Institut für Dynamik der Flugsysteme	Bl.: IB.Nr.: 515-83/2

A Numerical Solution of State/Control-Constraint Optimal Control Problems with Piecewise Continuous Derivatives Using RKF45T

Freigabe:	Die Bearbeiter:	Unterschriften:
	Dr. M.K. Horn	i.A. Klaus Uch Well
	Dr. K.H. Well Der Abteilungsleiter	Klaus Uch Well
	Der stellv. Institutsdirektor:	
	Dr.Ing.J.Ackermann Der Institutsdirektor:	J. Ackermann

Dieser Bericht enthält

68 Blatt davon

45 ~~Bilder~~ Bl.Rechnerprotokoll

4 Diagramme

Ort Oberpfaffenhofen	Datum: Dezember 1982	Bearbeiter: Dr.Horn	Zeichen rm

Constraint optimal control problems, numerical analysis, Runge-Kutta algorithm, ordinary differential equations, discontinuous systems

A Numerical Solution of State/Control-Constraint Optimal Control Problems with Piecewise Continuous Derivatives Using RKF45T

Summary

The state/control-constraint optimal control problem is analyzed for a system having discontinuities in the state equations. As an example, an inequality constraint is treated by construction an additional, discontinuous differential equation. This differential equation is defined as a multi-branched function. The integration package permits the switching between branches as the points of discontinuity are isolated. Any function which may be expressed in terms of the independent variable, dependent variables, and/or hte first derivatives may be used to define the points of discontinuity, which will be located automatically within integration precision. The procedure required to define the branching and the points of discontinuity is presented, as well as an example showing the effect of the method on the convergence rate and the accuracy.

Beschränkte optimale Steuerungsprobleme, numerische Mathematik, Runge-Kutta Verfahren, gewöhnliche Differentialgleichungen mit Unstetigkeiten

Numerische Lösung von zustands-/steuerungs-beschränkten optimalen Steuerungsproblemen mit stückweise stetigen rechten Seiten mittels RKF45T

Übersicht

Das beschränkte optimale Steuerungsproblem mit unstetigen Differentialgleichungen wird analysiert. Als Beispiel wird eine Ungleichungsbeschränkung durch Hinzufügen einer zusätzlichen unstetigen Differentialgleichung behandelt. Diese Differentialgleichung besteht aus mehreren Zweigen. Das Integrationsverfahren erlaubt das Umschalten nach automatischem Anhalten an den Unstetigkeitsstellen. Jede Funktion, die von der unabhängigen Variablen, den abhängigen Variablen, oder deren Ableitungen abhängt, kann zur Definition der Unstetigkeitsstellen benutzt werden. Diese Stellen werden mit Integrationsgenauigkeit gefunden. Die Prozedur, die die Unstetigkeitsstellen isoliert, wird beschrieben; ebenso werden Ergebnisse bezüglich Genauigkeit und Konvergenzgeschwindigkeit diskutiert.

CONTENTS

PREFACE

This report is one of a series of four volumes which are designed to treat state/control-constraint optimal control problems involving piecewise continuous system equations including the extensive use of equation expressions written in terms of linearly interpolated tabular data. The titles of the volumes are listed below:

Volume 1 A FORTRAN Program for Solving State/Control-Constraint Optimal Control Problems with System Equations Having Expressions Involving Tabular Data

in which extensive use of linearly interpolated tabular data is made, treating the system truly as a piecewise continuous problem by halting the integration for equation updates as each table grid point is isolated. (See reference [1].)

Volume 2 A Numerical Solution of State/Control-Constraint Optimal Control Problems with Piecewise Continuous Derivatives Using RKF45T

in which constraint violation boundary crossings are isolated, and in which discontinuities in the derivatives occur. (Current report)

Volume 3 RKF45T--a Runge-Kutta 4/5 Software Package with User-Supplied Stops Involving the Dependent Variables and First Derivatives

in which the user may actually halt the integration at any point which may be described as a function of the independent variable, the dependent variables, and the first derivatives. (See reference [2].)

Volume 4 Subroutines for Handling Tabular Data Used in System Equations

in which a table structure is defined consistent with the example in Volume 1, and in which practical routines are provided for adjusting and analyzing tabular functions. (See reference [3].)

1. INTRODUCTION

The solution of the state/control-constraint optimal control problem (OCP) often involves the solution of piecewise continuous ordinary differential equations (ODEs). Two difficulties arise in handling such problems. First, the points of discontinuity are often defined by analytic expressions involving the dependent variables and the first derivatives. Locating the corresponding value of the independent variable generally requires the use of an iterative procedure. Secondly, the solution of the ODE system is generally embedded in a more complicated software package for generating cost functions or gradients, giving the user almost no control over the integration problem during execution. The convergence properties of an optimization package, however, may depend heavily upon the precise location of the points of discontinuity.

This report is one of a set of four volumes designed to treat the state/control-constraint optimal control problem in which discontinuities exist in the ODE system. The procedure involves stopping the integration at the points of discontinuity and reevaluating the ODEs so that they describe the problem correctly. This particular application, which concerns a system of equations having relatively few discontinuities, contains constraint boundaries whose points of violation are isolated in order to control the extent of the boundary violation.

The core of the software package system is a fifth order Runge-Kutta integrator, RKF45T, adapted to stop the integration whenever any component of a user-supplied vector of stopping conditions vanishes, giving the user the opportunity to redefine the ODE system before the integration continues. The points of discontinuity, along with the constraint boundary conditions, are supplied as the stopping conditions for the integration, with the ODEs defined as multi-branched functions. At each user-defined stopping point, the branch of the ODEs may be redefined. RKF45T is a modification of the RKF45 program (RKF45 being written H.A. Watts and L.F. Shampine [6]) and becomes the RKF45 package if no auxiliary functions are treated.

The TOMP software package due to D. Kraft [4] is designed to be used for trajectory optimization in connection with nonlinear programming algorithms. The subroutine TOMP solves the initial value problem of a given dynamic system with control parameters expressed as cubic or exponential splines (either continuous or discontinuous). In order to stop the integration at vanishing points of an auxiliary vector function, PHI, TOMP (Trajectory Optimization by Mathematical Programming) has been modified into TROMP (TOMP with zero-TRapping capabilities). The modifications to TOMP are minimal, basically involving the labeling of operation flags. (A user, not wishing to supply additional stopping conditions, may reference the TROMP package with RKF45T as the ODE solver in the "non-trapping" mode and obtain the same solution that would have been generated by using RKF45.)

The main additional feature of the TROMP is the ability to reference the RKF45T package to solve the ODE problem, giving the user the opportunity to monitor the integration process after each step and to interact with the ODE system as each point of discontinuity is isolated.

TROMP and RKF45T work together as a unit to provide the user with the cost function and/or the gradients. Control functions are described as splines (cubic or B-splines), with the defining constants to be adjusted by the opti-

mization package to locate the optimal conditions subject to constraints. The choice of optimization procedure is not restricted by the TROMP/RKF45T system. The SLLSQP package (Sequential Linear Least Squares Program) [5] has been chosen for the application due to its rapid convergence rate, although other packages may be used.

Before one can solve a trajectory optimization problem, one must first be able to solve the differential equations involved. Thus, the first portion of this report involves a detailed description of the treatment of a particular ODE system, showing how the problem of discontinuities is handled and how the updates are interpreted once the discontinuities are isolated. The TOMP software package is described in [4]. This report assumes that the reader is familiar with TOMP, and gives only the additional parameters needed to convert TOMP into TROMP. TROMP is used to solve a particular constrained trajectory optimization problem. Emphasis is given to the handling of constraint equations by discontinuous functions rather than by continuous functions which have contributions even when the constraints are not violated. A listing of TROMP is given in the Appendix A (with modifications from TOMP clearly marked). The RKF45T package is documented in Volume 3 of this series of reports [2] giving numerous applications. The user may be referred to that report for further details in describing the integration stopping conditions. A more extensive application of the RKF45T package for the state/control-constraint OCP involving linearly interpolated tabular data is given in Volume 1 [1] of this series. While [1] may be of interest to the reader, it is not essential for understanding of the current application.

2. TREATMENT OF CONSTRAINT EQUATIONS

An example of particular importance in optimization applications is the handling of constraint equations. One may treat a constraint, $g \leq g^*$ by defining an additional solution component of y, whose derivative is zero whenever the constraint is *not* violated but which is positive whenever the constraint *is* violated, e.g.,

$$\frac{dy_{n+1}}{dt} = \begin{cases} 0 & , \; g \leq g^* \\ k\,(g - g^*)^2 & , \; g > g^* \end{cases} \tag{2.1}$$

where g and g^* may both depend upon t, y, and y'. If the values of t can be located for which $g=g^*$, y_{n+1} can be evaluated quite accurately and will give a good measure of the amount of the constraint violation. The user includes the boundary condition that $y_{n+1}= 0$ at the initial conditions and that $|y_{n+1}| < EPS$ at the final time as a criterion for adjusting the control parameters. The amount of computing time required to isolate the discontinuity points, $g=g^*$, is the price one pays for using such a function.

Another means of expressing this additional ODE is by using a continuous function, such as hyperbolic tangent, e.g.,

$$(2.2) \qquad \frac{dy}{dt}_{n+1} = (g - g^*)^2 \, (\tanh(k(g - g^*)) + 1) \, ,$$

where g and g* may both depend upon t, y, and y'. The right hand side is continuous, and so requires no integration stops to activate different branches of the function. The function, however, also has a contribution when the constraint is not violated. Comparisons are made between the use of the two types of equations.

3. HANDLING DISCONTINUOUS DIFFERENTIAL EQUATION SYSTEMS

If the discontinuities of an ODE system can be expressed as a function of t, y, and y', the user, with the help of the RKF45T package, can halt the integration at these points and restart the integration using the new branch of the function. The user must supply a subroutine SUBPHI which defines the discontinuity points as the zeros of a particular function, PHI. (A vector of stopping conditions of any dimension can be supplied.) (See §5.)

To handle the isolation of the points of discontinuity efficiently, the right hand sides (RHS) of the ODE system must be "smooth". Thus, if a part of an ODE component has multiple algebraic expressions, (e.g., Eqn. 2.1), the integrator must "see" only one expression until the discontinuity has been isolated. Then an update will be made, and the integrator will again "see" only one expression, this time a new one. Thus, the integrator actually steps beyond the discontinuity, the user-supplied PHI "detector" sees the error, and an iterative procedure isolates the point of discontinuity. The user then restarts the integration from that point. Updating the ODE system occurs in the user-supplied subroutine SUBPHI, so that flags may be held in common with the RHS evaluator, assuring that the correct "branch" of the functions is being used. (See §5.)

As an example, consider Eqns. (4.9.1)-(4.9.3) which define the parameter a_z. Discontinuities in slope occur at V=150 and V=536.36. The branches of this ODE expression are branch 1 for $V \leq 150$, branch 2 for $150 < V \leq 536.36$, and branch 3 for $V > 536.36$. The initial conditions impose branch 1, Eqn.(4.9.1). The integration is continued until the point V=150 is isolated, at which time the branching is changed to branch 2, Eqn.(4.9.2). During the "isolation" of V=150, points for which V>150 are studied with branch 1 still assigned. Only when the value of T is isolated for which V=150, will the branch assignment be changed. These changes take place within the RKF45T system (in the user-supplied subroutine, SUBPHI, described in §6). The integration continues with the new branch assignment until V crosses the V=150 boundary again or the V=536.36 boundary. When the new discontinuity is located, the branch assignment is again changed. The subroutine evaluating the ODE *makes no judgements* concerning the current value of V, accepting the assigned branching value supplied through common statements.

4. TRAJECTORY PROBLEM

The equations of motion for the flight of a missile are given by

$$V' = a_x \cos\alpha + a_z \sin\alpha \tag{4.1}$$

$$\gamma' = (a_x \sin\alpha - a_z \cos\alpha) / V \tag{4.2}$$

$$x' = V \cos\gamma \tag{4.3}$$

$$z' = - V \sin\gamma \quad . \tag{4.4}$$

The equations

$$I'_\alpha = \begin{cases} 0 , & |\alpha| \le \alpha_m \quad (4.5.1) \\ (\alpha_m - |\alpha|)^2 , & |\alpha| > \alpha_m \quad (4.5.2) \end{cases}$$

may be used for handling the constraint, $|\alpha| \le \alpha$, with

$$\gamma'' = (a'_x \sin\alpha - a'_z \cos\alpha + (\alpha' - \gamma') V') / V \tag{4.6}$$

for specifying boundary constraints on γ', but neither of these equations is needed to solve (4.1)-(4.4). The control parameter, a_z, is expressed as a cubic spline, with coefficients changed by the optimization package until the cost function is minimal subject to the given constraints. The remaining parameters in (4.1) through (4.6) are:

(4.7) angle of attack: $\alpha = + a_z / (C_\alpha V^2)$

where C_α = -2.33D-03,

(4.8.1) forward acceleration: $a_x = (F(t) - W(\alpha)) / m(t)$

with thrust:

$$F(t) = \begin{cases} F_0 + FP * t , & t \le t_b \quad (4.8.2.1) \\ 0 , & t > t_b \quad (4.8.2.2) \end{cases}$$

where t_b=2.7 s, F_0=29.87D+03 N, FP=7.9D+03 N/s,

and with mass:

$$m(t) = \begin{cases} m_0 + \chi\,(F_0 + 0.5*FP*t)\,t\,, & t \le t_b \quad (4.8.3.1) \\ m(t_b)\,, & t > t_b \quad (4.8.3.2) \end{cases}$$

where m_0=137.2kg, χ =-0.4133D-03kg/N/s

and

(4.8.4) Drag: $W(\alpha) = 0.5\ C_W\ \rho\ V^2\ S$

where $C_W = C_{W_0} + C_{W_{\alpha\alpha}}\ \alpha^2$

with C_{W_0}=0.3, $C_{W_{\alpha\alpha}}$=0.7/rad^2, ρ =1.22575kg/m^3, and S=0.0314m^2

In addition, a_{z_m}, the constraint on the control function a_z, is written:

$$a_{z_m} = \begin{cases} 0.002\ V^2\,, & \text{if } V \le V_1 \quad (4.9.1) \\ 45.0 + 0.66\ (V - 150)\,, & \text{if } V_1 < V \le V_2 \quad (4.9.2) \\ 300\,, & \text{if } V > V_2 \quad (4.9.3) \end{cases}$$

with V_1 = 150 m/s, V_2 = 536.3636363636364 m/s .

Together, the velocity and a_{z_m} define the constraint angle of attack:

$$\alpha_m = -\,a_{z_m} / (C_\alpha\ V^2) \quad . \quad (4.10)$$

The initial conditions for the example in §9 are:

V_0 = 58.6 m/s , γ_0 = 90°

x_0 = 8.7 * cos(γ) m , I_{α_0} = 0

z_0 = -8.7 * sin(γ) m , γ'_0 = 0.

Final boundary conditions are given in §9.2.

5. PHI VECTOR AND ITS RELATION TO THE ODE BRANCHES

The user-supplied subroutine SUBPHI, evaluates the vector PHI, whose zero points are being sought. The PHI vector is perhaps best described by example. Considering the problem in §4, one finds the following possible points of discontinuity: (1) $|\alpha| = \alpha_m$, (2) TIME = 2.7, (3) V = 150, and (4) V = 536.36 . Suitable PHI expressions for locating these points would be:

(5.1) PHI(1) = (ALPHAM - ALPHA)*(ALPHA - (-ALPHAM))

(5.2) PHI(2) = T * TFINAL - 2.70D0

(5.3) PHI(3) = (VUPPER - V)*(V - VLOWER)/VSCALE

where VLOWER and VUPPER are bounds on the velocity, and where VSCALE is a scaling factor used to give a relative error convergence test. PHI(1) and PHI(2) remain moderate, and so no scaling of these parameters is imposed. In (5.2) the T value is a normalized time value. (To compare the actual time value with the burn-out time, 2.70, T must be scaled by the final time. Parameters needed for generating the derivatives of the PHI components can be held in common blocks with the derivative generating subroutines so that a Newton-Rhapson estimate for the zero points will be used by the RKF45T package.

The choice of (5.2) is clear. The form of PHI(1) and PHI(3) has been selected because the branch switching in the ODEs may be identified with pairs of bounds, e.g.,

ALPHA = + ALPHAM or ALPHA = -ALPHAM .

The expression

PHI = (XHIGH - X) * (X - XLOW)

is always positive when XLOW < X < XHIGH and always negative when X < XLOW or X > XHIGH. Thus, such a structure is useful in identifying the boundaries sought. The PHI function is parabolic in X, and therefore the sign of the derivative with respect to X (not with respect to the independent variable) can be used to identify the upper or lower bound, which is useful in treating the velocity branches since the lower and upper bounds are shifted at each update to reflect conditions in the new region, i.e.,

for the first branch, VLOWER = 0 and VUPPER = 150;
for the second branch, VLOWER = 150 and VUPPER = 536.36363636364;
for the third branch, VLOWER = 536.36363636364 and VUPPER = V*,

where V* in an absurdly high value which will never be reached.

The user identifies the branch of the ODE to be used by the current values of PHI. Changes in the branches are permitted *only* at the update of a PHI component. Thus, the integrator steps over the discontinuity, the PHI analysis detects the difficulty and traps the zero point. The user may then update the branch being used, designated by the parameter IZWEIG(J), J=1,2,3, where branches for (4.5) are defined by J=1, those for (4.8.2) and (4.8.3) for J=2, and those for (4.9.1-3) by J=3. The sign of the PHI component being updated

reflects conditions in the new branch. Thus, at update, the new branches of I_α, Eqn. (4.5) are defined by:

IZWEIG(1) = 1, if PHI(1) > 0 , corresponding to Eqn. (4.5.1),
and IZWEIG(1) = 2, if PHI(1) < 0 , corresponding to Eqn. (4.5.2),

the branches of F(T) and MASSE(T) are both defined by:

IZWEIG(2) = 1, if PHI(2) < 0 , for Eqns. (4.8.2.1) and (4.8.3.1)
and IZWEIG(2) = 2, if PHI(2) > 0 , for Eqns. (4.8.2.2) and (4.8.3.2).

