010021-F

FINAL REPORT

FOR

NASA Marshall Space Flight Center

CONTRACT NAS 8-26929

CONJUGATE GRADIENT OPTIMIZATION PROGRAMS FOR SHUTTLE REENTRY

by

W. F. Powers, R. A. Jacobson, and D. A. Leonard

Department of Aerospace Engineering
The University of Michigan
Ann Arbor, Michigan 48104

May 1972

010021-F

FINAL REPORT

FOR

NASA Marshall Space Flight Center

CONTRACT NAS 8-26929

CONJUGATE GRADIENT OPTIMIZATION PROGRAMS FOR SHUTTLE REENTRY

by

W. F. Powers, R. A. Jacobson, and D. A. Leonard

Department of Aerospace Engineering
The University of Michigan
Ann Arbor, Michigan 48104

May 1972

ABSTRACT

Two computer programs for shuttle reentry trajectory optimization are listed and described. Both programs use the conjugate gradient method as the optimization procedure. The Phase I Program is developed in cartesian coordinates for a rotating spherical earth, and crossrange, downrange, maximum deceleration, total heating, and terminal speed and altitude are included in the performance index. The Phase II Program is developed in an Euler angle system for a nonrotating spherical earth, and crossrange, downrange, total heating, maximum heat rate, and terminal speed, altitude, and flight path angle are included in the performance index. The programs make extensive use of subroutines so that they may be easily adapted to other atmospheric trajectory optimization problems.

TABLE OF CONTENTS

	Page
Abstract	i
Chapter 1 Introduction	1
Chapter 2 The Conjugate Gradient Method	4
2.1 Finite-Dimensional Conjugate Gradient Method	4
2.2 Infinite-Dimensional Conjugate Gradient: Unconstrained	10
2.3 Infinite-Dimensional Conjugate Gradient: Constrained	16
Chapter 3 Phase I Program	19
3.1 Basic Description	19
3.2 Subroutine Map	21
3.3 Subroutine Description	22
3.4 Phase I Program Notes	24
Chapter 4 Phase II Program	25
4.1 Basic Description	25
4.2 Subroutine Map	27
4.3 Subroutine Descriptions	28
4.4 Phase II Program Notes	30
Chapter 5 Conclusions and Recommendations	31
5.1 Summary	31
5.2 Conclusions and Recommendations	31
References	34
Appendix A Listing of Phase I Program	36
Appendix B Listing of Phase II Program	67

CHAPTER 1
INTRODUCTION

The forthcoming Space Shuttle Program will involve vehicles which possess both rocket and aircraft characteristics. Because of the interplay of gravitational, thrusting, and aerodynamic forces, the trajectories that the vehicle will fly are more complicated than the trajectories of the Saturn-Apollo class. Thus, the need exists for efficient, reliable shuttle trajectory optimization programs.

This report describes two computer programs which were generated for shuttle reentry. During the time period of this contract, the national emphasis shifted from a small, low-crossrange, straight-wing orbiter to a larger, high-crossrange, delta-wing orbiter. This definitely influences the reentry trajectory in that the straight-wing trajectory usually encounters the 3g deceleration constraint whereas the delta-wing trajectory rarely (if ever) encounters the 3g-constraint but instead has high heat-rate problems. Thus, instead of making a large cumbersome program for all possible vehicles, two programs were developed. Since the Phase II-deck was developed after the Phase I-deck, it has the advantage of some improvements learned in the development of the Phase I-program.

It has been noted by numerous investigators in the last two years that shooting (or initial Lagrange multiplier guessing) iteration schemes have been almost useless in determining shuttle reentry trajectories. There exist other techniques which might be applicable to the problem and they are briefly described below:

1) Classical Gradient Method: This method iterates on the total control function and does not require any second-order information (i.e., second-derivatives of the Hamiltonian). This method is well-known for having excellent properties far away from the solution, but slow convergence near the solution. With respect to boundary conditions, either penalty functions[1*] or projections[2] may

*Numbers refer to listings in the References section.

be employed. A modified gradient projection approach for shuttle reentry is under development at TRW-Systems[3].

2) Second-Order Gradient Method: This method is essentially a function space Newton's method which iterates on the total control function and requires full second-order information. The method is described in Ref. 4, and a shuttle-version of the program is in use at NASA-Manned Spacecraft Center[5]. It has been found that although this program obtains accurate trajectories and control histories, the deck is difficult to work with and modify, and requires extremely long computer time.

3) Conjugate Gradient[6] and Function-Space Davidon[7] Methods: These methods iterate on the total control function and do not require any second-order information. These methods are mainly motivated by deficiencies in the classical gradient and second-order gradient (or function space Newton) methods. That is, they require only first-order information and may have better convergence characteristics near the minimum than the classical gradient method. This study involved the generation of two conjugate gradient programs. It appears that the function-space Davidon method needs further analysis before it should be employed in a shuttle computer program.

4) Parameter Optimization Methods: In the last decade a number of efficient parameter optimization techniques have been popularized, e.g., conjugate gradient (CG), Davidon-Fletcher-Powell (DFP) variable metric. These schemes have proven their worth, and the DFP method is probably the most popular parameter optimization scheme in use today. Both the CG and DFP methods are available in Fortran subroutines[8]. The DFP method has been applied successfully to shuttle optimization by Johnson and Kamm[9,10]. They represent the control variables by sequences of straight line segments and then use DFP to iterate for the optimal slopes of the segments subject to continuity and inequality constraints. By computing their gradients numerically, the deck is easily modified to

include additional parameters, different vehicles, and various missions. Thus, for design purposes, this is a very efficient approach.

From the discussion above of the various approaches to shuttle optimization, it would appear at first glance that parameter optimization is the superior iterative procedure. For preliminary design this is probably the case. However, the parameter optimization approach requires either prior knowledge of approximate optimal control histories or an undeterminable amount of working time devoted to selecting workable but accurate representations for the controls. In reentry this may be especially difficult because a change in terminal boundary conditions may cause a completely different bank angle control, and in many cases the bank angle would require a large number of segments to approximate it adequately. Thus, the parameter optimization approach is by no means automatic or even desirable in some cases.

Because of the deficiencies noted above for the parameter optimization approach, the need still exists for a relatively flexible and efficient function space technique. At the present time it appears that both the projected gradient and the conjugate gradient methods are the leading candidates for such a scheme, and which scheme is best is probably problem dependent. For example, the projected gradient technique is probably best for problems which are strongly influenced by boundary conditions and do not contain singular arcs. The conjugate gradient technique is probably best for problems with singular arcs and/or problems which exhibit slow convergence near the minimum with a standard gradient technique. However, not as much work has been done with the conjugate gradient technique as the projected gradient technique, so improvements in the conjugate gradient approach are occurring more frequently than in the projected gradient method. It should be noted that the conjugate gradient and gradient projection technique have been combined[11], but the results were not promising. However, there may exist more efficient ways of combining the two techniques, and, thus, a "projected conjugate gradient" technique may be feasible.

CHAPTER 2

THE CONJUGATE GRADIENT METHOD

In this chapter, a tutorial treatment of the conjugate gradient method will be given in both finite- and infinite-dimensional spaces. The methods for treating inequality constraints in the programs are discussed, also.

2.1 Finite-Dimensional Conjugate Gradient Method

Consider the problem of minimizing

$$f(x_1, \ldots, x_n) , \tag{2.1}$$

where $x \equiv (x_1, \ldots, x_n)$ is an element of a bounded, connected, open subset of R^n and $f \in C^1$. If equality and/or inequality constraints are present, it is assumed that they are incorporated into (2.1) by means of penalty functions.

Before we develop the algorithm, let us consider a few general remarks about the minimization of a quadratic function. Consider

$$q \equiv x^T A x, \tag{2.2}$$

where $x \in R^n$, A is a positive definite matrix. The contours of constant q-values are n-dimensional ellipsoids centered on the global minimum $x = 0$. In 2-space, the eccentricity of the elliptical contours is dependent upon the relative magnitudes of the eigenvalues of A; the contours are circular if the eigenvalues are equal and the contours become more eccentric as the ratio of the eigenvalues increases from one. Of course, similar results are true in n-space.

If the contours of (2.2) are noncircular, the gradient method (with a one-dimensional search) will take an infinite number of iterates to converge to the minimum if the method does not converge on the first iterate. (If the initial guess is on a principal axis of the n-dimensional ellipsoid, then a single gradient step results in $x = 0$.) On the other hand, no matter what the eigenvalues are, Newton's method will converge in one iterate.

The reason why the quadratic problem is of interest is that in the terminal stages of an iterative minimization of many nonlinear functions, the

the function may be well-approximated by a second-order expansion. Thus, an efficient algorithm for general functions should have good convergence characteristics for quadratic functions. As noted above for quadratic functions, Newton's method is excellent in all cases, while the gradient method is strongly problem dependent. However, Newton's method requires the computation of second-order information while the gradient method requires only first-order. In addition, for general nonlinear functions, Newton's method may diverge whereas the gradient method will, at least, never result in an iterate which increases the quantity to be minimized.

Because of the properties discussed above, researchers in the 1950's attempted to develop techniques which combined the advantages of the gradient and Newton methods while minimizing their disadvantages. With respect to the quadratic minimization problem, two techniques with the following properties were developed: (i) the methods are stable, (ii) the minimum is obtained in at most n iterations, (iii) no second-order information is required. The methods are the conjugate gradient method[12] and the Davidon variable metric method[13] (or Davidon-Fletcher-Powell[14] method).

With respect to general nonlinear functions, the methods retain properties (i) and (ii) mentioned above. For certain classes of functions, rates of convergence are known for all of the methods mentioned except the DFP method. These show that when the methods work, Newton should be faster than the CG method, and the CG method should be faster than the gradient method. However, Newton's method does not possess either property (i) or (ii) mentioned above.

The CG formula will now be stated, the sequence of steps required in the development of the formula will be outlined, and then the steps will be developed in detail. The CG algorithm is as follows:

(1) Guess x_0. Define $g \equiv f_x$.

(2) $$p_0 \equiv g_0, \quad p_{J+1} = g_{J+1} + p_J\left(\frac{g_{J+1}^T\, g_{J+1}}{g_J^T\, g_J}\right) \quad (J = 0, 1, \ldots). \tag{2.3}$$

(3) $$x_{J+1} = x_J - \alpha_J p_J. \quad (\alpha_J \geqq 0) \tag{2.4}$$

In the formula above, x_J is an n-vector, g_J is the n-vector gradient, p_J is the n-vector search direction, and α_J is a scalar.

The formula development involves the following sequence of steps:

(A) Assume $x_{J+1} = x_J - \alpha_J p_J$ with $p_J = g_J + b_J$, and devise a means for defining b_J.

(B) Show that $g_J^T b_J = 0$ implies the method will be stable.

(C) Show that the largest decrease in f is obtained if $g_{J+1}^T p_J = 0$.

(D) Note that (B) and $b_J \equiv C_J p_{J-1}$ imply (C), where C_J is a constant to be defined.

(E) Show that finite convergence for the quadratic function $f = x^T Ax$ is guaranteed if $p_I^T A p_J = 0$ $(I \neq J)$, i.e., the search directions are "A-conjugate."

(F) Combine all of the above steps to show that

$$b_J = (\langle g_J, g_J\rangle / \langle g_{J-1}, g_{J-1}\rangle) p_{J-1} , \tag{2.5}$$

where $\langle g_J, g_J\rangle \equiv g_J^T g_J$ is an inner product in R^n. The inner product notation will be used from here on instead of the transpose notation.

Let us now develop the results noted in steps (A) to (F). First, we assume a form for the update formula

$$x_{J+1} = x_J - \alpha_J p_J \tag{2.6}$$

$$p_J = g_J + b_J . \tag{2.7}$$

The motivation for this form is that the method is basically a gradient method with a correction vector (i.e., b_J) which, hopefully, will aid the convergence characteristics of the gradient method in the neighborhood of the solution. The only undefined quantities in Eqs (2.6) and (2.7) are α_J and b_J. The scalar α_J will be determined by a 1-D search in each iteration, so the only quantity which must be characterized is the n-vector b_J.

<u>PROPERTY 1</u>: If

$$\langle g_J, b_J\rangle = 0 \tag{2.8}$$

then the method is stable.

Proof: Expand $f(x_{J+1})$ about $f(x_J)$ to first-order:

$$f(x_{J+1}) = f(x_J) + \langle g(x_J),\ x_{J+1} - x_J \rangle$$

$$f(x_{J+1}) = f(x_J) - \alpha_J \langle g_J, g_J \rangle - \alpha_J \langle g_J, b_J \rangle . \tag{2.9}$$

For α_J small, a sufficient condition for $f(x_{J+1})$ to be less than or equal to $f(x_J)$ is $\langle g_J, b_J \rangle = 0$. ∎

Note that Property 1 is a sufficient condition for stability. Thus, there exist numerous possibilities for techniques which could also be stable; one need only insure that the interaction of the two terms on the right-hand side of Eq. (2.9) be negative (when the first-order expansion is valid).

PROPERTY 2: Let α_J be the value of the search parameter which minimizes $f(x_J + \alpha p_J)$. Then,

$$\langle g_{J+1}, p_J \rangle = 0. \tag{2.10}$$

Proof: By definition of α_J:

$$\left.\frac{df}{d\alpha}\right|_{\alpha_J} = \left[\frac{\partial f}{\partial x_{J+1}}^T \frac{\partial x_{J+1}}{\partial \alpha}\right]_{\alpha_J} = \langle g_{J+1}, p_J \rangle = 0. \quad ∎$$

PROPERTY 3: If Eq. (2.8) holds and

$$b_J = C_J p_{J-1} \qquad (J = 1, 2, \ldots) \tag{2.11}$$

(where $C_J \neq 0$ is a scalar to be defined), then Eq. (2.10) is satisfied.

Proof: By Eq. (2.8):

$$\langle g_{J+1}, b_{J+1} \rangle = 0 \Rightarrow \langle g_{J+1}, C_{J+1} p_J \rangle = 0.$$

Thus, Eq. (2.10) is satisfied when $C_{J+1} \neq 0$. ∎

Because of Property 3, we shall assume that the correction vector, b_J, is linearly related to the previous search direction, i.e., we shall assume that b_J is defined by (2.11). In this case, the only thing that remains is the characterization of the constant C_J.

PROPERTY 4: Consider $f = x^T Ax$, where A is positive definite. If the search directions are A-conjugate (i.e., $p_I^T Ap_J = 0$, $I \neq J$) and Eqs. (2.6) and (2.10) hold (or, equivalently, (2.6), (2.8), (2.11)), then the global minimum $x = 0$ of f is obtained in at most n iterations.

Proof: At the unique global minimum of f, the gradient $g = Ax$ must equal zero. If in the application of the algorithm either $g_0, g_1, \ldots,$ or $g_{n-1} = 0$, then the property is proved. Thus, assume $g_0, \ldots, g_{n-1} \neq 0$. At each iterate, we have

$$g_J = Ax_J. \tag{2.12}$$

By repeated use of Eq. (2.6) we have:

$$x_n = x_{J+1} + \sum_{i=J+1}^{n-1} \alpha_i p_i$$

for any $J \in \{0, \ldots, n-2\}$. From Eq. (2.11):

$$g_n = g_{J+1} + \sum_{i=J+1}^{n-1} \alpha_i Ap_i. \tag{2.13}$$

The inner product of g_n and p_J is

$$<g_n, p_J> = <g_{J+1}, p_J> + \sum_{i=J+1}^{n-1} \alpha_i <p_i, Ap_J>. \tag{2.14}$$

The first term in this equation vanishes because of Eq. (2.10), and the summation vanishes because of the A-conjugacy property. Thus,

$$<g_n, p_J> = 0. \quad (J = 0, 1, \ldots, n-2) \tag{2.15}$$

By Eq. (2.10) we also have

$$<g_n, p_{n-1}> = 0. \tag{2.16}$$

Equations (2.15) and (2.16) may be written in matrix form as

$$[p_0 p_1 \cdots p_{n-1}] g_n = 0. \tag{2.17}$$

It can be shown that n A-conjugate vectors are linearly independent (note that A-conjugate is a generalization of orthogonality), and thus, Eq. (2.17) implies

$$g_n = 0. \quad \blacksquare \tag{2.18}$$

We now have enough information to define the constant C_J in Eq. (2.11).

PROPERTY 5: Consider $f = x^T Ax$, where A is positive definite. If: the update formula is defined by Eqs. (2.6) and (2.11), the search directions are A-conjugate, and α_J and C_J are chosen to give the maximum decrease in the function f, then

$$C_J = \langle g_J, g_J\rangle / \langle g_{J-1}, g_{J-1}\rangle . \tag{2.19}$$

Proof: At a given iteration f is given by

$$f[x_J + \alpha_J g_J + \alpha_J C_J p_{J-1}].$$

At a minimum of f with respect to α_J, C_J:

$$f_{\alpha_J} = 0 \Rightarrow \langle g_{J+1}, p_J\rangle = 0 \tag{2.20}$$

$$f_{C_J} = 0 \Rightarrow \langle g_{J+1}, p_{J-1}\rangle = 0. \tag{2.21}$$

Expansion of Eq. (2.20), noting $g_{J+1} = g_J + \alpha_J Ap_J$, $p_J = g_J + C_J p_{J-1}$, gives

$$\langle g_J, g_J\rangle + C_J\langle p_{J-1}, g_J\rangle + \alpha_J\langle p_J, Ap_J\rangle = 0,$$

which implies

$$\alpha_J = -\langle g_J, g_J\rangle / \langle p_J, Ap_J\rangle . \tag{2.22}$$

Before we obtain the desired result, note that Eqs. (2.20) and (2.21) imply

$$\langle g_{J+1}, g_J\rangle = 0. \tag{2.23}$$

To obtain the expression for C_J, we first form the inner product of $g_J = p_J - C_J p_{J-1}$ with Ap_{J-1}:

$$\langle g_J, Ap_{J-1}\rangle = \langle p_J, Ap_{J-1}\rangle - C_J\langle p_{J-1}, Ap_{J-1}\rangle . \tag{2.24}$$

The first inner product on the right vanishes because of A-conjugacy. The desired result is obtained by substituting $(g_J - g_{J-1})/\alpha_{J-1}$ for Ap_{J-1} on the left and $\langle g_{J-1}, g_{J-1}\rangle / \alpha_{J-1}$ for $-\langle p_{J-1}, Ap_{J-1}\rangle$ on the right:

$$<g_J, g_J>/\alpha_{J-1} - <g_J, g_{J-1}>/\alpha_{J-1} = C_J<g_{J-1}, g_{J-1}>/\alpha_{J-1}$$

or,

$$C_J = <g_J, g_J>/<g_{J-1}, g_{J-1}>. \quad \blacksquare$$

As noted previously, the algorithm defined above, along with the Davidon-Fletcher-Powell method, are available as Fortran subroutines in Ref. 8.

2.2 Infinite-Dimensional Conjugate Gradient: Unconstrained

In this chapter the conjugate gradient method is treated separately in finite- and infinite-dimensional spaces because of applications. However, one could describe the method in a Hilbert space setting and, thus, cover both the finite- and infinite-dimensional cases in one development. Such is the approach taken in Refs. 15, 16, and 17.

The main references for Sections 2.2 and 2.3 are Refs. 6 and 18. In this section we shall consider problems which do not possess control or state variable inequality constraints; these will be included in the next section.

The infinite-dimensional problem which we are mainly concerned with is the following:

BASIC PROBLEM: Determine the control $u^*(t)$, $t \in [t_0, t_f]$, which minimizes:

$$J[u] = \tilde{\phi}(t_f, x_f) + \int_{t_0}^{t_f} L(t,x,u)dt \tag{2.25}$$

subject to:

$$\dot{x} = f(t,x,u) \quad , \quad x(t_0) = x_0 \tag{2.26}$$

$$\psi(t_f, x_f) = 0 \quad , \quad (\text{p-vector; } p \leqq n + 1) \tag{2.27}$$

where x is an n-vector, u is an m-vector.

The algorithms in this report treat all of the terminal conditions (i.e., Eq. (2.27)) except one by the method of penalty functions; the remaining condition is employed as a stopping condition. Without loss of generality, assume that

$$\psi(t_f, x_f) \equiv \begin{bmatrix} x_{1f}(t\ ) \ - \ x_{1f} \\ \psi_2(t_f, x_{2f}, \ldots, x_{nf}) \\ \cdot \\ \cdot \\ \cdot \\ \psi_p(t_f, x_{2f}, \ldots, x_{nf}) \end{bmatrix} = 0 \,, \tag{2.28}$$

and that $x_1(t)$ is a variable which: (i) cannot reach the value x_{1f} until the terminal portion of the trajectory (e.g., a specified altitude or Mach number in reentry), (ii) will always be reached in a reasonable time, and (iii) will probably have a nonzero derivative at t_f. In this case, $x_1(t_f) = x_{1f}$ is a suitable stopping condition for the iterations.

Define

$$\phi(t_f, x_f) \equiv \tilde{\phi}(t_f, x_f) + \sum_{i=2}^{p} P_{i-1}\, \psi_i(t_f, x_f)^2 \,, \tag{2.29}$$

where it is assumed, also, that $\tilde{\phi}(t_f, x_f)$ does not depend upon x_{1f} (this is the usual case in trajectory analysis; the assumption is not restrictive, however) and the

$$P_i > 0 \quad (i = 1, \ldots, p - 1) \tag{2.30}$$

are selected by the investigator. With the definitions (2.28) and (2.29) we have the following problem:

<u>BASIC PROBLEM WITH PENALTY FUNCTIONS</u>: Determine the control $u^*(t)$, $t \in [t_0, t_f]$, which minimizes

$$J[u] = \phi(t_f, x_f) + \int_{t_0}^{t_f} L(t, x, u)\,dt \tag{2.31}$$

subject to:

$$\dot{x} = f(t, x, u), \quad x(t_0) = x_0 \tag{2.32}$$

$$x_1(t_f) = x_{1f} \,.$$

(<u>Note</u>: t_f is usually not specified.)

Before we list the formulas in the conjugate gradient method, we shall define a Hamiltonian function and adjoint variables which are useful in any function space iteration scheme. First, define

$$H \equiv L(t,x,u) + \lambda^T f(t,x,u), \tag{2.34}$$

where the n-vector $\lambda(t)$ will be characterized later. With this definition we have:

$$J[u] = \phi(t_f, x_f) + \int_{t_0}^{t_f} [H(t,x,u,\lambda) - \lambda^T \dot{x}]dt , \tag{2.35}$$

where the performance index (2.31) has been augmented to include $\int_{t_0}^{t_f} \lambda^T(f-\dot{x})dt$.

Let $u^{(0)}(t)$ be an initial control estimate, and integrate $\dot{x} = f[t,x,u^{(0)}(t)]$ forward from $x(t_0) = x_0$ to form a corresponding trajectory, $x^{(0)}(t)$. Suppose there exists a vector $\lambda^{(0)}(t)$ and define

$$u^{(1)}(t) = u^{(0)}(t) + \delta u(t) \tag{2.36}$$

$$x^{(1)}(t) = x^{(1)}(t) + \delta x(t). \tag{2.37}$$

$$t_f^{(1)} = t_f^{(0)} + dt_f \tag{2.38}$$

Expand $J[u^{(1)}]$ about $J[u^{(0)}]$ to first-order:

$$\begin{aligned} J[u^{(1)}] = J[u^{(0)}] &+ \phi_{t_f}^{(0)} dt_f + \sum_{i=2}^{n} \phi_{x_{if}}^{(0)} dx_{if} \\ &+ [H(t_f^{(0)}) - \lambda^{(0)T}(t_f^{(0)})\dot{x}^{(0)}(t_f^{(0)})]dt_f \\ &+ \int_{t_0}^{t_f^{(0)}} [H_x^{(0)T} \delta x + H_u^{(0)T} \delta u - \lambda^{(0)T} \delta\dot{x}]dt . \end{aligned} \tag{2.39}$$

Integration by parts of the third term in the integrand gives

$$\begin{aligned} \Delta J[\delta u] = J[u^{(1)}] - J[u^{(0)}] = (\phi_{t_f}^{(0)} + H^{(0)})_{t_f^{(0)}} dt_f - \lambda_1^{(0)}(t_f^{(0)}) dx_{1f} \\ + \sum_{i=2}^{n} (\phi_{x_{if}}^{(0)} - \lambda_i^{(0)})_{t_f^{(0)}} dx_{if} \\ + \int_{t_0}^{t_f^{(0)}} [(H_x^{(0)} + \dot{\lambda}^{(0)})^T \delta x + H_u^{(0)T} \delta u]dt \end{aligned} \tag{2.40}$$

subject to:

$$dx_{1f} = 0. \tag{2.41}$$

We now characterize $\lambda^{(0)}(t)$ so that a stable iterative algorithm is defined.

$$\text{SPECIFY:} \quad \lambda_i^{(0)}(t_f^{(0)}) \equiv \phi_{x_{if}}^{(0)} \qquad (i = 2, \ldots, n) \tag{2.42}$$

$$\lambda_1^{(0)}(t_f^{(0)}) \equiv -(\phi_{t_f}^{(0)} + H^{(0)} + \sum_{i=2}^{n} \lambda_i^{(0)} \dot{x}_i^{(0)})_{t_f^{(0)}} / \dot{x}_1^{(0)}(t_f^{(0)}) \tag{2.43}$$

$$\dot{\lambda}^{(0)}(t) \equiv -H_x[t, x^{(0)}(t), u^{(0)}(t), \lambda^{(0)}(t)]. \tag{2.44}$$

Definitions (2.42), (2.43), (2.44) uniquely define the vector $\lambda^{(0)}(t)$ and it is formed by a backward integration.

Substitution of Eqs. (2.41)-(2.44) into Eq. (2.40) gives

$$\Delta J[\delta u] = \int_{t_0}^{t_f} H_u^{(0)T} \delta u \, dt. \tag{2.45}$$

The quantity $H_u^{(0)}(t)$ is the gradient in function space for this iteration, and the gradient method is defined by

$$u^{(J+1)}(t) = u^{(J)}(t) - \alpha_J H_u^{(J)}(t). \tag{2.46}$$

(Note that if $t_f^{(J+1)} > t_f^{(J)}$, then a scheme must be devised to define $u^{(J+1)}(t)$ on the interval $[t_f^{(J)}, t_f^{(J+1)}]$, but there are numerous ways of doing this.) As with the parameter optimization problem, there exist numerous techniques which result in a stable method, e.g., one need only guarantee that the first-order expansion term dominate the expansion for $J[u^{(J+1)}]$ and that δu be chosen in such a way that

$$\int_{t_0}^{t_f} H_u^{(0)T}(t) \, \delta u(t) \leqq 0. \tag{2.47}$$

In analogy with parameter optimization, a possible choice for δu is

$$\delta u^{(J)}(t) = -\alpha_J [H_u^{(J)}(t) + C_J p^{(J-1)}(t)], \tag{2.48}$$

where $\alpha_J > 0$ is the search parameter, $p^{(J-1)}(t)$ is the previous search direction with the property $\int_{t_0}^{t_f} H_u^{(J)T} p^{(J-1)} dt = 0$, and C_J is a constant to be defined. As shown in Ref. 6, the following function space conjugate gradient scheme satisfies these conditions:

<u>UNCONSTRAINED CONJUGATE GRADIENT ALGORITHM</u>

1) Guess $u^{(0)}(t)$ on $[t_0, t_f^{(0)}]$.
2) Compute:

$$X^{(J)}(t),\ \lambda^{(J)}(t),\ H_u^{(J)}(t)$$

$$p^{(J)}(t) = H_u^{(J)}(t) + \frac{\int_{t_0}^{t_f} H_u^{(J)^T} H_u^{(J)} dt}{\int_{t_0}^{t_f} H_u^{(J-1)^T} H_u^{(J-1)^T} dt}\, p^{(J-1)}(t) \tag{2.49}$$

$$(p^{(0)}(t) \equiv H_u^{(0)}(t))$$

3) Perform 1-D search to determine α_J in the formula

$$u^{(J+1)}(t) = u^{(J)}(t) - \alpha_J p^{(J)}(t). \tag{2.50}$$

4) Check on appropriate cutoff criterion (e.g., $\left| \frac{dJ}{d\alpha_J}\Big|_{\alpha_J = 0} \right| \leqq \epsilon$);
5) Return to 2).

