRNS,DS,Y50,Y60
       COMMON/CMN9/DURTOL
       ICNT=1
       P1=DBLE(P2S)+.00001
       P2=DBLE(P2S)
       D=DBLE(DS)
       CALL DURATN(P1,D1)
       CALL DURATN(P2,D2)
   10  ICNT=ICNT+1
       IF(ICNT.GT.30)GO TO 40
       IF(DABS(D2-D).LT.DABS(D1-D))GO TO 20
       D1T=D1
       P1T=P1
       D1=D2
       P1=P2
       D2=D1T
       P2=P1T
   20  SLOPE=(D2-D1)/(P2-P1)
       P2DLT=(D-D2)/SLOPE
       P2DLT=DSIGN(DMIN1(.02,DABS(P2DLT)),P2DLT)
       P1=P2
       D1=D2
       P2=P2+P2DLT
       CALL DURATN(P2,D2)
       IF(DABS(D2-D).GT.DURTOL)GO TO 10
       P2S=SNGL(P2)
       RETURN
   40  WRITE(6,1000)
       RETURN 1
 1000  FORMAT(1H ,39HTHIS THRUST DURATION CANNOT BE ATTAINED)
       END
```

```
C                          SUBROUTINE FNEW
C
C
C       THE SUBROUTINE WHICH COMPUTES THE RIGHT HAND SIDES OF
C       EQS. (2-100)-(2-109)
C
C
$IBFTC FNEWB   XR7,M94,DECK
       SUBROUTINE FNEW
       COMMON/CMN1/AR,CR,PA,EA,THETA,CRT,DBLDOT,CPRO,PIE,P1,P2,PSI,DRN,D,
      1 Y50,Y60
       COMMON/CMN2/Y(9),P(9),XD(4),IWRT(4),TP(4),X,DX,N,TB,IP1,LN,IFRD
       COMMON/CMN12/NOROT,PRAYER,NOCOAS,SIGMA9,JOEMAX,TST3MX
       REAL LN
       PHI=ATAN2(Y(3),Y(2))
       E=Y(2)/(Y(1)*COS(PHI))
       EFF=Y(4)-PHI
       CF=COS(EFF)
       SF=SIN(EFF)
       H=(1.+E*CF)*Y(1)**2
       U=H*E*Y(1)*SF
       V=H*H/Y(1)
       C4=COS(Y(4))
       C7=COS(Y(7))
       S4=SIN(Y(4))
       S7=SIN(Y(7))
       IF(IP1.EQ.1.OR.IP1.EQ.3)GO TO 20
   10  P(1)=0.
       P(2)=0.
       P(3)=0.
       P(4)=V
       P(5)=0.
       P(6)=2.*U*Y(6)+Y(8)*S7-LN*C7
       P(7)=2.*V+(Y(8)*C7+LN*S7)/Y(6)
       P(8)=3.*Y(6)*H**3*S7
       P(9)=(H-Y(9))/Y(6)*(Y(8)*S7-LN*C7)+H*U-Y(9)*(2.*U)
       IF(IFRD.EQ.1)RETURN
       DO 15 I=1,9
   15  P(I)=-P(I)
       RETURN
   20  AD=AR*Y(5)
       ZSQOH=Y(1)**2/H
       P(1)=-AD*ZSQOH*C7
       P(2)=AD*(C4*C7*(1.+ZSQOH)+S4*S7)
       P(3)=AD*(S4*C7*(1.+ZSQOH)-C4*S7)
       P(4)=V
       P(5)=AD*Y(5)/CR
       P(6)=2.*U*Y(6)+Y(8)*S7-LN*C7
       P(7)=2.*V+(Y(8)*C7+LN*S7)/Y(6)
       P(8)=Y(6)*(3.*H**3*S7-AD*H)
       P(9)=(H-Y(9))/Y(6)*(Y(8)*S7-LN*C7)+H*U-Y(9)*(2.*U+AD/CR)
       IF(IFRD.EQ.1)RETURN
       DO 25 I=1,9
   25  P(I)=-P(I)
       RETURN
       END
```

```
C                          SUBROUTINE WRTOUT
C
C
C       THE SUBROUTINE WHICH COMPUTES THE HAMILTONIAN AND PRINTS OUT
C       THE IMPORTANT CHARACTERISTICS OF THE FINITE THRUST MINIMUM
C       FUEL TRANSFER TRAJECTORY AS A FUNCTION OF TIME
C
C
$IBFTC WRTGRL  XR7,M94,DECK
       SUBROUTINE WRTOUT(IP)
       COMMON/CMN1/AR,CR,PA,EA,THETA,CRT,DBLDOT,CPRO,PIE,P1,P2,PSI,DRN,D,
      1 Y50,Y60
       COMMON/CMN2/Y(9),P(9),XD(4),IWRT(4),TP(4),X,DX,N,TB,IP1,LN,IFRD
       DIMENSION W(9)
       REAL LN
       IF(IP)10,100,100
   10  CALL F
       WRITE(6,1003)
       WRITE(6,1000)LN,P(9)
       WRITE(6,1008)
       GO TO 110
  100  CALL F
  110  C4=COS(Y(4))
       C7=COS(Y(7))
       S4=SIN(Y(4))
       S7=SIN(Y(7))
       H=(1.+Y(2)*C4)/Y(1)
       PR=SQRT(Y(1))
       U=Y(2)*S4*H/PR
       HO=U*(-Y(8)+LN*(1.+Y(2)*C4)/(Y(2)*S4)+H*Y(6)/PR*(S7/S4*(C4+Y(2)*
      1 COS(2.*Y(4)))-2.*C7*(1.+Y(2)*C4)))
       PSE=0.