Since the VLOWER and VUPPER bounds are shifted at each update, identification of the branching in (4.9.1-3) is handled through the derivative of PHI with respect to V, rather than by using the the sign of PHI. PHI(3) is parabolic in V with nose up. Thus,

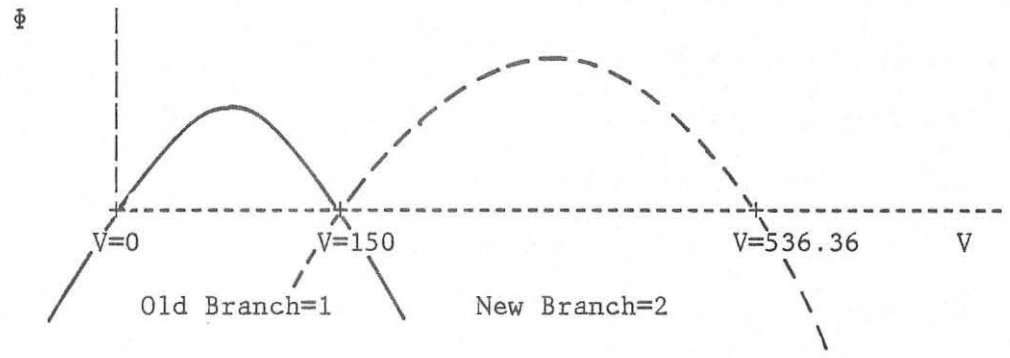
if D(PHI(3))/DT < 0, conditions are at the upper boundary, and
if D(PHI(3))/DT > 0, conditions are at the lower boundary,

giving the following criterion for update:

If D(PHI(3))/ DV < 0, $\quad IZWEIG(3)_{new} = IZWEIG(3)_{old} + 1$.

If D(PHI(3))/ DV > 0, $\quad IZWEIG(3)_{new} = IZWEIG(3)_{old} - 1$.

With the new branch identified, the user must then set the the VLOWER, VUPPER, and VSCALE values corresponding to the region. (See Figure 1.)



Before update: Branch=IZWEIG(3)=1 VLOWER=0 VUPPER=150
Slope = dΦ/dV < 0

After update: Branch=IZWEIG(3)=2 VLOWER=150 VUPPER=536.36

Figure 1. Branching assignment at a velocity discontinuity. The new branching at the point V=150 is defined.

6. SUBPHI, THE SUBROUTINE FOR EVALUATING PHI

The user is required to supply the subroutine SUBPHI for evaluating the PHI vector and its derivatives. (If no derivatives or derivative estimates are available, the user must set the PHIP values to 0.0D0 .) SUBPHI has two basic sections: (1) the evaluation of PHI and PHIP, and (2) the updating portion for vanished PHI components. The structure of SUBPHI is shown in Figure 2. In each of the basic sections the user has particular identifying flags so that a great deal of analysis can be carried on in SUBPHI if this is necessary. If the integrator is embedded deep within another software package, SUBPHI gives the user access to the integration at each step, so that he may monitor the solution as it proceeds.

6.1 Calling Sequence for SUBPHI

The calling Sequence for SUBPHI is:

```
      SUBROUTINE SUBPHI(NPHI,INDEX,NEQN,T,Y,YP,PHI,PHIP,KOUNTR,
     1                  UPDATE,IVAN,BOUNCE,ABSER)
```

If SUBPHI is used only to evaluate the PHI components, most of the parameters in the calling sequence will not be needed. For analysis of discontinuous functions during the integration, however, most of the parameters are of importance. These parameters are defined below, with their use better described in the following two sections.

Parameters identifying the ODE are:

T the independent variable

Y the dependent variable, dimensioned (NEQN)

YP the derivative of Y, dimensioned (NEQN), and

NEQN the dimension of the system.

where Y and YP always correspond to the given value of T. Parameters related to the PHI vector are:

PHI the vector whose zeros are sought, dimensioned (NPHI),

PHIP the derivative of PHI, dimensioned (NPHI), and

ABSER the convergence tolerance or the "vanishing" criteria for the PHI components. ABSER may be set by the user when KOUNTR=0. If no value is set, the integration tolerance will be imposed.

with the indicators or counters:

INDEX designates the component of PHI being studied.

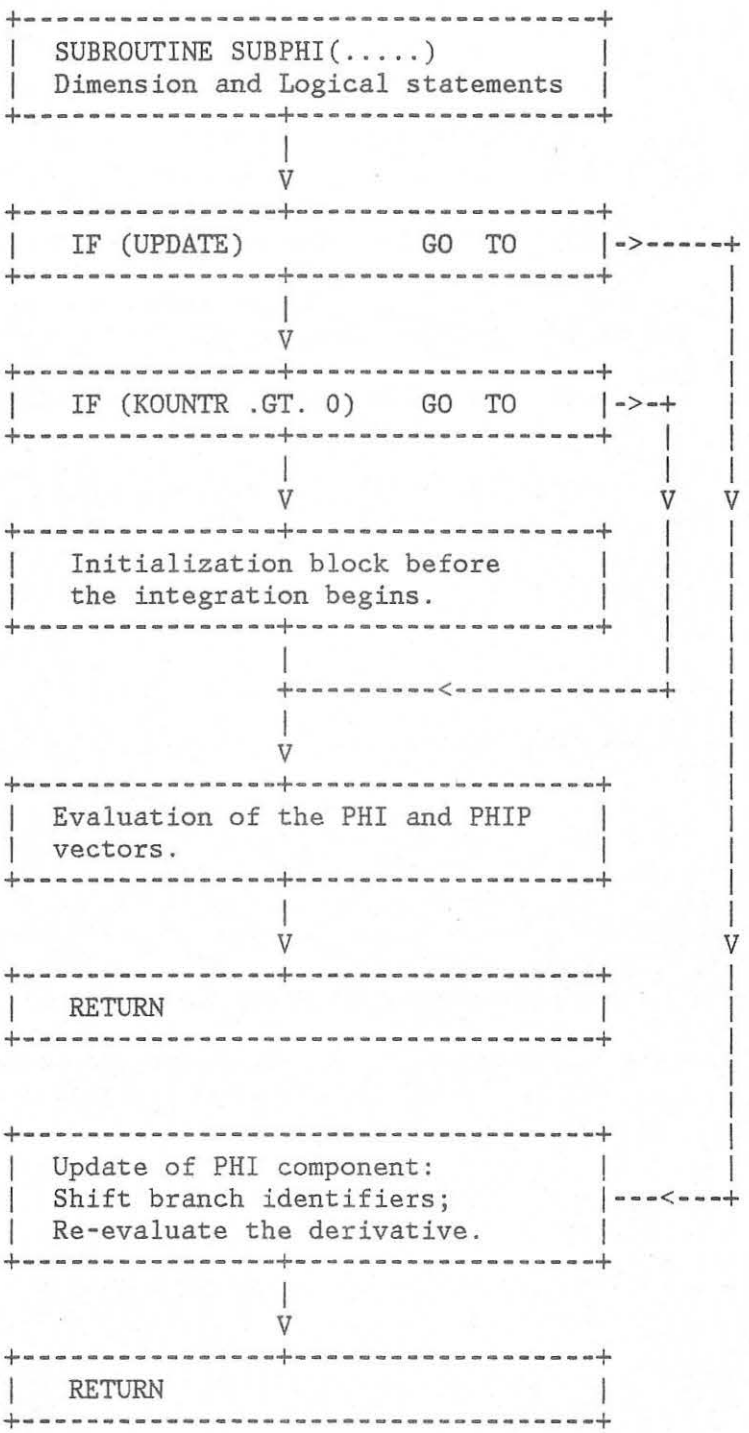


Figure 2. The structure of the SUBPHI subroutine.

KOUNTR counts the number of updates of the PHI vector (KOUNTR=0, indicates the initial call to SUBPHI before the first integration step is taken.) If KOUNTR < 0, the value is a dummy value indicating special conditions at update and will be reset to its previous value after the update.

LOGICAL parameters are:

UPDATE = .FALSE. ==> The PHI and PHIP values are to be computed for RKF45T analysis.

UPDATE = .TRUE. ==> PHI(INDEX) is being updated.

IVAN = .FALSE. ==> PHI(INDEX), which is being updated, *did not* vanish throughout the integration step.

IVAN = .TRUE. ==> PHI(INDEX), which is being updated, *did* vanish throughout the integration step.

BOUNCE = .FALSE. ==> PHI(INDEX), which is being updated, *did not* "bounce" on a zero on the previous integration step.

BOUNCE = .TRUE. ==> PHI(INDEX), which is being updated, *did* "bounce" on a zero on the previous integration step.

6.2 Evaluation of the PHI and PHIP Vectors

If the RKF45T analysis needs the values of PHI and PHIP, SUBPHI is referenced with UPDATE = .FALSE. . The user must return values of PHI(J) and PHIP(J), J=1,...,NPHI corresponding to the given values of T, Y, and YP. If derivatives of PHI(K) (or derivative estimates) are not available for any value of K, the user must set PHI(K)=0.0D0. (This activates a false-position estimate of the zeros of PHI(K) instead of a Newton-Rhapson estimate.)

The user may want to monitor the progress of the integration which is possible by understanding certain "flags" given in the calling sequence. SUBPHI is referenced after each integration step.

The following conditions hold when UPDATE = .FALSE., i.e., when the user is expected to supply the PHI and PHIP vectors.

INDEX = 0 The integration is proceeding, no zeros have been detected by RKF45T

INDEX > 0 The existence of a zero has been detected and the analysis has been shifted into TRAPPD to locate the zero of PHI(INDEX).

KOUNTR = 0 The integrator is calling SUBPHI before the integration begins. YP has *already* been evaluated.

The user should set ABSER if he wishes to use a tolerance different than the integration tolerance as the convergence tolerance for PHI.

Any initialization needed for PHI may be set. KOUNTR will be set to unity upon return to RKF45T.

The user *must* supply PHI and PHIP.

The PHI components for the example in §4 are given in §5. For this problem, derivatives are available, with needed parameters held in common blocks from KRODE, the subroutine for evaluating the ODE expressions. (See Appendix B.)

6.3 Update Portion of SUBPHI

Once a zero of PHI has been isolated, an update call is made to SUBPHI, with INDEX indicating the component of PHI which has vanished. If several components of PHI vanish at the same value of T, a *separate* update call will be made for each component that has vanished with INDEX corresponding to the subscript of the PHI component being updated at each call. The user should set a flag, "IF (UPDATE .EQ. .TRUE.) GO TO ...". In TROMP, updates are important, because the integration is not returned to the driving program after each step. Thus, the update calls are the user's only access to the ODE system when discontinuities are isolated.

6.3.1 PHI Functions in the "Vanished" Region

The treatment of the PHI function near "zeros" is thoroughly described in [2]. A brief description is provided here, so that the user may understand the sign convention at update. An additional problem, the "bouncing" PHI function is described, since ODE components such as (4.5) could involve a PHI function which has the same sign on both sides of a zero.

6.3.1.1 The PHI Component near a Zero

The RKF45T system is designed to trap the zeros of a vector PHI. The "zero detection" criterion is that the function change sign as it passes through zero (or that, by chance, the integrator step on a zero). Also, since a vanishing point is located within a specified tolerance, a neighborhood exists around each zero, in which any point is an acceptable "zero" of the PHI function. Any point within this neighborhood will be referred to as a "vanishing point" of PHI. The user is generally seeking only one such "vanishing point" within a neighborhood of a specific zero.

The RKF45T package assumes that a PHI component will change sign as it passes through a zero point. When a "vanishing point" is located, the PHI component may not yet have passed through the actual zero, and, therefore, would not yet have changed sign. The sign of PHI at the "vanished" point will be changed artificially if the zero point has not been crossed so that the sign reflects conditions in the new region.

THE SIGN AT UPDATE INDICATES CONDITIONS ACROSS THE BOUNDARY.

(Any artificial sign change is made BEFORE the component is updated, giving the user the chance to change the PHI value before the integration

continues.) If the integration step size is sufficiently small, several points may be located all defining the same zero. If a PHI components vanishes at both ends of an integration step and at the intermediate points studied by the RKF45T package, the component is considered to have vanished throughout the step. In such a case, the sign at the beginning of the step will be retained, since this sign was chosen to reflect conditions in the new region. A PHI component which has vanished throughout the step will be updated with the parameter IVAN=.TRUE., indicating that the same zero has been detected. Thus, in solving the example in §4 if IVAN = .TRUE., an immediate return from SUBPHI (KRPHI) is activated (with no user changes). An example of a PHI function with several "vanished points" representing the same zero is given in §6.3.1.3).

6.3.1.2 The "Bouncing" PHI Component

A "bouncing" PHI component is one which fails to change sign as it passes "through" zero, i.e., the function bounces on the zero. (A function which "bounces" on +EPS or -EPS, where |EPS| is less than the given tolerance, is also considered to be a bouncing function.) The RKF45T analysis is not designed to handle bouncing functions in general. One important example, however, which may occur in the constrained optimization problem, can be treated by the RKF45T package. This example concerns the handling of inequality constraints. More specifically, a constrained optimization problem may study solution estimates which violate the inequality constraints, with the amount of the boundary violation measured through the use of expressions such as (4.5). As the solution is driven back to the boundary, it enters a "vanished" region. Operation near the inequality boundary is often essential in optimization problems, which can lead to solution estimates that remain close to the boundary for a while and then diverge in either direction (since the optimization analysis does not guarantee that the inequality conditions are satisfied until convergence is achieved). Thus, bouncing functions could appear in such applications.

The RKF45T package will attach the incorrect sign to a PHI component that bounces. (See §6.3.1.1.) (If several "vanished" points, corresponding to this zero, are detected, the imposed sign will still be incorrect since the sign is determined by the "new region" which has been incorrectly identified.) This sign error on the bouncing component is first detected when that component leaves the "vanished" region. (The conditions are that: |PHI| at the beginning of the step is less than tolerance, |PHI| at the end of the step is greater than tolerance, with the incorrectly imposed sign indicating a sign change that does not actually exist.) The bouncing function is detected by the RKF45T package, before the iterative process attempts to trap the indicated zero, and the sign of PHI at the beginning of the step is changed. An update call is made to SUBPHI with UPDATE=.TRUE. and BOUNCE=.TRUE. for parameter PHI(INDEX). This call updates a previously detected zero, informing the user that an incorrect sign has been given. The value of KOUNTR is set equal to -1. If the user wants the step to be repeated (with PHI now properly identified), he *must change* KOUNTR to -2 and update the ODE system using the correct information. If KOUNTR=-2, an immediate return to the integrator is activated and the step is repeated. (A step would need to be repeated if an improper branch of the ODE had been identified because of the sign error on PHI.)

In the example in §4, the problem of a "bouncing" function will not generally occur. For PHI(3) to "bounce", the velocity would have to reach a local maximum or minimum at the values of 150 or 536.36. PHI(2), involving the burn-out

time, will never "bounce". The problem of handling the inequality constraint, $|\alpha| < \alpha_m$, however, could encounter such difficulties, as explained above. The user may want to ignore the bouncing possibility (and analysis of the warnings message from RKF45T), but he should at least print a message if BOUNCE=.TRUE. is encountered.

If a bouncing PHI component is identified in the problem stated in §4, the user should approach the difficulty in the following manner. He should shift the ODE branch indicator to reflect the proper conditions, and should set KOUNTR=-2. He must then reevaluate the ODE since the expressions have been changed. The step will then be repeated with the correct expression for the ODE.

6.3.1.3 An Example of Zeros of a PHI Component

The identification of the zeros of the function depicted in Figure 3 and the sign imposed on PHI will be discussed. Each "*" corresponds to the value of the function at an integration step. The function depicted in Figure 3 is

$$f(t) = |g(t,y(t),y'(t))| - g^*$$

where g* can vary with t, y, and y', but is presented as a constant to simplify the drawing. Figure 4 shows the associated PHI component as a function of T. In the problem given in §4, α would be g, and α_m would be g* with g* no longer a constant. If f < 0, branch 1 of Y(I) is to be used, if f > 0, branch 2 of Y(I) is to be used. (For the problem in §4, $Y(5)=I_\alpha$ has such branching.) The PHI component related to the f function is

$$PHI(1) = (g^* - g)\ (g + g^*)$$

which is a parabola in g, nose up, centered at g=0, with zeros at g=-g* and g=+g*. The zeros correspond to those of f . The sign of PHI(1) is positive if f < 0 and negative if f > 0.