In Eq. (2.49) above, the constant which multiplies $p^{(J-1)}$ may be written as

$$< H_u^{(J)}, H_u^{(J)} >_1 / < H_u^{(J-1)}, H_u^{(J-1)} >_1 ,$$

where

$$<a(t), b(t)>_1 \equiv \int_{t_0}^{t_f} a(t)^T b(t) dt \tag{2.51}$$

is an inner product on the function space whereas

$$<a, b> \equiv a^T b \tag{2.52}$$

is an inner product on R^n. Thus, the formula is the same as the finite-dimensional formula; one need only interpret properly the gradient and inner product functions.

In Ref. 6, a few theorems concerned with the convergence of the function space conjugate gradient method are presented for both general functionals and functionals which result from linear-quadratic optimal control problems. For general functionals, the convergence theorem (which is only sufficient for convergence) essentially requires that one show that the second variation is "strongly positive" (i. e., there exists a constant $M > 0$ such that for all admissible $u, \delta u$, $\delta^2 J(u; \delta u, \delta u) \geqq M \| \delta u \|^2$).

As with parameter optimization, the quadratic case plays an important role in functional optimization; again, the argument being that when the solution is approached the general optimal control problem may be well-approximated by a linear-quadratic optimal control problem (formed by expanding the differential equations and boundary conditions to first-order, and the performance index to second-order). The theorems in Ref. 6 assume the resultant quadratic functional to be of the form

$$\Delta J = \langle \delta u, A \delta u \rangle_1 \quad , \tag{2.53}$$

where A is a positive definite, self-adjoint linear operator. Note that since δu is infinite-dimensional there is no reason to expect finite convergence even in a small neighborhood of the solution. (Of course, on a digital computer, one is really only interested in a good rate of convergence since problems are never converged to the limit.) Reference 7 shows how one may transform a class of linear-quadratic problems into the form of Eq. (2.53).

For the case of Eq. (2.53), Ref. 6 shows that the conjugate gradient method has certain desirable features which the classical gradient method does not possess. However, it has never been proved mathematically that the conjugate gradient step is better than the gradient step on every iterate. In fact the statement is probably untrue because of numerical experience which indicates that a gradient step every so often in a conjugate gradient algorithm (i. e., a "reset" step) improves the convergence characteristics. Finally, as with general functionals, to show that the linear operator A in

Eq. (2.53) is positive definite usually requires a conjugate point test if the operator results from linearization of a nonlinear optimal control problem.

2.3 Infinite-Dimensional Conjugate Gradient: Constrained

In this section, the modifications of the Basic Problem With Penalty Functions (Eqs 2.31-2.33) and the Unconstrained Conjugate Gradient Algorithm to include state variable inequality constraints (SVIC) and control inequality constraints will be presented.

First, suppose that in addition to the equality constraints (2.32), (2.33), the problem contains the SVIC's:

$$S_i(t,x) \geqq 0. \qquad (i = 1,\dots,q) \tag{2.54}$$

There are two main ways of treating an SVIC:

(i) Transform the problem into a multiple-arc problem with intermediate point equality constraints; this is the approach of Ref. 19.

(ii) Augment the performance index to include the SVIC's by means of penalty functions; this is the approach of Ref. 1.

The main goal of the computer programs described in this report is to generate reasonable, near-optimal reentry trajectories with a minimal amount of guessing and analysis required of the user. The (i) approach above requires knowledge of the location and the number of times the inequality constraint boundary is encountered, which requires both analysis and additional programming by the user. Thus, the (ii) approach was chosen since this requires no additional programming and only the initial penalty coefficients must be estimated.

If the SVIC's (2.54) are present, then the performance index (2.31) is modified to

$$J[u] = \phi(t_f, x_f) + \int_{t_0}^{t_f} [L(t,x,u) + \sum_{i=1}^{q} S_i(t,x)^2 H_i(S_i)]dt, \tag{2.55}$$

where

$$H_i(S_i) = \begin{cases} C_i > 0 \text{ if } S_i < 0 \\ 0 \qquad \text{if } S_i \geqq 0 , \end{cases} \tag{2.56}$$

and the constant penalty coefficients $C_i (i = 1, \ldots, q)$ are selected by the investigator.

Second, suppose that inequality constraints

$$G_i(t,x,u) \geqq 0 \quad (i = 1, \ldots, r) \tag{2.57}$$

which satisfy the following constraint condition are present:

$$\begin{bmatrix} \frac{\partial G_1}{\partial u_1} & \cdots & \frac{\partial G_1}{\partial u_m} & G_1 & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & 0 & G_2 & & 0 \\ \vdots & & \vdots & \vdots & & \ddots & \vdots \\ \frac{\partial G_r}{\partial u_i} & \cdots & \frac{\partial G_r}{\partial u_m} & 0 & 0 & \cdots & G_r \end{bmatrix} \text{ has rank } r. \tag{2.58}$$

This condition is required to guarantee that the control may be determined from the appropriate $G_i = 0$ when a boundary is encountered. The condition is satisfied trivially if the constraints only contain control variables, e.g., $G = 1 - u^2 \geqq 0$, and are independent.

Since the adjoint variables (or Lagrange multipliers) are continuous across corners where control boundaries are encountered, constraints of the form (2.57) may be treated directly with little modification of the program. Let us first describe the procedure for treating a constraint of the form (2.57) before we justify the method.

CONTROL CONSTRAINED CONJUGATE GRADIENT ALGORITHM

Suppose the control is a scalar and $|u| \leqq 1$; the generalization to more than one control and other control constraints is straightforward:

1) At the beginning of the J^{th} iteration, we have a control $u^{(J)}(t)$, $J \in \{0,1,2,\ldots\}$. Define $W_J \equiv \{t: |u^{(J)}(t)| = 1\}$, i.e., the set of points where $u^{(J)}(t)$ is on the boundary. Integrate forward $\dot{x} = f[t,x,u^{(J)}(t)]$ to the stopping condition and set $\lambda^{(J)}(t_f^{(J)})$.
2) Integrate $\dot{\lambda}^{(J)} = -H_x[t,x^{(J)}(t),\lambda^{(J)}, u^{(J)}(t)]$ backwards from $t_f^{(J)}$. Evaluate $H_u^{(J)}(t)$ in the usual way on $[t_0,t_f^{(J)}]$. However, the inner product $<H_u^{(J)}, H_u^{(J)}>$ is defined by:

$$\langle H_u^{(J)}, H_u^{(J)} \rangle = \int_{[t_0, t_f]-W_J} H_u^{(J)^2} \, dt. \tag{2.59}$$

3) Perform the 1-D search with Eqs. (2.49), (2.50).
   In the search, truncate $u^{(J+1)}$ at the boundary if $|u^{(J+1)}(t)| > 1$, i.e., for a trial α_J:

$$\text{if } u^{(J)}(t) - \alpha_J p^{(J)}(t) > 1, \text{ set } u^{(J+1)}(t) = 1 \tag{2.60}$$

$$\text{if } u^{(J)}(t) - \alpha_J p^{(J)}(t) < 1, \text{ set } u^{(J+1)}(t) = -1.$$

   (This step gives us the means for adjusting the set W_J from iteration to iteration.) Return to (1) after α_J and W_{J+1} are determined.

Let us now justify the approach listed above; the method is developed in Ref. 18. The main difficulties in generating a method for treating control constraints are: (i) ensuring that the method is defined in such a way that it can converge to the true minimum (and not a false minimum), and (ii) developing a method consistent with (i) for defining $\langle H_u^{(J)}, H_u^{(J)} \rangle / \langle H_u^{(J-1)}, H_u^{(J-1)} \rangle$ when the iterate has bounded subarcs.

First, we shall consider how the algorithm should behave near the minimum. Suppose that the set W is known beforehand, i.e., the points $t \in [t_0, t_f]$ for which $u^*(t) = \pm 1$ are known. Then, the algorithm need only be concerned with "fine-tuning" the interior control segments. In this regard, we would want $H_u^{(J)}$ and $p^{(J)}$ to be such that it only changes the interior control segments and not the boundary segments. Thus, in the computation of the coefficient of $p^{(J-1)}(t)$ in Eq. (2.49), the effect of the boundary arcs is not included because of the form of Eq. (2.59), and this rule is consistent with requirement (i) above.

In reality, we do not know the set W beforehand, so we must devise a mechanism for the sets W_J to change from iterate to iterate and such that $W_J \to W$. This is accomplished by Step (3) of the procedure defined above; that is, the set W_J is modified in the 1-D search.

CHAPTER 3

PHASE I PROGRAM

3.1 Basic Description

The Phase I Program is designed to minimize a weighted performance index which includes the following effects:

i. Crossrange

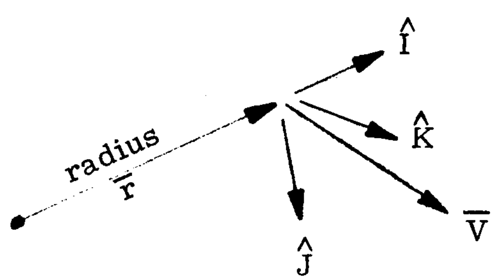
ii. Downrange

iii. Aerodynamic loading

iv. Terminal total heat

v. Terminal altitude

The equations of motion are written in a cartesian coordinate system defined by:

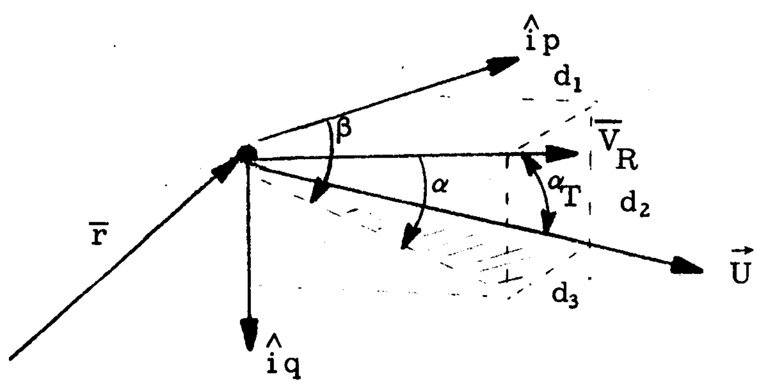


$$\hat{I} = \frac{\bar{r}}{|\bar{r}|}$$

$$\hat{K} = \frac{\bar{r} \times \bar{V}}{|\bar{r} \times \bar{V}|}$$

$$\hat{J} = \hat{K} \times \hat{I}$$

The Aerodynamic Angles are defined by the following coordinate system:



$$\hat{i}p = \frac{\bar{r} \times \bar{V}_R}{|\bar{r} \times \bar{V}_R|} \qquad \hat{i}q = \hat{i}p \times \frac{\bar{V}_R}{V_R}$$

$$d_1 = \frac{\bar{u}}{|u|} \cdot \frac{\bar{V}_R}{V_R}, \quad d_2 = \frac{\bar{u}}{|u|} \cdot \hat{i}q, \quad d_3 = \frac{\bar{u}}{|u|} \cdot \hat{i}p$$

$$\tan \alpha = \frac{d_2}{d_1}, \quad \tan \beta = \frac{d_2}{d_3}, \quad \tan \alpha_t = \frac{\sqrt{d_2^2 + d_3^2}}{d_1}.$$

The state equations are[20]:

$$\dot{\bar{r}} = \bar{V}$$

$$\dot{\bar{V}} = -\frac{\mu}{|\bar{r}|^3}\bar{r} + \rho A |V_R|^2 C_{L_\alpha}\left[-\left(\frac{C_A}{C_{L_\alpha}} + 2\eta\right)\frac{\bar{V}_R}{V_R} + \left[I + (2\eta - 1)\frac{\bar{V}_R\bar{V}_R^T}{V_R}\right]\frac{\bar{u}}{|\bar{u}|}\right]$$

$$\dot{Q} = C_q \rho^{\frac{1}{2}} V_R^{3.15}$$

$$\bar{V}_R = \bar{V} - \bar{V}_A$$

where $\bar{V}_A \equiv$ inertial velocity of the atmosphere. The equations involve the following assumptions:

a. The relative velocity vector $\bar{V}_R$ is in the plane of the vehicle that produces the greatest lift.

b. No aerodynamic moments exist about the center of mass.

The performance index is

$$J = C_1 r_c + C_2 r_d + P_1(h - \bar{h}_f)^2_{t_f} + P_2(Q)^2_{t_f}$$

$$+ P_4 \int_{t_0}^{t_f} \left(\frac{L^2 + D^2}{m^2} - 9g^2\right) \cdot U\left(\frac{L^2 + D^2}{m^2} - 9g^2\right) dt$$

where

r_c = cross range

r_d = down range

$$U(\cdot) = \begin{cases} 1 \text{ if } (\cdot) > 0 \\ 0 \text{ if } (\cdot) \leqq 0 . \end{cases}$$

The Hamiltonian is

$$H = \lambda_r^T \dot{\bar{r}} + \lambda_V^T \dot{\bar{V}} + \lambda_Q \dot{Q} + P_4\left(\frac{L^2 + D^2}{m^2} - 9g^2\right) \cdot U\left(\frac{L^2 + D^2}{m^2} - 9g^2\right)$$

3.2 Subroutine Map

```
MAIN —— ADOTB —— ACROSB
 |
WPRJCG
 |                      ACROSB —— ADOTB           ACROSB —— ADOTB
 |                        |                          |
FWDINT —— RK713 —— COSTFN —— XLAMFN —— OUTPUT
 |          |                   |
 |          |                 ADOTB
 |          |
 |        DERIV1 —— ACROSB —— ADOTB —— ATMOS —— AEROD
 |
 |                      ADOTB —— ACROSB
 |                        |
BAKINT —— RK713 —— GRADFN
 |          |
 |        DERIV2 —— DERIV1 —— ADOTB —— ACROSB
 |                    |
 |                  ACROSB —— ADOTB —— ATMOS —— AEROD
 |
SEKALF
```

3.3 Subroutine Descriptions

MAIN: Reads in all necessary data, sets integration coefficients, computes initial values, and calls on the conjugate gradient subroutine (WPRJCG). On Return, MAIN prints out a message concerning the results of the iteration and prints out the control obtained by that iteration.

A. Namelist Input Data

PI = π

RE = earth's radius

XMU = μ, gravitational constant

OMEGE(3) = angular velocity of the earth

AREA = aerodynamic reference area

ECOEF = heating coefficient

XO(3) = initial position vector

VO(3) = initial velocity vector

TO = initial time

ALTF = desired final altitude

XMACH = desired final Mach number

FLTANG = desired final flight path angle

QMAX = desired final heating value

XMASS = vehicle mass

IOUT = print frequency for forward integration

IOUT2 = print frequency for backward integration

IPRINT1 = print control flag

IPRINT2 = print control flag

DELTS = integration stepsize

IKEY = call flag for output (see FWDINT)

ERRMX = error tolerance for integration routine

ERRMN = not used

TCUT = upper time limit on trajectory

EPST = cutoff tolerance for norm of control change

EPSTF = not used

EPSA = cutoff tolerance for integration altitude cutoff

EPSIT = cutoff tolerance on gradient norm

ERR = cutoff tolerance for small cost change

ITMAX = limit on number of conjugate gradient iterations

ITMX = limit on steps in 1-D search

KOUNTM = limit on iterations for altitude cutoff

CSTR = guess of final cost value

B = control bound (see SEKALF)

PFUN(4) = penalty coefficient vector

CCOST(2) = coefficients in cost functional

DTFM = maximum allowable final time change

XDTFM = fraction of DTFM used to start 1-D search

B. Control Vector Data

IJKU = total number of control points

U(IJKU,4) = control vector and time point

WPRJCG: This subroutine controls the application of the conjugate gradient algorithm. It calls the forward and backward integration routines, directs the one dimensional search, and updates the control vector and terminal time. It checks for algorithm termination on small cost change, total number of iterations, errors in the 1-D search, failure to generate an admissible trajectory on the first trial.

SEKALF (One-Dimensional Search Subroutine): Determines the parameters for the new control value in the conjugate gradient algorithm. Fits a cubic in α to known values of $J(\alpha)$, $\partial J/\partial\alpha$, to obtain min $J(\alpha)$ and then α^* for J min.

FWDINT: Subroutine performs the forward integration of the state variables and calls the subroutines to evaluate the cost functional and final multiplier values.

RK713: 7th Order Runga-Kutta integration scheme called by both FWDINT and BAKINT.

BAKIWT: Subroutine performs the backward integration of the state variables and multiplier equations, and calls on GRADFW to calculate the gradients and store the value at each integration step. The subroutine also determines the new search direction.

DERIV1: Subroutine which calculates the time derivatives of the state variables.

DERIV2: Subroutine which calculates the time derivatives of the multipliers.

ATMOS: Calculates atmospheric parameters.

AEROD: Calculates aerodynamic parameters.

XLAMFN: Computes final multiplier values.

COSTFN: Computes cost functional.

OUTPUT: Subroutine called by FWDINT which prints out desired trajectory data.

3.4 Phase II Program Notes

i) The program obtains the state for the multiplier equations by integrating the state backward from the terminal conditions of the forward state integration (as opposed to storing the state in the forward integration).

ii) Each iterate is terminated on an assumed t_f, which is part of the iteration procedure. The value of t_f for the base trajectory is determined by the trajectory as the time when the desired altitude is reached (thus, the program also has an altitude-cutoff capability).

iii) See Appendix A for a listing of the Phase I Program.

CHAPTER 4

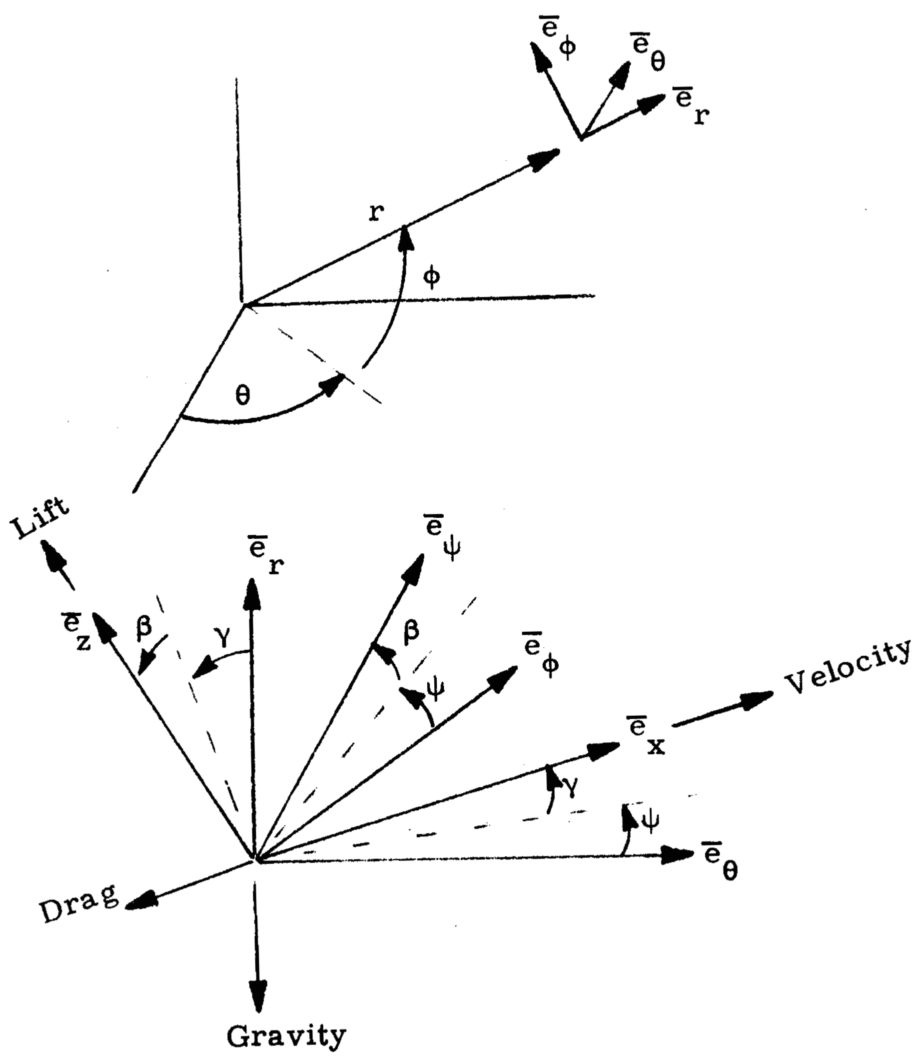
PHASE II PROGRAM

4.1 Basic Description

The Phase II Program is designed to minimize a performance index which includes the following effects:

1. Crossrange
2. Downrange
3. Total heat
4. Peak heating rate
5. Final speed and flight path angle boundary conditions.

Phase II Program uses a nonrotating earth centered spherical coordinate system with an Euler angle body-axis system to define the aerodynamic forces.



The equations of motion assuming a nonrotating earth and no aerodynamic moments are

$$\dot{R} = V \sin\gamma$$

$$\dot{\theta} = \frac{V \cos\gamma \cos\psi}{R \cos\phi}$$

$$\dot{\phi} = \frac{V \cos\gamma \sin\psi}{R}$$

$$\dot{V} = \frac{-\mu \sin\gamma}{R^2} - \frac{D}{m}$$

$$\dot{\gamma} = -\frac{\mu \cos\gamma}{R^2 V} + \frac{V \cos\gamma}{R} + \frac{L}{mV} \cos\beta$$

$$\dot{\psi} = \left[-\frac{V \cos\gamma \cos\psi \sin\phi}{R \cos\phi} - \frac{L \sin\beta}{mV \cos\gamma} \right]$$

where the drag (D) and lift (L) are defined by

$$L = \tfrac{1}{2} \rho S V^2 C_L(\alpha, M)$$

$$D = \tfrac{1}{2} \rho S V^2 C_D(\alpha, M)$$

The cost functional to be minimized is:

$$J = C(1) R_e \phi_f^2 + C(2) R_e \theta_f^2 + C(3)(V(t_f) - \overline{V}_f)^2 + C(4)[\gamma(t_f) - \overline{\gamma}_f]^2$$

$$+ C(5) \int_{t_0}^{t_f} \dot{q}\, dt + C(6) \int_{t_0}^{t_f} \ddot{q}^2\, dt$$

The term $\int_{t_0}^{t_f} \ddot{q}^2 dt$ is an approximate method for minimizing the peak heat rate.

The Hamiltonian is

$$H = C(5)\dot{q}(R, V) + C(6)\ddot{q}^2(R, V, \gamma) + \lambda_1 (V \sin\gamma)$$

$$+ \lambda_2 \left(\frac{V \cos\gamma \cos\psi}{R \cos\phi} \right) + \lambda_3 \left(\frac{V \cos\gamma \sin\psi}{R} \right)$$

$$+ \lambda_4 \left(-\frac{\mu \sin\gamma}{R^2} - \frac{D}{m} \right) + \lambda_5 \left(-\frac{\mu \cos\gamma}{R^2 V} + \frac{V \cos\gamma}{R} + \frac{L}{mV} \cos\beta \right.$$

$$\left. + \lambda_6 \left(-\frac{V \cos\gamma \cos\psi \sin\phi}{R \cos\phi} - \frac{L \sin\beta}{mV \cos\gamma} \right) \right)$$

Forward integration of the state variables is cutoff on a desired altitude.

4.2 Subroutine Map

```
MAIN
  |
WPRJCG
  |
FWDINT — DERIV1 — ATMOS — SETUP — SPLINE
  |        |
  |      XLAMFN — COSTFN — OUTPUT
  |
BAKINT — DERIV2 — ATMOS — SETUP — SPLINE
  |        |
  |      GRADFN
  |
SEKALF
```

4.3 Subroutine Descriptions

MAIN: Reads in all necessary input parameters, sets up spline interpolation of aerodynamic coefficients and calls the conjugate gradient subroutine WPRJCG. On Return, MAIN prints out message concerning the results of the iteration and prints out the control obtained by that iteration.

A. Namelist Data

PI = π

RE = radius of the earth

XMU = μ, gravitational constant

OMEGE = not used

AREA = aerodynamic reference area

ECOEF = heating coefficient

DELT = integration stepsize

IKEY = call flag for OUTPUT

ERRMX = not used

ERRMN = not used

TCUT = upper time limit on trajectory

EPST = cutoff tolerance for norm of control change

EPSTF = not used

EPSA = cutoff tolerance for integration altitude cutoff

EPSIT = cutoff tolerance on gradient norm

ERR = cutoff tolerance for small cost change

ITMAX = limit on number of conjugate gradient iterations

ITMX = limit on steps in 1-D search

KOUNTM = limit on iterations for altitude cutoff

CSTR = guess of final cost value

B = control bound (see SEKALF)

C(7) = coefficients in cost functional

DTFM = not used

XDTFM = not used

SVARO(6) = initial state variables

TO = initial time

ALTF = cutoff altitude

XMACH = not used

FLTANG = not used

GAMMF = final flight path angle

VF = final velocity

XMASS = vehicle mass

IOUT = print frequency for forward integration

IOUT2 = print frequency for backward integration

IPRINT1 = print control flag

IPRINT2 = print control flag

B. Control Data

IJKU = total number of control points

U(IJKU,3) = control vector and time points

C. Aerodynamic Data

N1, N2 = dimensions of coefficient array

Y(N1,N2,2) = coefficient array

(See sample program for input format)

See Chapter 3.3 for descriptions of:

WPRJCG

SEKALF

DERIV1

DERIV2

ATMOS

XLAMFN

GRADFN

COSTFN

OUTPUT

FWPIWT

BAKINT

SETUP - SPLINE - Subroutine computes aerodynamic coefficients based upon piecewise cubic spline interpolation. Input is angle of attack (α) and Mach number (M); returned are the values of $C_L, C_D, \partial C_L/\partial M, \partial C_D/\partial M, \partial C_D/\partial \alpha$.

(For test runs the aerodynamics were approximated by:

$$C_D = 2.2 \sin^3\alpha + .08$$

$$C_L = 2.2 \sin^2\alpha \cos \alpha + .01.)$$

4.4 Phase II Program Notes

i) The state values for the backward integration of the multiplier equations are stored during the forward integration (as opposed to backward integration for the state). The program currently can store the state at 999 time points.

ii) All trajectories terminate at a specified, desired altitude. The modification to the transversality conditions is discussed in Chapter 2. Since the terminal time of the N + 1 trajectory, say $t_f^{(N+1)}$, may be larger than $t_f^{(N)}$ (since h_f is the cutoff condition), a linear extrapolation of the control is used on $[t_f^{(N)}, t_f^{(N+1)}]$.

iii) See Appendix B for a listing of the Phase II Program.

CHAPTER 5

CONCLUSIONS AND RECOMMENDATIONS

5.1 Summary

Two computer programs for shuttle reentry optimization have been developed. The programs make extensive use of subroutines so that they may be adapted to other atmospheric optimization problems with little difficulty. Because of contract budget restrictions and the long flight times of realistic shuttle reentry trajectories, the programs have only been checked out with respect to programming errors. In the next section, suggestions for a study of the convergence properties of the programs will be presented. Also, our limited experience obtained with the programs will be discussed. However, with respect to these comments, it should be remembered that no controlled study was performed, and thus, the comments are somewhat tenuous.