       IF(Y(9).GE.0.)PSE=1.
       HAMLTN=HO+AR*PSE*Y(9)*Y(6)*Y(5)
       DO 120 I=1,9
  120  W(I)=Y(I)
       W(3)=W(3)/PIE
       W(4)=W(4)/PIE
       W(5)=CR*ALOG(W(5))
       W(7)=W(7)/PIE
       WRITE(6,1010)X,(W(I),I=1,9),P(9),HAMLTN
  200  RETURN
 1000  FORMAT(8F14.8)
 1003  FORMAT(1H ,6X,2HLN,11X,5HY9DOT)
 1008  FORMAT(1H ,6X,1HX,6X,4HY(1),6X,4HY(2),8X,4HY(3),9X,4HY(4),6X,4HCRV
      1L,5X,4HY(6),6X,4HY(7),8X,4HY(8),7X,4HY(9),8X,5HY9DOT,4X,6HHAMLTN)
 1010  FORMAT(1X,3F10.7,2F13.7,2F9.6,F12.6,F11.7,E13.5,F8.4,E13.5)
       END
```

```
C                          SUBROUTINE MINV
C
C
C       THE SUBROUTINE WHICH PERFORMS MATRIX INVERSION
C
C
$IBFTC MINVOR  DECK,XR7,M94
       SUBROUTINE MINV(A,B,N)
C
C            MATRIX INVERSION SUBROUTINE
C
C      CALLING SEQUENCE...
C
C      CALL MINV(A,B,N)
C
C         A IS THE INPUT MATRIX (DIMENSIONED N X N)
C         B WILL BE THE INVERSE MATRIX (DIMENSIONED N X N)
C         N IS THE DIMENSION OF A AND B
C
C      NOTE...
C      CALL MINV(A,A,N)
C         CAUSES THE MATRIX A TO BE REPLACED BY ITS INVERSE.
C
C      209 CELLS OF BLANK COMMON ARE USED.
C
       DIMENSION   A(N,N), B(N,N)
       COMMON    I, I1, I2, IP, J, J1, NN, PE, TPE
       COMMON    K(100), P(100)
       DATA    NMAX/ 100/
C
       NN=N
       IF (NN-NMAX) 10,10,500
 10    IF (NN) 500,500,20
 20    DO 30 I=1,NN
       K(I)=I
       DO 30 J=1,NN
 30    B(I,J)=A(I,J)
       DO 330 I=1,NN
       I2=NN-I+1
       PE=0.