Each integration step is identified on the t axis by a letter (a through o, with f and g deleted to avoid confusion). A "vanishing region" about f=0 is depicted by a band ("- - - -"). Points falling within this region correspond to |PHI| values less than tolerance and will activate an update in the RKF45T package. The RKF45T analysis of the functions in Figure 3 would be as follows:

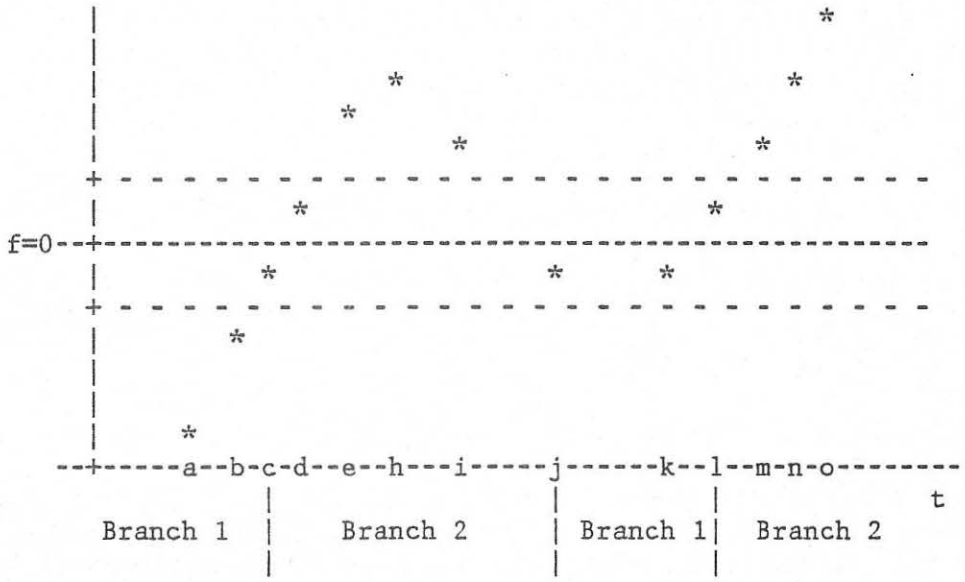
Points a,b — Integration proceeds normally

Point c — Zero is isolated; branch change from 1 to 2; the sign of PHI is changed to "-" even though the integration has not crossed the boundary

Point d — The same "vanished" point is isolated, IVAN=.T.; the sign of PHI is not altered (now correct); no update changes (the same zero was detected).

Points e,h,i — Integration proceeds normally

Point j — Zero is isolated; branch change from 2 to 1; the sign of PHI is changed to "+" (point is already across the boundary)



- - - - bounds the "vanished" region about f = 0, i.e., corresponding to "vanished" PHI values.

* designates results from an integration step

Figure 3 A "bouncing" function with additional standard zeros.

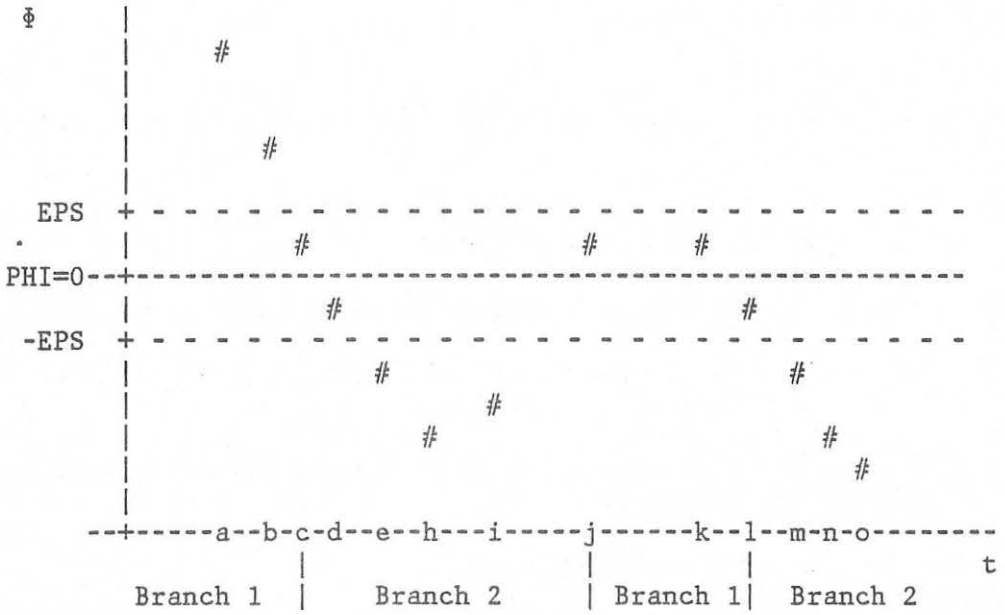
Point k The same "vanished" point is isolated, IVAN=.T., the sign of PHI is not altered (still correct); no update changes (the same zero was detected).

Point 1 The function BOUNCES, but the bouncing goes goes undetected. The same "vanished" point is isolated, IVAN=.T., The sign of PHI is not altered (remains + although it should be -); no update changes (the same zero was detected).

Point m PHI is negative, a sign change is detected, and RKF45T begins zero search; "bouncing" is detected before the zero iteration process begins; update call, BOUNCE=.T., allows the user to change branching from 1 to 2 and update the ODE; the integration step is repeated with correct branching

Points n,o Integration proceeds normally.

A summary of conditions at each point is given in Figure 4, where the reader should note that f and PHI have opposite signs if the analysis is proceeding correctly. An * denotes an artificial sign change (See §6.3.1.1.)



designates results from an integration step

Point	IVAN	BOUNCE	f	PHI	Branch change
a,b			-	+	
c	F	F	-	-*	from 1 to 2
d	T	F	+	-	none
e,h,i			+	-	
j	F	F	+	+	from 2 to 1
k,l	T	F	-	+	none
m	T	F	+	+*	first none then from 1 to 2
n	F	F	+	-	bouncing will be detected in RKF45T
o,p			+	-	

* denotes an artificial sign change

Figure 4. PHI as a Function of T, corresponding to f in Figure 3, with summary of conditions at integration points.

6.3.2 Logical Parameters at Update

The logical parameters at update are summarized below. For each example UPDATE=.TRUE.

IVAN=BOUNCE=.FALSE. Normal update. INDEX informs the user of the component of PHI which has vanished.

IVAN = .TRUE. (BOUNCE=.FALSE.) The solution vanished throughout the entire integration step. The user has discovered the same zero. (The sign will be kept the same as at the beginning of the step.)

The user may want to activate a return without changing conditions.

BOUNCE = .TRUE. KOUNTR=-1, IVAN=.FALSE.

A component of PHI has bounced on a zero rather than passing through zero. An incorrect sign may have been imposed.

This zero has already been updated. KOUNTR has been given a dummy value to indicate that this is a second "update" to inform the user of a possible problem.

If the PHI component indicates that an error existed in the ODE system, because of this bouncing error, the user may force the STEP TO BE REPEATED, by setting KOUNTR=-2. In this case, the user should make corrections to the ODE system.

For the problem in §4, the standard update (UPDATE=.TRUE., IVAN=.FALSE., BOUNCE=.FALSE.), requires a change in branches of the appropriate ODE component. These branches are identified by IZWEIG(J) descibed in §5. If IVAN=.TRUE., no update is permitted, since the zero from the previous step has again been detected. If BOUNCE=.TRUE., the branches are reset, the derivatives are recomputed, and the integration step is repeated.

7. SUBROUTINES ZWEIGE AND WARN

The user is required to provide subroutine ZWEIGE which defines the branches of the ODEs to be used at the initial time. The ZWEIGE call is an insertion in the original TOMP package which gives the user access to the problem before the integration package is called. (When ZWEIGE is referenced, y' is not yet available at t.) An optional subroutine WARN is also included to print warning messages if the wrong branch of a function is being used. The user defines the structure of both subroutines.

7.1 Subroutine ZWEIGE

The calling sequence of ZWEIGE is: ZWEIGE(NEQN,T,Y) with any other required parameters supplied through common blocks. In the example given, branch information, denoted IZWEIG(J), J=1,2,3, is held in common block BRANCH in subroutines F, SUBPHI, and ZWEIGE, (named KRODE, KRPHI, and ZWEIGE, respectively). (See Appendix B.)

TROMP (or TOMP) may be used in either a forward or a backward differencing mode. In the forward differencing mode, the integration always occurs in the forward direction. In this case, ZWEIGE may be simple in structure, if the user can recognize the branch assignments. In the backward differencing mode, the integration occurs in either the forward direction (T = 0 to T = 1) or in the backward direction (T = 1 to T = 0). The initial conditions for the backward integration are the final conditions from the forward integration. The user must provide the branch information at TF for the backward integration. In general, the final step of the forward integration will give the correct branching information for the start of the backward integration. If, however, RKF45T isolates a zero at T=1, the branch update could give the wrong information, i.e., a point of discontinuity update at TF would change the branch information for continued integration in the forward direction. The user should print warnings if incorrect branches are given in ZWEIGE.

For the example presented, IZWEIG(1) must be computed for each integration since the choice of branching depends upon the control function, a_z, which changes at each integration. IZWEIG(2) and IZWEIG(3) are known and are defined for all integrations. ZWEIGE is listed in Appendix B for the example (and initial conditions) in §4.

7.2 Subroutine WARN

Subroutine WARN is supplied to print warning messages if conditions are outside the current bounds. Since the trapping procedure is identified (through non-zero values of INDEX) only after one step across the boundary has been made, a "time-lag" has been inserted, stating that two steps over the boundary must be made before the warning is activated. WARN has an additional mode (MODE=4) which checks branching at the initial conditions or at update, where the time lag is not to be imposed. Sufficient comment cards are present in the subroutine listing (Appendix B) so that the user can understand the structure of the program. This subprogram is not essential to the TROMP analysis but is a good safety feature to include.

8. SUBROUTINE TROMP--THE CONVERSION FROM TOMP

The user is required to supply specified information for the TOMP software package. (See [4].) The user must also provide the following information for the TROMP package:

- subroutine SUBPHI, given in an external reference statement, the subroutine which evaluates the PHI components whose zeros are being sought,

- subroutine ZWEIGE, given in an external reference statement, the subroutine for evaluating the "branch" identifiers for the ODE expressions at the initial conditions,

- the values of NPHI, the dimension of the PHI vector, and the logical parameter, TRAP to identify the use of the zero trapping option.

If the user wishes to reference TROMP as TOMP, i.e., without the zero-trapping option, he must supply dummy subroutines for SUBPHI and ZWEIGE (which will not be referenced) and set NPHI equal to a positive value (preferably unity for less wasted storage). The only integration package, currently existing for use in TROMP, is the RKF45T package.

The communication through the TROMP calling sequence is the same as that in TOMP (described in [4]), with the additional subroutine names, SUBPHI and ZWEIGE added. The CTROMP common block has additional parameters, TRAP and NPHI. TRAP is set equal to .TRUE. if the trapping option is to be used in conjunction with the RKF45T integrator; NPHI states the number of components in the supplied stopping vector, PHI. Otherwise, the common block CTROMP, which conveys parameters needed for TROMP, contains the same information as CTOMP (which is the common block link between TOMP and the user). The parameters in CTROMP *have been reordered* in the standard form, i.e., double precision, single precision, integer, and logical.

An additional common block IDENT has been included to identify the particular integration (as a debugging aide). The form for IDENT is

```
COMMON/IDENT/ITER8,KNT,KNTFI,KNTBI
```

ITER8 identifies the current iteration in the optimization package (not supplied in TROMP),

KNT = 1, indicates the first forward integration,

KNTFI = J, indicates the Jth perturbed forward integration,

KNTBI = 1, indicates the backward integration.

Otherwise, TROMP and TOMP are used identically. TROMP is listed in Appendix A with the modifications listed between comment cards of the form "C----------".

9. TEST RESULTS

The problem presented in § 4 must be considered in two parts. First, the individual integrations with stops at the discontinuities, and then the application an optimization code using the cost function and gradient evaluations from TROMP to optimize the trajectory subject to given constraints. Input parameters are given in the driving program and are again printed in

the output listing (Appendix C). The final time is a fixed value, TF = 3.2 s (with a burn-out time of 2.7 s).

9.1 A Sample Integration

To illustrate the isolation of the points of discontinuity, the first integration is presented with "dense" printing. The solution is printed after each acceptable integration step, with output at discontinuity points of the ODE (zeros of the PHI function) showing the branch of the ODE to be used as the integration continues. The initial choice of branches is also printed. The solution is listed in Appendix C.

The detected points of discontinuity involve the velocity (PHI(3)=0), corresponding to the branching in az , and the burn-out time (PHI(2) or TIME=2.7), corresponding to the branching in the thrust and mass equations. The $\alpha > \alpha$ boundary is never violated during this sample integration, meaning that $I\alpha'$ does not switch branches during the first integration. Conditions not pertaining directly to the integration have been deleted in the output in Appendix C.

The initial branching is IZWEIG(1)=1, i.e., Eqn. 4.5.1, IZWEIG(2)=1, i.e., Eqns. 4.8.2.1, 4.8.3.1, and IZWEIG(3)=1, i.e., Eqn. 4.9.1. At the following T values, discontinuities in the ODEs are isolated and the following branching changes are made:

Time:	Parameter:	New Branch:	Corresponding Equation:
T= 0.1384	V= 150.	IZWEIG(3)=2	Eqn. 4.9.2
T= 0.5501	V= 536.$\overline{36}$	IZWEIG(3)=3	Eqn. 4.9.3
T= 0.8438	TIME=2.7	IZWEIG(2)=2	Eqn. 4.8.2.2

The integration required 201 derivative evaluations of which only 9 were used to isolate the discontinuities.

9.2 The Constrained Trajectory Optimization Problem

The Sequential Linear Least Squares Programming [5] is applied to the problem in §4 with initial conditions the same as those in the sample integration in §9.1. A description of the OCP may be found in [4]. The trapping option is again used (with printing suppressed) to locate discontinuities in the differential equations (i.e., the zeros of the PHI function in the trapping routine). Test results are presented in Appendix C, giving the cost function and constraint violations after each iteration, along with the altered control function, a_z.

With maximum velocity as the cost function,

$$F = -Y(1)$$

and equality constraints:

$$\gamma_f = -1.74533 \text{ rad},$$

$$z_f = -160.0 \text{ m},$$

$$x_f = 15000.0 \text{ m},$$

and inequality constraint:

$$I_{\alpha_f} < ACC = 1.D\text{-}04$$

at the final condition, TF = 3.2,

the optimization procedure converged in 15 iterations, with the requested accuracy of 1.D-04 . The total derivative count required to solve the problem was 67519 . Of this, 66159 were required to solve the integration problems (including the PHI vector analysis), with the additional evaluations being required by the TROMP package during the cost function evaluations. Of the 66159 derivative evaluations used during the integration, 4357 evaluations were used to isolate the zeros of the PHI functions (or the points of discontinuity). Computations were performed in double precision on the IBM 3081, MVS system at the DFVLR-Oberpfaffenhofen, using the extended H-compiler.

The use of the double branched I_α function, Eqn. (4.5) as opposed to a continuous function, which also has contributions when the constraint boundaries are not violated, is justified in this example. If the ODE system in §4 is solved using the I_α expression from Eqn. 2.2, along with the control function a_z from the converged solution, the I_α value is

$$I_\alpha = 0.1827D+0 \quad \text{const: k=1}$$

$$I_\alpha = 0.1663D-1 \quad \text{const: k=5},$$

$$I_\alpha = 0.2896D-2 \quad \text{const: k=10},$$

with the final boundary constraints naturally satisfied. Thus, the SLLSQP program could not have converged to the given solution. On the other hand, a solution which satisfies $I_\alpha <$ ACC, (the convergence tolerance for SLLSQP), using Eqn. (2.2) will naturally satisfy, the bound using (4.5). Basically, *the use of (2.2) prohibits the solution from staying in particular regions which are not yet across the boundary even though such regions could be important to the optimal solution.*

In addition, the problem in §4 with the SLLSQP package was run without the trapping option for isolating discontinuities. A well written ODE package should be able to reduce the step size near discontinuities, performing a "natural" trapping procedure for locating the discontinuities. This process gives reasonable accuracy in some applications but can cause difficulties in other problems. More specifically, at each point of discontinuity a number of steps are attempted, each being rejected until the step size is small enough that the detected error is less than integration tolerance. This does not state that the point of discontinuity has been isolated within integration tolerance. The step size estimation process about a discontinuity (in the "natural" trapping approach) is less efficient than the trapping proce-

dure in RKF45T, which can converge to the point of discontinuity using a Newton iteration. In an optimization process, the integration "noise" resulting from the "natural" trapping causes convergence difficulties, since the slight perturbations in the optimal solution estimates are contaminated with more integration errors. The test results verify this argument. The number of iterations for convergence with the "natural" trapping increased from 15 to 25, with the total derivative count equal to 140526. (The drastic increase in derivative evaluations is, in part, the result of a gradual build up of such costs from each integration, i.e., each integration with "natural trapping" requires more derivative evaluations than that required by the RKF45T trapping procedure although never a striking amount. Naturally, the ten additional iterations increase the derivative evaluation count immensely.) The CPU time for the RKF45T run with trapping procedure is 18.71 s while that for "natural" trapping procedure (run to convergence) is 30.76.

10. CONCLUSIONS

The use of the RKF45T with trapping option for for isolating discontinuities in systems of ordinary differential equations, can be an effective tool for reducing computing time. The user must supply additional subroutines to identify the points of discontinuity as zeros of a particular function, PHI, where PHI may be a vector of any dimension. The user defines the ODE system as a multi-branched system, holding the branches "fixed" until update information indicates that a point of discontinuity has been isolated. Then the "branching" may be redefined. The cost of applying such an iterative procedure for isolating such discontinuity points was less than 7 per cent of the total derivative count required by the RKF45T package, and increases in efficiency if compared to "natural" isolation by a well-written software code, since the latter procedure wastes more time in isolating discontinuities and does not necessarily isolate the points within specified tolerance.

Acknowledgements

The author wishes to thank K.H. Well for his support throughout the development of this system of reports and D. Kraft for his help in the use of the TOMP and SLLSQP packages as well as for posing the problem presented in this report. Thanks are also extended to R. Dierstein for his help in the use of the report writing package for the final draft of the manuscript.