5.2 Conclusions and Recommendations

1. Before an extensive analysis of the optimization of reentry trajectories is undertaken, it is recommended that a carefully controlled study of numerical integration procedures be performed for reentry problems in which: (a) the controls are piecewise linear (or possibly higher-order splines) in an integration step, (b) the aerodynamic data is given in tabular form, and (c) the vehicle is a relatively low-drag vehicle (e.g., the high-crossrange shuttle). Much of our time was devoted to determining an acceptable numerical integration package while the optimization procedure was the major goal of the study. We found that RK 7-13 was an excellent scheme with constant aerodynamics and smooth controls; however, with piecewise linear controls and spline-fit aerodynamics, its performance was reduced substantially. For this reason, a fourth-order, predictor-corrector scheme with fixed stepsize is employed in the Phase II-Program. Research should be conducted to make the problem suitable for use with RK 7-13 (or some other high-order scheme) to shorten the long integration times.

2. Because of the relatively low-drag characteristics of the

high-crossrange shuttle, arbitrary initial estimates of the controls in any optimization program may cause highly oscillatory trajectories. This is due to the fact that the path angle may become positive (positive path angle is above the local horizontal) and oscillate about zero degrees. Thus, it is recommended that, if possible, initial control estimates be chosen so that γ remains negative. Some investigators have used artificial means to insure $\gamma \leqq 0$, e.g., impose a state variable inequality constraint, add damping to the initial iterates, increase the drag in the initial iterates. This problem may be accentuated by an inaccurate numerical integration scheme because $\dot{\gamma}$ is essentially the difference between two terms of the same order of magnitude. Thus, $\dot{\gamma}$ may become positive because of numerical error when its true physical value is negative.

3. Neither program uses nondimensional variables. If the rate of convergence is slow in simulations, nondimensionalization of the variables may improve the rate.

4. Most of the investigations which have applied the conjugate gradient method to optimal control problems have been of low-dimension, near-linear, and fixed final time. Two exceptions are Refs. 21 and 22. In these studies, it was found that the method did not perform satisfactorily on a problem with tight terminal conditions[21] and a free-final time problem[22]. Since the two programs of this report treat the free final time problem in two different ways, trends as to which method is best would be useful information.

5. In the Phase II-Program, $\int_{t_0}^{t_f} \ddot{q}^2 dt$ is used in the performance index to penalize large heat rate slopes, and, thus, should aid in "flattening-out" the heating rate. This conjecture should be tested since if it serves to flatten the peak heating rate, it might be a simple way of controlling peak heating rate in an on-board, optimization oriented guidance scheme.

6. A convenient test problem for reentry is the maximum crossrange

problem. In this problem, the optimal control should consist of an angle of attack which causes $(L/D)_{max}$ and a bank angle which is initially near 90° and which decreases (nearly linearly) toward 0° as time increases to t_f. In our limited testing of the Phase II-Program on the IBM 360/67, a typical iterate (including the 1-D search) required about one minute of CPU time for a double precision, 2000 second (real-time) trajectory with a fixed stepsize of four seconds.

7. As noted above, a typical iterate requires approximately one minute of CPU time. Of course, the large amount of computer time is mainly due to the numerical integration requirements. Hopefully, more efficient numerical integration schemes will be developed for use in conjunction with function-space gradient-type algorithms. In this development one should keep in mind that both forward and backward integrations are required, and this heavily influences the choice of a variable stepsize integration scheme. A possibility in this direction is spline numerical integration schemes since they result in "global" information as opposed to discrete data.

REFERENCES

1.) H. J. Kelley, "Method of Gradients," in Optimization Techniques (G. Leitmann, Ed.), Academic Press, New York, 1962, Chapter 6.

2.) A.E. Bryson and Y.C. Ho, Applied Optimal Control, Blaisdell, Waltham, Mass., 1969, Chapter 7.

3.) P. Salvato, Personal Communication, TRW-Systems, Houston, Texas, 1971.

4.) H. J. Kelley, B. R. Uzzell, and S.S. McKay II, "Rocket Trajectory Optimization by a Second-Order Numerical Technique," AIAA J., Vol.7, No.5, pp.879-884, May 1969.

5.) H.C. Sullivan and B.R. Uzzell, "Optimal Boost Trajectories for the Shuttle Vehicle," MSC Internal Note No.71-FM-4, January 13, 1971.

6.) L. S. Lasdon, S.K. Mitter, and A.D. Waren, "The Conjugate Gradient Method for Optimal Control Problems," IEEE Trans. on A.C., Vol.AC-12, pp.132-138, April 1967.

7.) L.B. Horwitz and P.E. Sarachik, "Davidon's Method in Hilbert Space," SIAM J. Appl. Math., Vol.16, No.4, 1968, pp. 676-695.

8.) IBM System/360, Scientific Subroutine Package, (360A-CM-03X) Version III (1968) IBM Technical Publication Dept., pp.220-225. (Corrected by IBM Memo, May 1969.)

9.) I.L. Johnson and J.L. Kamm, "Parameter Optimization and the Space Shuttle," Proceedings of the 1971 JACC, St.Louis, Mo., August 1971, pp. 776-781.

10.) J.L.Kamm and I.L. Johnson, "Optimal Shuttle Trajectory-Vehicle Design Using Parameter Optimization," AAS/AIAA Paper No.329, presented at AAS/AIAA Astrodynamics Conference, Ft. Lauderdale, Fla., August 1971. (Also, see Paper No.328.)

11.) J.K. Willoughby and B.L. Pierson, "A Constraint-Space Conjugate Gradient Method for Function Minimization and Optimal Control Problems," to appear in International Journal of Control, 1972.

12.) M.R. Hestenes and E. Stiefel, "Method of Conjugate Gradients for Solving Linear Systems," J. of Res. of the Nat. Bur. of Stds., Vol.

49, No.6, pp. 409-436, 1952.

13.) W.C. Davidon, "Variable Metric Method for Minimization," A.E.C. Research and Development Report ANL-5990 (Rev.), November 1959.

14.) R. Fletcher and M.J.D. Powell, "A Rapidly Convergent Descent Method for Minimization," Computer J., Vol.6, No.2, 1963, pp.163-168.

15.) R.M. Hayes, "Iterative Methods of Solving Linear Problems on Hilbert Space," Natl. Bur. of Stds. App. Math. Series, Vol.39, pp. 71-104, 1954.

16.) H. Antosiewicz and W. Rheinboldt, "Numerical Analysis and Functional Analysis," in Survey of Numerical Analysis (J.Todd, Ed.), McGraw-Hill, 1962, Chap.14.

17.) J.W. Daniel, "The Conjugate Gradient Method for the Numerical Solution of Linear and Nonlinear Operator Equations," SIAM J. on Num. Anal., Vol.4, No.1, pp. 10-26, 1967.

18.) B.Pagurek and C.M. Woodside, "The Conjugate Gradient Method for Optimal Control Problems with Bounded Control Variables," Automatica, Vol.4, pp. 337-349, 1968.

19.) W.F. Denham and A. E. Bryson, "Optimal Programming Problems with Inequality Constraints II: Solution by Steepest-Ascent," AIAA J., Vol.2, No.1, pp. 25-34, Jan.1964.

20.) K.R.Brown, E. F. Harrold, and G. W. Johnson, "Some New Results on Space Shuttle Atmospheric Ascent Optimization," AIAA Paper No. 70-978, August 1970.

21.) R.K. Mehra and A.E. Bryson, "Conjugate Gradient Methods with an Application to V/STOL Flight-Path Optimization," AIAA J. Aircraft, Vol.6, No.2, pp. 123-128, 1969.

22.) C.L. Pu, "Numerical Solution of Dynamic Optimization Problems," MIT Draper Lab Report E-2549, Feb. 1971.