       DO 120 I1=1,I2
       TPE=B(I1,1)
       IF (ABS(PE)-ABS(TPE)) 100,100,120
 100   PE=TPE
       IP=I1
 120   CONTINUE
       IF (PE) 160,510,160
 160   DO 170 J=2,NN
 170   P(J-1)=B(IP,J)/PE
       P(NN)=1.0/PE
       IP=K(IP)
       I2=0
       DO 310 J=1,NN
       I1=J-I2
       K(I1)=K(J)
       IF (K(J)-IP) 260,250,260
 250   I2=1
```

```
      GO TO 310
260   TPE=-B(J,1)
      DO 300 J1=2,NN
300   B(I1,J1-1)=B(J,J1)+TPE*P(J1-1)
      B(I1,NN)=TPE*P(NN)
310   CONTINUE
      DO 320 J=1,NN
320   B(NN,J)=P(J)
330   K(NN)=IP
      DO 410 I=1,NN
      DO 400 J=1,NN
      I1=K(J)
400   P(I1)=B(I,J)
      DO 410 J=1,NN
410   B(I,J)=P(J)
      RETURN
500   WRITE (6,1000) NN
      CALL FXEM(500)
510   WRITE (6,1001)
      CALL FXEM(510)
      RETURN
1000  FORMAT (3H0N=,I12,33H IS INCORRECT FOR SUBROUTINE MINV)
1001  FORMAT (40HOSUBROUTINE MINV FINDS MATRIX A SINGULAR)
      END
```

```
C                          SUBROUTINE F
C
C
C       THE SUBROUTINE WHICH COMPUTES THE FIRST DERIVATIVE OF THE
C       SWITCHING FUNCTION S
C
C
$IBFTC FFR      XR7,M94,DECK
       SUBROUTINE F
       COMMON/CMN1/AR,CR,PA,EA,THETA,CRT,DBLDOT,CPRO,PIE,P1,P2,PSI,DRN,D,
      1 Y50,Y60
       COMMON/CMN2/Y(9),P(9),XD(4),IWRT(4),TP(4),X,DX,N,TB,IP1,LN,IFRD.
       REAL LN
       C4=COS(Y(4))
       C7=COS(Y(7))
       S4=SIN(Y(4))
       S7=SIN(Y(7))
       H=(1.+Y(2)*C4)/Y(1)
       PR=SQRT(Y(1))
       U=Y(2)*S4*H/PR
       V=PR*H*H
    20 AD=AR*Y(5)
       P(9)=(H-Y(9))/Y(6)*(Y(8)*S7-LN*C7)+H*U-Y(9)*(2.*U+AD/CR)
       IF(IFRD.EQ.1)RETURN
       P(9)=-P(9)
       RETURN
       END
```

```
C                          SUBROUTINE COAST
C
C
C       THE SUBROUTINE WHICH PERFORMS THE CANONICAL TRANSFORMATION TO
C       ELIMINATE INTEGRATION ON THE COAST INTERVAL
C
C
$IBFTC COASTE  XR7,M94,DECK
       SUBROUTINE COAST(Y,X,LN,C,THETA2,IFRD,*)
       DIMENSION Y(9)
       REAL LH,LU,LV,LN,LR,LUR,LUT,LA,LB,LZ,L,LMU,MU
       INTEGER COUNT
       PIE=3.1415926/180.
       PH1T=-1.
       JIG=0
       JACK=0
       COUNT=0
       SGN=1.
       FAC=1.
       P=Y(1)
       E=Y(2)
       PHI=Y(3)
       EFF=Y(4)
       MU=Y(5)
       PRIMA=Y(6)
       PSI=Y(7)
       CRT=Y(8)
       EF1=EFF
       THETA=PHI+EFF
       CF=COS(EFF)
       SF=SIN(EFF)
       H=(1.+E*CF)/P
       SQ=SQRT(P)
       U=H*E*SF/SQ
       V=H*H*SQ
       LH=(CRT+3.*LN*SF/E)/H