11. REFERENCES

[1] Horn, M.K.
A FORTRAN Program for Solving State/Control-Constraint Optimal Control Problems with System Equations Having Expressions Involving Tabular Data.
DFVLR-IB Nr. 515-83/1

[2] Horn, M.K.
RKF45T - A Runge-Kutta 4/5 Software Package with User-Supplied Stops Involving the Dependent Variables and the First Derivatives.
DFVLR-IB Nr. 515-83/3

[3] Horn, M.K.
Subroutines for Handling Tabular Data Used in System Equations (in Appendix).
DFVLR-IB 515-82/16 (to be printed).

[4] Kraft, D.
FORTRAN-Programme zur numerischen Lösung optimaler Steuerungsprobleme.
DFVLR-Mitteilung 80-03, 1980.

[5] Kraft, D.
Schittkowski, K.
Theorie und Anwendung der Sequentiellen Quadratischen Programmierung in Steuerungs- und Regelungsaufgaben.
DFVLR-Forschungs-Bericht, 1982
(to be published).

[6] Shampine, L.F.
Watts, H. A.
Practical Solution of Ordinary Differential Equations by Runge-Kutta Methods.
SAND 76-0585, Sandia Laboratories, Albuquerque, New Mexico, December 1976.

APPENDIX A. TROMP LISTING

```
C-----------------------------------------------------------------------
C     TOMP WITH MINOR MODIFICATIONS TO USE RKF45T AS ODE SOLVER
C                     MODIFICATIONS--K.HORN 6.12.81
C
C     COPIED 28.08.82--CORRECT VERSION
C
C
C-----------------------------------------------------------------------
      SUBROUTINE TROMP(NX,MG,MQ,X,F,DF,G,DG,LDG,FCT,RHS,SLV,
     1                 SUBPHI,ZWEIGE)
C-----------------------------------------------------------------------
C
C   TOMP    TRAJECTORY OPTIMIZATION BY MATHEMATICAL PROGRAMMING
C           -          -                -           -
C-----------------------------------------------------------------------
C   TROMP:  INTEGRATION "TRAPPING" CAPABILITIES ADDED TO TOMP
C                        --                              ---
C-----------------------------------------------------------------------
C
C   PURPOSE:
C     EVALUATION OF COST-FUNCTION F, CONSTRAINTS G,
C     GRADIENT VECTOR DF AND MATRIX OF CONSTRAINT NORMALS DG
C     AS FUNCTIONS OF PARAMETER VECTOR X.
C     TOMP SOLVES AN INITIAL VALUE PROBLEM (VIA SLV) OF A GIVEN
C     DYNAMICAL SYSTEM (IN RHS) WITH GIVEN INITIAL CONDITIONS (IN Z0)
C     FOR A GIVEN PARAMETRIC CONTROL MODEL U, WHICH IS TEMPORARYLY
C     REPRESENTED BY CONTINUOUS OR DISCONTINUOUS CUBIC OR EXPONENTIAL
C     SPLINE FUNCTIONS (SPLINE).
C     TOMP MAY BE USED FOR TRAJECTORY OPTIMIZATION IN CONNECTION WITH
C     A NONLINEAR PROGRAMMING ALGORITHM (E.G. CFMIN OR CQUNEW),
C     SEE REFERENCE /1/.
C
C-----------------------------------------------------------------------
C     TROMP HAS THE ABILITY TO STOP THE INTEGRATION WHENEVER ANY
C     COMPONENT OF A USER-SUPPLIED, CONSTRAINT FUNCTION VANISHES.
C     THIS CAPABILITY IS USEFUL IN ISOLATING DISCONTINUITIES IN THE
C     ODE SYSTEM.
C-----------------------------------------------------------------------
C
C   INPUT ARGUMENTS:
C     NX    : NUMBER OF PARAMETERS CHARACTERIZING THE PARAMETRIC CONTROL
C             MODEL TOGETHER WITH CERTAIN DESIGN PARAMETERS (NX > 0)
C     MG    : TOTAL NUMBER OF CONSTRAINTS COMPRIZED IN VECTOR G
C             INCLUDING BOUNDS ON CONTROL PARAMETERS X (MG >= 0)
C     MQ    : NUMBER OF BOUNDARY VALUES OF THE GIVEN PROBLEM (FURNISHED
C             IN FCT), WITH EQUALITY CONSTRAINTS FIRST (MQ >= 0)
C     X     : VECTOR OF DIMENSION NX CONTAINING THE CONTROL PARAMETERS
C             AND DESIGN PARAMETERS IN THIS ORDER
C     LDG   : LEADING DIMENSION OF MATRIX OF CONSTRAINT NORMALS DG
C     THETA, VECTORS OF DIMENSION MG EACH CONTAINING APPROXIMATIONS TO
C     SIGMA : LAGRANGE-MULTIPLIERS AND PENALTIES, RESPECTIVELY. THESE
C             VECTORS ARE IMPORTANT ONLY IN CONNECTION WITH MULTIPLIER
C             METHODS FOR CONSTRAINED OPTIMIZATION (CFMIN,E.G., IN CASE
C             OF WHICH VALUES OF THETA & SIGMA ARE FURNISHED BY THESE)
```

```
C
C   OUTPUT ARGUMENTS:
C     F     : SCALAR VARIABLE CONTAINING VALUE OF COST FUNCTION;
C             F IS EVALUATED IN USER FURNISHED SUBROUTINE FCT WITH
C             X AS ARGUMENT (SEE NEXT PARAGRAPH)
C     DF    : VECTOR OF DIMENSION NX OF PARTIALS OF COST FUNCTION F
C             WITH RESPECT TO ARGUMENT VECTOR X
C     G     : VECTOR OF DIMENSION MQ OF CONSTRAINT FUNCTIONS (BOUNDARY
C             VALUES OF THE PROBLEM); G IS EVALUATED IN FCT WITH X AS
C             ARGUMENT (SEE NEXT PARAGRAPH)
C     DG    : JACOBI-MATRIX OF DIMENSION (LDG,MG) OF CONSTRAINT NORMALS
C
C             DF & DG ARE GENERATED BY FORWARD DIFFERENCES OR BY
C             IMPULSIVE RESPONSE-FUNCTIONS DEPENDING ON PARAMETER
C             IMPULS (SEE COMMON /CTROMP/ DESCRIPTION AND REFERENCE /1/)
C
C   SUBROUTINES AS ARGUMENTS:
C     FCT   : USER SUPPLIED SUBROUTINE WITH PARAMETER LIST
C             FCT(P,X,Z,F,G,FP,FZ,GP,GZ,IFLAG)
C               PURPOSE:
C                 COMPUTE COST FUNCTION & CONSTRAINT VECTOR OR THEIR
C                 GRADIENT
C               INPUT ARGUMENTS:
C                 P     : DESIGN PARAMETERS (SECOND PART OF X)
C                 X     : PARAMETRIC CONTROL MODEL & DESIGN PARAMETERS
C                         (SEE X ABOVE)
C                 Z     : VECTOR OF DIMENSION NZ (SEE COMMON /CTROMP/)
C                         CONTAINING THE VALUES OF THE STATE VARIABLES
C                         AT THE END OF THE FORWARD INTEGRATION (AS
C                         RESULT OF THE INITIAL VALUE PROBLEM SOLVED IN
C                         SLV)
C                 IFLAG : INTEGER FLAG, INDICATING WHETHER FCT IS USED
C                         FOR EVALUATING COST & CONSTRAINTS (IFLAG=0)
C                         OR FOR GIVING INITIAL CONDITIONS FOR BACKWARD
C                         INTEGRATION OF ADJOINT VARIABLES Y (IFLAG=1)
C                         NOTE, THAT IN CASE OF GRADIENT GENERATION BY
C                         FORWARD DIFFERENCES (IMPULS=.FALSE.!) FCT IS
C                         USED IN THE "IFLAG=0" MODE ONLY
C               OUTPUT ARGUMENTS:
C                 FOR IFLAG=0:
C                 F     : COST FUNCTION
C                 G     : BOUNDARY VALUES (EQUALITY CONSTRAINTS FIRST!)
C                 FOR IFLAG=1 (IMPULS=TRUE ONLY!):
C                 FP    : VECTOR OF DIMENSION NP (SEE COMMON /CTROMP/)
C                         OF PARTIALS OF COST FUNCTION F WITH RESPECT
C                         TO DESIGN PARAMETERS P
C                 FZ    : VECTOR OF DIMENSION NZ OF PARTIALS OF COST
C                         FUNCTION F WITH RESPECT TO STATE VARIABLES Z
C                 GP    : JACOBI-MATRIX OF (CURRENTLY) DIMENSION (25,25)
C                         OF PARTIALS OF G WITH RESPECT TO P
C                 GZ    : JACOBI-MATRIX OF (CURRENTLY) DIMENSION (25,25)
C                         OF PARTIALS OF G WITH RESPECT TO Z
C     RHS   : USER SUPPLIED SUBROUTINE WITH PARAMETER LIST
C             RHS(T,Z,DZ)
C               PURPOSE:
C                 DESCRIPTION OF THE MATHEMATICAL MODEL OF THE DYNAMICAL
C                 SYSTEM AS A SYSTEM OF ORDINARY DIFFERENTIAL EQUATIONS
C                 OF FIRST ORDER WITH RHS SUPPLYING THE RIGHT HAND SIDES
```

```
C                                .
C                OF THE SYSTEM Z=F(T,Z,U); NOTE THAT THE INITIAL VALUES
C                ARE GIVEN IN Z0 OF COMMON /CTROMP/.
C
C                IN MODE IMPULS=TRUE RHS ALSO MUST FURNISH THE RIGHT
C                HAND SIDES OF THE HOMOGENEOUS ADJOINT SYSTEM
C                                .
C                DEFINED BY Y=-(DF/DZ)'*Y. THIS SYSTEM HAS TO BE INTE-
C                GRATED BACKWARD FROM TF, WITH SUITABLE 'INITIAL CON-
C                DITIONS' RESULTING FROM THE BOUNDARY VALUES AND FOR-
C                MULATED BY THE USER IN SUBROUTINE FCT IN MODE IFLAG=1.
C                FOR FURTHER INFORMATION SEE REFERENCE /1/.
C              INPUT ARGUMENTS:
C                T     : INDEPENDENT VARIABLE (TIME)
C                Z     : CURRENT SYSTEM STATE (VECTOR OF DIMENSION NZ)
C              OUTPUT ARGUMENTS:
C                DZ    : CURRENT STATE DERIVATIVES (WITH RESPECT TO T)
C                        VECTOR OF DIMENSION (NZ)
C     SLV    : USER SUPPLIED STANDARD SOFTWARE SUBROUTINE WITH PARAMETER
C              LIST SLV(RHS,NZ,Z,T0,T1,RELTOL,ABSTOL,IFLAG,WORK,IWORK)
C                PURPOSE:
C                SUBROUTINE SLV INTEGRATES A SYSTEM OF NZ FIRST ORDER
C                ORDINARY DIFFERENTIAL EQUATIONS OF THE FORM
C                   DZ(I)/DT = F(T,Z(1),Z(2), ... , Z(NZ))
C                WHERE THE Z(I) ARE GIVEN AT T
C              INPUT ARGUMENTS:
C                NZ,Z,T0,T1,RELTOL,ABSTOL,IFLAG
C              OUTPUT ARGUMENTS:
C                Z
C              SUBROUTINE SUPPLIED AS ARGUMENT:
C                RHS (SEE ABOVE)
C              ALL ARGUMENTS ARE DESCRIBED IN REFERENCE /2/
C            REMARK: RKF78 IS URGENTLY RECOMMENDED AS SLV HERE,
C                    BOTH WITH RESPECT TO ACCURACY & EFFICIENCY
C            REMARK: RKF45T MUST BE USED FOR TRAPPING CAPABILITIES
C                    UNTIL RKF78T IS AVAILABLE.
C-----------------------------------------------------------------------
C     SUBPHI: USER SUPPLIED, STANDARD SOFTWARE SUBROUTINE WITH
C             PARAMETER LIST SUBPHI(NPHI,INDEX,NEQN,T,Y,YP,PHI,PHIP,
C                                   UPDATE,IVAN,BOUNCE,ABSER)
C             PURPOSE:
C               SUBROUTINE SUBPHI EVALUATES THE PHI VECTOR WHOSE
C               ZERO ARE SOUGHT DURING THE INTEGRATION.  UPDATES
C               AT ZERO LOCATIONS ARE POSSIBLE, PROVIDING USER
C               CONTROL OVER THE ODE SYSTEM DURING THE INTEGRATION.
C               SUBPHI IS REFERENCED AFTER EACH SUCCESSFUL INTEGRA-
C               TION STEP AND AT PHI VANISHING POINTS.  (PHI MAY BE
C               A VECTOR OF STOPPING CONDITIONS--DIMENSIONED NPHI)
C             INPUT ARGUMENTS:
C               NPHI,INDEX,NEQN,T,Y,YP,KOUNTR,UPDATE,BOUNCE,ABSER
C             OUTPUT ARGUMENTS:
C               PHI, PHIP
C             ALL ARGUMENTS ARE DESCRIBED IN REFERENCE /XX/
C
C     ZWEIGE: USER SUPPLIED, STANDARD SOFTWARE SUBROUTINE WITH
C             PARAMETER LIST ZWEIGE(NEQN,T,Y)
C
C             PURPOSE:
```

```
C               SUBROUTINE ZWEIGE IDENTIFIES THE BRANCHES OF EXPRES-
C               SIONS IN RHS WHICH ARE TO BE USED AT THE INITIAL
C               CONDITIONS.  THE STRUCTURE IS GIVEN BY THE USER
C               WITH ADDITIONAL PARAMETERS NEEDED HELD IN COMMON
C               WITH RHS, SUBPHI, AND ZWEIGE.  INITIAL CONDITIONS
C               FOR THE BACKWARD INTIGRATION WILL BE THE FINAL CON-
C               DITIONS FROM THE FORWARD INTEGRATION.
C               ZWEIGE MUST IDENTIFY THESE BRANCHES CORRECTLY.
C             INPUT ARGUMENTS:
C               NPHI,T, Y  (YP HAS NOT YET BEEN EVALUATED IN TROMP)
C             OUTPUT ARGUMENTS:
C               USER-SUPPLIED--IN USER SUPPLIED COMMON BLOCKS
C
C-----------------------------------------------------------------------
C
C   COMMON AREA:
C   CTROMP : THE MAIN TASK OF COMMON /CTROMP/ IS TO RESERVE STORAGE AND
C            TO COMMUNICATE VALUES OF THE PARAMETRIC CONTROL MODEL U.
C            A CONSEQUENCE THEREOF IS THAT COMMON /CTROMP/ IS USED AT
C            LEAST IN SUBROUTINE RHS AND IN A SUPERVISING PROGRAM.
C
C            THE CONTROL MODEL U IS EXAMPLIFIED ONHAND OF THE
C            FOLLOWING FIGURE:
C
C       A
C       |             |                |         |                |
C   A   ------------------------------------------------------------*
C   |   |  UU(I,1,1)  |                |         | UU(I,1,4)      |
C   |   |             |                |         |                |
C       |             |             + +         +  U(I,1,4)   +  |
C   U   +             |         +      |         |  +         +   |
C       |+            |     +          |         |      + +       |
C       | + U(I,1,1)  |  +             *++++++++++                |
C       |   +         |+            *  |         |                **
C       |      +      +          *    *         |          *     |
C       |          +  |        *       | *       |      *         |
C       |             +      *         |   *     |   *            |
C       |             |     * U(I,2,2)  |         * * U(I,2,4)     |
C       ***************   *            |         |*               |
C       |   U(I,2,1)  | *              |         |                |
C       |             *                |         *                |
C       |             |                |         |                |
C       +-------------------------------------------------------------->
C   GRID(1,1)    GRID(KNOT(1),1)                      GRID(KNOT(INTERV),I
C                =GRID(1,2)         GRID(1,3)
C       T0                                    T ---->          TF
C
C
C            THE INTERVALL (T0,TF) OF THE INDEPENDENT VARIABLE T
C            IS DIVIDED INTO I SECTIONS (I=1, ... ,INTERV; 0<INTERV<6)
C            WITH J GRIDPOINTS IN EACH SECTION (J=1, ... ,KNOT(I);
C            0<KNOT(I)<16). TO EACH GRIDPOINT IN THE TOTAL MESH K CON-
C            TROL PARAMETERS U(J,K,I) ARE RELATED (K=1, ... ,KONTRL;
C            0<KONTRL<6).NEITHER THE SECTIONS NOR THE GRIDS ARE CON-
C            STRAINED TO BE EQUIDISTANT.
C
C            TWO DISCONTINUOUS CONTROL FUNCTIONS U VERSUS T ARE SHOWN
C            (MARKED BY + AND *, RESP.) TOGETHER WITH AN UPPER BOUND UU
```

```
C             ON U(I,J,K) FOR ALL GRIDPOINTS I IN EVERY INTERVAL K FOR
C             THE J'TH OF KONTRL CONTROL FUNCTIONS (MARKED BY _).
C
C             CUBIC OR EXPONENTIAL INTERPOLATING SPLINE FUNCTIONS ARE
C             CURRENTLY USED FOR APPROXIMATING THE PARAMETRIC CONTROL
C             MODEL U OVER THE MESH, WITH ST(I,J), I=1, .. ,KNOT(INTERV)
C             ,J=1, ... ,KONTRL, AS STIFFNESS PARAMETERS FOR THE EXPO-
C             NENTIAL SPLINE. VALUES FOR STIFF ARE GENERATED IN SUBROU-
C             TINE GENER. IF STIFF=0 FOR ALL GRIDPOINTS AND ALL CONTROLS
C             CUBIC SPLINE FUNCTIONS ARE USED. FOR FURTHER EXPLANATIONS
C             SEE SUBROUTINE SPLINE AND REFERENCES /1,3,4/.
C
C             SUMMARIZED, THE USER MUST PROVIDE VALUES FOR
C               GRID   : GRIDPOINTS
C               STIFF  : STIFFNESS PARAMETERS