APPENDIX A

LISTING OF PHASE I PROGRAM

```
C   MAIN CONTROLLING ROUTINE
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION OMEGE(3),PFUN(4),CCOST(2),XO(3),VO(3),URXVO(3),
     1 GRAD(999,4),SERCH(999,4),U(999,4),TEMP(3),OMEGO(3,3),UTM(3)
      DIMENSION ALPH(13), BETA(13,12), CH(13)
      COMMON/RKCOE/ALPH,BETA,CH
      COMMON/CONS1/PI,RE,XMU,OMEGE,AREA,FCOEF,GNOT,OMEGO
      COMMON/CONS2/DELTS,      ERRMX,ERRMN,ICUT,EPST,EPST-,EPSA,EPSIT,
     1 ERR,ITMAX,ITMX,KOUNTM,IKEY
      COMMON/CONS3/CSTP,B,PFUN,CCOST,DTFM,XDTFM
      COMMON/STATO/XO,VO,XOMAG,VOMAG,URXVO,TO
      COMMON/STATF/ALTF,XMACH,FLTANG,QMAX,SINCR,COSCR,SINDR,COSDR
      COMMON/STATE/ALT,XMASS,UTM,UTMAG
      COMMON/CNTRL/GRAD,SERCH,U,ASTR,STF,TF,KJIS,IJKU,ISTAR
      COMMON/PRINT/IOUT,IOUT2,IPRNT1,IPRNT2
      NAMELIST/ANAME/PI,RE,XMU,OMEGE,AREA,FCOEF,DELTS,IKEY,ERRMX,ERRMN,
     1 ICUT,EPST,EPSTF,EPSA,EPSIT,ERR,ITMAX,ITMX,KOUNTM,CSTR,B,PFUN,
     2 CCOST,DTFM,XDTFM,XO,VO,TO,ALTF,XMACH,FLTANG,QMAX,XMASS
     3 ,IOUT,IOUT2,IPRNT1,IPRNT2
C   READ IN DATA
    1 READ(5,ANAME)
      READ(7,700) IJKU
      READ(7,701) ((U(I,J),J=1,4),I=1,IJKU)
      WRITE(6,ANAME)
C    COMPUTE INITIAL VARIABLES
      XOMAG = DSQRT(ADOTB(XO,XO))
      VOMAG = DSQRT(ADOTB(VO,VO))
      CALL ACROSB(XO,VO,TEMP,1,URXVO)
      GNOT = XMU/RE**2
      DO 5 I=1,3
    5 OMEGO(I,I) = 0.000
      OMEGO(2,1) = OMEGE(3)
      OMEGO(3,1) = -OMEGE(2)
      OMEGO(3,2) = OMEGE(1)
      OMEGO(1,2) = -OMEGE(3)
      OMEGO(1,3) = OMEGE(2)
      OMEGO(2,3) = -OMEGE(1)
C     CONSTANTS FOR INTEGRATION SUBROUTINE
      DO 260 I=1,13
      DO 250 J=1,12
  250 BETA(I,J)=0.
      ALPH(I)=0.
  260 CH(I)=0.
      CH(6)=34./105.
      CH(7)=9./35.
      CH(8)=CH(7)
      CH(9)=9./280.
      CH(10)=CH(9)
      CH(12)=41./840.
      CH(13)=CH(12)
      ALPH(2)=2./27.
      ALPH(3)=1./9.
      ALPH(4)=1./6.
      ALPH(5)=5./12.
      ALPH(6)=.5
      ALPH(7)=5./6.
      ALPH(8)=1./6.
      ALPH(9)=2./3.
      ALPH(10)=1./3.
      ALPH(11)=1.
```

```
      ALPH(13)=1.
      BETA(2,1)=2./27.
      BETA(3,1)=1./36.
      BETA(4,1)=1./24.
      BETA(5,1)=5./12.
      BETA(6,1)=.05
      BETA(7,1)=-25./108.
      BETA(8,1)=31./300.
      BETA(9,1)=2.
      BETA(10,1)=-91./108.
      BETA(11,1)=2383./4100.
      BETA(12,1)=3./205.
      BETA(13,1)=-1777./4100.
      BETA(3,2)=1./12.
      BETA(4,3)=1./8.
      BETA(5,3)=-25./16.
      BETA(5,4)=-BETA(5,3)
      BETA(6,4)=.25
      BETA(7,4)=125./108.
      BETA(9,4)=-53./6.
      BETA(10,4)=23./108.
      BETA(11,4)=-341./164.
      BETA(13,4)=BETA(11,4)
      BETA(6,5)=.2
      BETA(7,5)=-65./27.
      BETA(8,5)=61./225.
      BETA(9,5)=704./45.
      BETA(10,5)=-976./135.
      BETA(11,5)=4496./1025.
      BETA(13,5)=BETA(11,5)
      BETA(7,6)=125./54.
      BETA(8,6)=-2./9.
      BETA(9,6)=-107./9.
      BETA(10,6)=311./54.
      BETA(11,6)=-301./82.
      BETA(12,6)=-6./41.
      BETA(13,6)=-289./82.
      BETA(8,7)=13./900.
      BETA(9,7)=67./90.
      BETA(10,7)=-19./60.
      BETA(11,7)=2133./4100.
      BETA(12,7)=-3./205.
      BETA(13,7)=2193./4100.
      BETA(9,8)=3.
      BETA(10,8)=17./6.
      BETA(11,8)=45./82.
      BETA(12,8)=-3./41.
      BETA(13,8)=51./82.
      BETA(10,9)=-1./12.
      BETA(11,9)=45./164.
      BETA(12,9)=3./41.
      BETA(13,9)=33./164.
      BETA(11,10)=18./41.
      BETA(12,10)=6./41.
      BETA(13,10)=12./41.
      BETA(13,12)=1.
C CALL CONJUGATE GRADIENT ROUTINE
      CALL WPRJCG(IER)
      GO TO (10,20,30,40,50,60,70,80,90,100),IER
   10 CONTINUE
```

```
   20 WRITE(6,520)
      GO TO 101
   30 WRITE(6,530)
      GO TO 101
   40 WRITE(6,540)
      GO TO 101
   50 WRITE(6,550)
      GO TO 101
   60 WRITE(6,560)
      GO TO 101
   70 WRITE(6,570)
      GO TO 101
   80 WRITE(6,580)
      GO TO 101
   90 WRITE(6,590)
      GO TO 101
  100 WRITE(6,600)
  101 CONTINUE
      WRITE(8,625) IJKU
      WRITE(8,650) ((U(K,L),L=1,3),K=1,IJKU)
      STOP
 500  FORMAT(2I4)
 505  FORMAT(8F10.0)
  520 FORMAT(1H0,5X,'ONE-D SEARCH FAILED TO FIND A MINIMUM')
  530 FORMAT(1H0,5X,'COST IS NOT DECREASING IN SEARCH DIRECTION')
  540 FORMAT(1H0,5X,'CONVERGENCE ON SMALL CONTROL CHANGE')
  550 FORMAT(1H0,5X,'LITTLE COST CHANGE IN LAST TWO ITERATIONS')
  560 FORMAT(1H0,5X,'FAILED TO CONVERGE IN ITMAX ITERATIONS')
  570 FORMAT(1H0,5X,'INITIAL TRAJECTORY FAILED TO REACH CUT-OFF ALT')
  580 FORMAT(1H0,5X,'TOO MANY INTEGRATIONS STEPS REQUIRED ')
  590 FORMAT(1H0,5X,'BACKWARD INTEGRATED TRAJECTORY ERRORS')
  600 FORMAT(1H0,5X,'CONVERGENCE ON ZERO GRADIENT NORM')
 625  FORMAT(' ',I5)
 650  FORMAT(' ',3D26.16)
700   FORMAT(I5)
 750  FORMAT(3D26.16)
      END
```

```
C  SUBROUTINE WPRJCG
      SUBROUTINE WPRJCG(IER)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION XJ(8,1),XLAMF(7),SERCH(999,4),U(999,4),TEMPU(999,3)
     1,PFUN(4),CCOST(2),GRAD(999,4)
      COMMON/CONS2/DELTS,·      ERRMX,ERRMN,TCUT,EPST,EPSTF,EPSA,EPSIT,
     1 ERR,ITMAX,ITMX,KOUNTM,IKEY
      COMMON/CONS3/CSTR,B,PFUN,CCOST,DTFM,XDTFM
      COMMON/CNTRL/GRAD,SERCH,U,ASTR,STF,TF,KJIS,IJKU,ISTAR
      ITER=0
C PERFORM FORWARD INTEGRATION TO ALTITUDE CUT-OFF
8     ISTAR=0
      IFLAG=1
      CALL FWDINT(COST,XJ,TF,XLAMF,DCDTF,IFLAG)
      IF(IFLAG .NE. 1) GO TO 94
C PERFORM BACKWARD INTEGRATION
9     CALL BAKINT(XJ,XLAMF,TF,ITER,DCOST,XNORMS,DCDTF)
      IF(ITER-ITMAX) 7,95,93
7     CSAVE=COST
      ITNUM = ITER + 1
      WRITE(6,603) ITNUM
  603 FORMAT(1H0,5X,'ITERATION NUMBER',I5//)
C ENTER 1-D SEARCH
      ISTAR=1
      IFLAG=2
      KFLG = 0
      JKNT = 0
      TFS=TF
      IFLG=0
      IF(DTFM .LE. 0.D00) GO TO 10
C HAVE A RESTRICTION ON FINAL TIME CHANGES, COMPUTE FIRST GUESS
      ASTR=DABS(XDTFM*DTFM/STF)
      IFLG=-1
10    CALL SEKALF(COST,DCOST,ASTR,CSTR,XNORMS,B,TO,TF,ITMX,IFLG)
      JKNT = JKNT + 1
      IF(IPRNT1.GT.0) WRITE(6,600) JKNT,ASTR
  600 FORMAT(5X,'1-D SEARCH TRIAL =',I5,5X,'PARAMETER =',D24.16)
      IF(IFLG .GT. ITMX) GO TO 11
      DTF=ASTR*STF
      IF(DTFM .LE. 0.D00) GO TO 15
      IF(DABS(DTF) .LE. DTFM) GO TO 15
      WRITE(6,200)
200   FORMAT(1H0,5X,'PARAMETER VALUE CAUSES LARGE TF CHANGE')
      ASTR=DABS(DTFM/STF)
      DTF=ASTR*STF
15    TF=TFS-DTF
      CALL FWDINT(COST,XJ,TF,XLAMF,DCDTF,IFLAG)
      GO TO 10
11    IF(IFLG .GT.ITMX+1)GO TO 99
C   CHECK FOR SMALL CONTROL NORM CHANGE
      UNORM = ASTR*XNORMS
      IF(UNORM.LT.EPST)  GO TO 98
      DTF = ASTR*STF
      TF = TFS - DTF
C PERFORM FORWARD INTEGRATION
      IFLAG=3
      CALL FWDINT(COST,XJ,TF,XLAMF,DCDTF,IFLAG)
      IF(COST.LT.CSAVE) GO TO 12
      IFLAG = 2
      KFLG = KFLG + 1
```

```
      IF(KFLG.GE.ITMX) GO TO 99
      IFLG = 1
      GO TO 10
C HAVE FOUND INTERPOLATED VALUE, UPDATE CONTROL AT FREQUENCY OF SEARCH
12    KTAU=1
      DO 60 L=1,KJIS
      TAU=SERCH(L,4)
52    IF(U(KTAU,4) .GT. TAU) GO TO 54
      IF(KTAU .GE. IJKU) GO TO 57
      KTAU=KTAU+1
      GO TO 52
C TAU LIES BETWEEN U(KTAU-1,4) AND U(KTAU,4)
54    IF(KTAU .EQ. 1) GO TO 56
C USE LINEAR INTERPOLATION IN CNTRL DIRECTION GENERATION
      DO 55 K=1,3
      TEMPU(L,K)=-ASTR*SERCH(L,K)+(U(KTAU,K)-U(KTAU-1,K))*(TAU-U(KTAU-1,
     24))/(U(KTAU,4)-U(KTAU-1,4))+U(KTAU-1,K)
55    CONTINUE
      GO TO 60
C     USE LINEAR EXTRAPOLATION
56    KTAU=2
57    DO 58 K=1,3
      TEMPU(L,K)=-ASTR*SERCH(L,K)+(U(KTAU,K)-U(KTAU-1,K))*(TAU-U(KTAU,4)
     2)/(U(KTAU,4)-U(KTAU-1,4))+U(KTAU,K)
58    CONTINUE
60    CONTINUE
C COMPUTE NEW FINAL TIME
      TF=TFS-ASTR*STF
      DO 62 L=1,KJIS
      DO 61 M=1,3
61    U(L,M)=TEMPU(L,M)
62    U(L,4)=SERCH(L,4)
      IJKU=KJIS
   65 CONTINUE
      ISTAR=0
C CHECK CHANGE IN COST VALUES
      IF(DABS(COST-CSAVE) .LT. ERR) GO TO 96
      ITER=ITER+1
      GO TO 9
C 1-D SEARCH ERRORS
99    IER=2
      IF(IFLG .EQ. ITMX+2)IER=3
      RETURN
C  HAVE CONVERGENCE DUE TO SMALL CONTROL NORM CHANGE
   98 IER = 4
      RETURN
C HAVE CONVERGENCE DUE TO NO COST CHANGE
96    IER=5
      RETURN
C HAVE EXCEEDED PERMITTED NUMBER OF CG STEPS
95    IER=6
      RETURN
C HAVE FAILED TO REACH ALTITUDE CUT-OFF
94    IER=7
      RETURN
93    IF(ITER-(ITMAX+2)) 92,91,90
C NOT ENOUGH STORAGE SPACE FOR GRADIENT
92    IER=8
      RETURN
C CANNOT FIND INTEGRATION CUT-OFF POINT
```

```
91     IER=9
       RETURN
C HAVE CONVERGED ON GRADIENT NORM
90     IER=10
       RETURN
       END
```

```
      SUBROUTINE SEKALE(COST,DCOST,ASTAR,CSTAR,SNORM,B,TO,TF,ITMAX,
     1 IFLAG)
      DOUBLE PRECISION COST, DCOST, ASTAR, CSTAR, SNORM, B, TO ,TF,
     1FUNT,ALF,BM,G,DETERM,AA,BB,CC,XNORM,BSTAR
      DIMENSION FUNT(20),ALF(20),BM(3,3),G(3)
      IF(IFLAG.GT.0) GO TO 20
      IF(DCOST.GE.0.0D0) GO TO 15
      IF(ASTAR .NE. 0.0D0)  GO TO 11
C   COMPUTE FIRST PARAMETER
      ASTAR = 2.0D0*(CSTAR - COST)/DCOST
      IF(B.LE.0.0D0) GO TO 10
      XNORM=B*DSQRT(TF-TO)/SNORM
 11   IF(ASTAR.LE.0.0D0.OR.ASTAR.GT.XNORM) ASTAR=XNORM
      FUNT(1)=COST
      ALF(1) = 0.0D0
      IFLAG = 1
      RETURN
 10   XNORM=1.0D0/SNORM
      GO TO 11
C   SLOPE OF COST IS NOT NEGATIVE
   15 WRITE(6,100) DCOST
  100 FORMAT(1H0,10X,'THE VALUE OF THE NON-NEGATIVE SLOPE IS',D24.16)
      IFLAG = ITMAX + 2
      RETURN
C   COMPUTE SECOND PARAMETER
   20 IF (IFLAG.GT.1) GO TO 30
      ALF(2) = ASTAR
      FUNT(2) = COST
      IF(FUNT(2).LE.FUNT(1)) GO TO 25
      ASTAR = ALF(2)/2.0D0
      IFLAG = 2
      RETURN
   25 IFLAG = 2
      GO TO 31
C   COMPUTE THIRD PARAMETER
   30 IF (IFLAG.LT.3) GO TO 59
      ALF(IFLAG) = ASTAR
      FUNT(IFLAG) = COST
      IF(FUNT(IFLAG).GT.FUNT(IFLAG-1)) GO TO 50
   31 ASTAR = ALF(2)*(2.0D0)**(IFLAG-1)
      IF(IFLAG.GE.ITMAX) GO TO 40
      IFLAG = IFLAG + 1
      RETURN
C   CANNOT FIND A MINIMUM
   40 WRITE(6,101)
  101 FORMAT(1H0,10X,'SEARCH HAS EXCEEDED MAXIMUM NUMBER OF STEPS')
      IFLAG = ITMAX + 2
      RETURN
C   GET DATA FOUR POINT INTERPOLATION
   50 IF(IFLAG.EQ.3) GO TO 60
      IFLAG = IFLAG - 3
      DO 51 I=1,3
      BM(I,3) = ALF(IFLAG) - ALF(IFLAG+I)
      BM(I,2) =(ALF(IFLAG) + ALF(IFLAG+I))*BM(I,3)/2.0D0
      BM(I,1) =(ALF(IFLAG)**3 - ALF(IFLAG+I)**3)/3.0D0
   51 G(I) = FUNT(IFLAG) - FUNT(IFLAG+I)
      GO TO 70
C GET DATA FOR  THREE POINT AND SLOPE INTERPOLATION
59    ALF(3) = ASTAR
      FUNT(3) = COST
```

```
   60 G(1) = (ALF(3) - ALF(2))*(ALF(3)*ALF(2))**2
      G(2) = FUNT(2)*ALF(3)**2 - FUNT(3)*ALF(2)**2 - ALF(2)*ALF(3)
     1 *(ALF(3)-ALF(2))*DCOST - (ALF(3)**2 - ALF(2)**2)*FUNT(1)
      G(2) = -3.0D0*G(2)/G(1)
      G(3) = FUNT(2)*ALF(3)**3 - FUNT(3)*ALF(2)**3 - ALF(2)*ALF(3)
     1 *(ALF(3)**2 -ALF(2)**2)*DCOST - (ALF(3)**3-ALF(2)**3)*FUNT(1)
      G(3) = 2.0D0*G(3)/G(1)
      AA = G(2)
      BB = G(3)
      CC= DCOST
      GO TO 71
C   SOLVE FOR COEFFICIENTS BY CRAMER'S RULE
   70 DETERM = BM(1,1)*(BM(2,2)*BM(3,3)-BM(3,2)*BM(2,3))
     1 + BM(1,2)*(BM(3,1)*BM(2,3) - BM(3,3)*BM(2,1))
     2 + BM(1,3)*(BM(2,1)*BM(3,2) - BM(3,1)*BM(2,2))
      AA = (G(1) *(BM(2,2)*BM(3,3) - BM(3,2)*BM(2,3))
     1    + G(2)*(BM(3,2)*BM(1,3) - BM(1,2)*BM(3,3))
     2    + G(3)*(BM(1,2)*BM(2,3) - BM(2,2)*BM(1,3)))/DETERM
      BB = (G(1)*(BM(2,3)*BM(3,1) - BM(3,3)*BM(2,1))
     1     + G(2)*(BM(1,1)*BM(3,3) - BM(3,1)*BM(1,3))
     2     + G(3)*(BM(2,1)*BM(1,3) - BM(2,3)*BM(1,1)))/DETERM
      CC = (G(1)*(BM(2,1)*BM(3,2) - BM(3,1)*BM(2,2))
     1     + G(2) *(BM(1,2)*BM(3,1) -BM(3,2)*BM(1,1))
     2     + G(3)*(BM(1,1)*BM(2,2) - BM(2,1)*BM(1,2)))/DETERM
C   COMPUTE MINIMIZING ALPHA
   71 IF(BB.GT.0.0D0) GO TO 73
      ASTAR = (-BB + DSQRT(BB**2 - 4.0D0*AA*CC))/AA/2.0D0
   72 IFLAG = ITMAX + 1
      RETURN
   73 ASTAR = -2.0D0*CC/(BB + DSQRT(BB**2 - 4.0D0*AA*CC))
      GO TO 72
      END
```

```
C     SUBROUTINE FWDINT
      SUBROUTINE FWDINT(COST,XJ,TF,XLAMF,DCDTF,IFLAG)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION XJ(8,1),YPP(8,4,1),DPSAVE(8,1),ER(8,1),DV(8,1),P(8,1),
     2U(3,1),RPAR(1),IPAR(1),XO(3),VO(3),XSAVE(8),XLAMF(7),PFUN(4)
     3,OMEGE(3),OMEGO(3,3),CCOST(2),URXVO(3),UTM(3)
      COMMON/CONS1/PI,RE,XMU,OMEGE,AREA,FCOEF,GNOT,OMEGO
      COMMON/CONS2/DELTS,ERRMX,ERRMN,TCUT,EPST,EPSTF,EPSA,EPSIT,ERR,
     1 ITMAX,ITMX,KOUNTM,IKEY
      COMMON/CONS3/CSTR,B,PFUN,CCOST,DTFM,XDTFM
      COMMON/STATO/XO,VO,XOMAG,VOMAG,VRXVO,TO
      COMMON/STATF/ALTF,XMACH,FLTANG,QMAX,SINCR,COSCR,SINDR,COSDR
      COMMON/STATE/ALT,XMASS,UTM,UTMAG
      COMMON/PRINT/IOUT,IOUT2,IPRNT1,IPRNT2
      EXTERNAL  DERIV1
C     INITIALIZATION
      TEST=XOMAG-RE
      TEND=TCUT
      LMN=0
      IPAR(1)=0
      IF(IFLAG .NE. 1)  TEND=TF
      DELT=DELTS
      DO 10  I=1,3
   10 XJ(I,1)=XO(I)
      DO  11  I=4,6
 11   XJ(I,1)=VO(I-3)
      XJ(7,1)=0.0D0
      XJ(8,1)=0.0D0
      TM=TO
      INX=1
      TOL=ERRMX
      CALL DERIV1(TM,XJ,P,6,1,8,1,RPAR,IPAR,1)
      COST=0.0D0
      DCDTF=0.0D0
      KOUNT=0
C     PERFORM INTEGRATION AND COMPUTE COST
   20 CONTINUE
      DO  21  I=1,8
   21 XSAVE(I)=XJ(I,1)
      TESTP=TEST
      LTEST=MOD(LMN,IOUT)
      IF(IFLAG.EQ.2.AND.IPRNT1.LT.2) LTEST=1
      IF(LTEST.EQ.0)  WRITE(6,600) TM,(XJ(I,1),I=1,8),(UTM(J),J=1,3)
      IF(IFLAG.NE.2.AND.IKEY.GT.0.AND.LTEST.EQ.0) CALL OUTPUT(XJ,UTM)
      TI=TM
      TM=TM+DELT
      CALL RK713(INX,8,200,M,TOL,TI,TM,XJ,XJ,DV,P,DERIV1)
      LMN=LMN+1
  600 FORMAT(1H0,5X,'TIME=',1PD24.16/32X,'STATE'/6X,1P4D24.16/
     1 6X,1P4D24.16/39X,'CONTROL'/6X,1P3D24.16)
   23 CONTINUE
      IF(TM-TEND)22,90,91
   22 IF(IFLAG .NE. 1) GO TO 20
      ALT=DSQRT(XJ(1,1)**2+XJ(2,1)**2+XJ(3,1)**2)-RE
      TEST=ALT-ALTF
      IF(TEST) 70,80,20
C     FINAL ALTITUDE ITERATION
   70 CONTINUE
      INDX=4
 71   IF(DABS(TEST) .LT. EPSA)  GO TO 80
```

```
      IF(KOUNT .GT. KOUNTM) GO TO 101
      IF(TEST)  78,80,72
 72   DO 73 I=1,8
 73   XSAVE(I)=XJ(I,1)
      TESTP=TEST
      GO TO 75
 78   DO 79  I=1,8
 79   XJ(I,1)=XSAVE(I)
      TN=TM
      TESTN=TEST
      TM=TM-DELT
 75   DELT=TESTP*(TN-TM)/(TESTP-TESTN)
      TI=TM
      TM=TM+DELT
      CALL RK713(INX,8,200,M,TOL,TI,TM,XJ,XJ,DV,P,DERIV1)
      KOUNT=KOUNT+1
      ALT=DSQRT(XJ(1,1)**2+XJ(2,1)**2+XJ(3,1)**2)-RE
      TEST=ALT-ALTF
      GO TO 71
 90   GO TO(100,105,81),IFLAG
 91   IF(IFLAG .EQ. 1) GO TO 100
C     GET FINAL POINT
      DO 92  I=1,8
 92   XJ(I,1)=XSAVE(I)
      TM=TM-DELT
      DELT=TF-TM
      TI=TM
      TM=TM+DELT
      CALL RK713(INX,8,200,M,TOL,TI,TM,XJ,XJ,DV,P,DERIV1)
      IF(IFLAG .EQ. 2) GO TO 105
      GO TO 81
 80   TF=TM
 81   COST=COSTFN(XJ)
      CALL XLAMFN(XLAMF,XJ)
C     COMPUTE COST CHANGE WITH RESPECT TO TF
      CALL DERIV1(TF,XJ,P,1,1,8,1,RPAR,IPAR,1)
      IF(DTFM.LE.0.0D0.AND.XDTFM.GT.0.0D0) GO TO 104
      DO  82 I=1,7
 82   DCDTF=DCDTF+XLAMF(I)*P(I,1)
      IF(P(8,1).LT.0.0D0) GO TO 104
      DCDTF=DCDTF+PFUN(4)*P(8,1)
 104  WRITE(6,600)  TM,(XJ(I,1),I=1,8),(UTM(J),J=1,3)
      IF(IFLAG.NE.2.AND.IKEY.GT.0)  CALL OUTPUT(XJ,UTM)
      WRITE(6,601) COST,(XLAMF(I),I=1,7),ALT
 601  FORMAT(1H0,5X,'COST FUNCTION=',1PD24.16/35X,'FINAL MULTIPLIERS'/
     1 6X,1P4D24.16/6X,1P3D24.16/6X,'ALTITUDE =',1PD24.16)
      RETURN
 105  COST=COSTFN(XJ)
      IF(IPRNT1.GE.2)  GO TO 104
      RETURN
 100  WRITE(6,210)
      IFLAG=IFLAG+1
      RETURN
 101  WRITE(6,220)
      IFLAG=IFLAG+3
      RETURN
 210  FORMAT(1H0,5X,'EXCEEDED CUTOFF TIME ON RUN WITH ALTITUDE CUTOFF')
 220  FORMAT(1H0,5X,'EXCEEDED MAXIMUM NUMBER OF ITERATIONS IN TERMINAL
     1 CUTOFF')
      END
```

```
C     SUBROUTINE BAKINT
      SUBROUTINE BAKINT(XJ,XLAMF,TG,ITER,DCOST,XNORMS,DCDTF)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION XS(14,1),YPR(14,4,1),DPSAVE(14,1),FR(14,1),DV(14,1),
     2P(14,1),TE(14,1),RPAR(1),XJ(8,1),XLAMF(7),XSAVE(14),GRAD(999,4),
     3G(999),SERCH(999,4),TEMPS(999,4),U(999,4),IPAR(1)
     4,XO(3),VO(3),URXVO(3)
      COMMON/STATO/XO,VO,XOMAG,VOMAG,URXVO,TO
      COMMON/CNTRL/GRAD,SERCH,U,ASTR,STF,TF,KJIS,IJKU,ISTAR
      COMMON/CONS2/DELTS,       ERRMX,ERRMN,TCUT,EPST,EPSTF,EPSA,EPSIT,
     1 ERR,ITMAX,ITMX,KOUNTM,IKEY
      COMMON/PRINT/IOUT,IOUT2,IPRNT1,IPRNT2
      EXTERNAL DERIV2
C INTEGRATION INITIALIZATION
      LMN = 0
C  REMOVED FIRST STEPER CALL
      DELT=DELTS
      DO 10 I=1,7
10    XS(I,1)=XJ(I,1)
      DO 11 I=8,14
11    XS(I,1)=XLAMF(I-7)
      TM=0.0D0
C  REMOVED THE SECOND STEPER CALL
      INX = 1
      TOL = ERRMX
      CALL DERIV2(TM,XS,P,6,1,14,1,RPAR,IPAR,1)
      IJK=999
C PERFORM INTEGRATION AND GRADIENT COMPUTATION
20    CONTINUE
      TEST = TF - TM
      IF(TEST.LE.TO) GO TO 70
      CALL GRADEN(XS,TM,IJK)
      LTEST = MOD(LMN,IOUT2)
      IF(LTEST.EQ.0) WRITE(6,600) TEST,(XS(I,1),I=1,14),(GRAD(IJK,J),
     1 J=1,3)
      LMN = LMN + 1
      DO 21 I=1,14
21    XSAVE(I)=XS(I,1)
      IJK=IJK-1
      IF(IJK .LT. 1) GO TO 90
C  REPLACED STEPER WITH RK713
      TI = TM
      TM = TM + DELT
      CALL RK713(INX,14,200,MK,TOL,TI,TM,XS,XS,DV,P,DERIV2)
      GO TO 20
C TERMINAL ITERATION
70     IF(TEST.EQ.TO) GO TO 30
      DO 79 I=1,14
79    XS(I,1)=XSAVE(I)
      TM=TM-DELT
      DELT=(TF-TO)-TM
C  REPLACED STEPER WITH RK713
      TI = TM
      TM = TM + DELT
      CALL RK713(INX,14,200,MK,TOL,TI,TM,XS,XS,DV,P,DERIV2)
      TEST = TF - TM
30    CALL GRADEN(XS,TM,IJK)
      WRITE(6,600) TEST,(XS(I,1),I=1,14),(GRAD(IJK,J),J=1,3)
  600 FORMAT(1H0,5X,'TIME =',1PD24.16/47X,'STATE VARIABLES'/6X,
     11P4D24.16/6X,1P3D24.16/48X,'MULTIPLIERS'/6X,1P4D24.16/6X,
```

```
     21P3D24.16/38X,'GRADIENT'/6X,1P3D24.16)
C SHIFT GRADIENT STORAGE
      KJI=1000-IJK
      DO 31 L=1,KJI
      DO 31 M=1,4
31    GRAD(L,M)=GRAD(IJK+L-1,M)
C FORM GRADIENT QUADRATURE BY TRAPEZOIDAL RULE
      DO 40 K=1,KJI
      G(K)=GRAD(K,1)**2+GRAD(K,2)**2+GRAD(K,3)**2
40    CONTINUE
      BETAN=0.0D0
      DO 41 L=2,KJI
41    BETAN=BETAN+(G(L)+G(L-1))*(GRAD(L,4)-GRAD(L-1,4))/2.0D0
      BETAN = BETAN + DCDTF**2
      IF(BETAN .LE. EPSIT) GO TO 101
C GET DERIVATIVE OF COST WITH RESPECT TO PARAMETER
      DCOST=-BETAN
C GET NORM OF SEARCH DIRECTION
      IF(ITER .EQ. 0) GO TO 42
      XNORMS=DSQRT(BETAN+(BETAN*XNORMS/BETAD)**2)
      GO TO 43
42    XNORMS=DSQRT(BETAN)
43    CONTINUE
      IF(IPRNT2.GT.0) WRITE(6,601)BETAN,XNORMS,DCOST,DCDTF
  601 FORMAT(1H0,5X,'GRADIENT NORM SQUARED =',1PD24.16/
     16X,'SEARCH DIRECTION NORM =',D24.16/6X,'COST SLOPE IN SEARCH
     2DIRECTION =',1PD24.16/6X,'COST DERIVATIVE WITH RESPECT TO TF =',
     31PD24.16)
C GET NEW SEARCH DIRECTION
      IF(ITER .NE. 0) GO TO 51
      DO 50 K=1,KJI
      DO 50 L=1,4
50    SERCH(K,L)=GRAD(K,L)
      STF=DCDTF
      GO TO 80
51    KTAU=2
      DO 60 L=1,KJI
      TAU=GRAD(L,4)
52    IF(TAU .LT. SERCH(KTAU-1,4)) GO TO 54
      IF(TAU .LE. SERCH(KTAU,4)) GO TO 57
      IF(KTAU .GE. KJIS) GO TO 56
      KTAU=KTAU+1
      GO TO 52
C TAU IS BELOW THE LOWER LIMIT
54    IF(KTAU .LE. 2) GO TO 56
      KTAU=KTAU-1
      GO TO 52
C FIND SEARCH DIRECTION BY LINEAR EXTRAPOLATION
56    DO 53 K=1,3
      TEMPS(L,K)=GRAD(L,K)+(BETAN/BETAD)*(SERCH(KTAU,K)+(SERCH(KTAU,K)-S
     2ERCH(KTAU-1,K))*(TAU-SERCH(KTAU,4))/(SERCH(KTAU,4)-SERCH(KTAU-1,4)
     3))
53    CONTINUE
      GO TO 60
C FIND SEARCH DIRECTION BY LINEAR INTERPOLATION
57    DO 55 K=1,3
      TEMPS(L,K)=GRAD(L,K)+(BETAN/BETAD)*((SERCH(KTAU,K)-SERCH(KTAU-1,K)
     2)*(TAU-SERCH(KTAU-1,4))/(SERCH(KTAU,4)-SERCH(KTAU-1,4))+SERCH(KTAU
     3-1,K))
55    CONTINUE
```

```
60     CONTINUE
C STORE SEARCH DIRECTION
       DO 62 L=1,KJI
       DO 61 M=1,3
61     SERCH(L,M)=TEMPS(L,M)
62     SERCH(L,4)=GRAD(L,4)
       STF=DCDTF          +(BETAN/BETAD)*STF
80     KJIS=KJI
       BETAD=BETAN
       RETURN
90     WRITE(6,200)
       ITER=ITMAX+1
       RETURN
200    FORMAT(1H0,5X,'HAVE EXCEEDED ALLOTTED STORAGE SPACE FOR GRADIENT')
101    WRITE(6,220)
       ITER=ITMAX+3
       RETURN
220    FORMAT(1H0,5X,'GRADIENT NORM LESS THAN TOLERANCE')
       END
```

```
C     SUBROUTINE RK713
      SUBROUTINE RK713(INDX,N,KT,M,TOL,TI,TF,XI,X,XDUM,TE,DERIV)
C     SEVENTH ORDER RUNGE-KUTTA INTEGRATION WITH STEPSIZE CONTROL
C     M IS THE NUMBER OF STEPS NEEDED
C     N IS THE NUMBER OF DIFFERENTIAL EQUATIONS
C     KT IS MAX NUMBER OF ITERATIONS
C     ARRAY F STORES THE 13 EVALUATIONS OF THE DIFFERENTIAL EQUATIONS
C     SUBSCRIPTS FOR ALPHA,BETA, AND CH ARE +1 GREATER THAN FEHLBERGS
C     F(0) IN FEHLBERGS REPORT IS IN F(1,J)
C     F(I) IS IN F(I+1,J)
C     PARAMETERS FOR DEQ SUBROUTINE MUST BE STORED IN COMMON
C     DIMENSIONS MUST AGREE WITH NUMBER OF DIFFERENTIAL EQUATIONS AND
C     NUMBER OF CONSTANTS IN THE PARTICULAR FEHLBERG FORMULA USED
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION F(13,25),XDUM(N),TE(N),XI(N),X(N),ALPH(13),
     1 BETA(13,12),CH(13),RPAR(1),IPAR(1)
      COMMON/RKCOE/ALPH,BETA,CH
      IPAR(1) = 0
      T=TI
      DT=TF-T
      M=0
      DO 10 I=1,N
   10 X(I)=XI(I)
   20 CALL DERIV(T,X,TE,INDX,MM,N,1,RPAR,IPAR,1)
      DO 30 I=1,N
   30 F(1,I)=TE(I)
      DO 70 K=2,13
      DO 40 I=1,N
   40 XDUM(I)=X(I)
      NN=K-1
      DO 50 I=1,N
      DO 50 J=1,NN
   50 XDUM(I)=XDUM(I)+DT*BETA(K,J)*F(J,I)
      TDUM=T+ALPH(K)*DT
      CALL DERIV(TDUM,XDUM,TE,INDX,MM,N,1,RPAR,IPAR,1)
      DO 60 I=1,N
   60 F(K,I)=TE(I)
   70 CONTINUE
      DO 80 I=1,N
   80 XDUM(I)=X(I)
      DO 90 I=1,N
      DO 90 L=1,13
   90 X(I)=X(I)+DT*CH(L)*F(L,I)
      DO 120 I=1,N
      IF (X(I)) 110,100,110
  100 A=1.
      GO TO 120
  110 A=X(I)
  120 TE(I)=DT*(F(1,I)+F(11,I)-F(12,I)-F(13,I))*41./840./A
      ER=DABS(TE(1))
      DO 140 I=2,N