       DEL=H**6*E*E
       SP=SIN(PSI)
       CP=COS(PSI)
       LU=PRIMA*SP       +LN*(V*V-H**3)*V/DEL
       LV=PRIMA*CP       -LN*(V*V+H**3)*U/DEL
       LR=-H*H*LH-U*H*LU-V*H*LV+H**4*U*V/DEL*LN
       LUR=H*LU-(H*V**3-H**4*V)/DEL*LN
       LUT=H*LV+(H**4*U+H*V**2*U)/DEL*LN
       L=SQRT(LUR**2+LUT**2)
       LMU=-C*L/MU
       CT=COS(THETA)
       ST=SIN(THETA)
       Z=H*H/V
       A=Z*E*COS(PHI)
       B=Z*E*SIN(PHI)
       LA=-Z*CT/H**2*LR+ST*LUR+CT*LUT
       LB=-Z*ST/H**2*LR-CT*LUR+ST*LUT
       LZ=-(1./(Z*H)+Z/H**2)*LR+LUT
       THETA=THETA2
    10 CT=COS(THETA)
       ST=SIN(THETA)
```

```
      COUNT=COUNT+1
      IF(COUNT.GT.25)RETURN 1
      UT=Z+A*CT+B*ST
      LUR=LA*ST-LB*CT
      LUT=-Z*LZ/UT+(1.+Z/UT)*(LA*CT+LB*ST)
      L=SQRT(LUR**2+LUT**2)
      SP=LUR/L
      CP=LUT/L
      H1=L+LMU/C*MU
      THETDG=THETA/PIE
      IF(ABS(H1).LT..0000002.AND.PH1T.GT.0..AND.IFRD.EQ.1)GO TO 200
      IF(ABS(H1).LT..0000002.AND.PH1T.LT.0..AND.IFRD.EQ.0)GO TO 200
      PLURT=LA*CT+LB*ST
      PUTT=-A*ST+B*CT
      PLUTT=Z/UT**2*(LZ-LA*CT-LB*ST)*PUTT-(1.+Z/UT)*(LA*ST-LB*CT)
      PLT=(LUR*PLURT+LUT*PLUTT)/L
      PH1T=PLT
      IF(PH1T.LT.0..AND.IFRD.EQ.1)GO TO 100
      IF(PH1T.GT.0..AND.IFRD.EQ.0)GO TO 100
      JACK=1
      DELTHT=-H1/PH1T
      DELTHT=SIGN(AMIN1(5.*PIE,ABS(DELTHT)),DELTHT)
      THETA=THETA+DELTHT
      GO TO 10
  100 IF(JIG.GT.5)RETURN 1
      IF(JACK.GT.0)RETURN 1
      SGN=-SGN
      IF(JIG.GT.1)FAC=2.
      IF(JIG.GT.3)FAC=4.
      JIG=JIG+1
      THETA=THETA2+FAC*SGN*PIE*18.
      GO TO 10
  200 EFF=THETA-PHI
      CF=COS(EFF)
      SF=SIN(EFF)
      H=(1.+E*CF)/P
      U=H*E*SF/SQ
      V=H*H*SQ
      LR=Z*H*(LA*CT+LB*ST-LZ)
      DEL=H**6*E*E
      LH=-LR/H**2-LUR*U/H**2-LUT*V/H**2+LN*3.*H**2*U*V/DEL
      LU=LUR/H+LN*V*(V*V-H**3)/DEL
      LV=LUT/H-LN*U*(V*V+H**3)/DEL
      PRIMA=(LU-LN*(V*V-H**3)*V/DEL)/SP
      PSI=ATAN2(LUR,LUT)
      CRT=H*LH-3.*LN*SF/E
      Y(4)=EFF
      Y(6)=PRIMA
      Y(7)=PSI
      Y(8)=CRT
      IF(IFRD.EQ.0)GO TO 300
      E1=2.*ATAN(SQRT((1.-E )/(1.+E ))*TAN(EF1/2.))
      E2=2.*ATAN(SQRT((1.-E )/(1.+E ))*TAN(EFF/2.))
      GO TO 350
  300 E2=2.*ATAN(SQRT((1.-E )/(1.+E ))*TAN(EF1/2.))
      E1=2.*ATAN(SQRT((1.-E )/(1.+E ))*TAN(EFF/2.))
  350 IF(E1.LT.0.)E1=E1+2.*PIE*180.
```

```
IF(E2.LT.0.)E2=E2+2.*PIE*180.
IF(E1.GT.E2)E2=E2+2.*PIE*180.