C               U      : INITIAL VALUES FOR THE CONTROL MODEL
C               UL,UU  : LOWER AND UPPER BOUNDS FOR THE CONTROLS
C                        (NECESSARY ONLY IF LBOUND OR UBOUND ARE TRUE
C                        FOR THE CONSIDERED CONTROL)
C               INTERV : NUMBER OF INTERVALS OR SECTIONS. IN EACH SECTN
C                        A SEPERATE SPLINE FUNCTION APPROXIMATES THE
C                        CONTROL MODEL, THUS ALLOWING FOR DISCONTINUOUS
C                        CONTROLS. IF THE USER CONJECTURES A CONTINUOUS
C                        SOLUTION INTERV=1 IS RECOMMENDED.  0<INTERV<6
C               KONTRL : TOTAL NUMBER OF CONTROL FUNCTIONS, 0<KONTRL<6
C               KNOT(I): AN INTEGER ARRAY PRESCRIBING THE NUMBER OF
C                        GRIDPOINTS OR KNOTS IN INTERV I.  0<KNOT(I)<16
C               LBOUND(I),
C               UBOUND(I), I=1, ... ,KONTRL, ARRAY OF LOGICAL VARIABLES
C                        INDICATING WHETHER A LOWER OR UPPER BOUND ON
C                        CONTROL(I)IS PRESCRIBEB (E.G. LBOUND(2)=TRUE)
C               P(I)   : I=, ... ,NP, FREE DESIGN PARAMETERS SUCH AS
C                        SWITCHING TIME FOR DISCONTINUOUS CONTROLS,
C                        FREE FINAL TIME TF, OR AIRCRAFT DESIGN PARA
C                        METERS (WING LOADING A.O.)
C               NP     : NUMBER OF FREE DESIGN PARAMETERS, 0<NP<11
C               TOL    : RELATIVE=ABSOLUTE LOCAL ERROR TOLERANCES
C                        FOR SUBROUTINE SLV (SEE RELTOL,ABSTOL ABOVE)
C               Z0(I)  : I=1, ... ,NZ, INITIAL CONDITIONS FOR THE STATE
C                        VARIABLES OF THE ANALYZED SYSTEM
C               NZ     : NUMBER OF STATE VARIABLES, 0<NZ<21
C               IMPULS : LOGICAL VARIABLE INDICATING WHETHER GRADIENTS
C                        ARE GENERATED VIA IMPULSIVE RESPONSE FUNCTIONS
C                        (IMPULS=TRUE) OR FORWARD DIFFERENCES (FALSE)
C               STORE,STORB: LOGICAL VARIABLES TO BE SET TRUE IF INITIAL
C                        TRAJECTORIES ARE TO BE PRINTED; BOTH ARE RESET
C                        FALSE AFTER PRINTING
C               LPLOT  : LOGICAL VARIABLE TO BE SET TRUE IF PLOTTING IS
C                        DESIRED; OTHERWISE IT MUST BE FALSE
C               MQP    : NUMBER OF SETS OF ADJOINT VARIABLES TO BE
C                        INTEGRATED BACKWARD, INITIALIZED IN TOMP
C                        AND TRANSFERED TO RHS.
C                        MQP = 1 FOR MINIMIZATION WITH CFMIN IN VERSION
C                              NEWTON=.FALSE.
C                        MQP = 1+MQ IN ALL OTHER CASES.
C
C             A,B,C,UH,AH,BH,CH ARE INTERNAL ARRAYS OF VARIABLES USED
C             BY SUBROUTINES SPLINE, SPLINT & GENER. THEY MUST NOT BE
```

```
C               ALTERED BY THE USER.
C               FORWRD : INTERNAL LOGICAL VARIABLE NOT TO BE CHANGED BY
C               THE USER.
C
C     CCFMIN: COMMON /CCFMIN/ IS FULLY DESCRIBED IN SUBROUTINE CFMIN
C           DUMMY(15) : DOUBLE PRECISION ARRAY OF PRIVATE VARIABLES
C                       THEY MUST NOT BE CHANGED BY THE USER
C           IG    : INTEGER*4 VARIABLE, SET BY CFMIN HELPING TO
C                   REDUCE COMPUTATION TIME
C           IG = 0: FUNCTION EVALUATION ONLY
C           IG = 1: FUNCTION AND GRADIENT EVALUATION (IMPLICIT FORM)
C           IG = 2: FUNCTION AND GRADIENT EVALUATION (EXPLICIT FORM)
C           IG = 3: GRADIENT OF CONSTRAINTS ONLY
C                   IF TOMP IS USED IN CONNECTION WITH OTHER NONLINEAR
C                   PROGRAMMING ALGORITHMS THAN CFMIN IG=2 IS RECOMMD.
C           LFL   : LOGICAL VARIABLE SET TRUE BY TOMP IF STEPSIZE IS
C                   TO BE REDUCED IN CFMIN, OTHERWISE LFL=FALSE; IT
C                   MAY NOT BE MANIPULATED BY THE USER.
C
C   REMARKS:
C-----------------------------------------------------------------------
C   0.SUBROUTINE TROMP IS A SLIGHT MODIFIECATION OF SUBROUTINE TOMP
C     DESCRIBED BELOW.  CHANGES ARE PLACED BETWEEN "C---------------"
C     CARDS FOR EASY RECOGNITION.  TROMP HAS THE ADDED FEATURE THAT
C     THE INTEGRATION WILL BE HALTED WHENEVER USER-SUPPLIED AUXIL-
C     LIARY FUNCTIONS VANISH.  THIS CAN BE USED TO IDENTIFY DISCON-
C     TINUITIES IN THE RHS SUBROUTINE.  DETECTION OF THE VANISHING
C     POINTS IS ACCOMPLISHED THROUGH THE USER SUPPLIED SUBROUTINE
C     SUBPHI.  SUBROUTINE ZWEIGE IS REFERENCED BEFORE EACH INTEGRA-
C     TION STEP SO THAT THE USER MAY IDENTIFY THE PROPER BRANCH OF
C     THE RHS SUBROUTINE EXPRESSIONS.  THE APPLICATION OF TROMP IS
C     DESCRIBED IN /XX/.
C-----------------------------------------------------------------------
C   1.SUBROUTINE TOMP IS BEST SUITED FOR TRAJECTORY OPTIMIZATION
C     IN CONNECTION WITH NONLINEAR PROGRAMMING AND REVERSE COMMUNICATION
C     (SEE REMARKS IN SUBROUTINE CFMIN)
C     EXAMPLE:
C             DIMENSION W(350,E.G.)
C                     .
C                     .
C             EXIT = DIM(W)
C          10 CALL CFMIN (.... ,X,F,G,FX,GX, ... ,W, ... ,EXIT,..)
C                     .
C                     .
C             CALL TOMP  (.... ,X,F,G,FX,GX, .... )
C             IF(EXIT.GT.5 .AND. EXIT.LT.10) GOTO 10
C                     .
C                     .
C   2.THE EQUATION NUMBERS IN THE COMMENTS WITHIN THE PROGRAM FLOW
C     REFER TO REFERENCE /1/.
C
C   METHOD:
C     (I) SOLUTION OF  1 INITIAL VALUE PROBLEM TO DEFINE F & G
C    (II) SOLUTION OF NX PERTURBED INITIAL VALUE PROBLEMS FOR DF & DG
C         BY FORWARD DIFFERENCES
C   (III) SOLUTION OF MQP FINAL VALUE PROBLEM TO DEFINE DF & DG
C         BY IMPULS RESPONSE FUNCTIONS DEFINED BY THE ADJOINT SYSTEM.
C         NOTE THAT MQP IS ONE IN CASE WORKING IN CONNECTION WITH CFMIN
```

```
C          WHILE MQP=MQ+1 FOR POWELL'S METHOD FOR INSTANCE;
C          SEE REFERENCE /1/
C     (IV) SOLUTION OF 1 INITIAL VALUE PROBLEM TO GRAPH & STORE THE SOLU-
C          TION FOR PLOTTING IN MODE IG<0
C
C   REFERENCES:
C     /1/ KRAFT,D. (1979) COMPARING MATHEMATICAL PROGRAMMING ALGORITHMS
C           BASED ON LAGRANGIAN FUNCTIONS FOR SOLVING OPTIMAL CONTROL
C           PROBLEMS. DFVLR IB 552-79/7.
C     /2/ SHAMPINE,L.F.,WATTS,H.A. (1976) PRACTICAL SOLUTION OF ORDINARY
C           DIFFERENTIAL EQUATIONS BY RUNGE-KUTTA METHHODS. SANDIA LABS.
C           REPORT SAND 76-0585.
C     /3/ BULIRSCH,R.,RUTISHAUSER,H. (1968) INTERPOLATION UND GENAEHERTE
C           QUADRATUR. IN:SAUER,R.,SZABO,I.(EDS.) MATHEMATISCHE HILFS-
C           MITTEL DES INGENIEURS,VOL.III. BERLIN-HEIDELBERG-NEW YORK:
C           SPRINGER.
C     /4/ RENTROP,P. (1979) AN ALGORITHM FOR THE COMPUTATION OF THE
C           EXPONENTIAL SPLINE. TO APPEAR IN 'HANDBOOK SERIES APPROXI-
C           MATION. BERLIN-HEIDELBERG-NEW YORK: SPRINGER.
C
C   IMPLEMENTED BY:
C      KRAFT,D., DFVLR - INSTITUT FUER DYNAMIK DER FLUGSYSTEME
C                D-8031 OBERPFAFFENHOFEN
C
C   STATUS: 05. DECEMBER 1979, REVISED 05. APRIL 1982
C
C-----------------------------------------------------------------------
C   MINOR MODIFICATIONS (FOR TRAPPING CAPABILITIES) BY:
C      HORN, M.K., DFVLR - INSTITUT FUER DYNAMIK DER FLUGSYSTEME
C                  D-8031 OBERPFAFFENHOFEN
C-----------------------------------------------------------------------
C
C   SUBROUTINES REQUIRED: * = DIRECT CALL
C     *GENER,*SPLINE,*BILD,*GRAPH,*FCT,RHS,*SLV,SWW2,ZPFORM,IKL
C     IF RKF78 IS USED AS SLV ADDITIONALLY RKFS AND FEHL ARE REQUIRED
C
C
      DOUBLE PRECISION A,B,C,F,G,H,P,U,X,Z,
     .AH,BH,CH,DF,DG,DH,DT,FP,FZ,GP,GZ,TF,TT,T0,T1,UH,UL,UU,Z0,
     .DEL,ONE,PM6,TOL, GRID,TOL1,WORK,ZERO, DSQRT,DUMMY,SIGMA,STIFF,
     .THETA, FSTORE,GSTORE,SPLINT,HFF
C
      INTEGER I,J,K,L,M,N, ID,IG,I1,I2,I3,I4,I5,MG,MQ,NN,NP,NX,NZ,
     .LDG,MQP,MQ1, ISPL,JUMP,KNOT,LONG,NABS,NORD, IFLAG,IWORK,KNOTI,
     .LSAVE,NGRID,NPLOT,NXNMP,INTERV,KONTRL,KNOTI2,IOUT,JOUT
C-----------------------------------------------------------------------
     .,IFLAGS,NPHI,KNT,KNTFI,KNTBI,ITER8,NFE,NEXT
C-----------------------------------------------------------------------
C
      LOGICAL IMPULS,FORWRD,STORE,STORB,LBOUND,UBOUND,IFL,JFL,KFL,LFL,
     *NUL,OUT,NOBOUN,LPLOT,SAMESC/F/,HOLD,LFCT
C-----------------------------------------------------------------------
     *,TRAP
C-----------------------------------------------------------------------
C
      DIMENSION X(NX),DF(NX),G(01),DG(LDG,01),SIGMA(01),THETA(01),FP(10)
     *,FZ(20),GSTORE(20),GP(10,99),GZ(20,99),Z(250),
C-----------------------------------------------------------------------
```

```
     * WORK(3253),IWORK(10)
C-----------------------------------------------------------------------
C
      REAL ORD0,ORD1,ORD,ABS,ABST,T2,T3, UF(402,5),ZF(402,20),HF(402,5)
C
C     COMMON /CCFMIN/ DUMMY(15),IG,LFL
      COMMON /C OUT / IOUT,JOUT,HFF(5)

      COMMON /CTROMP/
     1  GRID(15,5),STIFF(15,5),U(15,5,5),A(15,5,5),B(15,5,5),C(15,5,5),
     2  UL(15,5,5),UU(15,5,5),UH(15,15,5,5),AH(15,15,5,5),BH(15,15,5,5),
     3  CH(15,15,5,5),P(10),Z0( 6),TOL,INTERV,KONTRL,KNOT(5),NP,NZ,
C-----------------------------------------------------------------------
     4  MQP,IG,NPHI,LFL,
     5  IMPULS,FORWRD,STORE,STORB,LPLOT,LBOUND(5),UBOUND(5),LFCT,TRAP
C
      COMMON/IDENT/ITER8,KNT,KNTFI,KNTBI,NFE,NEXT
C-----------------------------------------------------------------------
C
C-----------------------------------------------------------------------
      EXTERNAL RHS,SUBPHI,ZWEIGE
C-----------------------------------------------------------------------
C
      DATA ZERO,ONE,PM6/0D0,1D0,1D-6/,IFL,JFL,KFL/3*F/,ID/400/,NN/17/,
     *ORD0,ORD1,ORD,ABS,ABST,NORD,NABS,NUL,OUT/
     * 0E0, 0E0,6E0,15E0,4E0,   4,   10,   T,   F/,LONG/053/,NGRID/080/,
     *NPLOT/8/
C
C##########################################################################
C**  1.0:  TRAJECTORIES & BOUNDARY VALUES FOR OPTIMAL CONTROL PROBLEMS #
C##########################################################################
C
      TOL1 = TOL
C**  ADJUSTMENT OF TOLERANCE FOR GRADIENT GENERATION
C**  BY DIFFERENCE SCHEMES
C     IF(.NOT. IMPULS) TOL1 = 1D-2*TOL
      IF(IG.GT.2 .AND. .NOT. LPLOT)                              GOTO 280
      LFL = .FALSE.
C
C**  RESTORE CONTROL-PARAMETERS & EVALUATE SPLINE FOR NOMINAL CONTROL **
C**  GENER GENERATES TENSION PARAMETERS FOR EXPONENTIAL SPLINES  **
C
      ISPL = 1
      IF(STIFF(1,1).NE.ZERO .AND. .NOT.JFL) ISPL = 2
      DO 110 L=1,ISPL
      LSAVE = 0
      DO 110 I=1,INTERV
      KNOTI = KNOT(I)
      DO 110 J=1,KONTRL
      DO 100 K=1,KNOTI
      LSAVE = LSAVE+1
  100 U(K,J,I) = X(LSAVE)
      IF(L.EQ.2)
     *CALL GENER (KNOTI,GRID(1,I),U(1,J,I),STIFF(1,I),C(1,J,I))
  110 CALL SPLINE(KNOTI,GRID(1,I),U(1,J,I),STIFF(1,I),
     *            A(1,J,I),B(1,J,I),C(1,J,I))
      IF(NP.LE.0)                                                GOTO 130
      NXMNP = NX-NP
```

```
C
C**  RESTORE DESIGN-PARAMETERS  **
C
      DO 120 I=1,NP
  120 P(I) = X(NXMNP+I)
  130           ASSIGN 220 TO JUMP
C-----------------------------------------------------------------------
      KNT = 0
      KNTFI = -1
      KNTBI = 0
      NFE   = 0
      NEXT  = 0
C-----------------------------------------------------------------------
      IF(LPLOT) ASSIGN 690 TO JUMP
C
C**  ENTRY POINT FROM 2.2  **
C**  FORWARD INTEGRATION OF SYSTEM STATE   EQ.(2.2)  **
C
C-----------------------------------------------------------------------
      KNT = KNT + 1
      KNTFI = KNTFI + 1
C-----------------------------------------------------------------------
  140 IF(LFCT)                                                GOTO 215
      FORWRD = .TRUE.
C**  INITIAL VALUES OF STATE VECTOR  EQ.(2.3)  **
      DO 150 I=1,NZ
  150 Z(I) = Z0(I)
      T0 = GRID(1,1)
C-----------------------------------------------------------------------
      IFLAGS = 1
      IF (TRAP) IFLAGS = 10
      IF (TRAP) CALL ZWEIGE(NZ,T0,Z)
      IFLAG = 1 * IFLAGS
      IF(.NOT.JFL) IFLAG = -1 * IFLAGS
C-----------------------------------------------------------------------
      IF(.NOT.STORE)                                          GOTO 160
      JOUT = 1
C     PRINT 998
C     PRINT 996, T0,(Z(J),J=1,NZ)
      CALL OUTPUT(T0,Z,WORK)
  160 CALL RHS(T0,Z,WORK)
      CALL FCT(P,X,Z,FSTORE,GSTORE,FP,FZ,GP,GZ,2,T0,WORK)
      DO 210 I=1,INTERV
      KNOTI=KNOT(I)
      DO 210 J=2,KNOTI
      T1 = GRID(J,I)
C
C**  SOLVING THE INITIAL VALUE PROBLEM  **
C
CURRENTLY RKF78 IS RECOMMENDED AS SLV
C
      KOUNT = 0
  169 CONTINUE
      KOUNT = KOUNT + 1
C-----------------------------------------------------------------------
  170 CALL SLV(RHS,SUBPHI,NPHI,NZ,Z,T0,T1,TOL1,TOL1,IFLAG,WORK,IWORK)
C-----------------------------------------------------------------------
      IF (IFLAG .LT. 0 .AND. T0 .NE. T1)                  GO TO 170
```

```
      IF (IFLAG .EQ. 3)                                          GO TO 170
      IF (IFLAG .GT. 2 .AND. IFLAG .LE. 8)                       GO TO 180
      IF (IFLAG .GT. 25)                                         GO TO 180
      GO TO 190
C-----------------------------------------------------------------------
  180 PRINT 997, IFLAG,FORWRD
C-----------------------------------------------------------------------
      IF (IFLAG .GT. 8)     STOP
C-----------------------------------------------------------------------
C
C**  SET FLAG FOR STEP-SIZE REDUCTION IN CFMIN  **
C
      LFL = .TRUE.
                                                                 GOTO 800
  190 IF(.NOT.STORE)                                             GOTO 200
C     TT = T0*P(I)
C     PRINT 996, T0,(Z(K),K=1,NZ)
      CALL OUTPUT(T0,Z,WORK)
  200 CONTINUE
CRUDE IMPLEMENTATION OF MULTIPOINT BOUNDARY VALUE PROBLEMS
      CALL FCT(P,X,Z,FSTORE,GSTORE,FP,FZ,GP,GZ,2,T0,WORK)
C-----------------------------------------------------------------------
  205 IF(IFLAG.EQ.-2 * IFLAGS)                                   GOTO 170
      IF(.NOT.JFL) IFLAG = -2 * IFLAGS
C-----------------------------------------------------------------------
  210 CONTINUE
C-----------------------------------------------------------------------
      KNT = -1
      NFE = NFE + IWORK(1)
      NEXT = NEXT + IWORK(3)
C-----------------------------------------------------------------------