      IF(DABS(TE(I))-ER) 140,140,130
  130 ER=DABS(TE(I))
  140 CONTINUE
      DT1=DT
      M=M+1
      AK=.8
      DT=AK*DT1*(TOL/ER)**.125
      IF (ER-TOL) 150,150,180
  150 T=T+DT1
```

```
      IF (DT-(TF-T)) 170,170,160
  160 DT=TF-T
  170 CONTINUE
      GO TO 200
  180 DO 190 I=1,N
  190 X(I)=XDUM(I)
  200 IF (M-KT) 210,220,220
  210 IF (T-TF) 20,220,220
  220 RETURN
      END
```

```
C     SUBROUTINE DERIV1
      SUBROUTINE DERIV1(T,X,P,L,M,N,NE,RPAR,IPAR,ND)
      IMPLICIT REAL*8(A-H,O-Z)
      REAL*8 LOADF
      DIMENSION X(N,NE),P(N,NE),RPAR(ND),IPAR(ND),OMEGE(3),TEMP(3),
     2TEMM(3),VR(3),COEF(2),COEFM(3,3),GRAD(999,4),SERCH(999,4),U(999,4)
     3,OMEGO(3,3)
      COMMON/CONST/PI,RE,XMU,OMEGE,AREA,ECOFF,GNOT,OMEGO
      COMMON/CNTRL/GRAD,SERCH,U,ASTR,STE,TF,KJIS,IJKU,TSTAR
      COMMON/DERIVS/RMAG1,VR,VRMAG,RHO,DRHO,VS,DVS,CLA,CA,ETA,DCLA,
     1 DCA,DETA
      COMMON/STATE/ALT,XMASS,TEMP,TEMPM
      P(1,1)=X(4,1)
      P(2,1)=X(5,1)
      P(3,1)=X(6,1)
      RMAG2=0.0D0
      DO 10 I=1,3
10    RMAG2=RMAG2+X(I,1)*X(I,1)
      RMAG1=DSQRT(RMAG2)
      RMAG3=RMAG2*RMAG1
      RMAG3=RMAG2*RMAG1
C COMPUTE ACCELERATIONS DUE TO GRAVITY
      DO 11 I=4,6
11    P(I,1)=-XMU*X(I-3,1)/RMAG3
C COMPUTE RELATIVE VELOCITY
      CALL ACROSB(OMEGE,X,VR,0,UNITC)
      DO 12 I=1,3
12    VR(I)=X(I+3,1)-VR(I)
      VRMAG=DSQRT(ADOTB(VR,VR))
C COMPUTE ATMOSPHERIC QUANTITIES AND AERODYNAMIC PARAMETERS
      ALTI=RMAG1-RE
      CALL ATMOS(ALTI,TEMPR,PRES,RHO,VS,DVS,DRHO,DPRES)
      RHO = DABS(RHO)
      XMACH=VRMAG/VS
      CALL AEROD(XMACH,CLA,CA,ETA,DCLA,DCA,DETA)
C COMPUTE AERODYNAMIC COEFFICIENTS
      COEF(1)=RHO*AREA*VRMAG*VRMAG*CLA/XMASS/2.0D0
      COEF(2)=-(2.0*ETA+CA/CLA)/VRMAG
      DO 13 I=1,3
      DO 13 J=1,3
13    COEFM(I,J)=VR(I)*VR(J)*(2.0D0*ETA-1.0D0)/VRMAG**2
      DO 14 I=1,3
14    COEFM(I,I)=COEFM(I,I)+1.0D0
C ADD AERODYNAMIC ACCELERATIONS TO GRAVITY
      DO 15 I=4,6
15    P(I,1)=P(I,1)+COEF(1)*COEF(2)*VR(I-3)
      IF(IPAR(1).EQ.1) GO TO 28
C FIND CONTROL VECTOR FROM TABLE
      GO TO (20,70,70,70,70,70),L
   22 IF(KT.GE.IJKU) GO TO 25
      KT = KT + 1
      IF(T.LE.U(KT,4)) GO TO 30
      GO TO 22
   70 KT = 2
   20 IF(T.GT.U(KT,4)) GO TO 22
      IF(T.GE.U(KT-1,4)) GO TO 30
      IF(KT.LE.2) GO TO 25
      KT =  KT - 1
      GO TO 20
C INTERPOLATE FOR CONTROL WHICH LIES IN INTERVAL U(KT-1,4),U(KT,4)
```

```
30     DO 31 I=1,3
31     TEMP(I)=U(KT-1,I)+(U(KT,I)-U(KT-1,I))*(T-U(KT-1,4))/(U(KT,4)-U(KT-
      21,4))
33     TEMPM=DSORT(ADOTB(TEMP,TEMP))
       DO 32 I=1,3
32     TEMP(I)=TEMP(I)/TEMPM
       GO TO 40
C TIME LIES OUTSIDE CONTROL ARRAY USE LINEAR EXTRAPOLATION
25     DO 26 I=1,3
       TEMP(I)=U(KT,I)+(U(KT,I)-U(KT-1,I))*(T-U(KT,4))/(U(KT,4)-U(KT-1,4)
      2)
26     CONTINUE
       GO TO 33
C CHECK ON CONTROL OPTION FLAG
40     IF(ISTAR .EQ. 0) GO TO 28
C FIND SEARCH DIRECTION FROM TABLE
       GO TO (50,53,61,53,61,60),L
50     IF(T .LT. SERCH(KTS-1,4)) GO TO 60
52     IF(T .LE. SERCH(KTS,4)) GO TO 65
       IF(KTS-KJIS) 51,55,55
51     KTS=KTS+1
       GO TO 52
60     KTS=2
61     IF(T .LT. SERCH(KTS-1,4)) GO TO 55
       GO TO 52
53     IF(T .LT. SERCH(KTS-1,4)) GO TO 55
       GO TO 65
C INTERPOLATE FOR SEARCH DIRECTION
65     DO 66 I=1,3
       TEM(I)=SERCH(KTS-1,I)+(SERCH(KTS,I)-SERCH(KTS-1,I))*(T-SERCH(KTS-1
      2,4))/(SERCH(KTS,4)-SERCH(KTS-1,4))
66     CONTINUE
       GO TO 68
55     DO 56 I=1,3
       TEM(I)=SERCH(KTS,I)+(SERCH(KTS,I)-SERCH(KTS-1,I))*(T-SERCH(KTS,4))
      3/(SERCH(KTS,4)-SERCH(KTS-1,4))
56     CONTINUE
C FORM CONTROL
68     DO 69 I=1,3
69     TEMP(I)=TEMP(I)-ASTR*TEM(I)
       TEMPM=DSORT(ADOTB(TEMP,TEMP))
       DO 67 I=1,3
67     TEMP(I)=TEMP(I)/TEMPM
C ADD CONTROL ACCELERATIONS
   28  CONTINUE
       DO 41 I=4,6
       P(I,1)=P(I,1)+COEF(1)*(COEFM(I-3,1)*TEMP(1)+COEFM(I-3,2)*TEMP(2)+C
      2OEFM(I-3,3)*TEMP(3))
41     CONTINUE
C COMPUTE HEATING DERIVATIVE
       RHOO = 1.2250
       P(7,1) = ECOEF*DSORT(RHO/RHOO)*(1.262D-4*VRMAG)**3.15
C COMPUTE INTEGRATED COST DERIVATIVE
       CALL ACROSB(VR,TEMP,TEM,0,UNITC)
       DO 42 I=1,3
42     TEM(I)=TEM(I)/VRMAG
       ALF2=ADOTB(TEM,TEM)
       LOADF=(RHO*VRMAG**2*AREA/2.0)**2*(CA**2+(CLA**2+2.0*ETA*CA*CLA)
      1*ALF2 + (ETA*CLA*ALF2)**2)
       LOADF = LOADF/XMASS**2
```

```
P(8,1)=LOADF-(3.0D0*XMU/RE/RE)**2
IF(P(8,1).LT.0.0D0) P(8,1) = 0.0D0
RETURN
END
```

```
C     SUBROUTINE DERIV2
      SUBROUTINE DERIV2(T,X,P,L,M,N,NE,RPAR,IPAR,ND)
      IMPLICIT REAL*8(A-H,O-Z)
      REAL*8 LDRHO,LDCLA,LDCA,LDETA
      DIMENSION X(N,NE),P(N,NE),RPAR(ND),IPAR(ND),UTM(3),XS(8,1),
     1PS(8,1),OMEGO(3,3),OMEGE(3),GRAD(999,4),SERCH(999,4),U(999,4),
     2VR(3),VRVRT(3,3),DMDR(3),DMDV(3)
     3,PFUN(4),CCOST(2),VECVR(3),VECU(3),VECLV(3),UNITC(3)
      COMMON/CONS1/PI,RE,XMU,OMEGE,AREA,ECOEF,GNOT,OMEGO
      COMMON/CONS3/CSTR,B,PFUN,CCOST,DTEM,XDTEM
      COMMON/DERIVS/RMAG1,VR,VRMAG,RHO,DRHO,VS,DVS,CLA,CA,ETA,DCLA,
     1 DCA,DETA
      COMMON/STATE/ALT,XMASS,UTM,UTMAG
      COMMON/CNTRL/GRAD,SERCH,U,ASTR,STF,TF,KJIS,IJKU,ISTAR
C COMPUTE FORWARD TIME
      TM=TF-T
C FIND CONTROL VECTOR
      GO TO (10,19,10,19,10,8),L
    5 IF(KT.GE.IJKU) GO TO 15
      KT = KT + 1
      IF(TM.LE.U(KT,4)) GO TO 15
      GO TO 5
8     KT=IJKU
10    IF(TM.GT.U(KT,4)) GO TO 5
      IF(TM .GE. U(KT-1,4)) GO TO 15
      KT=KT-1
      IF(KT .LE. 1) GO TO 25
      GO TO 10
C FIND CONTROL BY INTERPOLATION OR EXTRAPOLATION
15    DO 16 I=1,3
      UTM(I)=U(KT-1,I)+(U(KT,I)-U(KT-1,I))*(TM-U(KT-1,4))/(U(KT,4)-U(KT-
     21,4))
16    CONTINUE
17    UTMAG=DSQRT(ADOTB(UTM,UTM))
      DO 18 I=1,3
18    UTM(I)=UTM(I)/UTMAG
      GO TO 19
C FIND CONTROL BY EXTRAPOLATION
25    KT=2
      GO TO 15
C PREPARE FOR DERIV1 CALL
19    DO 20 I=1,7
20    XS(I,1)=X(I,1)
      IPAR(1)=1
      CALL DERIV1(TM,XS,PS,L,M,8,1,RPAR,IPAR,1)
C COMPUTE BACKWARD STATE DERIVATIVES
      DO 30 I=1,7
30    P(I,1)=-PS(I,1)
C COMPUTE MACH NUMBER DERIVATIVE WITH RESPECT TO R
      CALL ACROSB(OMEGE,VR,VECVR,0,UNITC)
      DO 31 I=1,3
   31 DMDR(I) =-VRMAG*DVS*X(I,1)/VS**2/RMAG1 + VECVR(I)/VS/VRMAG
C COMPUTE MACH NUMBER DERIVATIVE WITH RESPECT TO V
      DO 32 I=1,3
32    DMDV(I)=VR(I)/VS/VRMAG
C  COMPUTE DOT PRODUCTS
      XLVR =   0.0D0
      XLVRS =  0.0D0
      XLVU = 0.0D0
      DO 40 J=11,13
```

```
      XLVU = XLVU + X(J,1)*UTM(J-10)
      XLVR = XLVR + X(J,1)*VR(J-10)/VRMAG
   40 XLVRS = XLVRS + X(J,1)*X(J-10,1)
      VRU = ADOTB(VR,UTM)/VRMAG
C COMPUTE COEFFICIENT TERMS IN MULTIPLIER EQUATIONS
      CFA = RHO*AREA*CLA/XMASS/2.0D0
      HA = CFA*(2.0*XLVU - (CA/CLA + 2.0*ETA)*XLVR)
      HB = CFA*(-VRMAG*(CA/CLA + 2.0*ETA) + (2.0*ETA -1.0)*VRU*VRMAG)
      HC = CFA*VRMAG*(2.0*ETA -1.0)*XLVR
C COMPUTE CROSS PRODUCTS FOR POSITION MULTIPLIER EQUATIONS
      CALL ACROSB(OMEGE,UTM,VECU,0,UNITC)
      VECLV(1) = OMEGE(2)*X(13,1) - OMEGE(3)*X(12,1)
      VECLV(2) = OMEGE(3)*X(11,1) - OMEGE(1)*X(13,1)
      VECLV(3) = OMEGE(1)*X(12,1) - OMEGE(2)*X(11,1)
      HD =((CFA*VRMAG**2/RHO)*(XLVU +((2.0*ETA-1.0)*VRU -(CA/CLA +2.0
     1 *ETA))*XLVR)*DRHO
      HE =(CFA*VRMAG**2/CLA)*(XLVU*DCLA +(((2.0*ETA -1.0)*VRU -2.0
     1 *ETA)*DCLA - DCA + 2.0*CLA*(VRU-1.0)*DETA)*XLVR)
C    COMPUTE STATE MULTIPLIERS
      DO 41 J=8,10
      P(J,1) = -(XMU/RMAG1**3)*(X(J+3,1) - 3.0*XLVRS*X(J-7,1)/RMAG1**2)
     1 + HA*VECVR(J-7) + HB*VECLV(J-7) + HC*VECU(J-7)
     2 + HD*X(J-7,1)/RMAG1 + HE*DMDR(J-7)
   41 CONTINUE
C   COMPUTE VELOCITY MULTIPLIERS
      DO 42 J=11,13
      P(J,1) = X(J-3,1) + HB*X(J,1) + HA*VR(J-10) + HC*UTM(J-10)
     1+HE*DMDV(J-10)
   42 CONTINUE
C   ADD IN HEATING EFFECTS
      RHOO = 1.22501
      COFB=ECOEF*(1.262D-4*VRMAG)**3.15*DRHO/2.0/DSQRT(RHO*RHOO)
      COFA = 3.15*ECOEF*DSQRT(RHO/RHOO)*(1.262D-4*VRMAG)**3.15/VRMAG**2

      DO 82 I=8,10
   82 P(I,1) = P(I,1) + X(14,1)*(COFB*X(I-7,1)/RMAG1 + COFA*VECVR(I-7))
      DO 83 I=11,13
   83 P(I,1) = P(I,1) + X(14,1)*COFA*VR(I-10)
C CHECK INEQUALITY ON COST INTEGRAND
      IF(PS(8,1) .LE. 0.0D0) GO TO 300
C COMPUTE AERODYNAMIC PARTIALS
      QQ = (RHO*AREA/2.0)**2*VRMAG**3
      DOT = ADOTB(VR,UTM)
      LDRHO =(2.0*QQ/RHO)*((CA**2 + 4.0*ETA*CLA*CA +(4.0*ETA**2+1.0)
     1 *CLA**2)*VRMAG - DOT*(4.0*ETA*CLA*(CA -2.0*ETA*CLA)
     2 - (4.0*ETA**2-1.0)*CLA**2*DOT/VRMAG))
      LDCLA = QQ*((4.0*ETA*CA + 2.0*(4.0*ETA**2+1.0)*CLA)*VRMAG - DOT
     1 *(4.0*ETA*(CA - 4.0*ETA*CLA) - 2.0*(4.0*ETA**2-1.0)*CLA*DOT
     2 /VRMAG))
      LDCA = QQ*((2.0*CA + 4.0*ETA*CLA)*VRMAG - 4.0*ETA*CLA*DOT)
      LDETA = QQ*(4.0*CLA*(CA + 2.0*ETA*CLA)*VRMAG - DOT*(4.0*CLA
     1 *(CA - 4.0*ETA*CLA) - 8.0*ETA*CLA**2*DOT/VRMAG))
      DLDVR = (RHO*AREA/2.0)**2*(4.0*(CA**2 + 4.0*ETA*CLA*CA + CLA**2
     1 *(4.0*ETA**2 +1.0))*VRMAG**3 - 3.0*(4.0*ETA*CLA*CA - 8.0*ETA**2
     2 *CLA**2)*VRMAG**2*DOT + 2.0*(4.0*ETA**2 -1.0)*CLA**2*VRMAG
     3 *DOT**2)
      DLDVRU =QQ*(-4.0*ETA*CLA*(CA - 2.0*ETA*CLA) + 2.0*(4.0*ETA**2
     1 -1.0)*CLA**2*DOT/VRMAG)
      TERM = LDCLA*DCLA + LDCA*DCA + LDETA*DETA
```

```
C   ADD AERODYNAMIC LOAD TO MULTIPLIERS
      DO 90 I=8,10
      P(I,1) = P(I,1) +(LDRHO*DRHO*X(I-7,1)/RMAG1 + TERM*DMDR(I-7)
     1 + DLDVR*VECVR(I-7)/VRMAG + DLDVRU*VECU(I-7))*PFUN(4)/XMASS**2
   90 CONTINUE
      DO 92 I=11,13
      P(I,1) = P(I,1) + (TERM*DMDR(I-10) + DLDVR*VR(I-10)/VRMAG
     1 + DLDVRU*UTM(I-10))*PFUN(4)/XMASS**2
   92 CONTINUE
300   CONTINUE
      P(14,1)=0.0D0
      RETURN
      END
```

```
C     SUBROUTINE GRADEN
      SUBROUTINE GRADEN(XS,TM,IJK)
      IMPLICIT REAL*8(A-H,O-Z)
      REAL*8 LOADF
      DIMENSION XS(14,1),GRAD(999,4),B(3,3),VR(3),TEMP(3),VRVRT(3,3),
     1TMTRX(3,3),DLDU(3),DDDU(3),XLAM(3),XO(3),VO(3),URXVO(3),OMEGE(3),
     2UTM(3),PFUN(4),C(3,3),XLIFT(3),DRAG(3)
     3,OMEGO(3,3),CCOST(2),SERCH(999,4),U(999,4)
      COMMON/CONS1/PI,RE,XMU,OMEGE,AREA,ECOEF,GNOT,OMEGO
      COMMON/CONS3/CSTR,O,PFUN,CCOST,DTEM,XDTEM
      COMMON/STATE/ALT,XMASS,UTM,UTMAG
      COMMON/STATO/XO,VO,XOMAG,VOMAG,URXVO,TO
      COMMON/CNTRL/GRAD,SERCH,U,ASTR,STF,TF,KJIS,IJKU,ISTAR
C CHECK GRADIENT TIME POINT - EFFECT OF VARIABLE STEPSIZE
      TME = TF - TM
      IF(IJK.GE.999) GO TO 9
      IF(TME.LT.GRAD(IJK+1,4)) GO TO 9
    7 IJK = IJK + 1
      IF(TME.LT.GRAD(IJK+1,4)) GO TO 9
      IF(IJK.EQ.998) GO TO 8
      GO TO 7
    8 IJK = 999
C SET TIME
9     GRAD(IJK,4)= TF - TM
C COMPUTE RELATIVE VELOCITY
      CALL ACROSB(OMEGE,XS,TEMP,O,UNITC)
      DO 10 J=1,3
10    VR(J)=XS(J+3,1) - TEMP(J)
      VRMAG=DSORT(ADOTB(VR,VR))
C COMPUTE ALTITUDE
      ALTI=DSORT(ADOTB(XS,XS))-RE
C COMPUTE ATMOSPHERE AND AERO DYNAMIC QUANTITIES
#REPRODUCE
C COMPUTE ATMOSPHERE AND AERO DYNAMIC QUANTITIES
      CALL ATMOS(ALTI,TEMPR,PRES,RHO,VS,DVS,DRHO,DPRES)
      XMACH=VRMAG/VS
      CALL AEROD(XMACH,CLA,CA,ETA,DCLA,DCA,DETA)
C  COMPUTE DOT PRODUCTS
      DVRU = ADOTB(VR,UTM)/VRMAG
      DLVU = 0.0D0
      DLVVR = 0.0D0
      DO 11 I=1,3
      DLVU = DLVU + XS(10+I,1)*UTM(I)
   11 DLVVR = DLVVR + XS(10+I,1)*VR(I)/VRMAG
C  COMPUTE CONSTANTS
      CKA = RHO*AREA*VRMAG**2*CLA/XMASS/2.0D0
      CKB = (2.0*ETA-1.0)*DLVVR/UTMAG/VRMAG
      CKC = -(DLVU + (2.0*ETA-1.0)*DLVVR*DVRU)/UTMAG
C  COMPUTE GRADIENT
      DO 12 J=1,3
   12 GRAD(IJK,J) =(XS(10+J,1)/UTMAG + CKB*VR(J) + CKC*UTM(J))*CKA
C  COMPUTE AERODYNAMIC LOAD
      CKD = (RHO*AREA*VRMAG**2/2.0D0)**2
      LOADF = CA**2 + 4.0*ETA*CLA*CA + (4.0*ETA**2 + 1.0)*CLA**2
      LOADF = LOADF - 4.0*ETA*CLA*(CA-2.0*ETA*CLA)*DVRU
      LOADF = (LOADF + (4.0*ETA**2-1.0)*CLA**2*DVRU**2)*CKD/XMASS**2
      LOADF = LOADF -(3.0*GNOT)**2
C  CHECK LOAD MAGNITUDE
      IF(LOADF.LE.0.0D0) RETURN
C  COMPUTE GRADIENT OF LOAD
```

```
      CKE = (RHO*AREA*VRMAG/2.0)**2/UTMAG
      CKE = CKE*(2.0*(4.0*ETA**2-1.0)*CLA**2*DVRU*VRMAG
     1 -4.0*ETA*CLA*(CA -2.0*ETA*CLA)*VRMAG)
      DO 13 I=1,3
   13 TEMP(I) = CKE*(VR(I) - DVRU*UTM(I)*VRMAG)
C  ADD LOAD GRADIENT TO TOTAL GRADIENT
      DO 14 I=1,3
   14 GRAD(IJK,I) = GRAD(IJK,I) + TEMP(I)/XMASS**2
      RETURN
      END
```

```
C     FUNCTION COSTFN
      DOUBLE PRECISION FUNCTION COSTFN(XI)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION X(3),X0(3),TEMP(3),URXV0(3),UNDR(3),UNZ(3),CCOST(2),
     1PFUN(4),V0(3),XI(8,1),OMEGE(3),OMEGO(3,3),UTM(3)
      COMMON/CONS1/PI,RE,XMU,OMEGE,AREA,FCOEF,GNOT,OMEGO
      COMMON/CONS3/CSTR,B,PFUN,CCOST,DTEM,XDTEM
      COMMON/STATE/ALT,XMASS,UTM,UTMAG
      COMMON/STAT0/X0,V0,X0MAG,V0MAG,URXV0,T0
      COMMON/STATF/ALTF,XMACH,FLTANG,QMAX,SINCR,COSCR,SINDR,COSDR
      DO 10 I=1,3
10    X(I)=XI(I,1)
C COMPUTE DOWNRANGE AND CROSS RANGE
      CALL ACROSR(URXV0,X,TEMP,1,UNDR)
      CALL ACROSR(UNDR,URXV0,TEMP,1,UNZ)
      COSDR=ADOTB(X0,UNDR)/X0MAG
      SINDR=ADOTB(X0,UNZ)/X0MAG
      DRANG=DATAN2(SINDR,COSDR)
      XMAG=DSQRT(ADOTB(X,X))
      SINCR=ADOTB(X,URXV0)/XMAG
      COSCR=ADOTB(X,UNZ)/XMAG
      CRANG=DATAN2(SINCR,COSCR)
      DRANGE=DRANG*RE
      CRANGE=CRANG*RE
C COMPUTE ALTITUDE
      ALT=XMAG-RE
C FORM COST VALUE
      COSTFN=CCOST(1)*CRANGE+CCOST(2)*DRANGE+PFUN(1)*(ALT-ALTF)**2+XI(8,
     11)*PFUN(4)+PFUN(2)*(XI(7,1)-QMAX)**2
      RETURN
      END
```

```
C     SUBROUTINE XLAMEN
      SUBROUTINE XLAMEN(XLAMF,XJ)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION XJ(8,1),XLAMF(7),DHDRF(3),DCDRF(3),DDDRF(3),
     1 XO(3),VO(3),PFUN(4),CCOST(2),OMEGE(3),OMEGO(3,3),UTM(3),
     3 URXVO(3)
      COMMON/CONS1/PI,RE,XMU,OMEGE,AREA,ECOEF,GNOT,OMEGO
      COMMON/CONS3/CSTR,B,PFUN,CCOST,DTFM,XDTFM
      COMMON/STATF/ALTF,XMACH,FLTANG,QMAX,SINCR,COSCR,SINDR,COSDR
      COMMON/STATE/ALT,XMASS,UTM,UTMAG
      COMMON/STATO/XO,VO,XOMAG,VOMAG,URXVO,TO
C SET FINAL VELOCITY MULTIPLIERS
      DO 10 I=4,6
10    XLAMF(I)=0.0D0
C SET FINAL HEATING MULTIPLIER
      XLAMF(7)=2.0D0*PFUN(2)*(XJ(7,1)-QMAX)
C COMPUTE FINAL POSITION MULTIPLIERS
      RF = ALT + RE
      DO 20 I=1,3
20    DHDRF(I)=XJ(I,1)/RF
      DO 21 I=1,3
21    DCDRF(I)=-SINCR*XJ(I,1)/COSCR/RF**2+URXVO(I)/COSCR/RF
      DO 22 I=1,3
      DDDRF(I)=XJ(I,1)*COSDR/SINDR/RF**2-(ADOTB(XO,VO)*XO(I)/VOMAG/XOMA
     2G**2-VO(I)/VOMAG)/SINDR/RF
22    CONTINUE
      CON=2.0D0*PFUN(1)*(ALT-ALTF)
      DO 23 I=1,3
      XLAMF(I)=CCOST(1)*RE*DCDRF(I)+CCOST(2)*RE*DDDRF(I)+CON*DHDRF(I)
23    CONTINUE
      RETURN
      END
```

```
C   VECTOR OPERATIONS SUBPROGRAMS
C    CROSS PRODUCT
      SUBROUTINE ACROSB(A,B,C,IUNIT,UNITC)
      DOUBLE PRECISION A,B,C,UNITC,CMAG,ADOTB
      DIMENSION A(1),B(1),C(1),UNITC(1)
      C(1)=A(2)*B(3)-A(3)*B(2)
      C(2)=A(3)*B(1)-A(1)*B(3)
      C(3)=A(1)*B(2)-A(2)*B(1)
      IF(IUNIT .LE. 0) RETURN
      CMAG=DSQRT(ADOTB(C,C))
      DO 1 K=1,3
1     UNITC(K)=C(K)/CMAG
      RETURN
      END
C   DOT PRODUCT
      DOUBLE PRECISION FUNCTION ADOTB(A,B)
      DOUBLE PRECISION A,B,ADOTB
      DIMENSION A(1),B(1)
      ADOTB=0.0D0
      DO 1 K=1,3
1     ADOTB=ADOTB+A(K)*B(K)
      RETURN
      END
```

```
C     SUBROUTINE OUTPUT
      SUBROUTINE OUTPUT(X,UTM)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION X(8,1),UTM(3),OMEGE(3),OMEGO(3,3),UNITC(3),VR(3),
     1 UNITP(3),UNITO(3),UNITK(3),UNITJ(3),XLIFT(3)
      COMMON/CONS1/PI,RE,XMU,OMEGE,AREA,ECOEF,GNOT,OMEGU
C     COMPUTE UNIT VECTORS
      CALL ACROSB(OMEGE,X,VR,0,UNITC)
      DO 12 I=1,3
   12 VR(I) = X(I+3,1) - VR(I)
      CALL ACROSB(X,VR,UNITC,1,UNITP)
      CALL ACROSB(UNITP,VR,UNITC,1,UNITO)
      VRMAG = DSQRT(ADOTB(VR,VR))
      D1 = ADOTB(UTM,VR)/VRMAG
      D2 = ADOTB(UTM,UNITO)
      D3 = ADOTB(UTM,UNITP)
C    COMPUTE AERODYNAMIC ANGLES
      ALFA = DATAN2(D2,D1)*180.0D0/PI
      BETA = DATAN2(D2,D3)*180.0D0/PI
      ALFAT = DATAN(DSQRT(D2**2 + D3**2)/D1)*180.0D0/PI
C    COMPUTE UNIT VECTORS FOR TRAJECTORY PLANE
      UNITK(1) = X(2,1)*X(6,1) - X(3,1)*X(5,1)
      UNITK(2) = X(3,1)*X(4,1) - X(1,1)*X(6,1)
      UNITK(3) = X(1,1)*X(5,1) - X(2,1)*X(4,1)
      UNIT = DSQRT(ADOTB(UNITK,UNITK))
      DO 15 I=1,3
   15 UNITK(I) = UNITK(I)/UNIT
      CALL ACROSB(UNITK,X,UNITC,1,UNITJ)
C    COMPUTE UNIT VECTOR IN LIFT DIRECTION
      CALL ACROSB(VR,UTM,XLIFT,0,UNITC)
      CALL ACROSB(XLIFT,VR,UNITC,1,XLIFT)
      RMAG = DSQRT(ADOTB(X,X))
      ZETA1 = ADOTB(XLIFT,UNITK)
      ZETA2= ADOTB(XLIFT,X)/RMAG
      ZETA3 = ADOTB(XLIFT,UNITJ)
C    COMPUTE LIFT VECTOR OUT OF PLANE ANGLE
      PHIOUT = DATAN(ZETA1/DSQRT(ZETA2**2 + ZETA3**2))*180. /PI
      ZETA4 = ADOTB(UTM,UNITK)
      ZETA5 = ADOTB(UTM,X)/RMAG
      ZETA6 = ADOTB(UTM,UNITJ)

C COMPUTE ANGLES OF VEHICLES AXIS
      PSIOUT = DATAN(ZETA4/DSQRT(ZETA5**2+ZETA6**2))*180.D0/PI
      PSIIN = DATAN2(ZETA5,ZETA6)*180.D0/PI
C    COMPUTE FLIGHT PATH ANGLE
      ZETA7 = (X(1,1)*X(4,1)+X(2,1)*X(5,1)+X(3,1)*X(6,1))/RMAG
      ZETA8 = X(4,1)*UNITJ(1)+X(5,1)*UNITJ(2)+X(6,1)*UNITJ(3)
      GAMMA = DATAN2(ZETA7,ZETA8)*180.D0/PI
      WRITE(6,600) ALFA,BETA,ALFAT
  600 FORMAT(1H0,5X,'IN PLANE ANGLE OF ATTACK =',1PD24.16,5X,
     1 'ANGLE OF SIDESLIP =',1PD24.16/6X,'TOTAL ANGLE OF ATTACK =',
     2 1PD24.16)
      WRITE(6,601) PHIOUT
  601 FORMAT(1H0,5X,'OUT OF PLANE ANGLE OF LIFT VECTOR =',1PD24.16)
      WRITE(6,602) PSIOUT,PSIIN
  602 FORMAT(1H0,5X,'BODY AXIS OUT OF PLANE ANGLE =',1PD24.16/
     1 6X,'BODY AXIS IN PLANE ANGLE =',1PD24.16)
      WRITE(6,603) GAMMA
  603 FORMAT(1H0,5X,'FLIGHT PATH ANGLE =',1PD24.16)
      RETURN
      END
```

```
C     SUBROUTINE AEROD  -  POLYNOMIAL FIT
      SUBROUTINE AEROD(XMACHI,CLAO,CAO,ETAO,DCLAO,DCAO,DETAO)
      DOUBLE PRECISION XMACHI,CLAO,CAO,ETAO,DCLAO,DCAO,DETAO
      DIMENSION A(5), B(5), C(5), D(6)
      DATA         A       /9.115648E-2,-3.040159E-2,5.068882E-3,
     1 -4.18355E-4,1.363262E-5/,      B       /5.063782E-1,2.126489E-1,
     2 -7.215163E-2,9.065345E-3,-3.72305E-4/,    C          /4.038559E-1,
     3 2.533609E-1,3.731828E-2,-1.001608E-2,5.174269E-4/,D/
     4 6.373618E+0,-6.245387E+0,2.82442E+0,-6.414284E-1,7.210993E-2,
     5 -3.191962E-3/
      XMACH = SNGL(XMACHI)
      IF(XMAGH.GT.1.0E+1) GO TO 10
      CA = A(1) + XMACH*(A(2) + XMACH*(A(3) + XMACH*(A(4) + A(5)
     1 *XMACH)))
      DCA = A(2) + XMACH*(2.0*A(3) + XMACH*(3.0*A(4) + 4.0*A(5)*XMACH))
      GO TO 11
   10 CA = 1.2E-2
      DCA = 0.0E0
   11 IF(XMACH.GT.9.0E0) GO TO 20
      ETA = C(1) + XMACH*(C(2) + XMACH*(C(3) + XMACH*(C(4) + C(5)
     1 *XMACH)))
      DETA = C(2) +XMACH*(2.0*C(3) + XMACH*(3.0*C(4)+4.0*C(5)*XMACH))
      GO TO 21
   20 ETA = 1.8E+0
      DETA = 0.0E0
   21 IF(XMACH.GT.5.8E+0) GO TO 30
      CLA = D(1) + XMACH*(D(2) + XMACH*(D(3) + XMACH*(D(4) + XMACH
     1 *(D(5) + D(6)*XMACH))))
      DCLA = D(2) + XMACH*(2.0*D(3) + XMACH*(3.0*D(4) + XMACH*(4.0*D(5)
     1 + 5.0*D(6)*XMACH)))
      GO TO 40
   30 IF(XMACH.GT.1.0E+1) GO TO 31
      CLA = B(1) + XMACH*(B(2) + XMACH*(B(3) + XMACH*(B(4) + B(5)*
     1 XMACH)))
      DCLA = B(2) + XMACH*(2.0*B(3) + XMACH*(3.0*B(4) + 4.0*B(5)*XMACH))
      GO TO 40
   31 CLA = 7.6E-1
      DCLA = 0.0E0
   40 CLAO  =DBLE(CLA)
      CAO   =DBLE(CA)
      ETAO  =DBLE(ETA)
      DCLAO =DBLE(DCLA)
      DCAO  = DBLE(DCA)
      DETAO = DBLE(DETA)
      RETURN
      END
```

```
      SUBROUTINE ATMOS(ALTI,TEMP ,PRES ,RHO ,VS ,DVS ,DRHO ,DPRES )
      IMPLICIT REAL*8(A-H,O-Z)
      DATA A0,A1,A2,A3,A4,A5,A6,A7,A8,A9,A10,A11,A12,A13,A14,A15,A16,
     1     A17,A18,A19,A20,A21,A22,A23,B0,B1,B2,B3,B4,B5,B6,B7,B8,D1,
     2     D2,D3,D4/-1.0902039D-7,6356.77D0,1.787026D-4,-1.680485D0,
     3     1.3949832D-5,284.017680D0,1.3327563D-4,29.8950600,924.136D0,
     4     8.3168074D-4,.3777736SD-1,.5646783D0,1.6002510-4,189.5201D0,
     5     9665.295D0,1.1637071D-3,.38184967D-1,3.61840940D0,5.562892D-5,
     6     420.11368D0,45675.466D0,1.284404D-4,.25387008D-1,5.33271460D0,
     7     2.8247930810+2,-5.24057299200,-1.2660105950-1,1.87329383GD-2,
     8    -5.1047465330-4,6.0501864060-6,-3.5501627350-8,1.014102927
     9     D-10,1.1244496190-13,3483.6763560D,.2021698 261D-1,5.80334458
     A     91D0,.40187430086D+3/
C NOTE THAT FORMULAS ARE NOT ACCURATE FOR ALTITUDE OUTSIDE 0 TO 200 KM
C ALT MUST BE IN METERS
C TEMP IS IN DEGREES KELVIN
C PRES IS IN NEWTONS/M**2
C RHO IS IN KG/M**3
C VS IS IN METERS PER SECOND
C DRHO,DPRES, AND DVS ARE IN SAME UNITS AS RHO, PRES, AND DVS OVER MTS
      ALT=ALTI
      Z=ALT*1.0D-3
      IF(Z)1,2,2
1     Z=0.0
2     CONTINUE
      IF(Z-2.0D2)3,3,4
4     Z=200.D0
3     CONTINUE
      Z2=Z*Z
      E1=Z+A1
      E2=Z+A3
      E3=Z-A5
      E4=Z2-A7*Z+A8
      E5=Z2-A13*Z+A14
      E6=Z2-A19*Z+A20
      A=A0/E1+A2*DLOG(E2)-A4*DLOG(-E3)+A6*DLOG(E4)+A9*DATAN(A10*Z-A11)
     1-A12*DLOG(E5)+A15*DATAN(A16*Z-A17)-A18*DLOG(E6)+A21*DATAN(A22*Z
     2-A23)
      AAB=0.018031036
      AAC=-0.060803123
      AAD=-0.028629767
      DA=-A0/(E1*E1)+A2/E2-A4/E3+(2.*A6*Z+AAB)/E4-(2.*A12*Z+AAC)/E5
     1-(2.*A18*Z+AAD)/E6
      DA=DA*0.001
      TEMP=B0+Z*(B1+Z*(B2+Z*(B3+Z*(B4+Z*(B5+Z*(B6+Z*(B7-B8*Z)))))))
      DTEMP=B1+Z*(2.*B2+Z*(3.*B3+Z*(4.*B4+Z*(5.*B5+Z*(6.*B6+Z*(7.*B7-
     18.*B8*Z))))))
      DTEMP=DTEMP*0.001
      PRES=DEXP(-D1*A)
      RHO=D2*PRES/TEMP
      PRES=D3*PRES
      VS=DSQRT(D4*TEMP)
      DRHO=-RHO*(D1*DA+DTEMP/TEMP)
      DVS=0.5D0*D4*DTEMP/VS
      DPRES=-D1*PRES*DA
      IF(ALT-2.0D2/1.0D-3) 5,5,6
6     A=ALT-2.0D2/1.0D-3
      RHO=RHO+DRHO*A
      PRES=DPRES*A+PRES
      VS=VS+DVS*A
```

```
5       IF(ALT)7,8,8
7       RHO=RHO+DRHO*ALT
        PRES=PRES+DPRES*ALT
        VS=VS+DVS*ALT
8       CONTINUE
        RETURN
        END
```