X=X+(P/(1.-E*E))**1.5*(E2-E1-E*(SIN(E2)-SIN(E1)))
RETURN
END
```

```
C                          SUBROUTINE PRMITR
C
C
C       THE SUBROUTINE WHICH CONTROLS THE CONVERGENCE BASED ON THE
C       SENSITIVITY MATRIX (G) AND COMPUTES THE MATRIX (G) ON THE
C       SECOND AND SUBSEQUENT ITERATIONS
C
C
$IBFTC PRMTR2  XR7,M94,DECK
       SUBROUTINE PRMITR
       REAL KAK,MAXDIF
       COMMON/CMN1/AR,CR,PA,EA,THETA,CRT,DBLDOT,CPRO,PIE,P1,P2,PSI,DRN,D,
      1 Y50,Y60
       COMMON/CMN2/Y(9),W(9),XD(4),IWRT(4),TP(4),X,DX,N,TB,IP1,LN,IFRD
      1,DSTVAR(3,3)
       COMMON/CMN7/E0,P0,ET,PT,EF,PF,PHI1R,PHI2R,DLV1,DLV2,DN1R,WMBR,
      1 PSIA1,PSIA2,ASX80V,MAXDIF,ICNTMX,D2TMN,DL8I,XDAR1,XDARP1,JMP2
      2 ,COASTN,IMPTRD,JMP1
       COMMON/CMPH/KAK(3),PAR(3),HOM(3,3),EPS(3),C(3),DLT2,JPH
       COMMON/CMN10/ICNT,WB(3),ERRMIN,KEY1
       COMMON/CMN12/NOROT,PRAYER,NOCOAS,SIGMA9,JOEMAX,TST3MX
       LOGICAL NOCOAS,NOCOST
       DIMENSION PMIN(3),P(3),R(3),A(3,3),TEST(3),AINV(3,3),DP(3)
       DO 10 I=1,3
       P(I)=KAK(I)
  10   PMIN(I)=KAK(I)
       CALL PARHOM($320)
       DO 20 I=1,3
  20   R(I)=WB(I)-PAR(I)
       DO 30 I=1,3
       DO 30 J=1,3
  30   A(I,J)=-HOM(J,I)/EPS(J)*.01
  40   ICNT=ICNT+1
       WRITE(6,1002)ICNT
       IOUT=1
       ERROR=0.
       DO 50 I=1,3
       TEST(I)=ABS(R(I))
       ERROR=ERROR+TEST(I)
  50   IF(TEST(I).GT.MAXDIF)IOUT=0
       WRITE(6,1003)(TEST(I),I=1,3)
       WRITE(6,1004)ERROR
       IF(ERROR.GT.2.)GO TO 200
       IF(ERROR.GT.ERRMIN)GO TO 70
       ERRMIN=ERROR
       DO 60 I=1,3
  60   PMIN(I)=P(I)
  70   IF(IOUT)80,80,300
  80   CALL MINV(A,AINV,3)
       DO 90 I=1,3
  90   DP(I)=0.
       DO 100 I=1,3
       DO 100 J=1,3
 100   DP(I)=DP(I)- AINV(I,J)*R(J)*.01
       WRITE(6,1006)(DP(I),I=1,3)
       YOZ=.01/AR
       IF(ABS(DP(1)).GT.YOZ.OR.ABS(DP(2)).GT.YOZ.OR.ABS(DP(3)).GT.YOZ)
```

```
     1 GO TO 210
       IF(ICNT.GT.ICNTMX)GO TO 310
       DO 110 I=1,3
  110 P(I)=P(I)+DP(I)
       THETA=P(1)
       P2=P(2)
       CRT=P(3)
       Y4DG=THETA/PIE
       Y7DG=P2/PIE
       WRITE(6,1007)
       WRITE(6,1008)Y4DG,Y7DG,CRT
       CALL ZIPPO(JPH,4,$320,$320)
       WRITE(6,1009)
       Y3DG=Y(3)/PIE
       CRVL=CR*ALOG(Y(5))
       WRITE(6,1008)Y(1),Y(2),Y3DG,CRVL
       DO 120 I=1,3
  120 R(I)=WB(I)-Y(I)
       SMSQDP=DP(1)**2+DP(2)**2+DP(3)**2
       DO 130 I=1,3
       DO 130 J=1,3
  130 A(I,J)=A(I,J)+R(I)*DP(J)*.01/SMSQDP
       GO TO 40
  200 WRITE(6,1010)
       GO TO 320
  210 WRITE(6,1011)
       GO TO 320
  300 THETA=P(1)
       P2=P(2)
       CRT=P(3)
       GO TO 330
  310 WRITE(6,1012)ICNTMX
  320 THETA=PMIN(1)
       P2=PMIN(2)
       CRT=PMIN(3)
       WRITE(6,1004)ERRMIN
  330 NOCOST = NOCOAS
       NOCOAS=.TRUE.