C     SPACE FOR USER SUPPLIED PRINT OUT  AFTER FORWARD INTEGRATIONS
C
C     IF KNT = 1 (KNTFI = 0)    , UNPERTURBED FORWARD INTEGRATION
C     IF KNTFI > 0  (KNT = 0)   , PERTURBED FORWARD INTEGRATION
C
C-----------------------------------------------------------------------
C
      JOUT = 0
      JFL = .TRUE.
      STORE = .FALSE.
C
C**  FUNCTION EVALUATION FOR NOMINAL & VARIED CONTROLS  EQ.(3.7)  **
C
  215 CALL FCT(P,X,Z,FSTORE,GSTORE,FP,FZ,GP,GZ,0,T0,WORK)
      G O T O JUMP,                                  (220,580,610,690)
  220 F = FSTORE
      IF(MQ.LE.0)                                                GOTO 235
      DO 230 I=1,MQ
  230 G(I) = GSTORE(I)
C
C##########################################################################
C**  1.1:  BOUNDS ON CONTROL VARIABLES  **
C##########################################################################
C
  235 M = MQ
      DO 270 I=1,INTERV
      KNOTI = KNOT(I)
```

```
      DO 270 J=1,KONTRL
      IF(.NOT.LBOUND(J))                                          GOTO 250
      DO 240 K=1,KNOTI
      M = M+1
  240 G(M) = (U(K,J,I)-UL(K,J,I))/DMAX1(DABS(UL(K,J,I)),PM6)
  250 IF(.NOT.UBOUND(J))                                          GOTO 270
      DO 260 K=1,KNOTI
      M = M+1
  260 G(M) = (UU(K,J,I)-U(K,J,I))/DMAX1(DABS(UU(K,J,I)),PM6)
  270 CONTINUE
      NOBOUN = M.EQ.MQ
      MQ1 = M
      IF(IG.EQ.0)                                                 GOTO 800
C
C##############################################################################
C**  2.0:  EVALUATION OF GRADIENT OF PENALTY LAGRANGIAN  **
C##############################################################################
C
  280 IF(.NOT.IMPULS)                                             GOTO 540
C
C##############################################################################
C**  2.1:  GRADIENTS BY IMPULSIVE RESPONSE FUNCTIONS  **
C##############################################################################
      DO 290 I=1,NX
      DF(I) = ZERO
      IF(MQ.LE.0)                                                 GOTO 290
      DO 285 J=1,MQ
  285 DG(J,I) = ZERO
  290 CONTINUE
C
C**  VARIED CONTROLS FOR QUADRATURES CALLED IN RHS   **
C     THESE ARE THE DELTA-S IN EQ.(3.9)
C
      DO 310 I=1,INTERV
      KNOTI = KNOT(I)
      DO 310 J=1,KONTRL
      DO 310 K=1,KNOTI
      DO 300 L=1,KNOTI
      H = U(L,J,I)
      IF(K.EQ.L) H = H+DSIGN(DMAX1(DABS(H),PM6),H)
  300 UH(L,K,J,I) = H
  310 CALL SPLINE(KNOTI,GRID(1,I),UH(1,K,J,I),STIFF(1,I),
     *            AH(1,K,J,I),BH(1,K,J,I),CH(1,K,J,I))
C
C**  BACKWARD INTEGRATION OF STATE, COSTATE & QUADRATURES  EQ.(2.6)  **
C
      FORWRD = .FALSE.
C**  INITIAL VALUES OF COSTATE  EQ.(3.10)  **
      IF(NP.EQ.0)                                                 GOTO 325
      DO 320 I=1,NP
      FP(I)   = ZERO
      IF(M.EQ.0)                                                  GOTO 320
      DO 315 J=1,M
  315 GP(I,J) = ZERO
  320 CONTINUE
  325 DO 335 I=1,NZ
      FZ(I)   = ZERO
      IF(M.EQ.0)                                                  GOTO 335
```

```
      DO 330 J=1,M
  330 GZ(I,J) = ZERO
  335 CONTINUE
C**  ONLY THE NON ZERO VALUES OF FP, ETC. HAVE TO BE OCCUPIED  EQ.(3.10)
      CALL FCT(P,X,Z,FSTORE,GSTORE,FP,FZ,GP,GZ,1,TO,WORK)
CARDINALITY OF SET OF BOUNDARY VALUES INCLUDING COST FUNCTION
COMPRISES AUGMENTED COST FUNCTION FOR MULTIPLIER METHOD : MQP = 1
CONTAINS MQP = MQ+1 VALUES FOR ALL OTHER METHODS - UNTIL A GENIUS COMES
      MQP = MQ+1
      IF(IG.EQ.1) MQP = MQP-MQ
      L = NZ
C
CONDITIONS NECESSARY FOR OCP TO BE OPTIMAL WRT CONTROL-PARAMETERS
C
      DO 380 K=1,MQP
      DO 370 I=1,NZ
      IF(K.GT.1)                                              GOTO 360
      H = ZERO
      IF(MQP.GT.1)                                            GOTO 350
      IF(MQ .EQ.0)                                            GOTO 350
      DO 340 J=1,MQ
  340 H = H+GZ(I,J)*(G(J)-THETA(J))*SIGMA(J)
  350 H = H+FZ(I)
                                                              GOTO 370
  360 H =   GZ(I,K-1)
  370 Z(I+L) = H
  380 L = L+NZ
      LSAVE = L
C**  INITIAL VALUES OF QUADRATURES  **
      DO 440 K=1,MQP
      DO 390 I=1,NX
  390 Z(I+L) = ZERO
      IF(NP.LE.0)                                             GOTO 440
C
CONDITIONS NECESSARY FOR OCP TO BE OPTIMAL WRT DESIGN-PARAMETERS
C
      DO 430 I=1,NP
      IF(K.GT.1)                                              GOTO 420
      H = ZERO
      IF(MQP.GT.1)                                            GOTO 410
      IF(MQ .EQ.0)                                            GOTO 410
      DO 400 J=1,MQ
  400 H = H+GP(I,J)*(G(J)-THETA(J))*SIGMA(J)
  410 H = H+FP(I)
                                                              GOTO 430
  420 H =   GP(I,K-1)
  430 Z(NXMNP+I+L) = -H
  440 L = L+NX
      IF(STORB) PRINT 999
      IF(STORB) PRINT 996, TO,(Z(J),J=1,L)
C-----------------------------------------------------------------------
      KNT = 0
      KNTFI = 0
      KNTBI = 1
C-----------------------------------------------------------------------
      IFLAGS = 1
      IF (TRAP)  IFLAGS = 10
      IF (TRAP)  CALL ZWEIGE(NZ,TO,Z)
```

```
      IFLAG = 1 * IFLAGS
      IF(.NOT.KFL) IFLAG = -1 * IFLAGS
C-----------------------------------------------------------------------
      DO 480 I=1,INTERV
      KNOTI = KNOT(I)
      DO 480 J=2,KNOTI
      T1 = GRID(KNOTI-J+1,INTERV-I+1)
C
C**  SOLVING THE INITIAL VALUE PROBLEM  **
C
C-----------------------------------------------------------------------
  450 CALL SLV(RHS,SUBPHI,NPHI,L,Z,T0,T1,TOL1,TOL1,IFLAG,WORK,IWORK)
C-----------------------------------------------------------------------
      IF (IFLAG .LT. 0 .AND. T0 .NE. T1)                    GO TO 450
      IF (IFLAG .EQ. 3)                                     GO TO 450
      IF (IFLAG .GT. 0 .AND. IFLAG .LE. 8)                  GO TO 460
      IF (IFLAG .GT. 25)                                    GO TO 460
      GO TO 470
C-----------------------------------------------------------------------
  460 PRINT 997,IFLAG,FORWRD
C-----------------------------------------------------------------------
      IF (IFLAG .GT. 8)  STOP
C-----------------------------------------------------------------------
      LFL = .TRUE.
                                                            GOTO 800
  470 IF(STORB) PRINT 996, T0,(Z(K),K=1,L)
C-----------------------------------------------------------------------
      IF(IFLAG.EQ.-2 * IFLAGS)                              GOTO 450
      IF(.NOT.KFL) IFLAG = -2 * IFLAGS
C-----------------------------------------------------------------------
  480 CONTINUE
      KFL = .TRUE.
      STORB = .FALSE.
C-----------------------------------------------------------------------
      NFE = NFE + IWORK(1)
      NEXT = NEXT + IWORK(3)
C-----------------------------------------------------------------------
C
C**  EVALUATION OF GRADIENTS FROM QUADRATURES  EQ.(3.9)  **
CHANGE OF SIGN DUE TO BACKWARD INTEGRATION !!
C
      DO 530 K=1,MQP
      L = 0
      DO 500 M=1,INTERV
      KNOTI = KNOT(M)
      DO 500 I=1,KONTRL
      DO 500 J=1,KNOTI
      L = L+1
      H = X(L)
      H = DSIGN(DMAX1(DABS(H),PM6),H)
      IF(K.GT.1)                                            GOTO 490
C     EQ.(3.9A)
      DF(L)       = -Z(L+LSAVE)/H
                                                            GOTO 500
  490 DG(K-1,L)   = -Z(L+LSAVE)/H
  500 CONTINUE
      IF(NP.LE.0)                                           GOTO 530
      DO 520 I=1,NP
```

```
      IF(K.GT.1)                                                    GOTO 510
C     EQ.(3.9B)
      DF(L+I)      = -Z(L+I+LSAVE)
                                                                    GOTO 520
  510 DG(K-1,L+I) = -Z(L+I+LSAVE)
  520 CONTINUE
  530 LSAVE = LSAVE+NX
                                                                    GOTO 630
C-----------------------------------------------------------------------
C     SPACE FOR USER SUPPLIED PRINT OUT  AFTER BACKWARD INTEGRATION
C
C     KNTBI = 1 (KNTFI = KNT = 0)
C
C-----------------------------------------------------------------------
C#######################################################################
C**  2.2:  GRADIENTS BY FORWARD DIFFERENCES  **
C#######################################################################
C
  540 ASSIGN 580 TO JUMP
C**  AUTOMATIC SELECTION OF DISTURBANCE PARAMETER DEL  **
      DEL = DSQRT(1D-1*TOL1)
      I1 = 0
      I2 = 0
  550 I2 = I2+1
      KNOTI2 = KNOT(I2)
      I3 = 0
  560 I3 = I3+1
      I4 = 0
  570 I4 = I4+1
      I1 = I1+1
C
C**  VARIED CONTROLS   **
C
C-----------------------------------------------------------------------
      KNTFI = KNTFI + 1
C-----------------------------------------------------------------------
      H = X(I1)
      DH = DSIGN(DMAX1(DABS(DEL*H),PM6),H)
      X(I1) = H+DH
      U(I4,I3,I2) = H+DH
      DH = ONE/DH
      CALL SPLINE(KNOTI2,GRID(1,I2),U(1,I3,I2),STIFF(1,I2),
     *            A(1,I3,I2),B(1,I3,I2),C(1,I3,I2))
                                                                    GOTO 140
C**  VARIED FUNCTIONS   **
C
  580 DF(I1)   = (FSTORE   -F   )*DH
      IF(MQ.LE.0)                                                   GOTO 595
      DO 590 I=1,MQ
  590 DG(I,I1) = (GSTORE(I)-G(I))*DH
  595 X(I1) = H
      U(I4,I3,I2) = H
      IF(I4.LT.KNOTI2)                                              GOTO 570
      CALL SPLINE(KNOTI2,GRID(1,I2),U(1,I3,I2),STIFF(1,I2),
     *            A(1,I3,I2),B(1,I3,I2),C(1,I3,I2))
      IF(I3.LT.KONTRL)                                              GOTO 560
      IF(I2.LT.INTERV)                                              GOTO 550
      IF(NP.LE.0)                                                   GOTO 630
```

```
C
C**  VARIED DESIGN PARAMETERS    **
C
      ASSIGN 610 TO JUMP
      I1 = 0
  600 I1 = I1+1
      I5 = NXMNP+I1
      H = P(I1)
      DH = DSIGN(DMAX1(DABS(DEL*H),PM6),H)
      P(I1) = H+DH
      DH = ONE/DH
                                                              GOTO 140
C**  VARIED FUNCTIONS    **
C
  610 DF(I5)    = (FSTORE    -F    )*DH
      IF(MQ.LE.0)                                             GOTO 625
      DO 620 I=1,MQ
  620 DG(I,I5) = (GSTORE(I)-G(I))*DH
  625 P(I1) = H
C-------------------------------------------------------------------------
      KNTFI = KNTFI + 1
C-------------------------------------------------------------------------
      IF(I1.LT.NP)                                            GOTO 600
C
C#########################################################################
C**  2.3:  GRADIENTS OF CONTROL BOUNDS  **
C#########################################################################
C
  630 IF(IFL.OR.NOBOUN)                                       GOTO 800
      IFL = .TRUE.
      K = MQ+1
      DO 640 I=1,NX
      DO 640 J=K,MQ1
  640 DG(J,I) = ZERO
      L = 0
      N = 0
      M = MQ
      DO 680 I=1,INTERV
      KNOTI = KNOT(I)
      DO 680 J=1,KONTRL
      IF(.NOT.LBOUND(J))                                      GOTO 660
      DO 650 K=1,KNOTI
      L = L+1
      M = M+1
  650 DG(M,L) =  ONE/DMAX1(DABS(UL(K,J,I)),PM6)
  660 IF(.NOT.UBOUND(J))                                      GOTO 680
      DO 670 K=1,KNOTI
      N = N+1
      M = M+1
  670 DG(M,N) = -ONE/DMAX1(DABS(UU(K,J,I)),PM6)
  680 CONTINUE
                                                              GOTO 800
C#########################################################################
C**  3.0:  FORWARD INTEGRATION OF SYSTEM STATE FOR GRAPHING  **
C#########################################################################
C
  690 FORWRD = .TRUE.
      K = 1
```

```
      DO 700 I=1,KONTRL
  700 UF(K,I) = U(1,I,1)
      DO 710 I=1,NZ
      ZF(K,I) = Z0(I)
  710 Z(I) = Z0(I)
      T0 = GRID(K,K)
C-----------------------------------------------------------------------
      IFLAGS = 1
      IF (TRAP)   IFLAGS = 10
      IF (TRAP)   CALL ZWEIGE(NZ,T0,Z)
      IFLAG = IFLAGS
C-----------------------------------------------------------------------
      T2 = T0
      TF = GRID(KNOT(INTERV),INTERV)
      T3 = TF
      DT = (TF-T0)/DFLOAT(ID)
      IOUT = 0
      JOUT = 0
      CALL OUTPUT(T0,Z,WORK)
      IF(IOUT.LE.0)                                          GOTO 720
      DO 715 I=1,IOUT
  715 HF(K,I)=HFF(I)
  720 K = K+1
      T1 = T0+DT
C
C**  SOLVING THE INITIAL VALUE PROBLEM  **
C
C-----------------------------------------------------------------------
  730 CALL SLV(RHS,SUBPHI,NPHI,NZ,Z,T0,T1,TOL1,TOL1,IFLAG,WORK,IWORK)
      IWORK(4) = 0
C-----------------------------------------------------------------------
      IF(IOUT.LE.0)                                          GOTO 736
      CALL OUTPUT(T1,Z,WORK)
      DO 734 I=1,IOUT
  734 HF(K,I)=HFF(I)
  736 DO 740 I=1,NZ
  740 ZF(K,I) = Z(I)
      DO 750 I=1,INTERV
      J = I
      IF(T0.LE.GRID(KNOT(I),I))                              GOTO 760
  750 CONTINUE
  760 TT = DMIN1(T0,TF)
      DO 770 I=1,KONTRL
  770 UF(K,I) = SPLINT(KNOT(J),GRID(1,J),U(1,I,J),STIFF(1,J),
     *                 A(1,I,J),B(1,I,J),C(1,I,J),TT)
      IF(T1.LT.TF-PM6)                                       GOTO 720
C
C**  STORE STATES AND CONTROLS FOR GRAPHING VIA PLTF (PLI-JOBSTEP)  **
C
      DO 780 I=1,NZ
  780 CALL BILD(1,K,ZF(1,I),ORD0,ORD1,ORD,ABS,ABST,NORD,NABS,NUL,OUT,NN)
      DO 790 I=1,KONTRL
  790 CALL BILD(1,K,UF(1,I),ORD0,ORD1,ORD,ABS,ABST,NORD,NABS,NUL,OUT,NN)
      IF(IOUT.LE.0)                                          GOTO 796
      DO 794 I=1,IOUT
  794 CALL BILD(1,K,HF(1,I),ORD0,ORD1,ORD,ABS,ABST,NORD,NABS,NUL,OUT,NN)
C
C**  GRAPHING ON LINE PRINTER  **
```

```
C
  796 NZ1=MIN0(NZ,10)
      CALL GRAPH(ZF,402,10,   NZ1,K,SAMESC,NGRID,T2,T3,NPLOT,LONG)
      CALL GRAPH(UF,402,05,KONTRL,K,SAMESC,NGRID,T2,T3,NPLOT,LONG)
      IF(IOUT.LE.0)                                          GOTO 800
      CALL GRAPH(HF,402,05,  IOUT,K,SAMESC,NGRID,T2,T3,NPLOT,LONG)
  800 RETURN
C
C### NON EXECUTABLE STATEMENTS ##########################################
C
  996 FORMAT(1H ,1P13D10.2)
  997 FORMAT(20H ***  TOMP :  FLAG = ,I2,L1,5H  ***)
  998 FORMAT('1STATE   VECTOR :')
  999 FORMAT('1COSTATE VECTOR :')
      END
```