1292 LINES PRINTED

APPENDIX B

LISTING OF PHASE II PROGRAM

NOTE: The Phase II Program is built to use either single aerodynamic approximations or spline-fit aerodynamics. Both listings are presented in this Appendix. To use the simple aerodynamic approximations, use the listing of pages 68-90; to use the spline-fit aerodynamics replace MAIN, DERIV1, and DERIV2 by the listings on pages 91-98 and add the subroutine SETUP (pages 93-94).

```
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION    C(7),SVARO(6),STVRS(999,6),U(999,3),TEMP(2),UTM(2),
     2XJ(8,1),XLAMF(6),GRAD(999,3),SERCH(999,3),Y(15,15,2)
      COMMON/CONS1/PI,RE,XMU,OMEGE,AREA,ECOEF,GNOT
      COMMON/CONS2/DELTS,TCUT,EPST,EPSTF,EPSA,EPSIT,ERR,ITMAX,ITMX,
     2KOUNTM,IKEY
      COMMON/CONS3/CSTR,B,C,DTEM,XDTEM
      COMMON/CNTRL/GRAD,SERCH,U,ASTR,STF,    KJIS,IJKU,ISTAR
      COMMON/STATO/SVARO,TO
      COMMON/STATE/ALT,XMASS,UTM,STVRS
      COMMON/STATF/ALTF,XMACH,FLTANG,VF,GAMMF,TF
      COMMON/PRINT/IOUT,IOUT2,IPRNT1,IPRNT2
      COMMON/STORE/DELSV,DELSE,DELGE,DELT,KEN
      NAMELIST/ANAME/PI,RE,XMU,OMEGE,AREA,ECOEF,DELTS,IKEY,TCUT,EPST,
     2EPSTF,EPSA,EPSIT,ERR,ITMAX,ITMX,KOUNTM,CSTR,B,C,DTEM,SVARO,TO,ALTF
     3,XMACH,FLTANG,GAMMF,XMASS,IOUT,IOUT2,IPRNT1,IPRNT2,VF
C     READ IN DATA
      READ(5,ANAME)
      READ(7,700) IJKU
      READ(7,750)  ((U(I,J),J=1,3),I=1,IJKU)
      WRITE(6,ANAME)
C CALL CONJUGATE GRADIENT ROUTINE
      CALL MPRJCG(IER)
      GO TO (10,20,30,40,50,60,70,80,90,100),IER
   10 CONTINUE
   20 WRITE(6,520)
      GO TO 101
   30 WRITE(6,530)
      GO TO 101
   40 WRITE(6,540)
      GO TO 101
   50 WRITE(6,550)
      GO TO 101
   60 WRITE(6,560)
      GO TO 101
   70 WRITE(6,570)
      GO TO 101
   80 WRITE(6,580)
      GO TO 101
   90 WRITE(6,590)
      GO TO 101
  100 WRITE(6,600)
  101 CONTINUE
      WRITE(8,625) IJKU
      WRITE(8,650) ((U(K,L),L=1,3),K=1,IJKU)
      STOP
 500  FORMAT(2I4)
 505  FORMAT(8F10.0)
  520 FORMAT(1H0,5X,'ONE-D SEARCH FAILED TO FIND A MINIMUM')
  530 FORMAT(1H0,5X,'COST IS NOT DECREASING IN SEARCH DIRECTION')
  540 FORMAT(1H0,5X,'CONVERGENCE ON SMALL CONTROL CHANGE')
  550 FORMAT(1H0,5X,'LITTLE COST CHANGE IN LAST TWO ITERATIONS')
  560 FORMAT(1H0,5X,'FAILED TO CONVERGE IN ITMAX ITERATIONS')
  570 FORMAT(1H0,5X,'INITIAL TRAJECTORY FAILED TO REACH CUT-OFF ALT')
  580 FORMAT(1H0,5X,'TOO MANY INTEGRATIONS STEPS REQUIRED ')
  590 FORMAT(1H0,5X,'BACKWARD INTEGRATED TRAJECTORY ERRORS')
  600 FORMAT(1H0,5X,'CONVERGENCE ON ZERO GRADIENT NORM')
 625  FORMAT(' ',I5)
 650  FORMAT(' ',3D26.16)
700   FORMAT(I5)
750   FORMAT(3D26.16)
      END
```

```
      SUBROUTINE WPRJCG(IER)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION XJ(8,1),XLAMF(7),SERCH(999,3),U(999,3),TEMPU(999,2),C(7)
     2,GRAD(999,3)
      COMMON/CONS2/DELTS,TCUT,EPST,EPSTF,EPSA,EPSIT,ERR,ITMAX,ITMX,
     2KOUNTM,IKEY
      COMMON/CONS3/CSTR,B,C,DTEM,XDTEM
      COMMON/CNTRL/GRAD,SERCH,U,ASTR,STF,    KJIS,IJKU,ISTAR
      COMMON/STATE/ALTE,XMACH,FLTANG,VE,GAMME,TF
      COMMON/PRINT/IOUT,IOUT2,IPRNT1,IPRNT2
      ITER=0
      ASTR=0.0D0
C PERFORM FORWARD INTEGRATION TO ALTITUDE CUT-OFF
8     ISTAR=0
      IFLAG=1
      KMX=0
      CALL FWDINT(COST,XJ,XLAMF,DCDTF,IFLAG)
      IF(IFLAG .NE. 1) GO TO 94
      IF(ITER.EQ.0) GO TO 9
C CHECK CHANGE IN COST VALUES
    5 IF(DABS(COST-CSAVE) .LT. ERR) GO TO 96
C PERFORM BACKWARD INTEGRATION
9     CALL BAKINT(XJ,XLAMF,TF,ITER,DCOST,XNORMS,DCDTF)
      IF(ITER-ITMAX) 7,95,93
7     CSAVE=COST
      ITNUM = ITER + 1
      WRITE(6,603) ITNUM
  603 FORMAT(1H0,5X,'ITERATION NUMBER',I5//)
C ENTER 1-D SEARCH
      KFLG = 0
    6 ISTAR = 1
      IFLAG=2
      JKNT = 0
      TFS=TF
      IFLG=0
   10 CALL SEKALF(COST,DCOST,ASTR,CSTR,XNORMS,B,TO,TFS,ITMX,IFLG)
      IF(KMX .GT. 5)  GO TO 100
      JKNT = JKNT + 1
      IF(IPRNT1.GT.0) WRITE(6,600) JKNT,ASTR
  600 FORMAT(5X,'1-D SEARCH TRIAL =',I5,5X,'PARAMETER =',D24.16)
      IF(IFLG .GT. ITMX) GO TO 11
      CALL FWDINT(COST,XJ,XLAMF,DCDTF,IFLAG)
      GO TO 10
11    IF(IFLG .GT.ITMX+1)GO TO 99
      IFLAG=1
      CALL FWDINT(COST,XJ,XLAMF,DCDTF,IFLAG)
      KMX=KMX+1
      IFLG=1
      IF(COST .GT. CSAVE) GO TO 10
C   CHECK FOR SMALL CONTROL NORM CHANGE
   14 UNORM = ASTR*XNORMS
      IF(UNORM.LT.EPST)  GO TO 98
      KMX=0
C HAVE FOUND INTERPOLATED VALUE, UPDATE CONTROL AT FREQUENCY OF SEARCH
12    KTAU=1
      DO 60 L=1,KJIS
      TAU=SERCH(L,3)
52    IF(U(KTAU,3) .GT. TAU) GO TO 54
      IF(KTAU .GE. IJKU) GO TO 57
      KTAU=KTAU+1
```

```
      GO TO 52
C TAU LIES BETWEEN U(KTAU-1,3) AND U(KTAU,3)
54    IF(KTAU .EQ. 1) GO TO 56
C USE LINEAR INTERPOLATION IN CNTRL DIRECTION GENERATION
      DO 55 K=1,2
      TEMPU(L,K)=-ASTR*SERCH(L,K)+(U(KTAU,K)-U(KTAU-1,K))*(TAU-U(KTAU-1,
     23))/(U(KTAU,3)-U(KTAU-1,3))+U(KTAU-1,K)
55    CONTINUE
      GO TO 60
C     USE LINEAR EXTRAPOLATION
56    KTAU=2
57    DO 58 K=1,2
      TEMPU(L,K)=-ASTR*SERCH(L,K)+(U(KTAU,K)-U(KTAU-1,K))*(TAU-U(KTAU,3)
     2)/(U(KTAU,3)-U(KTAU-1,3))+U(KTAU,K)
58    CONTINUE
60    CONTINUE
      DO 62 L=1,KJIS
      DO 61 M=1,2
61    U(L,M)=TEMPU(L,M)
62    U(L,3)=SERCH(L,3)
      IJKU=KJIS
   65 CONTINUE
      ITER=ITER+1
      GO TO 5
C 1-D SEARCH ERRORS
  100 IER = 2
      IF(IFLG .EQ. ITMX+2)IER=3
      RETURN
C  HAVE CONVERGENCE DUE TO SMALL CONTROL NORM CHANGE
   98 IER = 4
      RETURN
C HAVE CONVERGENCE DUE TO NO COST CHANGE
96    IER=5
      RETURN
C HAVE EXCEEDED PERMITTED NUMBER OF CG STEPS
95    IER=6
      RETURN
C HAVE FAILED TO REACH ALTITUDE CUT-OFF
94    IER=7
      RETURN
93    IF(ITER-(ITMAX+2)) 92,91,90
C NOT ENOUGH STORAGE SPACE FOR GRADIENT
92    IER=8
      RETURN
C CANNOT FIND INTEGRATION COT-OFF POINT
91    IER=9
      RETURN
C HAVE CONVERGED ON GRADIENT NORM
90    IER=10
      RETURN
   99 IF(IFLG-(ITMX+3)) 110,105,100
C  NEW SEARCH IN GRADIENT DIRECTION
  105 IF(KFLG.GT.0) GO TO 100
       KFLG = 1
      DO 101 II=1,KJIS
      DO 101 JJ=1,3
  101 SERCH(II,JJ) = GRAD(II,JJ)
      GO TO 6
C  CHECK FOR COST DECREASE
  110 CONTINUE
```

```
IFLAG = 1
CALL FWDINT(COST,XJ,TF,XLAME,DCDTF,IFLAG)
IF(COST.GE.CSAVE) GO TO 105
GO TO 14
END
```

```
      SUBROUTINE SEKALF(COST,DCOST,ASTAR,CSTAR,SNORM,B,TO,TF,ITMAX,
     1 IFLAG)
      DOUBLE PRECISION COST, DCOST, ASTAR, CSTAR, SNORM, B, TO ,TF,
     1FUNT,ALF,BM,G,DETERM,AA,BB,CC,XNORM,BSTAR
      DIMENSION FUNT(20),ALF(20),BM(3,3),G(3)
      IF(IFLAG.GT.0) GO TO 20
      IF(DCOST.GE.0.0D0) GO TO 15
      IF(ASTAR .NE. 0.0D0)  GO TO 11
C   COMPUTE FIRST PARAMETER
      ASTAR = 2.0D0*(CSTAR - COST)/DCOST
      BSTAR=B*DSQRT(2.D0*TF)/SNORM
      IF(ASTAR .GT. BSTAR)  ASTAR=BSTAR
      XNORM=1.D0/SNORM
      IF(ASTAR .LE.  0.0D0) ASTAR=XNORM
   11 FUNT(1)=COST
      ALF(1) = 0.0D0
      IFLAG = 1
      RETURN
C   SLOPE OF COST IS NOT NEGATIVE
   15 WRITE(6,100) DCOST
  100 FORMAT(1H0,10X,'THE VALUE OF THE NON-NEGATIVE SLOPE IS',D24.16)
      IFLAG = ITMAX + 2
      RETURN
C   COMPUTE SECOND PARAMETER
   20 IF (IFLAG.GT.1) GO TO 30
      ALF(2) = ASTAR
      FUNT(2) = COST
      IF(FUNT(2).LE.FUNT(1)) GO TO 25
      ASTAR = ALF(2)/2.0D0
      IFLAG = 2
      RETURN
   25 IFLAG = 2
      GO TO 31
C   COMPUTE THIRD PARAMETER
   30 IF (IFLAG.LT.3) GO TO 59
      ALF(IFLAG) = ASTAR
      FUNT(IFLAG) = COST
      IF(FUNT(IFLAG).GT.FUNT(IFLAG-1)) GO TO 50
   31 ASTAR = ALF(2)*(2.0D0)**(IFLAG-1)
      IF(IFLAG.GE.ITMAX) GO TO 40
      IFLAG = IFLAG + 1
      RETURN
C   CANNOT FIND A MINIMUM
   40 WRITE(6,101)
  101 FORMAT(1H0,10X,'SEARCH HAS EXCEEDED MAXIMUM NUMBER OF STEPS')
      IFLAG = ITMAX + 2
      RETURN
C   GET DATA FOUR POINT INTERPOLATION
   50 IF(IFLAG.EQ.3) GO TO 60
      IFLAG = IFLAG - 3
      DO 51 I=1,3
      BM(I,3) = ALF(IFLAG) - ALF(IFLAG+I)
      BM(I,2) =(ALF(IFLAG) + ALF(IFLAG+I))*BM(I,3)/2.0D0
      BM(I,1) =(ALF(IFLAG)**3 - ALF(IFLAG+I)**3)/3.0D0
   51 G(I) = FUNT(IFLAG) - FUNT(IFLAG+I)
      GO TO 70
C GET DATA FOR  THREE POINT AND SLOPE INTERPOLATION
 59   ALF(3) = ASTAR
      FUNT(3) = COST
   60 G(1) = (ALF(3) - ALF(2))*(ALF(3)*ALF(2))**2
```

```
      G(2) = FUNT(2)*ALF(3)**2 - FUNT(3)*ALF(2)**2 - ALF(2)*ALF(3)
     1 *(ALF(3)-ALF(2))*DCOST - (ALF(3)**2 - ALF(2)**2)*FUNT(1)
      G(2) = -3.0D0*G(2)/G(1)
      G(3) = FUNT(2)*ALF(3)**3 - FUNT(3)*ALF(2)**3 - ALF(2)*ALF(3)
     1 *(ALF(3)**2 -ALF(2)**2)*DCOST - (ALF(3)**3-ALF(2)**3)*FUNT(1)
      G(3) = 2.0D0*G(3)/G(1)
      AA = G(2)
      BB = G(3)
      CC= DCOST
      GO TO 71
C   SOLVE FOR COEFFICIENTS BY CRAMER'S RULE
   70 DETERM = BM(1,1)*(BM(2,2)*BM(3,3)-BM(3,2)*BM(2,3))
     1 + BM(1,2)*(BM(3,1)*BM(2,3) - BM(3,3)*BM(2,1))
     2 + BM(1,3)*(BM(2,1)*BM(3,2) - BM(3,1)*BM(2,2))
      AA = (G(1) *(BM(2,2)*BM(3,3) - BM(3,2)*BM(2,3))
     1    + G(2)*(BM(3,2)*BM(1,3) - BM(1,2)*BM(3,3))
     2    + G(3)*(BM(1,2)*BM(2,3) - BM(2,2)*BM(1,3)))/DETERM
      BB = (G(1)*(BM(2,3)*BM(3,1) - BM(3,3)*BM(2,1))
     1     + G(2)*(BM(1,1)*BM(3,3) - BM(3,1)*BM(1,3))
     2     + G(3)*(BM(2,1)*BM(1,3) - BM(2,3)*BM(1,1)))/DETERM
      CC = (G(1)*(BM(2,1)*BM(3,2) - BM(3,1)*BM(2,2))
     1     + G(2) *(BM(1,2)*BM(3,1) -BM(3,2)*BM(1,1))
     2     + G(3)*(BM(1,1)*BM(2,2) - BM(2,1)*BM(1,2)))/DETERM
C   COMPUTE MINIMIZING ALPHA
   71 IF(BB.GT.0.0D0) GO TO 73
      ASTAR = (-BB + DSQRT(BB**2 - 4.0D0*AA*CC))/AA/2.0D0
   72 IFLAG = ITMAX + 1
      RETURN
   73 ASTAR = -2.0D0*CC/(BB + DSQRT(BB**2 - 4.0D0*AA*CC))
      GO TO 72
      END
```

```
      SUBROUTINE FWDINT(COST,XJ,    XLAMF,DCDTF,IFLAG)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION YPR(8,4,1),DPSAVE(8,1),DV(8,1),P(8,1),TE(8,1),RPAR(1),
     2IPAR(1),SVARO(6),XLAMF(6),C(7),STVRS(999,6),DEP(8,1),UTM(2),A(4)
     3,B(4)
      COMMON/CONS1/PI,RE,XMU,OMEGE,AREA,ECOEF,GNOT
      COMMON/CONS2/DELTS,TCUT,EPST,EPSTF,EPSA,EPSIT,ERR,ITMAX,ITMX,
     2KOUNTM,IKEY
      COMMON/STATE/ALT,XMASS,UTM,STVRS
      COMMON/STATO/SVARO,TO
      COMMON/STATF/ALTF,XMACH,FLTANG,VF,GAMMF,TF
      COMMON/PRINT/IOUT,IOUT2,IPRNT1,IPRNT2
      COMMON/STORE/DELSV,DELSE,DELGE,DELT,KEN
C INITIALIZATION
      TEST=SVARO(1)-RE
      TEND=TCUT
      LMN=IOUT-1
      INX=2
      IPAR(1) = 0
   10     SIX     = 24.0D0
          ZERO    =  0.0D0
C
          A(1)    = -9.0D0/SIX
          A(2)    = 37.0D0/SIX
          A(3)    =-59.0D0/SIX
          A(4)    = 55.0D0/SIX
          B(1)    =  1.0D0/SIX
          B(2)    = -5.0D0/SIX
          B(3)    = 19.0D0/SIX
          B(4)    = -A(1)
C
          RATIO   = 19.0D0/270.0D0
          SIX     =  6.0D0
          TWO     =  2.0D0
C
      M=1
      L=1
      NE=1
      N=8
          M1      = 4
          M2      = 1
          M3      = 2
          M4      = 3
      DELT=DELTS
      DO 111  I=1,6
  111 DEP(I,1)=SVARO(I)
      DEP(7,1)=0.0D0
      DEP(8,1)=0.0D0
      K=1
      TM=TO
   20 ASSIGN 100 TO IPL5
          KOUNT   = 0
          DELBY6 = DELT/SIX
          DELBY2 = DELT/TWO
C
      DO 21 JAY = 1,NE
      DO 22 J   = 1,N
      DPSAVE(J,JAY) = DEP(J,JAY)
   22 DV(J,JAY) = DEP(J,JAY)
      CALL DERIV1(TM,DV,P,L, M ,N,NE,RPAR,IPAR,1)
```

```
   21 CONTINUE
C
C  EVALUATING THE DERIVATIVES AND STORING THE VALUES
C
      DO 25 JAY = 1,NE
      DO 26 J   = 1,N
   26  YPR(J,M1,JAY) = P(J,JAY)
   25 CONTINUE
      IF(INX .EQ. 3) GO TO 30
      COST=0.0D0
      DCDTF=0.0D0
      KCONT=0
C PERFORM INTEGRATION AND COMPUTE COST
  202 CONTINUE
      DO  320  J=1,8
  320 DPSAVE(J,1)=DEP(J,1)
      DO  322  J=1,6
  322 STVRS(K,J)=DEP(J,1)
      K=K+1
      IF(K .GT. 999) GO TO 651
      LMN = LMN + 1
      TESTP = TEST
      IF((LMN/IOUT)*IOUT .EQ. LMN)  WRITE(6,600) TM, (DEP(I,1),I=1,8),
     2(UTM(J),J=1,2)
C  REPLACED STEPER BY RK713
  30  V=TM
      TM=V+DELT
C
         GO TO IPL5,(100,200)
C
C  ENTRY ADAMS PREDICTIOR-CORRECTOR
C
C  APPLICATION OF THE PREDICTOR EQUATION AND THE DERIVATIVE EVALUATION
C
  200 DO 220 JAY=1,NE
      DO 210 J   = 1,N
      TE(J,JAY)=B(3)*YPR(J,M1,JAY)+B(2)*YPR(J,M2,JAY)+B(1)*YPR(J,M3,JAY)
      DV(J,JAY)=DEP(J,JAY)+ DELT*(A(4)*YPR(J,M1,JAY)+A(3)*YPR(J,M2,JAY))
      DV(J,JAY)=DV(J,JAY) + DELT*(A(2)*YPR(J,M3,JAY)+A(1)*YPR(J,M4,JAY))
  210 CONTINUE
      CALL DERIV1(TM,DV,P,L, M ,N,NE,RPAR,IPAR,1)
  220 CONTINUE
C
C  APPLICATION OF THE CORRECTOR EQUATION AND THE DERIVATIVE EVALUATION
C
      DO 240 JAY=1,NE
      DO 230 J   = 1,N
  230 DV(J,JAY) = DEP(J,JAY) + DELT*(B(4)*P(J,JAY) + TE(J,JAY))
      CALL DERIV1(TM,DV,P,L, M ,N,NE,RPAR,IPAR,1)
  240 CONTINUE
C
C  SECOND APPLICATION OF THE CORRECTOR EQUATION AND COMPUTING
C  THE SINGLE STEP ERROR
C
      DO 250 JAY=1,NE
      DO 250 J   = 1,N
      YPR(J,M4,JAY) = P(J,JAY)
      DEP(J,JAY)= DEP(J,JAY) + DELT*(B(4)*P(J,JAY) + TE(J,JAY))
      DV(J,JAY) = DEP(J,JAY)
  250 CONTINUE
```

```
5000     M0      = M4
         M4      = M3
         M3      = M2
         M2      = M1
         M1      = M0
      IF(INX .EQ. 2) GO TO 202
      IF(TM .GE. TEND) GO TO 400
      ALT=DEP(1,1)-RE
      TEST=ALT-ALTF
      IF(DABS(TEST) .LT. EPSA) GO TO 80
      IF(TEST) 70,80,90
  90  IF(INX .EQ. 1) GO TO 202
C FINAL ALTITUDE ITERATION
  70  KCONT=KCONT+1
      INX=3
      IF(KCONT .GT. KOUNTM) GO TO 101
      DO  79  I=1,8
  79  DEP(I,1)=DPSAVE(I,1)
      TM=TM-DELT
      DELT=DELT*DABS(TESTP/(TESTP-TEST))
      GO TO 20
  80  TF=TM
      DELSV=DELT
      KEN=K
      DO 380  J=1,6
 380  SIVRS(K,J)=DEP(J,1)
  81  COST=COSTFN(DEP)
      CALL XLAMEN(XLAMF,DEP,L,M)
 104  WRITE(6,600)   TM,(DEP(I,1),I=1,8),(UTM(J),J=1,2)
      IF(IFLAG.NE.2.AND.IKEY.GT.0) CALL OUTPUT(DEP,UTM)
      WRITE(6,601)   COST,(XLAMF(I),I=1,6)
600   FORMAT(1H0,5X,'TIME =',1PD24.16/32X,'STATE'/6X,1P4D24.16/
     1 6X,1P4D24.16/39X,'CONTROL'/6X,1P3D24.16)
 601  FORMAT(1H0,5X,'COST FUNCTION=',1PD24.16/35X,'FINAL MULTIPLIERS'/
     1 6X,1P4D24.16/6X,1P2D24.16/6X)
      RETURN
105   COST=COSTFN(DEP)
      IF(IPRNT1.GE.2) GO TO 104
      RETURN
 400  WRITE(6,410)
      IFLAG=IFLAG+1
      RETURN
101   WRITE(6,420)
      IFLAG=IFLAG+3
      RETURN
410   FORMAT(1H0,5X,'EXCEEDED CUTOFF TIME ON RUN WITH ALTITUDE CUTOFF')
420   FORMAT(1H0,5X,'EXCEEDED MAXIMUM NUMBER OF ITERATIONS IN TERMINAL C
     2UTOFF')
C  ENTRY RUNGE KUTTA
C
 100     VV      = V + DELBY2
C
      DO 120 JAY=1,NE
      DO 110 J  = 1,M
 110  DV(J,JAY) = DEP(J,JAY) + YPR(J,M1,JAY)*DELBY2
      CALL DERIV1(VV,DV,P,L, M ,N,NE,RPAR,IPAR,1)
 120  CONTINUE
C
      DO 140 JAY=1,NE
      DO 130 J  = 1,N
```

```
  130 DV(J,JAY) = DEP(J,JAY) + P(J,JAY)*DELBY2
      CALL DERIV1(VV,DV,TE,L,M,N,NF,RPAR,IPAR,1)
  140 CONTINUE
C
      DO 160 JAY=1,NE
      DO 150 J  = 1,N
      DV(J,JAY) = DEP(J,JAY) + TE(J,JAY)*DELT
  150 TE(J,JAY) = 2.0D0 * (TE(J,JAY) + P(J,JAY))
      CALL DERIV1(TM,DV,P,L, M ,N,NE,RPAR,IPAR,1)
  160 CONTINUE
C
      DO 180 JAY=1,NE
      DO 170 J  = 1,N
      DEP(J,JAY)= DEP(J,JAY) + DELBY6*(P(J,JAY)+TE(J,JAY)+YPR(J,M1,JAY))
  170 DV(J,JAY) = DEP(J,JAY)
      CALL DERIV1(TM,DV,P,L, M ,N,NF,RPAR,IPAR,1)
      DO 190 J  = 1,N
  190 YPR(J,M4,JAY) = P(J,JAY)
  180 CONTINUE
C
         KOUNT  = KOUNT + 1
         IF (KOUNT .LT. 3) GO TO 5000
         ASSIGN 200 TO IPL5
      INX=1
         GO TO 5000
651   WRITE(6,650)
      STOP
 650  FORMAT(' ','EXCEEDED STATE VAR. STORAGE')
      END
```

```
      SUBROUTINE BAKINT(XJ,XLAMF,TG,ITER,DCOST,XNORMS,DCDTF)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION YPR(6,4,1),DPSAVE(6,1),DV(6,1),P(6,1),TE(6,1),RPAR(1),
     2IPAR(1),TEMPS(999,3),XLAMF(6),C(7),DEP(6,1),A(4),B(4)
     4,GRAD(999,3),SERCH(999,3),U(999,3),G(999),STVRS(999,6),UTM(2)
      COMMON/CONS1/PI,RE,XMU,OMEGE,AREA,FCOEF,GNOT
      COMMON/CONS2/DELTS,TCUT,EPST,EPSTF,EPSA,EPSIT,ERR,ITMAX,ITMX,
     2KOUNTM,IKEY
      COMMON/CNTRL/GRAD,SERCH,U,ASTR,STF,    KJIS,IJKU,ISTAR
      COMMON/STATE/ALT,XMASS,UTM,STVRS
      COMMON/STATE/ALTF,XMACH,FLTANG,VF,GAMMF,TF
      COMMON/PRINT/IOUT,IOUT2,IPRNT1,IPRNT2
      COMMON/STORE/DELSV,DELSF,DELGE,DELT          ,KEN