       CALL ZIPPO(0,4,$340,$340)
       NOCOAS=NOCOST
  340 RETURN
 1000 FORMAT(93X,12HTHE MATRIX A)
 1001 FORMAT(70X,3F20.8)
 1002 FORMAT(1H0,22HNUMBER OF ITERATIONS =I2)
 1003 FORMAT(1X,15H(TEST(I),I=1,3),3F14.8)
 1004 FORMAT(1X,7HERROR =F11.8)
 1005 FORMAT(91X,15HTHE MATRIX AINV)
 1006 FORMAT(1X,11HDP(I),I=1,3,3F18.8)
 1007 FORMAT(2X,12HTRUE ANOMALY,3X,12HTHRUST ANGLE,6X,7HNU(CRT))
 1008 FORMAT(1X,8F15.7)
 1009 FORMAT(1X,13HSM-LAT-RECTUM,2X,12HECCENTRICITY,2X,13HARG OF PRCNTR,
     1 2X,13HCHAR VELOCITY)
 1010 FORMAT(1X,23HERROR TOO LARGE,SO STOP)
 1011 FORMAT(1X,23HDP(I) TOO LARGE,SO STOP)
 1012 FORMAT(1X,32HCONVERGENCE NOT ATTAINED AFTER  ,I3,12H  ITERATIONS)
       END
```

```
C                          SUBROUTINE PARHOM
C
C
C       THE SUBROUTINE WHICH IS USED FOR THE FIRST-TIME COMPUTATION
C       OF THE SENSITIVITY MATRIX (G).
C
C
$IBFTC PRHMGL  XR7,M94,DECK
       SUBROUTINE PARHOM(*)
       COMMON/CMN1/AR,CR,PA,EA,THETA,CRT,DBLDOT,CPRO,PIE,P1,P2,PSI,DRN,D,
      1 Y50,Y60
       COMMON/CMN2/Y(9),P(9),XD(4),IWRT(4),TP(4),X,DX,N,TB,IP1,LN,IFRD
       COMMON/CMPH/KAK(3),PAR(3),HOM(3,3),EPS(3),C(3),DLT2,JPH
       REAL KAK
       DIMENSION E(3,3)
       TP2RAT=.8
       DO 5 I=1,3
     5 TP(I)=0.
       DO 10 I=1,3
       DO 10 J=1,3
    10 E(I,J)=0.
       E(1,1)=EPS(1)
       E(2,2)=EPS(2)
       E(3,3)=EPS(3)
       THETA=KAK(1)
       P2=KAK(2)
       CRT=KAK(3)
       Y4DG=THETA/PIE
       Y7DG=P2/PIE
       WRITE(6,1002)
       WRITE(6,1001)Y4DG,Y7DG,CRT
       CALL ZIPPO(JPH,4,$100,$100)
       DO 20 I=1,3
    20 PAR(I)=Y(I)
       WRITE(6,1000)
       Y3DG=Y(3)/PIE
       CRVL=CR*ALOG(Y(5))
       WRITE(6,1001)Y(1),Y(2),Y3DG,CRVL,(TP(I),I=1,4)
       IF(TP(3)-TP(2).LT.TP2RAT*DLT2)RETURN 1
       DO 50 J=1,3
       THETA=KAK(1)+E(1,J)
       P2=KAK(2)+E(2,J)
       CRT=KAK(3)+E(3,J)
       CALL ZIPPO(JPH,4,$100,$100)
       DO 30 I=1,3
    30 HOM(J,I)=Y(I)-PAR(I)
       Y3DG=Y(3)/PIE
       CRVL=CR*ALOG(Y(5))
    50 WRITE(6,1001)Y(1),Y(2),Y3DG,CRVL,(TP(I),I=1,4)
       RETURN
   100 RETURN 1
  1000 FORMAT(7X, 9HPARAMETER,8X,9HECCNTRCTY,5X,15HARG OF PERICNTR,3X,
      1 13HCHAR VELOCITY,5X,5HTP(1),7X,5HTP(2),7X,5HTP(3),7X,5HTP(4))
  1001 FORMAT(4F17.8,4F12.8)
  1002 FORMAT(5X, 12HTRUE ANOMALY,5X,12HTHRUST ANGLE,10X,3HCRT)
       END
```

```
C                          SUBROUTINE ZIPPO
C
C
C       THE SUBROUTINE WHICH CONTROLS THE COMPUTATION OF A SPECIFIC
C       TRANSFER TRAJECTORY
C
C
$IBFTC ZIPP     XR7,M94,DECK
       SUBROUTINE ZIPPO(NOWRT,NP,*,*)
       COMMON/CMN1/AR,CR,PA,EA,THETA,CRT,DBLDOT,CPRO,PIE,P1,P2,PSI,DRN,D,
      1 Y50,Y60
       COMMON/CMN2/Y(9),P(9),XD(4),IWRT(4),TP(4),X,DX,N,TB,IP1,LN,IFRD
      1,DSTVAR(3,3)
       COMMON/CMN11/POLAN2
       COMMON/CMN12/NOROT,PRAYER,NOCOAS,SIGMA9,JOEMAX,TST3MX
       COMMON/NAMCOM/  NFNTR,NPRM,NZIP,NMIN,NFNEW,NTHT,NNU,NCOAST,NPSI,
      1 NPRHM,NDRN,NY7SET,NWRT,NFFR,NEX1,NEX2,NEX3,NEX4,NEX5,NEX6,NEX7
       DIMENSION XO(8)
       LOGICAL NOCOAS,NOROT
       REAL LN
       PRAYER=1.