APPENDIX B. PROGRAM LISTING FOR OCP APPLICATION

The following driving program and subroutines have been used in the solution of the state/control-constraint optimal control problem presented in this report. Computations have been performed in double precision.

The driving program gives the initial data for the problem and references the TROMP/RKF45T system and the SLLSQP package to solve the OCP.

FK1 — which evaluates the cost function, f, and constraint violations, g.

KRODE — which evaluates the right hand sides of the differential equations.

KRPHI — which evaluates the PHI functions, i.e., the vector of stopping conditions for the integration package and which permits updates in the differential equation branching as the discontinuities are isolated.

WARN — which prints a warning message if the incorrect branch of the ODE is being used.

ZWEIGE — which determines the branches to be used for the initial integration step.

The Driving program:

```
C-----------------------------------------------------------------------
      IMPLICIT REAL*8 (A-H,O-Z)
C-----------------------------------------------------------------------
C     COMMON BLOCKS:
C-----------------------------------------------------------------------
      COMMON /CTROMP/
     1  GRID(15,5),STIFF(15,5),U(15,5,5),A(15,5,5),B(15,5,5),C(15,5,5),
```

```
     2  UL(15,5,5),UU(15,5,5),UH(15,15,5,5),AH(15,15,5,5),BH(15,15,5,5),
     3  CH(15,15,5,5),P(10),ZO( 6),TOL,INTERV,KONTRL,KNOT(5),NP,NZ,
     4  MQP,IG,NPHI,LFL,
     5  IMPULS,FORWRD,STORE,STORB,LPLOT,LBOUND(5),UBOUND(5),LFCT,TRAP
C
      COMMON/CRKF45/IOPT,IDUM45(4)
      COMMON/IDENT/ITER8,KDUMM(3),NFE,NEXT
      COMMON/PLOTTD/TIME(500),APT(500),AMPT(500),DT,NPT,IPLOT
      COMMON/FINALT/TFINAL
      COMMON/CCFMIN/DUMMY(15),IGRD,KFLAG
      COMMON/FKOUNT/KF
C-----------------------------------------------------------------------
      DIMENSION XX(15),AZ(15),STIFFF(15)
      DIMENSION X(16),DF(16),G(40),DG(40,18)
      DIMENSION GY(10,99),GZ(20,99)
      DIMENSION XL(16),XU(16),W(5000),INDEX(100)
      REAL AA(500,2),TMAX,TMIN
C
      LOGICAL IMPULS,FORWRD,STORE,STORB,LPLOT,LBOUND,UBOUND,TRAP,LFCT
     1        ,LFL
C-----------------------------------------------------------------------
C     DATA STATEMENTS:
C-----------------------------------------------------------------------
      DATA XX/.0000000000000000D+00,.3703703703703704D-02,
     1        .1851851851851852D-01,.3703703703703704D-01,
     2        .7407407407407407D-01,.1111111111111111D+00,
     3        .2222222222222222D+00,.3703703703703704D+00,
     4        .4629629629629630D+00,.5555555555555556D+00,
     5        .6666666666666666D+00,.7777777777777777D+00,
     6        .8518518518518519D+00,.9259259259259259D+00,
     7       1.0000000000000000D+00/
      DATA  AZ/.0000000000000000D+00,1.0000000000000000D+00,
     1         5.0000000000000000D+00,10.000000000000000D+00,
     2         12.000000000000000D+00,15.000000000000000D+00,
     3         10.000000000000000D+00,80.000000000000000D+00,
     4         100.00000000000000D+00,120.00000000000000D+00,
     5         180.00000000000000D+00,220.00000000000000D+00,
     6         230.00000000000000D+00,240.00000000000000D+00,
     7         250.00000000000000D+00/
      DATA STIFFF/0.0D0,0.0D0,0.0D0,0.0D0,0.0D0,
     1            0.0D0,0.0D0,0.0D0,0.0D0,0.0D0,
     2            0.0D0,0.0D0,0.0D0,0.0D0,0.0D0/
      DATA XU/0.10D0,10.0D0,50.0D0,95.0D0,12*300.D0/
      DATA XL/13*0.0D0,-300.D0,-300.D0,-300.D0/
      DATA NN/15/
      DATA F/1.D1/,LW/5000/
C-----------------------------------------------------------------------
      EXTERNAL KRODE,KRPHI,FK1,ZWEIGE,RKF45T
C-----------------------------------------------------------------------
C     INITIALIZATION OF CONSTANTS FOR PLOTTING ALPHA AND DERIVATIVE
C     EVALUATION COUNTERS:
C-----------------------------------------------------------------------
      KF    = 0
      NEXTT = 0
      NFET  = 0
      IPLOT = 0
      NPT   = 0
C-----------------------------------------------------------------------
```

```
C     INITIALIZATION FOR TROMP:
C-----------------------------------------------------------------------
      INTERV =  1
      KONTRL =  1
      KNOT(1)= 15
      NP     =  0
      IMPULS = .FALSE.
      STORE  = .TRUE.
      STORB  = .FALSE.
      LPLOT  = .FALSE.
      LFCT   = .FALSE.
      LBOUND(1) = .FALSE.
      UBOUND(1) = .FALSE.
      TOL    = 1.D-05
      M      = 4
      L      = 4
      IGRD = 1
      IG = 0
C-----------------------------------------------------------------------
C     INITIAL VALUES--Z, GRID, AND U--NO VALUES FOR BOUNDL, BOUNDU
C                                     OR DESIGN PARAMETERS
C-----------------------------------------------------------------------
      Z0(1) =  58.6D0
      Z0(2) = 90.0D0 * 3.141592653589793D0 / 180.D0
      Z0(3) = 8.79D0 * DCOS(Z0(2))
      Z0(4) =-8.79D0 * DSIN(Z0(2))
      Z0(5) = 0.0D0
      Z0(6) = 0.0D0
      NZ = 6
      NPHI = 3
      TFINAL = 3.2D0
      TRAP = .TRUE.
C
      DO 10 J = 1,NN
      GRID(J,1) = XX(J)
      STIFF(J,1) = STIFFF(J)
      U(J,1,1)  = AZ(J)
   10 CONTINUE
C-----------------------------------------------------------------------
C     WRITE INITIAL CONDITIONS
C-----------------------------------------------------------------------
      PRINT 500,INTERV,KONTRL,KNOT(1),NP,NZ,IMPULS,STORE,STORB,
     1          LPLOT,LBOUND(1),UBOUND(1),LFCT,TOL,M
      PRINT 499
      PRINT 501,(J,Z0(J),J=1,NZ)
      PRINT 499
      PRINT 502,(J,STIFF(J,1),J=1,NN)
      PRINT 499
      PRINT 503,(J,GRID(J,1),J=1,NN)
      PRINT 499
      PRINT 504,(J,U(J,1,1),J=1,NN)
C
C-----------------------------------------------------------------------
C     PARAMETERS FOR THE SLLSQP PACKAGE:
C-----------------------------------------------------------------------
      NX = 15
      ME = L - 1
      LDG = 40
```

```
      ACC = 1.D-04
      MAXIT =   50
      MODE = 0
C-----------------------------------------------------------------------
C     WRITE SLLSQP PRAMETERS:
C-----------------------------------------------------------------------
      PRINT 1506,MODE,MAXIT,NX,ME,LDG,(J,XL(J),J,XU(J),J=1,NX)
C-----------------------------------------------------------------------
C     COPY CONTROL PARAMETERS INTO X FOR TOMP CALLING SEQUENCE
C-----------------------------------------------------------------------
      N  = KNOT(1)
      DO 15 I = 1,N
   15 X(I) = U(I,KONTRL,INTERV)
C-----------------------------------------------------------------------
C     NO. DESIGN PARAMETERS--X VECTOR CONTAINS ONLY CONTROL VALUES
C-----------------------------------------------------------------------
C
C-----------------------------------------------------------------------
C     CHOOSE MINIMIZER--SLLSPQ, PRINT HEADING:
C-----------------------------------------------------------------------
      PRINT 505
C-----------------------------------------------------------------------
C
  130 CONTINUE
      NPHI = 3
      CALL TROMP(N,M,M,X,F,DF,G,DG,LDG,
     1          FK1,KRODE,RKF45T,KRPHI,ZWEIGE)
C
      NFET  = NFET  + NFE
      NEXTT = NEXTT + NEXT
C
  132 CONTINUE
      CALL SLLSQP(M,ME,LDG,NX, X, XL, XU, F, G, DF, DG, ACC,
     1            MAXIT, MODE, W, LW, INDEX)
C
      IF (MODE .EQ. 0)   GO TO 150
      IF (MODE .GT. 1)   GO TO 150
      IF (MODE .EQ. -1)  GO TO 145
C-----------------------------------------------------------------------
C     MODE = 1   SET IG = 0 FOR COST FUNCTION EVALUATION
C-----------------------------------------------------------------------
      IG = 0
C     PRINT 558,MODE
      GO TO 130
  145 CONTINUE
C-----------------------------------------------------------------------
C     MODE = 2   SET IG = 2 FOR GRADIENT EVALUATION
C-----------------------------------------------------------------------
      IG = 2
C     PRINT 558,MODE
      GO TO 130
  150 CONTINUE
C-----------------------------------------------------------------------
C     EXIT ITERATION LOOP:  MODE = 0 IMPLIES CONVERGENCE
C-----------------------------------------------------------------------
      PRINT 558,MODE
      PRINT 559,(J,X(J),J=1,N)
      PRINT 560,F
```

```
C
  200 CONTINUE
C
      PRINT 1548,KF,NEXT
      PRINT 1548,NFET,NEXTT
 1548 FORMAT(///,' NUMBER OF DERIVATIVE EVALUATIONS:  TOTAL NFE = ',I6,
     1    /,'                                          NEXTRA    = ',I6,//)
C
C-----------------------------------------------------------------------
C     SET UP FOR PLOTTING RESULTS:
C     (IPLOT ACTIVATES STORAGE IN KRPHI FOR PLOTTING ALPHA AND ALPHAM)
C     (DT IS THE INCREMENT FOR POINTS IN TROMP)
C-----------------------------------------------------------------------
      PRINT 506
      STORE = .TRUE.
      LPLOT = .TRUE.
      IPLOT = 1
      DT = 1.D0/400.D0
      IG = 0
      CALL TROMP(N,M,M,X,F,DF,G,DG,LDG,
     1          FK1,KRODE,RKF45T,KRPHI,ZWEIGE)
C
      DO 400 J = 1,NPT
      AA(J,1) = SNGL(APT(J))
  400 AA(J,2) = SNGL(AMPT(J))
      TMAX = 1.0
      TMIN = 0.0
      I1 = 500
      I2 = 2
      NSET = 2
      N = NPT
      NGRID = 80
      NPLOT =  8
      LONG = 53
      CALL GRAPH(AA,I1,I2,NSET,N,.TRUE.,NGRID,TMIN,TMAX,NPLOT,LONG)
C
C-----------------------------------------------------------------------
C     FORMAT STATEMENTS:
C-----------------------------------------------------------------------
  500 FORMAT(//,' NUMBER OF INTERVALS FOR GRID = ',I3,/,
     1       ' NUMBER OF CONTROL PARAMETERS = ',I3,/,
     2       ' NUMBER OF KNOTS              = ',I3,/,
     3       ' NUMBER OF DESIGN PARAMETERS  = ',I3,/,
     4       ' NUMBER OF DEPENDENT VARIABLES= ',I3,/,
     5       ' LOGICAL PARAMETERS--IMPULS = ',L2,/,
     6   21X,'STORE  = ',L2,2X,' STORB  = ',L2,/,21X,'LPLOT  = ',L2,/,
     7   21X,'LBOUND = ',L2,2X,' UBOUND = ',L2,/,21X,'LFCT   = ',L2,/,
     8       ' INTEGRATION TOLERANCE        = ',D15.7,/,
     9       ' NUMBER OF CONSTRAINTS        = ',I3,/)
  499 FORMAT(/)
  501 FORMAT(17X,' Z0(',I2,') = ',D15.7)
  502 FORMAT(3X,' STIFFNESS FACTOR(',I2,') = ',D15.7)
  503 FORMAT(15X,' GRID(',I2,') = ',D15.7)
  504 FORMAT(2X,' CONTROL PARAMETER(',I2,') = ',D15.7)
  505 FORMAT(//,' MINIMIZER: SLLSQP',/,
     1  '            SEQUENTIAL LINEAR LEAST SQUARES PROGRAMMING',/)
  506 FORMAT(///,' FORWARD INTEGRATION BEFORE PLOTTING USING',
     1           ' OPTIMAL TRAJECTORY:',/)
```

```
  558 FORMAT(/,' MODE = ',I4)
  559 FORMAT(/,' AFTER RETURN FROM SLLSQP:',/,(' X(',I2,') = ',D15.7))
  560 FORMAT(//,' F = ',D15.7,//)
 1506 FORMAT(//,3X,' PARAMETERS FOR SLLSQP:',/,3X,
     1   '                                  MODE  = ',I4,/,3X,
     2   '                                 MAXIT  = ',I4,/,3X,
     3   ' NUMBER OF CONTROL GRID POINTS,  NX  = ',I4,/,3X,
     4   ' NUMBER OF EQUALITY CONSTRAINTS, ME  = ',I4,/,3X,
     5   ' LEADING DIMENSION OF DG MATRIX, LDG = ',I4,//,3X,
     6   ' BOUNDS ON CONTROL VALUES:',//,
     7    (5X,'XL(',I3,') = ',D15.7,3X,'XU(',I3,') = ',D15.7))
C
      STOP
      END
```

Subroutine FK1 for evaluating the cost function and constraint violation:

```
C
C-----------------------------------------------------------------------
      SUBROUTINE FK1(Y,X,Z,F,G,FY,FZ,GY,GZ,IFLAG,T,WORK)
C-----------------------------------------------------------------------
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION
     1         G(1),Y(1),X(1),Z(1),FY(1),FZ(1),GY(10,99),GZ(20,99),
     2         WORK(1)
      DIMENSION YP(6)
      DATA   VF/800.D0/,GAMF/-.174533D0/,ZF/-160.D0/,XF/1500.D0/
C
      COMMON /CTROMP/
     1  GRID(15,5),STIFF(15,5),U(15,5,5),A(15,5,5),B(15,5,5),C(15,5,5),
     2  UL(15,5,5),UU(15,5,5),UH(15,15,5,5),AH(15,15,5,5),BH(15,15,5,5),
     3  CH(15,15,5,5),P(10),Z0( 6),TOL,INTERV,KONTRL,KNOT(5),NP,NZ,
     4  MQP,IG,NPHI,LDUM(18)
C
      LOGICAL LDUM
C
C-----------------------------------------------------------------------
C     CALL DERIVATIVE ROUTINE:
C-----------------------------------------------------------------------
      CALL KRODE(T,Z,YP)
C
C-----------------------------------------------------------------------
C     IFLAG = 0   EVALUATE COST FUNCTION AND CONSTRAINTS
C-----------------------------------------------------------------------
C     COST FUNCTION:  MAXIMUM VELOCITY
C
      F = -Z(1)/VF
C
C-----------------------------------------------------------------------
C     RANDWERTE
C-----------------------------------------------------------------------
      G(1) = (Z(2)-GAMF)/GAMF
      G(2) = (Z(4)-ZF)/ZF
      G(3) = (Z(3)-XF)/XF
      G(4) = -Z(5) + 1.D-04
```

```
      RETURN
      END
```