C     INITIALIZATION
   10    SIX     = 24.0D0
         ZERO    =  0.0D0
C
         A(1)    = -9.0D0/SIX
         A(2)    = 37.0D0/SIX
         A(3)    =-59.0D0/SIX
         A(4)    = 55.0D0/SIX
         B(1)    =  1.0D0/SIX
         B(2)    = -5.0D0/SIX
         B(3)    = 19.0D0/SIX
         B(4)    = -A(1)
C
         RATIO   = 19.0D0/270.0D0
         SIX     =  6.0D0
         TWO     =  2.0D0
C
      L=IJKU
      NE=1
      N=6
         M1      = 4
         M2      = 1
         M3      = 2
         M4      = 3
      DELT=DELSV
      DO 111  I=1,6
  111 DEP(I,1)=XLAMF(I)
      TM=0.0
      NEK=KEN
      WRITE(6,900) NEK,KEN
  900 FORMAT(' ',2I4)
      IJK=999
      LMN=IOUT2-1
      INX=3
   20 ASSIGN 100 TO IPL5
         KOUNT   = 0
         DELBY6 = DELT/SIX
         DELBY2 = DELT/TWO
C
      DO 21 JAY = 1,NE
      DO 22 J    = 1,N
      DPSAVE(J,JAY) = DEP(J,JAY)
   22 DV(J,JAY) = DEP(J,JAY)
      CALL DERIV2(TM,DV,P,L,NEK,0,RPAR,IPAR)
   21 CONTINUE
C
      CALL GRADFN(DEP,TM,IJK)
```

```
      TEST=TF-TM
      LMN=LMN+1
      LTEST=MOD(LMN,IOUT2)
      IF(LTEST .EQ. 0) WRITE(6,600) TEST,(STVRS(NEK,K),K=1,6),(DV(I,1),
     2I=1,6),(GRAD(IJK+1,J),J=1,2)
C  EVALUATING THE DERIVATIVES AND STORING THE VALUES
C
      DO 25 JAY = 1,NE
      DO 26 J   = 1,N
  26  YPR(J,M1,JAY) = P(J,JAY)
   25 CONTINUE
C     PERFORM INTEGRATION AND COMPUTE GRAD.
  30  V=TM
      TM=V+DELT
      NEK=NEK-1
C
         GO TO IPL5,(100,200)
C
C  ENTRY ADAMS PREDICTIOR-CORRECTOR
C
C  APPLICATION OF THE PREDICTOR EQUATION AND THE DERIVATIVE EVALUATION
C
  200 DO 220 JAY=1,NE
      DO 210 J  = 1,N
      TE(J,JAY)=B(3)*YPR(J,M1,JAY)+B(2)*YPR(J,M2,JAY)+B(1)*YPR(J,M3,JAY)
      DV(J,JAY)=DEP(J,JAY)+ DELT*(A(4)*YPR(J,M1,JAY)+A(3)*YPR(J,M2,JAY))
      DV(J,JAY)=DV(J,JAY) + DELT*(A(2)*YPR(J,M3,JAY)+A(1)*YPR(J,M4,JAY))
  210 CONTINUE
      CALL DERIV2(TM,DV,P,L,NEK,O,RPAR,IPAR)
  220 CONTINUE
C
C  APPLICATION OF THE CORRECTOR EQUATION AND THE DERIVATIVE EVALUATION
C
      DO 240 JAY=1,NE
      DO 230 J   = 1,N
  230 DV(J,JAY) = DEP(J,JAY) + DELT*(B(4)*P(J,JAY) + TE(J,JAY))
      CALL DERIV2(TM,DV,P,L,NEK,O,RPAR,IPAR)
  240 CONTINUE
C
C  SECOND APPLICATION OF THE CORRECTOR EQUATION AND COMPUTING
C  THE SINGLE STEP ERROR
C
      DO 250 JAY=1,NE
      DO 250 J  = 1,N
      YPR(J,M4,JAY) = P(J,JAY)
      DEP(J,JAY)= DEP(J,JAY) + DELT*(B(4)*P(J,JAY) + TE(J,JAY))
      DV(J,JAY) = DEP(J,JAY)
  250 CONTINUE
 5000    MO      = M4
         M4      = M3
         M3      = M2
         M2      = M1
         M1      = MO
C     C
      DO  320  J=1,N
  320 DPSAVE(J,1)=DEP(J,1)
      TEST=TF-TM
      IF(TEST .LE. 1.0D-1) GO TO 500
      CALL DERIV2(TM,DV,P,L,NEK,O,RPAR,IPAR)
      CALL GRADEN(DEP,TM,IJK)
```

```
      LMN=LMN+1
      LTEST=MOD(LMN,IOUT2)
      IF(LTEST .EQ. 0) WRITE(6,600) TEST,(STVRS(NEK,K),K=1,6),(DV(I,1),
     2I=1,6),(GRAD(IJK+1,J),J=1,2)
      GO TO 30
C  ENTRY RUNGE KUTTA
C
  100     VV      = V + DELBY2
C
      DO 120 JAY=1,NE
      DO 110 J   = 1,N
  110 DV(J,JAY) = DEP(J,JAY) + YPR(J,M1,JAY)*DELBY2
      CALL DERIV2(VV,DV,P,L,NEK,1,RPAR,IPAR)
  120 CONTINUE
C
      DO 140 JAY=1,NE
      DO 130 J   = 1,N
  130 DV(J,JAY) = DEP(J,JAY) + P(J,JAY)*DELBY2
      CALL DERIV2(VV,DV,TE,L,NEK,1,RPAR,IPAR)
  140 CONTINUE
C
      DO 160 JAY=1,NE
      DO 150 J   = 1,N
      DV(J,JAY) = DEP(J,JAY) + TE(J,JAY)*DELT
  150 TE(J,JAY) = 2.0D0 * (TE(J,JAY) + P(J,JAY))
      CALL DERIV2(TM,DV,P,L,NEK,0,RPAR,IPAR)
  160 CONTINUE
C
      DO 180 JAY=1,NE
      DO 170 J   = 1,N
      DEP(J,JAY)= DEP(J,JAY) + DELBY6*(P(J,JAY)+TE(J,JAY)+YPR(J,M1,JAY))
  170 DV(J,JAY) = DEP(J,JAY)
      CALL DERIV2(TM,DV,P,L,NEK,0,RPAR,IPAR)
      DO 190 J   = 1,N
  190 YPR(J,M4,JAY) = P(J,JAY)
  180 CONTINUE
C
      IF(INX .EQ. 3) GO TO 800
         KOUNT  = KOUNT + 1
         IF (KOUNT .LT. 3) GO TO 5000
         ASSIGN 200 TO IPL5
      INX=1
         GO TO 5000
 800  INX=1
      DELT=DELTS
      GO TO 20
 500  CALL DERIV2(TM,DV,P,L,NEK,0,RPAR,IPAR)
      CALL GRADFN(DEP,TM,IJK)
      WRITE(6,600) TEST,(STVRS(NEK,K),K=1,6),(DEP(I,1),I=1,6),(GRAD(IJK+
     21,J),J=1,2)
  600 FORMAT(' ','TIME=',1PD24.16/47X,'STATE'/6X,1P4D24.16/6X,1P2D24.16/
     247X,'MULTIPLIERS'/6X,1P4D24.16/6X,1P2D24.16/30X,'GRADIENT'/6X,1P3
     3D24.16)
C     SHIFT GRADIENT STORAGE
      KJI=999-IJK
      DO 830  L=1,KJI
      DO 830  M=1,3
 830  GRAD(L,M)=GRAD(IJK+L,M)
C     FORM GRADIENT QUADRATURE BY TRAPEZOEDAL RULE
      DO 40 K=1,KJI
```

```
      G(K)=GRAD(K,1)*GRAD(K,1)+GRAD(K,2)*GRAD(K,2)
   40 CONTINUE
      BETAN=0.0
      JI=KJI-1
      DO 41  L=2,JI
 41   BETAN=BETAN+(G(L)+G(L-1))*DELTS/2.0D0
      BETAN=BETAN+(G(KJI)+G(KJI-1))*DELSV/2.D0
      IF(BETAN .LE. EPSIT)  GO TO 101
C     GET DERIVATIVE OF COST WITH RSEPECT TO  PARAMETER
      DCOST=-BETAN
C     GET NORM OF SEARCH DIRECTION
      IF(ITER .EQ. 0) GO TO 42
      XNORMS=DSQRT(BETAN+(BETAN*XNORMS/BETAD)**2)
      GO TO 43
   42 XNORMS=DSQRT(BETAN)
   43 CONTINUE
      IF(IPRNT2  .GT. 0) WRITE(6,601) BETAN,XNORMS,DCOST
  601 FORMAT(1H0,5X,'GRADIENT NORM SQUARED =',1PD24.16/
     216X,'SEARCH DIRECTION NORM =',D24.16/6X,'COST SLOPE IN SEARCH
     2DIRECTION =',1PD24.16)
C     GET NEW SEARCH DIRECTION
      IF((ITER/5)*5 .NE. ITER) GO TO 51
      DO 50 K=1,KJI
      DO  50 L=1,3
   50 SERCH(K,L)=GRAD(K,L)
      STF=DCDTF
      DELSE=DELSV
      GO TO 80
   51 DELG=DELTS
      DELS=DELTS
      DO 60  L=1,KJI
      IF(L .GE. KJIS)  GO TO 202
      IF(L .EQ. KJI)  GO TO 400
      DO 105 K=1,2
  105 TEMPS(L,K)=GRAD(L,K)+(BETAN/BETAD)*SERCH(L,K)
      GO TO 60
  202 DELS=DELSE
      IF(L .EQ. KJI)  DELG=DELSV
      DO 300  K=1,2
  300 TEMPS(L,K)=GRAD(L,K)+(BETAN/BETAD)*SERCH(KJIS,K)
      GO TO 60
  400 DELG=DELSV
      DO  501  K=1,2
  501 TEMPS(L,K)=GRAD(L,K)+(BETAN/BETAD)*(SERCH(L-1,K)+DELG/DELS*
     2(SERCH(L,K)-SERCH(L-1,K)))
   60 CONTINUE
C     STORE SEARCH DIRECTION
      DELSE=DELG
      DO 62 L=1,KJI
      DO 61 M=1,2
   61 SERCH(L,M)=TEMPS(L,M)
   62 SERCH(L,3)=GRAD(L,3)
      STF=DCDTF+(BETAN/BETAD)*STF
   80 KJIS=KJI
      BETAD=BETAN
      RETURN
  101 WRITE(6,225)
      ITER=ITMAX+3
      RETURN
  225 FORMAT(1H0,5X,'GRADIENT NORM LESS THAN TOLERANCE')
      END
```

```
      SUBROUTINE  DERIV1(T,X,P,L,M,N,NE,RPAR,IPAR,ND)
      IMPLICIT  REAL*8(A-H,O-Z)
      DIMENSION  X(N,NE),P(N,NE),RPAR(ND),IPAR(ND),TEMP(2),
     1TEM(2),TF(3,2),U(999,3),SERCH(999,3),GRAD(999,3)
      COMMON/CONS1/PI,RE,XMU,OMEGE,AREA,ECOEF,GNOT
      COMMON/CNTRL/GRAD,SERCH,U,ASTR,STF,    KJIS,IJKU,ISTAR
      COMMON/STATE/ALT,XMASS,TEMP
      COMMON/STATE/ALTF,XMACH,FLTANG,VF,GAMMF
C     T = TIME
C     X = STATE AND INTEGRATED COST
C     P = DERIVATIVES OF STATE AND INTEGRATED COST AT T
C     COMPUTE TRIG FUNCTIONS OF PHI,GAMMA,CHI
      TF(1,1)=DSIN(X(3,1))
      TF(1,2)=DCOS(X(3,1))
      TF(2,1)=DSIN(X(5,1))
      TF(2,2)=DCOS(X(5,1))
      TF(3,1)=DSIN(X(6,1))
      TF(3,2)=DCOS(X(6,1))
C     COMPUTE RELATIVE VELOCITY
      RMAG1=X(1,1)
      V=X(4,1)
      VRMAG=V
C     COMPUTE ATMOSPHERIC PARAMETERS
      ALTI=X(1,1)-RE
      CALL ATMOS(ALTI,TEMPR,PRES,RHO,VS,DVS,DRHO,DPRES)
      XMACH=VRMAG/VS
      RHO=DABS(RHO)
C     COMPUTE DERIVATIVES NO ATMOS.
      P(1,1)=V*TF(2,1)
      P(2,1)=(V*TF(2,2)*TF(3,2))/(RMAG1*TF(1,2))
      P(3,1)=(V*TF(2,2)*TF(3,1))/RMAG1
      P(4,1)=(-XMU*TF(2,1))/(RMAG1*RMAG1)
      P(5,1)=(-XMU*TF(2,2))/(RMAG1*RMAG1*V)+(V*TF(2,2))/RMAG1
      P(6,1)=(-V*TF(2,2)*TF(3,2)*TF(1,1))/(RMAG1*TF(1,2))
      IF(IPAR(1) .EQ. 1)  GO TO 100
C     FIND CONTROL
   20 IF(L .GE. IJKU)  GO TO  60
   30 IF(U(L+1,3) .LT. T)  GO TO  50
      IF(U(L,3) .GT. T)  GO TO 55
      TEMP(1)=U(L,1)+((U(L+1,1)-U(L,1))/(U(L+1,3)-U(L,3)))*(T-U(L,3))
      TEMP(2)=U(L,2)+((U(L+1,2)-U(L,2))/(U(L+1,3)-U(L,3)))*(T-U(L,3))
      L=L+1
      GO TO 40
   50 L=L+1
      GO TO 20
   55 L=L-1
      GO TO 20
   60 TEMP(1)=U(L,1)+((U(L,1)-U(L-1,1))/(U(L,3)-U(L-1,3))*(T-U(L,3)))
      TEMP(2)=U(L,2)+((U(L,2)-U(L-1,2))/(U(L,3)-U(L-1,3))*(T-U(L,3)))
   40 IF(ISTAR .EQ. 0)  GO TO 100
C     FIND SEARCH DIRECTION
   21 IF(M .GE. KJIS) GO TO 61
      IF(SERCH(M+1,3) .LT. T) GO TO 51
      IF(SERCH(M,3) .GT. T) GO TO 56
      DO 31 I=1,2
   31 TEM(I)=SERCH(M,I)+((SERCH(M+1,I)-SERCH(M,I))/(SERCH(M+1,3)
     2-SERCH(M,3)))*(T-SERCH(M,3))
      M=M+1
      GO TO 80
   51 M=M+1
```

```
      GO TO 21
   56 M=M-1
      GO TO 21
   61 DO 62 I=1,2
   62 TEM(I)=SERCH(M,I)+((SERCH(M,I)-SERCH(M-1,I))/(SERCH(M,3)-SERCH
     2(M-1,3)))*(T-SERCH(M,3))
   80 CONTINUE
C     FORCE CONTROL
      TEMP(1)=TEMP(1)-ASTR*TEM(1)
      TEMP(2)=TEMP(2)-ASTR*TEM(2)
  100 CONTINUE
C     COMPUTE AEROD . COEF.
      CL=2.2D0*DSIN(TEMP(1))**2*DCOS(TEMP(1))+1.0D-2
      CD=2.2D0*DSIN(TEMP(1))**3+8.0D-2
      Q=RHO*AREA*V*V/(XMASS*2.D0)
C     ADD CONTROL ACCELERATIONS
      P(4,1)=P(4,1)-Q*CD
      P(5,1)=P(5,1)+Q*CL*DCOS(TEMP(2))/V
      P(6,1)=P(6,1)-Q*CL*DSIN(TEMP(2))/(V*TE(2,2))
C     COMPUTE  HEATING AND HEAT RATE DERIVATIVES
      RHOO=1.225D0
      P(7,1)=ECOEF*DSQRT(RHO/RHOO)*(1.262D-4*VRMAG)**3.15
      DVRMAG=P(4,1)
      P(8,1)=DSQRT(RHO/RHOO)*3.15D0*1.262D-4*(1.262D-4*VRMAG)**2.15*
     1DVRMAG
      P(8,1)=P(8,1)+((1.262D-4*VRMAG)**3.15*DRHO)*P(1,1)/(2.D0*DSQRT(RHO
     1*RHOO))
      P(8,1)=ECOEF*P(8,1)*ECOEF*P(8,1)
      RETURN
      END
```

```
      SUBROUTINE DERIV2(TS,XS,P,L,M,NI,RPAR,IPAR)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION XS(6,1),STVRS(999,6),GRAD(999,3),SERCH(999,3),U(999,3),
     2TEMP(2),XP(6,1),X(6,1),P(6,1),C(7)
      COMMON/CONS1/PI,RE,XMU,OMEGE,AREA,ECOEF,GNOT
      COMMON/CONS3/CSTR,B,C,DTEM,XDTEM
      COMMON/CNTRL/GRAD,SERCH,U,ASTR,STF,    KJIS,IJKU,ISTAR
      COMMON/STATE/ALT,XMASS,TEMP,STVRS
      COMMON/STATE/ALTF,XMACH,FLTANG,VF,GAMMF,TF
      T=TF-TS
C     RETREIVE STATE VARIABLES FROM STORAGE
      IF(NI .EQ. 1)  GO TO 10
      DO 200 I=1,6
  200 X(I,1)=STVRS(M,I)
      GO TO 40
C     INTERPOLATE FROM STATE VAR.
   10 IF (M .EQ. 1)  GO TO 800
      DO 30  I=1,6
   30 X(I,1)=(STVRS(M,I)-STVRS(M-1,I))/2.D0+STVRS(M-1,I)
   40 CONTINUE
C     FIND CONTROL
   20 IF(L .GE. IJKU)  GO TO  60
      IF(U(L+1,3) .LT. T)  GO TO  50
      IF(U(L,3) .GT. T)  GO TO 55
      TEMP(1)=U(L,1)+((U(L+1,1)-U(L,1))/(U(L+1,3)-U(L,3)))*(T-U(L,3))
      TEMP(2)=U(L,2)+((U(L+1,2)-U(L,2))/(U(L+1,3)-U(L,3)))*(T-U(L,3))
      L=L+1
      GO TO 400
   50 L=L+1
      GO TO 20
   55 L=L-1
      GO TO 20
   60 IF(U(L-1,3) .GT. T)  GO TO 55
      TEMP(1)=U(L,1)+((U(L,1)-U(L-1,1))/(U(L,3)-U(L-1,3)))*(T-U(L,3)))
      TEMP(2)=U(L,2)+((U(L,2)-U(L-1,2))/(U(L,3)-U(L-1,3))*(T-U(L,3)))
  400 CONTINUE
C     COMPUTE TRIG QUANTITIES
      COSB=DCOS(TEMP(2))
      SINB=DSIN(TEMP(2))
      SINP=DSIN(X(3,1))
      COSP=DCOS(X(5,1))
      SING=DSIN(X(5,1))
      COSG=DCOS(X(5,1))
      SINC=DSIN(X(6,1))
      COSC=DCOS(X(6,1))
      R=X(1,1)
      RMAG2=R*R
      RMAG3=R*RMAG2
      V=X(4,1)
C     COMPUTE ATMOSPHERIC PARAMETERS
      ALTI=X(1,1)-RE
      CALL ATMOS(ALTI,TEMPR,PRES,RHO,VS,DVS,DRHO,DPRES)
      XMACH=V/VS
      RHO=DABS(RHO)
C     COMPUTE AEROD . COEF.
      Q=RHO*AREA*V*V/(2.D0*XMASS)
      CD=2.2D0*DSIN(TEMP(1))**3+8.0D-2
      CL=2.2D0*DSIN(TEMP(1))**2*DCOS(TEMP(1))+1.0D-2
      DCLM=0.0D0
      DCDM=0.0D0
```

```
      DCLA=2.2D0*(2.0D0*DSIN(TEMP(1))*DCOS(TEMP(1))**2-DSIN(TEMP(1))**3)
      DCDA=6.6D0*DSIN(TEMP(1))**2*DCOS(TEMP(1))
C     FIND AEROD.  PARTIALS
      DLDR=AREA*V*V*CL*DRHO/(2.D0*XMASS)
      DLDV=AREA*V*CL*RHO/XMASS+Q*DCLM/VS
      DDDR=Q*CD*DRHO/RHO
      DDDV=2.D0*Q*CD/V+Q*DCDM/VS
C     COMPUTE  MULTIPLIER DERIVATIVES
      P(1,1)=XS(2,1)*V*COSG*COSC/(RMAG2*COSP)+XS(3,1)*V*COSG*SINC/RMAG2
     2-XS(4,1)*(2.0D0*XMU*SING/RMAG3-DDDR)-XS(5,1)*(2.D0*XMU*COSG/(RMAG3
     3*V)-V*COSG/RMAG2+COSB*DLDR/V)-XS(6,1)*(V*COSG*COSC*SINP/(RMAG2*
     4COSP)-SINB*DLDR/(V*COSG))
      P(2,1)=0.0
      P(3,1)=-XS(2,1)*V*COSG*COSC*SINP/(R*COSP*COSP)+XS(6,1)*V*COSG*COSC
     2/(R*COSP*COSP)
      P(4,1)=-XS(1,1)*SING-(XS(2,1)*COSG*COSC)/(R*COSP)-XS(3,1)*COSG*SIN
     2C/R+XS(4,1)*DDDV-XS(5,1)*(XMU*COSG/(RMAG2*V*V)+COSG/R-Q*CL*COSB/
     3(V*V)+COSB*DLDV/V)-XS(6,1)*(-COSG*COSC*SINP/(R*COSP)+Q*CL*SINB/
     4(V*V*COSG)-SINB*DLDV/(V*COSG))
      P(5,1)=-XS(1,1)*V*COSG+XS(2,1)*V*SING*COSC/(R*COSP)+XS(3,1)*V*SING
     2*SINC/R+XS(4,1)*XMU*COSG/RMAG2-XS(5,1)*(XMU*SING/(RMAG2*V)-V*SING
     3/R)-XS(6,1)*(V*SING*COSC*SINP/(R*COSP)+Q*CL*SINB*SING/(V*COSG*COSG
     4))
      P(6,1)=XS(2,1)*V*COSG*SINC/(R*COSP)-XS(3,1)*V*COSG*COSC/R-XS(6,1)
     2*V*COSG*SINC*SINP/(R*COSP)
C     ADD HEATING EFFECTS
      RHOO=1.2250D00
      P(1,1)=P(1,1)-C(5)*ECOEF*(1.262D-4*V)**3.15*DRHO/(2.0D0*DSQRT(RHO
     2*RHOO))
      P(4,1)=P(4,1)-C(5)*ECOEF*3.15D0*(1.262D-4*V)**2.15*DSQRT(RHO/RHOO)
     2*(-XMU*SING/RMAG2-Q*CD)
      DO=2.0D0*ECOEF*((1.262D-4*V)**2.15)*((1.262D-4*V)*DRHO*V*SING/
     2(2.D0*DSQRT(RHO*RHOO))+3.15D0*DSQRT(RHO/RHOO)*(-XMU*SING/RMAG2
     3-Q*CD))*C(6)
      P(1,1)=P(1,1)   -DO*ECOEF*(1.262D-4*V)**2.15*(1.262D-4*V*V*(-DRHO/
     2(2.0D0*RHO)+DRHO*DRHO/RHO)/(2.0D0*DSQRT(RHOO*RHO))+3.15*((-XMU*SIN
     3G/RMAG2-Q*CD)/(2.0D0*DSQRT(RHOO*RHO))+DSQRT(RHO/RHOO)*(2.0D0*XMU*
     4SING/RMAG3-DDDR)))
      P(4,1)=P(4,1)   -DO*ECOEF*(1.262D-4*V)**1.15*(DRHO*SING*1.262D-4*V*
     2V*3.15D0/(2.0D0*DSQRT(RHOO*RHO))+3.15D0*DSQRT(RHO/RHOO)*((-XMU*
     3SING/RMAG2-Q*CD)*2.15D0+1.262D-4*V*(-XMU*SING/RMAG2-DDDV)))
      P(5,1)=P(5,1)       -DO*ECOEF*((1.262D-4*V)**2.15)*((1.262D-4*V)
     2*DRHO*V*COSG/(2.D0*DSQRT(RHO*RHOO))-3.15*DSQRT(RHO/RHOO)*XMU*COSG
     3/RMAG2)
      DO  300  I=1,6
 300  P(I,1)=-P(I,1)
      RETURN
 800  WRITE(6,600)
      STOP
 600  FORMAT(' ','EXCEEDED LOWER BOUND FOR STATE VAR. STORAGE')
      ENTRY GRADFN(XP,TM,IJK)
      GRAD(IJK,1)=-XP(4,1)*Q*DCDA+XP(5,1)*COSB*Q*DCLA/V-XP(6,1)*SINB*Q*
     2DCLA/(V*COSG)
      GRAD(IJK,2)=-XP(5,1)*Q*CL*SINB/ V       -XP(6,1)*Q*CL*COSB/(V*COSG)
      GRAD(IJK,3)=T
      IJK=IJK-1
      RETURN
      END
```

```
      SUBROUTINE OUTPUT(XJ,UTM)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION  XJ(8,1),UTM(2),SVARO(6)
      COMMON/CONS1/PI,RE,XMU,OMEGE,AREA,ECOEF,GNOT
      COMMON/STATO/SVARO,TO
      CRANG=(XJ(3,1)-SVARO(3))*RE
      DRANG=(XJ(2,1)-SVARO(2))*RE
      ALTI=XJ(1,1)-RE
      CALL ATMOS(ALTI,TEMPR,PRES,RHO,VS,DVS,DRHO,DPRES)
      XMACH=XJ(4,1)/VS
      HEAD=XJ(6,1)*180.D0/PI
      FLTA=XJ(5,1)*180.D0/PI
      ANGAT=UTM(1)*180.D0/PI
      BANK=UTM(2)*180.D0/PI
      WRITE(6,700) ALTI,XMACH,CRANG,DRANG,HEAD,FLTA,ANGAT,BANK
  700 FORMAT('0',6X,'FINAL ALTITUDE',19X,'MACH NUMBER',19X,'CROSS RANGE'
     2,20X,'DOWN RANGE'/' ',D24.16,10X,D24.16,10X,D24.16,10X,D24.16/
     3'0',7X,'HEADING ANGLE',15X,'FLIGHT PATH ANGLE',15X,'ANGLE OF ATTAC
     4K',20X,'BANK ANGLE'/' ',D24.16,10X,D24.16,10X,D24.16,10X,D24.16)
      RETURN
      END
```

```
      SUBROUTINE XLAMEN(XLAMF,XJ,L,M)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION XJ(8,1),XLAMF(6),SVARO(6),C(7),UTM(2),P(8,1),RPAR(1)
      COMMON/CONS1/PI,RE,XMU,OMEGE,AREA,ECOEF,GNOT
      COMMON/CONS3/CSTR,R,C,DTEM,XDTEM
      COMMON/STATO/SVARO,TO
      COMMON/STATE/ALT,XMASS,UTM
      COMMON/STATF/ALTF,XMACH,FLTANG,VF,GAMMF,TF
C     COMPUTE FINAL MULTIPLIERS
      XLAMF(6)=0.0D0
      XLAMF(5)=2.D0*C(4)*(XJ(5,1)-GAMMF)
      XLAMF(4)=2.D0*C(3)*(XJ(4,1)-VF)
      XLAMF(3)=C(1)*RE*XJ(3,1)*2.D0
      XLAMF(2)=C(2)*RE*XJ(2,1)*2.D0
      CALL DERIV1(TF,XJ,P,L,M,8,1,RPAR,0,1)
      XLAMF(1)=C(5)*P(7,1)+C(6)*P(8,1)
      DO 100  I=2,6
  100 XLAMF(1)=XLAMF(1)+XLAMF(I)*P(I,1)
      XLAMF(1)=-XLAMF(1)/P(1,1)
      RETURN
      END
```

```
      FUNCTION  COSTEN(XI)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION SVARO(6),C(7),XI(8,1),UTM(2)
      COMMON/CONS1/PI,RE,XMU,OMEGE,AREA,ECOEF,GNOT
      COMMON/CONS3/CSTR,B,C,DTEM,XDTEM
      COMMON/STATO/SVARO,TO
      COMMON/STATF/ALTF,XMACH,FLTANG,VF,GAMMF,TF
      DRANGE=RE*XI(2,1)*XI(2,1)