     2 N=9
       IZO=0
       IZIP=0
       DO 5 I=1,4
     5 TP(I)=0.
       X=0.
       IP1=1
       SGN=1.
       Y(1)=PA
       Y(2)=EA
       Y(3)=0.
       Y(4)=THETA
       Y(5)=Y50
       Y(6)=Y60
       Y(7)=P2
       Y(8)=CRT
       Y(9)=0.
C  COMPUTE LN
       C4=COS(Y(4))
       C7=COS(Y(7))
       S4=SIN(Y(4))
       S7=SIN(Y(7))
       H=(1.+Y(2)*C4)/Y(1)
       PR=SQRT(Y(1))
       A=-Y(8)/H+Y(6)/PR*(S7*(C4+Y(2)*COS(2.*Y(4)))/S4-2.*C7*(1.+Y(2)*C4)
      1)
       B=-(1.+Y(2)*C4)/(H*Y(2)*S4)
       LN = A/B
       IF(NOWRT)10,10,20
    10 JWRT=0
       CALL WRTOUT(-1)
    20 DX=XD(IP1)
       IF(X+DX.GT.TB)GO TO 80
       CALL MIMIN(1.)
       IZIP=IZIP+1
       IF(SGN*Y(9).LT.0.)GO TO 50
```

```
      IF(NP.GT.1)GO TO 25
      IF((IFRD.EQ.1.AND.Y(5).GT.3.).OR.(IFRD.EQ.0.AND.Y(5).LT..75))
     1 RETURN 1
      IF(X.GT.2.*D)RETURN 1
      GO TO 35
   25 IF((IFRD.EQ.1.AND.Y(5).GT.3.).OR.(IFRD.EQ.0.AND.Y(5).LT..75).OR.
     1 (Y(1).LT..2))GO TO 130
   35 IF(NOWRT)40,40,20
   40 JWRT=JWRT+1
      IF(JWRT.EQ.IWRT(IP1))GO TO 45
      GO TO 20
   45 CALL WRTOUT(0)
      JWRT=0
      GO TO 20
   50 CALL F
      IF(IZIP.LT.2)GO TO 200
      IZO=IZO+1
      IF(IZO.GT.25)RETURN 1
      DX=-Y(9)/P(9)
      SN=1.
      IF(DX.LT.0.)SN=-1.
      DX=SIGN(AMIN1(XD(IP1),SN*DX),DX)
      CALL MIMIN(SN)
      IF(ABS(Y(9)).GT.SIGMA9)GO TO 50
      IZO=0
   55 IF(NOWRT)60,60,70
   60 CALL WRTOUT(0)
      JWRT=0
   70 TP(IP1)=X
      IF(NP.EQ.IP1)RETURN
      IP1T=IP1
      IP1=1
      CALL F
      IP1=IP1T
      DO 75 I=1,3
      DSTVAR(I,IP1)=P(I)
   75 IF(IP1.EQ.2)DSTVAR(I,2)=-P(I)
      IF(IP1.EQ.3)GO TO 100
      SGN=-SGN
      IP1=IP1+1
      IF(NOCOAS)GO TO 20
      IF(IP1.EQ.2)GO TO 120
      GO TO 20
   80 DX=TB-X
      CALL MIMIN(1.)