      CRANGE=RE*XI(3,1)*XI(3,1)
C     FORM  COST  VALUE
      COSTEN=C(1)*CRANGE+C(2)*DRANGE+C(3)*(XI(4,1)-VF)**2+C(4)*(XI(5,1)-
     2GAMMF)**2+C(5)*XI(7,1)+C(6)*XI(8,1)
      RETURN
      END
```

```
      SUBROUTINE ATMOS(ALTI,TEMP ,PRES ,RHO ,VS ,DVS ,DRHO ,DPRES )
      IMPLICIT REAL*8(A-H,O-Z)
      DATA A0,A1,A2,A3,A4,A5,A6,A7,A8,A9,A10,A11,A12,A13,A14,A15,A16,
     1     A17,A18,A19,A20,A21,A22,A23,B0,B1,B2,B3,B4,B5,B6,B7,B8,D1,
     2     D2,D3,D4/-1.0902039D-7,6356.77D0,1.787026D-4,21.680485D0,
     3     1.3949832D-5,284.01768D0,1.3327563D-4,29.89506D0,924.136D0,
     4     8.3168074D-4,.37777365D-1,.56467830D0,1.600231D-4,189.5201D0,
     5     9665.295D0,1.1637071D-3,.38184967D-1,3.6184094D0,5.562892D-5,
     6     420.11368D0,45675.466D0,1.284404D-4,.25387008D-1,5.3327146D0,
     7     2.8247930810+2,-5.2405729920D0,-1.266010595D-..,1.873293836D-2,
     8    -5.104746533D-4,6.050186406D-6,-3.550162735D-8,1.014102927
     9     D-10,1.124449619D-13,3483.676356D0,.2021698826lD-1,5.80334458
     A     91D0,.40187430086D+3/
C NOTE THAT FORMULAS ARE NOT ACCURATE FOR ALTITUDE OUTSIDE 0 TO 200 KM
C ALT MUST BE IN METERS
C TEMP IS IN DEGREES KELVIN
C PRES IS IN NEWTONS/M**2
C RHO IS IN KG/M**3
C VS IS IN METERS PER SECOND
C DRHO,DPRES, AND DVS ARE IN SAME UNITS AS RHO, PRES, AND DVS OVER MTS
      ALT=ALTI
      Z=ALT*1.0D-3
      IF(Z)1,2,2
1     Z=0.0
2     CONTINUE
      IF(Z-2.0D2)3,3,4
4     Z=200.D0
3     CONTINUE
      Z2=Z*Z
      E1=Z+A1
      E2=Z+A3
      E3=Z-A5
      E4=Z2-A7*Z+A8
      E5=Z2-A13*Z+A14
      E6=Z2-A19*Z+A20
      A=A0/E1+A2*DLOG(E2)-A4*DLOG(-E3)+A6*DLOG(E4)+A9*DATAN(A10*Z-A11)
     1-A12*DLOG(E5)+A15*DATAN(A16*Z-A17)-A18*DLOG(E6)+A21*DATAN(A22*Z
     2-A23)
      AAB=0.018031036
      AAC=-0.060803123
      AAD=-0.028429767
      DA=-A0/(E1*E1)+A2/E2-A4/E3+(2.*A6*Z+AAB)/E4-(2.*A12*Z+AAC)/E5
     1-(2.*A18*Z+AAD)/E6
      DA=DA*0.001
      TEMP=B0+Z*(B1+Z*(B2+Z*(B3+Z*(B4+Z*(B5+Z*(B6+Z*(B7-B8*Z)))))))
      DTEMP=B1+Z*(2.*B2+Z*(3.*B3+Z*(4.*B4+Z*(5.*B5+Z*(6.*B6+Z*(7.*B7-
     18.*B8*Z))))))
      DTEMP=DTEMP*0.001
      PRES=DEXP(-D1*A)
      RHO=D2*PRES/TEMP
      PRES=D3*PRES
      VS=DSQRT(D4*TEMP)
      DRHO=-RHO*(D1*DA+DTEMP/TEMP)
      DVS=0.5D0*D4*DTEMP/VS
      DPRES=-D1*PRES*DA
      IF(ALT-2.0D2/1.0D-3) 5,5,6
6     A=ALT-2.0D2/1.0D-3
      RHO=RHO+DRHO*A
      PRES=DPRES*A+PRES
      VS=VS+DVS*A
```

```
5     IF(ALT)7,8,8
7     RHO=RHO+DRHO*ALT
      PRES=PRES+DPRES*ALT
      VS=VS+DVS*ALT
8     CONTINUE
      RETURN
      END
```

```
C      SUBROUTINES FOR SPLINE FIT OF AEROD. COEF.
       IMPLICIT REAL*8(A-H,O-Z)
       DIMENSION     C(7),SVARO(6),STVRS(999,6),U(999,3),TEMP(2),UTM(2),
      2XJ(8,1),XLAMF(6),GRAD(999,3),SERCH(999,3),Y(15,15,2)
       COMMON/CONS1/PI,RE,XMU,OMEGE,AREA,ECOEF,GNOT
       COMMON/CONS2/DELTS,TCUT,EPST,EPSTF,EPSA,EPSIT,ERR,ITMAX,ITMX,
      2KOUNTM,IKEY
       COMMON/CONS3/CSTR,B,C,DTEM,XDTEM
       COMMON/CNTRL/GRAD,SERCH,U,ASTR,STF,    KJIS,IJKU,ISTAR
       COMMON/STATO/SVARO,TO
       COMMON/STATE/ALT,XMASS,UTM,STVRS
       COMMON/STATF/ALTF,XMACH,FLTANG,VF,GAMMF,TF
       COMMON/PRINT/IOUT,IOUT2,IPRNT1,IPRNT2
       COMMON/STORE/DELSV,DELSE,DELGE,DELT,KEN
       NAMELIST/ANAME/PI,RE,XMU,OMEGE,AREA,ECOEF,DELTS,IKEY,TCUT,EPST,
      2EPSTF,EPSA,EPSIT,ERR,ITMAX,ITMX,KOUNTM,CSTR,B,C,DTEM,SVARO,TO,ALTF
      3,XMACH,FLTANG,GAMMF,XMASS,IOUT,IOUT2,IPRNT1,IPRNT2,VF
C      READ IN DATA
       READ(5,ANAME)
       READ(7,700) IJKU
       READ(7,750)   ((U(I,J),J=1,3),I=1,IJKU)
       WRITE(6,ANAME)
       READ(5,502) N1,N2
       DO 110 I=1,2
       DO 110  L=1,N2
  110  READ(5,505) (Y(J,L,I),J=1,N1)
       DO 150 J=1,2
       DO 150 I=1,N1
  150  Y(I,1,J)=Y(I,1,J)*PI/180.D0
       CALL SETUP(N1,N2,Y)
C CALL CONJUGATE GRADIENT ROUTINE
       CALL WPRJCG(IER)
       GO TO (10,20,30,40,50,60,70,80,90,100),IER
    10 CONTINUE
    20 WRITE(6,520)
       GO TO 101
    30 WRITE(6,530)
       GO TO 101
    40 WRITE(6,540)
       GO TO 101
    50 WRITE(6,550)
       GO TO 101
    60 WRITE(6,560)
       GO TO 101
    70 WRITE(6,570)
       GO TO 101
    80 WRITE(6,580)
       GO TO 101
    90 WRITE(6,590)
       GO TO 101
   100 WRITE(6,600)
   101 CONTINUE
       WRITE(8,625) IJKU
       WRITE(8,650) ((U(K,L),L=1,3),K=1,IJKU)
       STOP
  502  FORMAT(2I4)
  505  FORMAT(8F10.0)
   520 FORMAT(1H0,5X,'ONE-D SEARCH FAILED TO FIND A MINIMUM')
   530 FORMAT(1H0,5X,'COST IS NOT DECREASING IN SEARCH DIRECTION')
   540 FORMAT(1H0,5X,'CONVERGENCE ON SMALL CONTROL CHANGE')
```

```
550 FORMAT(1H0,5X,'LITTLE COST CHANGE IN LAST TWO ITERATIONS')
560 FORMAT(1H0,5X,'FAILED TO CONVERGE IN ITMAX ITERATIONS')
570 FORMAT(1H0,5X,'INITIAL TRAJECTORY FAILED TO REACH CUT-OFF ALT')
580 FORMAT(1H0,5X,'TOO MANY INTEGRATIONS STEPS REQUIRED ')
590 FORMAT(1H0,5X,'BACKWARD INTEGRATED TRAJECTORY ERRORS')
600 FORMAT(1H0,5X,'CONVERGENCE ON ZERO GRADIENT NORM')
625   FORMAT(' ',I5)
650   FORMAT(' ',3D26.16)
700   FORMAT(I5)
750   FORMAT(3D26.16)
      END
```

```
      SUBROUTINE SETUP(N1,N2,Y)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION  Y(15,15,2),A(15,15,2),DX(15,15,2),DY(15,15,2),T(15),
     2S(15,2),R(15,2),K(15),B(15,2),C(15,2),DM(15,2),DC(15,2),DB(15,2),
     3COFE(2),COEP(2),CODE(2)
C     INTERPOLATION BY PIECEWISE CUBIC SPLINES
C     INPUT:  N1,N2  NUMBER OF DATA POINTS
C                    Y(N1,N2,2)= DATA TABLE
      NN=N1-1
      MM=N2-1
      DO 40  J=1,2
      DO 40  L=2,N2
      INDX=L-1
      A(2,INDX,J)=0.0
      A(N1,INDX,J)=0.0
      NN1=NN-1
      DO 11  I=2,NN1
      DX(I,INDX,J)=Y(I+1,1,J) - Y(I,1,J)
   11 DY(I,INDX,J)=(Y(I+1,L,J)-Y(I,L,J))/DX(I,INDX,J)
      T(2)=0.0
      DO 12  I=3,NN
      PIV=2.D0*(DX(I-1,INDX,J)+DX(I,INDX,J))-DX(I-1,INDX,J)*T(I-1)
      T(I)=DX(I,INDX,J)/PIV
   12 A(I,INDX,J)=(DY(I,INDX,J)-DY(I-1,INDX,J)-DX(I-1,INDX,J)*A(I-1,INDX
     2,J))/PIV
      DO 13  IB=2,NN1
      I=N1-IB
   13 A(I,INDX,J)=A(I,INDX,J)-T(I)*A(I+1,INDX,J)
   40 CONTINUE
      RETURN
      ENTRY SPLINE(XX,YY,CL,CD,DCLDM,DCLDA,DCDDM,DCDDA)
      INDX=1
      K(INDX)=2
   21 IF(XX .LT. Y(K(INDX),1,1)) GO TO 23
      IF(XX .GE. Y(K(INDX)+1,1,1))  GO TO 24
      DO 28  J=1,2
      DO 28  L=2,N2
      M=L-1
      W=(XX-Y(K(INDX),1,J))/DX(K(INDX),M,J)
      V=1.D0-W
      S(M,J)=W*Y(K(INDX)+1,L,J)+V*Y(K(INDX),L,J)+DX(K(INDX),M,J)*DX(K(IN
     2DX),M,J)*((W*W*W-W)*A(K(INDX)+1,M,J)+(V*V*V-V)*A(K(INDX),M,J))
   28 R(M,J)=(Y(K(INDX)+1,L,J)-Y(K(INDX),L,J))/DX(K(INDX),M,J)+DX(K(INDX
     2),M,J)*((3.D0*W*W-1.D0)*A(K(INDX)+1,M,J)-(3.D0*V*V-1.D0)*A(K(INDX)
     2,M,J))
      GO TO 100
   23 K(INDX)=K(INDX)-1
      IF(K(INDX) .EQ. 1)  XX=Y(2,1,1)
      GO TO 21
   24 K(INDX)=K(INDX)+1
      IF(K(INDX) .EQ. N1) XX=Y(N1,1,1)
      GO TO 21
  100 CONTINUE
      IF(YY .GE. Y(1,N2,1)) GO TO 250
      DO 400  J=1,2
      B(1,J)=0.0
      B(MM,J)=0.0
      C(1,J)=0.0
      C(MM,J)=0.0
      K(INDX)=1
```

```
      MM1=MM-1
      DO 120  I=1,MM1
      DM(I,J)=Y(1,I+2,J)-Y(1,I+1,J)
      DB(I,J)=(S(I+1,J)-S(I,J))/DM(I,J)
120   DC(I,J)=(R(I+1,J)-R(I,J))/DM(I,J)
      T(1)=0.0
      DO 140  I=2,MM1
      PIV=2.DO*(DM(I-1,J)+DM(I,J))-DM(I-1,J)*T(I-1)
      T(I)=DM(I,J)/PIV
      B(I,J)=(DB(I,J)-DB(I-1,J)-DM(I-1,J)*B(I-1,J))/PIV
140   C(I,J)=(DC(I,J)-DC(I-1,J)-DM(I-1,J)*C(I-1,J))/PIV
      DO 130  IM=2,MM1
      I=M2-IM
      B(I,J)=B(I,J)-T(I)*B(I-1,J)
130   C(I,J)=C(I,J)-T(I)*C(I-1,J)
400   CONTINUE
 210  IF(YY .LT. Y(1,K(INDX)+1,1))   GO TO 230
      IF(YY .GE. Y(1,K(INDX)+2,1)) GO TO 240
      DO 280  J=1,2
      U=(YY-Y(1,K(INDX)+1,J))/DM(K(INDX),J)
      Z=1.DO-U
      COFF(J)=U*S(K(INDX)+1,J)+Z*S(K(INDX),J)+DM(K(INDX),J)*DM(K(INDX),J
     2)*((U*U*U-U)*B(K(INDX)+1,J)+(Z*Z*Z-Z)*B(K(INDX),J))
      COFP(J)=U*R(K(INDX)+1,J)+Z*R(K(INDX),J)+DM(K(INDX),J)*DM(K(INDX),J
     2)*((U*U*U-U)*C(K(INDX)+1,J)+(Z*Z*Z-Z)*C(K(INDX),J))
280   CODF(J)=(S(K(INDX)+1,J)-S(K(INDX),J))/DM(K(INDX),J)+DM(K(INDX),J)
     2*((3.DO*U*U-1.DO)*B(K(INDX)+1,J)-(3.DO*V*V-1.DO)*B(K(INDX),J))
      GO TO 300
250   DO  275  J=1,2
      COFF(J)=S(MM,J)
      COFP(J)=R(MM,J)
275   CODF(J)=0.0
300   CL=COFF(1)
      CD=COFF(2)
      DCLDM=CODF(1)
      DCDDM=CODF(2)
      DCLDA=COFP(1)
      DCDDA=COFP(2)
      RETURN
230   K(INDX)=K(INDX)-1
      IF(K(INDX) .EQ. 0) YY=Y(1,2,1)
      GO TO 210
240   K(INDX)=K(INDX)+1
      GO TO 210
      END
```

```
      SUBROUTINE  DERIV1(T,X,P,L,M,N,NE,RPAR,IPAR,ND)
      IMPLICIT  REAL*8(A-H,O-Z)
      DIMENSION  X(N,NE),P(N,NE),RPAR(ND),IPAR(ND),TEMP(2),
     1TEM(2),TF(3,2),U(999,3),SERCH(999,3),GRAD(999,3)
      COMMON/CONS1/PI,RE,XMU,OMEGE,AREA,ECOEF,GNOT
      COMMON/CNTRL/GRAD,SERCH,U,ASTR,STF,    KJIS,IJKU,ISTAR
      COMMON/STATE/ALT,XMASS,TEMP
      COMMON/STATF/ALTF,XMACH,FLTANG,VF,GAMMF
C     T = TIME
C     X = STATE AND INTEGRATED COST
C     P = DERIVATIVES OF STATE AND INTEGRATED COST AT T
C     COMPUTE TRIG FUNCTIONS OF PHI,GAMMA,CHI
      TF(1,1)=DSIN(X(3,1))
      TF(1,2)=DCOS(X(3,1))
      TF(2,1)=DSIN(X(5,1))
      TF(2,2)=DCOS(X(5,1))
      TF(3,1)=DSIN(X(6,1))
      TF(3,2)=DCOS(X(6,1))
C     COMPUTE RELATIVE VELOCITY
      RMAG1=X(1,1)
      V=X(4,1)
      VRMAG=V
C     COMPUTE ATMOSPHERIC PARAMETERS
      ALTI=X(1,1)-RE
      CALL ATMOS(ALTI,TEMPR,PRES,RHO,VS,DVS,DRHO,DPRES)
      XMACH=VRMAG/VS
      RHO=DABS(RHO)
C     COMPUTE DERIVATIVES NO ATMOS.
      P(1,1)=V*TF(2,1)
      P(2,1)=(V*TF(2,2)*TF(3,2))/(RMAG1*TF(1,2))
      P(3,1)=(V*TF(2,2)*TF(3,1))/RMAG1
      P(4,1)=(-XMU*TF(2,1))/(RMAG1*RMAG1)
      P(5,1)=(-XMU*TF(2,2))/(RMAG1*RMAG1*V)+(V*TF(2,2))/RMAG1
      P(6,1)=(-V*TF(2,2)*TF(3,2)*TF(1,1))/(RMAG1*TF(1,2))
      IF(IPAR(1) .EQ. 1)  GO TO 100
C     FIND CONTROL
   20 IF(L .GE. IJKU)  GO TO  60
   30 IF(U(L+1,3) .LT. T)  GO TO  50
      IF(U(L,3) .GT. T)  GO TO 55
      TEMP(1)=U(L,1)+((U(L+1,1)-U(L,1))/(U(L+1,3)-U(L,3)))*(T-U(L,3))
      TEMP(2)=U(L,2)+((U(L+1,2)-U(L,2))/(U(L+1,3)-U(L,3)))*(T-U(L,3))
      L=L+1
      GO TO 40
   50 L=L+1
      GO TO 20
   55 L=L-1
      GO TO 20
   60 TEMP(1)=U(L,1)+((U(L,1)-U(L-1,1))/(U(L,3)-U(L-1,3))*(T-U(L,3)))
      TEMP(2)=U(L,2)+((U(L,2)-U(L-1,2))/(U(L,3)-U(L-1,3))*(T-U(L,3)))
   40 IF(ISTAR .EQ. 0)  GO TO 100
C     FIND SEARCH DIRECTION
   21 IF(M .GE. KJIS) GO TO 61
      IF(SERCH(M+1,3) .LT. T) GO TO 51
      IF(SERCH(M,3) .GT. T) GO TO 56
      DO 31 I=1,2
   31 TEM(I)=SERCH(M,I)+((SERCH(M+1,I)-SERCH(M,I))/(SERCH(M+1,3)
     2-SERCH(M,3)))*(T-SERCH(M,3))
      M=M+1
      GO TO 80
   51 M=M+1
```

```
      GO TO 21
   56 M=M-1
      GO TO 21
   61 DO 62 I=1,2
   62 TEM(I)=SERCH(M,I)+((SERCH(M,I)-SERCH(M-1,I))/(SERCH(M,3)-SERCH
     2(M-1,3)))*(T-SERCH(M,3))
   80 CONTINUE
C     FORM CONTROL
      TEMP(1)=TEMP(1)-ASTR*TEM(1)
      TEMP(2)=TEMP(2)-ASTR*TEM(2)
  100 CONTINUE
C     COMPUTE AEROD . COEF.
      CALL SPLINE(TEMP(1),XMACH,CL,CD,DCLM,DCLA,DCDM,DCDA)
      Q=RHO*AREA*V*V/(XMASS*2.D0)
C     ADD CONTROL ACCELERATIONS
      P(4,1)=P(4,1)-Q*CD
      P(5,1)=P(5,1)+Q*CL*DCOS(TEMP(2))/V
      P(6,1)=P(6,1)-Q*CL*DSIN(TEMP(2))/(V*TF(2,2))
C     COMPUTE  HEATING AND HEAT RATE DERIVATIVES
      RHOO=1.22501D0
      P(7,1)=ECOEF*DSQRT(RHO/RHOO)*(1.262D-4*VRMAG)**3.15
      DVRMAG=P(4,1)
      P(8,1)=DSQRT(RHO/RHOO)*3.15D0*1.262D-4*(1.262D-4*VRMAG)**2.15*
     1DVRMAG
      P(8,1)=P(8,1)+((1.262D-4*VRMAG)**3.15*DRHO)*P(1,1)/(2.D0*DSQRT(RHO
     1*RHOO))
      P(8,1)=ECOEF*P(8,1)*ECOFF*P(8,1)
      RETURN
      END
```

```
      SUBROUTINE DERIV2(TS,XS,P,L,M,NI,RPAR,IPAR)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION XS(6,1),STVRS(999,6),GRAD(999,3),SERCH(999,3),U(999,3),
     2TEMP(2),XP(6,1),X(6,1),P(6,1),C(7)
      COMMON/CONS1/PI,RE,XMU,OMEGE,AREA,ECOEF,GNOT
      COMMON/CONS3/CSTR,B,C,DTEM,XDTEM
      COMMON/CNTRL/GRAD,SERCH,U,ASTR,STF,    KJIS,IJKU,ISTAR
      COMMON/STATE/ALT,XMASS,TEMP,STVRS
      COMMON/STATF/ALTF,XMACH,FLTANG,VF,GAMMF,TF
      T=TF-TS
C     RETREIVE STATE VARIABLES FROM STORAGE
      IF(NI .EQ. 1)  GO TO 10
      DO 200 I=1,6
  200 X(I,1)=STVRS(M,I)
      GO TO 40
C     INTERPOLATE FROM STATE VAR.
   10 IF (M .EQ. 1)  GO TO 800
      DO 30  I=1,6
   30 X(I,1)=(STVRS(M,I)-STVRS(M-1,I))/2.D0+STVRS(M-1,I)
   40 CONTINUE
C     FIND CONTROL
   20 IF(L .GE. IJKU)  GO TO  60
      IF(U(L+1,3) .LT. T)  GO TO  50
      IF(U(L,3) .GT. T)  GO TO 55
      TEMP(1)=U(L,1)+((U(L+1,1)-U(L,1))/(U(L+1,3)-U(L,3)))*(T-U(L,3))
      TEMP(2)=U(L,2)+((U(L+1,2)-U(L,2))/(U(L+1,3)-U(L,3)))*(T-U(L,3))
      L=L+1
      GO TO 400
   50 L=L+1
      GO TO 20
   55 L=L-1
      GO TO 20
   60 IF(U(L-1,3) .GT. T)  GO TO 55
      TEMP(1)=U(L,1)+((U(L,1)-U(L-1,1))/(U(L,3)-U(L-1,3))*(T-U(L,3)))
      TEMP(2)=U(L,2)+((U(L,2)-U(L-1,2))/(U(L,3)-U(L-1,3))*(T-U(L,3)))
  400 CONTINUE
C     COMPUTE TRIG QUANTITIES
      COSB=DCOS(TEMP(2))
      SINB=DSIN(TEMP(2))
      SINP=DSIN(X(3,1))
      COSP=DCOS(X(5,1))
      SING=DSIN(X(5,1))
      COSG=DCOS(X(5,1))
      SINC=DSIN(X(6,1))
      COSC=DCOS(X(6,1))
      R=X(1,1)
      RMAG2=R*R
      RMAG3=R*RMAG2
      V=X(4,1)
C     COMPUTE ATMOSPHERIC PARAMETERS
      ALTI=X(1,1)-RE
      CALL ATMOS(ALTI,TEMPR,PRES,RHO,VS,DVS,DRHO,DPRES)
      XMACH=V/VS
      RHO=DABS(RHO)
C     COMPUTE AEROD . COEF.
      Q=RHO*AREA*V*V/(2.D0*XMASS)
      CALL SPLINE(TEMP(1),XMACH,CL,CD,DCLM,DCLA,DCDM,DCDA)
C     FIND AEROD.  PARTIALS
      DLDR=AREA*V*V*CL*DRHO/(2.D0*XMASS)
      DLDV=AREA*V*CL*RHO/XMASS+Q*DCLM/VS
```

```
      DDDR=Q*CD*DRHO/RHO
      DDDV=2.D0*Q*CD/V+Q*DCDM/VS
C     COMPUTE  MULTIPLIER DERIVATIVES
      P(1,1)=XS(2,1)*V*COSG*COSC/(RMAG2*COSP)+XS(3,1)*V*COSG*SINC/RMAG2
     2-XS(4,1)*(2.0D0*XMU*SING/RMAG3-DDDR)-XS(5,1)*(2.D0*XMU*COSG/(RMAG3
     3*V)-V*COSG/RMAG2+COSB*DLDR/V)-XS(6,1)*(V*COSG*COSC*SINP/(RMAG2*
     4COSP)-SINB*DLDR/(V*COSG))
      P(2,1)=0.0
      P(3,1)=-XS(2,1)*V*COSG*COSC*SINP/(R*COSP*COSP)+XS(6,1)*V*COSG*COSC
     2/(R*COSP*COSP)
      P(4,1)=-XS(1,1)*SING-(XS(2,1)*COSG*COSC)/(R*COSP)-XS(3,1)*COSG*SIN
     2C/R+XS(4,1)*DDDV-XS(5,1)*(XMU*COSG/(RMAG2*V*V)-COSG/R-Q*CL*COSB/
     3(V*V)+COSB*DLDV/V)-XS(6,1)*(-COSG*COSC*SINP/(R*COSP)+Q*CL*SINB/
     4(V*V*COSG)-SINB*DLDV/(V*COSG))
      P(5,1)=-XS(1,1)*V*COSG+XS(2,1)*V*SING*COSC/(R*COSP)+XS(3,1)*V*SING
     2*SINC/R+XS(4,1)*XMU*COSG/RMAG2-XS(5,1)*(XMU*SING/(RMAG2*V)-V*SING
     3/R)-XS(6,1)*(V*SING*COSC*SINP/(R*COSP)+Q*CL*SINB*SING/(V*COSG*COSG
     4))
      P(6,1)=XS(2,1)*V*COSG*SINC/(R*COSP)-XS(3,1)*V*COSG*COSC/R-XS(6,1)
     2*V*COSG*SINC*SINP/(R*COSP)
C     ADD HEATING EFFECTS
      RHOO=1.2250D0
      P(1,1)=P(1,1)-C(5)*ECOEFF*(1.262D-4*V)**3.15*DRHO/(2.0D0*DSQRT(RHO
     2*RHOO))
      P(4,1)=P(4,1)-C(5)*ECOEFF*3.15D0*(1.262D-4*V)**2.15*DSQRT(RHO/RHOO)
     2*(-XMU*SING/RMAG2-Q*CD)
      DQ=2.0D0*ECOEFF*((1.262D-4*V)**2.15)*((1.262D-4*V)*DRHO*V*SINC/
     2(2.D0*DSQRT(RHO*RHOO))+3.15D0*DSQRT(RHO/RHOO)*(-XMU*SING/RMAG2
     3-Q*CD))*C(6)
      P(1,1)=P(1,1)   -DQ*ECOEFF*(1.262D-4*V)**2.15*(1.262D-4*V*V*(-DRHO/
     2(2.0D0*RHO)+DRHO*DRHO/RHO)/(2.0D0*DSQRT(RHOO*RHO))+3.15*((-XMU*SIN
     3G/RMAG2-Q*CD)/(2.0D0*DSQRT(RHOO*RHO))+DSQRT(RHO/RHOO)*(2.0D0*XMU*
     4SING/RMAG3-DDDR)))
      P(4,1)=P(4,1)   -DQ*ECOEFF*(1.262D-4*V)**1.15*(DRHO*SING*1.262D-4*V*
     2V*3.15D0/(2.0D0*DSQRT(RHOO*RHO))+3.15D0*DSQRT(RHO/RHOO)*((-XMU*
     3SING/RMAG2-Q*CD)*2.15D0+1.262D-4*V*(-XMU*SING/RMAG2-DDDV)))
      P(5,1)=P(5,1)       -DQ*ECOEFF*((1.262D-4*V)**2.15)*((1.262D-4*V)
     2*DRHO*V*COSG/(2.D0*DSQRT(RHO*RHOO))-3.15*DSQRT(RHO/RHOO)*XMU*COSG
     3/RMAG2)
      DO  300  I=1,6
  300 P(I,1)=-P(I,1)
      RETURN
  800 WRITE(6,600)
      STOP
  600 FORMAT(' ','EXCEEDED LOWER BOUND FOR STATE VAR. STORAGE')
      ENTRY GRADFN(XP,TM,IJK)
      GRAD(IJK,1)=-XP(4,1)*Q*DCDA+XP(5,1)*COSB*Q*DCLA/V-XP(6,1)*SINB*Q*
     2DCLA/(V*COSG)
      GRAD(IJK,2)=-XP(5,1)*Q*CL*SINB/(V*COSG)-XP(6,1)*Q*CL*COSB/(V*COSG)
      GRAD(IJK,3)=T
      IJK=IJK-1
      RETURN
      END
```

1437 LINES PRINTED