      IF(IP1.EQ.3)TP(3)=X
      IF(IP1.EQ.2)GO TO 140
  100 IF(NOWRT.EQ.1)RETURN
      DO 110 I=1,8
  110 XO(I)=Y(I)
      XO(3)=Y(3)/PIE
      XO(4)=Y(4)/PIE
      XO(7)=Y(7)/PIE
      CRVL=CR*ALOG(Y(5))
      CALL WRTOUT(1)
      WRITE(6,1000)
      WRITE(6,1001)(XO(I),I=1,8),CRVL
```

```
      RETURN
  120 CALL COAST(Y,X,LN,CR,POLAN2,IFRD,$140)
      GO TO 55
  130 TP(3)=X
      IF(NOWRT.EQ.1)RETURN
      CALL WRTOUT(1)
      RETURN
  140 IF(NP.LE.3)GO TO 150
      TP(2)=0.
      TP(3)=0.
      RETURN
  150 TP(2)=TB
      RETURN
  200 WRITE(6,1002)
      RETURN 2
 1000 FORMAT(3X,9HPARAMETER,5X,9HECCNTRCTY,3X,12HPERICNTR ARG,2X,
     1 12HTRUE ANOMALY,6X,5HXQ(5),9X,5HXQ(6),6X,12HTHRUST ANGLE,6X,
     2 5HXQ(8),5X,12HCHAR VELOCTY)
 1001 FORMAT(9F14.8)
 1002 FORMAT(1H0,50HSWITCHING FUNCTION DOES NOT BECOME POSITIVE AT T0+)
      END
```

BIBLOGRAPHY

1. Fraeijs DeVeubeke, B. "Canonical Transformations and the Thrust-Coast - Thrust Optimal Transfer Problem", Astronautica Acta, 11 (1965), p. 271-282.

2. McCue, G. A. "Quasilinearization Determination of Optimum, Finite-Thrust Orbital Transfers", AIAA Journal, 5 (1967), p. 755-763.

3. McCue, G. A. and Bender, D. F. "Numerical Investigation of Minimum Impulse Orbital Transfer", AIAA Journal, 3 (1965), p. 2328-2334.

4. Lee, G. "An Analysis of Two-Impulse Orbital Transfer", AIAA Journal, 2 (1964),p. 1767-1773.

5. Lawden, D. F. Optimal Trajectories for Space Navigation, Butterworths Scientific Publications Ltd., London, 1963 p. 54-60.

6. Robbins, H. M. "An Analytical Study of the Impulsive Approximation", AIAA Journal, 4 (1966),p. 1417-1423.

7. Battin, R. H. Astronautical Guidance, McGraw Hill, New York, 1964.

8. Fletcher, R. and Powell, M. J. D. "A Rapidly Convergent Descent Method for Minimization", British Computer Journal, 6 (1963),p. 163-168.

9. Robbins, H. M. "Optimality of Intermediate - Thrust Arcs of Rocket Trajectories", AIAA Journal, 3 (1965),p. 1094-1098.

10. Fraeijs DeVeubeke, B. "Optimal Steering and Cutoff-Relight Programs for Orbital Transfers", Astronautica Acta, 12 (1966), p. 323-328.

11. Kern, E. A. and Burghart, J. H. "An Efficient Algorithm for Minimum Impulse Orbital Transfer", AIAA Journal, 6 (1968), p. 1370-1372.

12. Pontryagin, V. G., Boltyanski, V. G., Gamkrelidze, R. V. and Mishehenko, E. F. The Mathematical Theory of Optimal Processes, John Wiley & Sons, Inc., New York, 1962.

13. Marchal, Christian "Transferts Optimaux Entre Orbites Elliptiques Coplanaires (Duree Indifferente)", Astronautica Acta, 11 (1965), p. 432-445.

14. Kopp, R. E. and Moyer, H. G. "Necessary Conditions for Singular Extremals", AIAA Journal, 3 (1965),p. 1439-1444.

15. Handelsman, M. "Optimal Free-Space Fixed-Thrust Trajectories Using Impulsive Trajectories as Starting Iteratives", AIAA Journal, 4 (1966), p. 1077-1082.

16. Pines, S. "Constants of the Motion for Optimum Thrust Trajectories in a Central Force Field", AIAA Journal, 2 (1964), p. 2010-2014.

17. Culp, R. D. "Contensou-Busemann Conditions for Optimal Coplanar Orbit Transfer", AIAA Journal, 5 (1967), p. 371-372.

18. Kulakowski, L. J. and Stancil, R. T. "Rocket Boost Trajectories for Maximum Burnout Velocity", ARS Journal, 30 (1960), p. 612-618.

19. Coddington, E. A. and Levinson, N. Theory of Ordinary Differential Equations, McGraw Hill, New York, 1955.