Subroutine KRODE for evaluating the ODE:

```
C
C-----------------------------------------------------------------------
      SUBROUTINE KRODE(T,Y,YP)
C-----------------------------------------------------------------------
C
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION Y(6),YP(6)
      COMMON/FKOUNT/KF
      COMMON/FINALT/TOUT
      COMMON/BRANCH/IZWEIG(3)
      COMMON/PHILIM/ALPHA,DALDT,ALPHAM,DALMDT
C
      COMMON /CTROMP/
     1  GRID(15,5),STIFF(15,5),U(15,5,5),A(15,5,5),B(15,5,5),C(15,5,5),
     2  UL(15,5,5),UU(15,5,5),UH(15,15,5,5),AH(15,15,5,5),BH(15,15,5,5),
     3  CH(15,15,5,5),P(10),Z0( 6),TOL,INTERV,KONTRL,KNOT(5),NP,NZ,
     4  MQP,IG,NPHI,LDUM(18)
C
      LOGICAL LDUM
      DATA KNOT1/15/
      DATA CALPHA/-2.33D-03/
      DATA TF,F0,FSTR,XM0,CHI/2.7D0,29.9D+03,7.9D+03,137.2D0,
     1                        -0.4133D-03/
      DATA CW0,CWAA,RHO,S/0.3D0,0.7D0,1.22575D0,0.0314D0/
C
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
C      T IS A NORMALIZED VALUE.  IF CORRECT TIME IS NEEDED IN ODE
C      TIME = TOUT*T
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
C     V = Y(1)
C     GAMMA = Y(2)
C     X = Y(3)
C     Z = Y(4)
C     CONSTRAINT VIOLATION ODE:  Y(5)
C     GAMMA" = Y(6)
C
C     DERIVATIVES ARE NORMALIZED AT THE END OF KRODE.  TERMS IN THE
C          ODE EXPRESSIONS ARE WRITTEN IN TERMS OF TIME NOT NORMALIZED
C          TIME.
C
C     D--DT DESIGNATES DERIVATIVE WRT TIME, E.G.,DVDT,DAZMDT, ETC.
C
C-----------------------------------------------------------------------
C     DERIVATIVE COUNTER
C-----------------------------------------------------------------------
      KF = KF + 1
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
C     COMPUTE PARAMETERS NEEDED FOR EVALUATING THE ODE
C-----------------------------------------------------------------------
```

```
C-----------------------------------------------------------------------
C
C-----------------------------------------------------------------------
C     COMPUTE ALPHA, AZ, ALPHAM WITH AZ REPRESENTED AS A CUBIC SPLINE
C-----------------------------------------------------------------------
C
      AZ    = SPLINT(KNOT1,GRID(1,1),U(1,1,1),STIFF(1,1),
     1               A(1,1,1),B(1,1,1),C(1,1,1),T)
      DAZDT = DSPLNT(KNOT1,GRID(1,1),U(1,1,1),STIFF(1,1),
     1               A(1,1,1),B(1,1,1),C(1,1,1),T)
      DAZDT = DAZDT/TOUT
C
      TIME = TOUT*T
      V = Y(1)
      GAMMA = Y(2)
      ALPHA = AZ/(CALPHA*V*V)
C-----------------------------------------------------------------------
C     DETERMINE THRUST AND MASS--DOUBLE BRANCH FUNCTION IZWEIG(2):
C-----------------------------------------------------------------------
      IF (IZWEIG(2) .EQ. 2)   GO TO 10
C-----------------------------------------------------------------------
C     T < TF AT BEGINNING OF STEP--USE BRANCH 1 OF F AND XMASSE
C       =                           EXPRESSIONS
C-----------------------------------------------------------------------
      F = F0 + FSTR * TIME
      DFDT = FSTR
      XMASSE = XM0 + CHI * (F0 + 0.5D0 * FSTR * TIME) * TIME
      DXMDT  = CHI*(F0 + FSTR*TIME)
      GO TO 12
   10 CONTINUE
C-----------------------------------------------------------------------
C     T > TF AT BEGINNING OF STEP--USE BRANCH 2 OF F AND XMASSE
C                                 EXPRESSIONS
C-----------------------------------------------------------------------
C
      F = 0.0D0
      XMASSE = XM0 + CHI * (F0 + 0.50D0 * FSTR * TF) * TF
      DFDT = 0.0D0
      DXMDT = 0.0D0
   12 CONTINUE
C-----------------------------------------------------------------------
C     COMPUTE DRAG:
C-----------------------------------------------------------------------
      CW = CW0 + CWAA * ALPHA * ALPHA
      WIDER = 0.50D0 * CW * RHO * V * V * S
C-----------------------------------------------------------------------
C     COMPUTE AX USING THRUST, DRAG, AND MASS:
C-----------------------------------------------------------------------
      AX = (F - WIDER)/XMASSE
C-----------------------------------------------------------------------
C     DETERMINE AZM--TRIPLE BRANCH FUNCTION IZWEIG(3):
C-----------------------------------------------------------------------
C
      IF (IZWEIG(3) .EQ. 1) AZM = 0.002D0 * V * V
      IF (IZWEIG(3) .EQ. 2) AZM = 45.0D0 + 0.660D0 * (V - 150.0D0)
      IF (IZWEIG(3) .EQ. 3) AZM = 300.0D0
C-----------------------------------------------------------------------
C     COMPUTE ALPHAM
```

```
C-----------------------------------------------------------------------
      ALPHAM = -AZM /(CALPHA * V * V)
C
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
C     EVALUATE ODE  WRT  TIME:
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
      COSA = DCOS(ALPHA)
      SINA = DSIN(ALPHA)
      COSG = DCOS(GAMMA)
      SING = DSIN(GAMMA)
C
      YP(1) =  AX * COSA + AZ * SINA
      YP(2) = (AX * SINA - AZ * COSA) / V
      YP(3) =  V * COSG
      YP(4) = -V * SING
      IF (IZWEIG(1) .EQ. 1) YP(5) = 0.0D0
      IF (IZWEIG(1) .EQ. 2) YP(5) = (ALPHAM - DABS(ALPHA))**2
C-----------------------------------------------------------------------
C     COMPUTE D ALPHA / DT  AND D ALPHAM /DT   AND TERMS NEEDED
C         FOR COMPUTING YP(6):
C-----------------------------------------------------------------------
      DVDT = YP(1)
      DALDT = DAZDT/(CALPHA*V*V) -2.0D0*AZ/(CALPHA*V*V*V) * DVDT
      IF (IZWEIG(3) .EQ. 1) DAZMDT = 0.004D0 * V * DVDT
      IF (IZWEIG(3) .EQ. 2) DAZMDT = 0.66D0 * DVDT
      IF (IZWEIG(3) .EQ. 3) DAZMDT = 0.0D0
      DALMDT = -DAZMDT/(CALPHA*V*V) +2.0D0*AZM/(CALPHA*V*V*V)*DVDT
      DWIDDT = CW*RHO*S*V* DVDT  + RHO*V*V*S*(CWAA*ALPHA*DALDT)
      DAXDT=(DFDT - DWIDDT)/XMASSE - (F - WIDER)/(XMASSE*XMASSE)*DXMDT
C-----------------------------------------------------------------------
C     COMPUTE SECOND DERIVATIVE OF GAMMA
C-----------------------------------------------------------------------
      YP(6) = (DAXDT*SINA - DAZDT*COSA + (DALDT - YP(2))*DVDT) / V
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
C     COMPUTE DERIVATIVES WRT NORMALIZED TIME:
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
C
      YP(1) = YP(1)*TOUT
      YP(2) = YP(2)*TOUT
      YP(3) = YP(3)*TOUT
      YP(4) = YP(4)*TOUT
      YP(5) = YP(5)*TOUT
      YP(6) = YP(6)*TOUT
C
      RETURN
      END
```

Subroutine KRPHI for evaluating the PHI components and udating the ODE:

```
C
C-----------------------------------------------------------------------
C     PHI SUBROUTINE FOR KRAFT PROBLEM
C-----------------------------------------------------------------------
C
```

```
      SUBROUTINE KRPHI(NPHI,INDEX,NEQN,T,Y,YP,PHI,PHIP,KOUNTR,UPDATE,
     1                 IVAN,BOUNCE,ABSER)
C
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION Y(NEQN),YP(NEQN),PHI(NPHI),PHIP(NPHI)
      DATA TF/2.7D0/
      DATA IPRINT/0/
      DATA ZAPP/1.D-10/
      DATA MODE1/1/,MODE2/2/,MODE4/4/
C
      COMMON/FKOUNT/KF
      COMMON/FINALT/TOUT
      COMMON/PLOTTD/TTTT(500),APT(500),AMPT(500),DT,NPT,IPLOT
      COMMON/VBD/V,VLOWER,VUPPER
      COMMON/BRANCH/IZWEIG(3)
      COMMON/PHILIM/ALPHA,DALDT,ALPHAM,DALMDT
C
      LOGICAL UPDATE,IVAN,BOUNCE
C
      IF (UPDATE)  GO TO 100
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
C     INTEGRATION IN PROGRESS:
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
      IF (KOUNTR .GT. 0)  GO TO 8
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
C     INITIALIZATION BLOCK:
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
      NPT = 0
      DO 105  J = 1,500
      APT(J) = 0.0D0
      AMPT(J) = 0.0D0
  105 CONTINUE
C
      CALL WARN(MODE1,PHI,NPHI,T,Y,NEQN,INDEX)
C
      IF (IPRINT .EQ. 1) PRINT 1544,IZWEIG(1),IZWEIG(2),IZWEIG(3)
 1544 FORMAT(' INITIAL BRANCHES:  ',3(1X,I3))
C-----------------------------------------------------------------------
C     SET BOUNDS FOR PHI(3)  (VELOCITY RELATED DISCONTINUITY):
C-----------------------------------------------------------------------
      IZ3 = IZWEIG(3)
      GO TO (5,6,7), IZ3
    5 CONTINUE
C
C     IZWEIG(3) = 1:
C
      VLOWER = 0.0D0
      VUPPER = 150.0D0
      VSCALE = 150.0D0
      GO TO 8
    6 CONTINUE
C
C     IZWEIG(3) = 2:
C
```

```
      VLOWER = 150.0D0
      VUPPER = 536.3636363636364D0
      VSCALE = 536.3636363636364D0
      GO TO 8
    7 CONTINUE
C
C     IZWEIG(3) = 3:
C
      VLOWER = 536.3636363636364D0
      VLOWER = 4000.0D0
      VSCALE = 536.3636363636364D0
C
    8 CONTINUE
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
C     EVALUATE PHI COMPONENTS
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
      V = Y(1)
      DVDTAU = YP(1)
C
      PHI(1) =  ALPHAM*ALPHAM - ALPHA*ALPHA
      PHI(2) =  T * TOUT - TF
      PHI(3) = (VUPPER - V)*(V - VLOWER)/VSCALE
C
      PHIP(1) = 2.0D0*(ALPHAM*DALMDT - ALPHA*DALDT)*TOUT
      PHIP(2) = TOUT
      PHIP(3) = DVDTAU * (-2.D0*V + VUPPER + VLOWER) / VSCALE
C-----------------------------------------------------------------------
C     CALL WARN TO CHECK BRANCHING:
C-----------------------------------------------------------------------
      IF (KOUNTR .EQ. 0)  CALL WARN(MODE4,PHI,NPHI,T,Y,NEQN,INDEX)
      CALL WARN(MODE2,PHI,NPHI,T,Y,NEQN,INDEX)
C
      IF (IPLOT .EQ. 0)  RETURN
C-----------------------------------------------------------------------
C     SAFE ALPHA AND ALPHAM VALUES FOR PLOTTING (IF AT A PLOTTING POINT)
C-----------------------------------------------------------------------
      IF (INDEX .GT. 0)  RETURN
      IF (IZWEIG(1) .EQ. 1 .AND. DABS(ALPHA) .GT. ALPHAM)  RETURN
      IF (IZWEIG(1) .EQ. 2 .AND. DABS(ALPHA) .LT. ALPHAM)  RETURN
      TTT = DFLOAT(NPT)*DT
      IF (DABS(T - TTT)  .GT.  1.D-06)    RETURN
      NPT = NPT + 1
      APT(NPT) = DABS(ALPHA)
      AMPT(NPT)= ALPHAM
      TTTT(NPT)= T
C
      RETURN
C
  100 CONTINUE
C
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
C     UPDATE PORTION
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
C
```

```
      IF (IVAN)   RETURN
      IF (BOUNCE)  GO TO 200
C
      GO TO (110,120,130),  INDEX
C
  110 CONTINUE
C
C-----------------------------------------------------------------------
C     DISCONTINUITY IN CONSTRAINT ODE ISOLATED:  ALPHA = ALPHAM
C-----------------------------------------------------------------------
C     SIGN OF PHI(1) REFELCTS THE NEW BRANCH:
C
C     IF |ALPHA| < |ALPHAM|, PHI(1) > 0 AND CONSTRAINT ODE
C                            SHOULD NOT BE ACTIVE ===> IZWEIG(1) = 1
C     IF |ALPHA| > |ALPHAM|, PHI(1) < 0 AND CONSTRAINT ODE
C                            SHOULD BE ACTIVE     ===> IZWEIG(1) = 2
C              PHI(1) WILL NOT BE EXACTLY ZERO.
C
      IF (PHI(1) .GT. 0.0D0)  IZWEIG(1) = 1
      IF (PHI(1) .LT. 0.0D0)  IZWEIG(1) = 2
C
      IF (IPRINT .EQ. 1)       PRINT 1515,ALPHA,ALPHAM,IZWEIG(1)
C
C     THE DIFFERENTIAL EQUATIONS HAVE BEEN CHANGED--REQUIRING NEW YP
C
      CALL KRODE(T,Y,YP)
      KF = KF - 1
      RETURN
C
  120 CONTINUE
C
C-----------------------------------------------------------------------
C     DISCONTINUITY IN F(T) AND M(T) ISOLATED:  TIME = TF = 2.7
C-----------------------------------------------------------------------
C     SIGN OF PHI(2) REFELCTS THE NEW BRANCH:
C
C              PHI(2) < 0  IF TIME < TF ==> IZWEIG(2)=1
C              PHI(2) > 0  IF TIME > TF ==> IZWEIG(2)=2
C              PHI(2) WILL NOT BE EXACTLY ZERO.
C
      IF (PHI(2) .LT. 0.0D0)  IZWEIG(2) = 1
      IF (PHI(2) .GT. 0.0D0)  IZWEIG(2) = 2
C
C     THE DIFFERENTIAL EQUATIONS HAVE BEEN CHANGED--REQUIRING NEW YP
C
      CALL KRODE(T,Y,YP)
      KF= KF - 1
C
      TIME = T*TOUT
      IF (IPRINT .EQ. 1)       PRINT 1511,T,TIME,IZWEIG(2)
      RETURN
C
  130 CONTINUE
C-----------------------------------------------------------------------
C     DISCONTINUITY IN AZM ISOLATED:  AZM = 45 OR  AZM = 300
C     MUST SHIFT VUPPER AND VLOWER BOUNDS: (0, 150), (150, 536.36), OR
C                                          (536.36, ABSURD LIMIT)
C-----------------------------------------------------------------------
```

```
C     SIGN OF PHI(3) REFELCTS THE NEW BRANCH:
C
C     BOUNDARY HAS BEEN REACHED:  |(VUPPER-V)*(V-VLOWER)| < TOLERANCE
C
C     VUPPER AND VLOWER MUST BE CHANGED TO REFLECT THE NEW REGION:
C
C     IF  D(PHI(3)/DV > 0      IZWEIG(3)    =  IZWEIG(3)    - 1
C                                     NEW            OLD
C
C     IF  D(PHI(3)/DV < 0      IZWEIG(3)    =  IZWEIG(3)    + 1
C                                     NEW            OLD
C
C-----------------------------------------------------------------------
C     CLARIFICATION:
C
C     PHI(3), AS A FUNCTION OF V, IS A PARABOLA WITH NOSE UP.
C     THE DERIVATIVE OF PHI(3) WRT V, INDICATES WHETHER THE
C     BRANCH NUMBER INCREASES OR DECREASES.
C
C     EXAMPLE WITH BRANCH BEFORE UPDATE:  IZWEIG(3) = 2
C
C     IF D(PHI(3))/DV > 0,  YOU ARE AT THE "LEFT"  BOUNDARY,
C                           NEW BRANCH:  IZWEIG(3) = 2 - 1  = 1
C
C     IF D(PHI(3))/DV < 0,  YOU ARE AT THE "RIGHT" BOUNDARY,
C                           NEW BRANCH:  IZWEIG(3) = 2 + 1  = 3
C
C
C    EXAMPLE: BRANCH BEFORE UPDATE, IZWEIG(3)=2
C
C
C          POSITIVE SLOPE      .          .  NEGATIVE SLOPE
C                           .    *    *    .
C                         .   *          *   .
C                       .   *              *   .
C    +------------------0-----------------------0-
C      BRANCH 1        .       BRANCH 2            .      BRANCH 3
C                    .                                .
C                  .   *                            *   .
C                .                                       .
C              .    *                                 *     .
C
C-----------------------------------------------------------------------
C
      SLOPE = (-2.D0*V + VUPPER + VLOWER) / VSCALE
      ISGN = +1
      IF  (SLOPE .GT. 0)   ISGN = -1
      IZWEIG(3) = IZWEIG(3) + ISGN
C
      IZW = IZWEIG(3)
      GO TO (131,132,133),IZW
C
  131 CONTINUE
C
C     IZWEIG(3) = 1   AFTER UPDATE
C
      VLOWER = 0.0D0
      VUPPER = 150.0D0
```

```
      VSCALE = 150.0D0
      GO TO 135
C
  132 CONTINUE
C
C     IZWEIG(3) = 2   AFTER UPDATE
C
      VLOWER = 150.D0
      VUPPER = 536.36363636364D0
      VSCALE = 536.36363636364D0
      GO TO 135
C
  133 CONTINUE
C
C     IZWEIG(3) = 3   AFTER UPDATE
C
      VLOWER = 536.36363636364D0
      VUPPER = 4000.0D0
      VSCALE = 536.36363636364D0
C
  135 CONTINUE
C
C     THE DIFFERENTIAL EQUATIONS HAVE BEEN CHANGED--REQUIRING NEW YP
C
      CALL KRODE(T,Y,YP)
      KF = KF - 1
      PHI(INDEX) = + ZAPP
C
      IF (IPRINT .EQ. 1)       PRINT 1512,T,V,IZWEIG(3)
      RETURN
C
  200 CONTINUE
C-----------------------------------------------------------------------
C     A BOUNCING FUNCTION HAS OCCURRED: SPECIAL ANALYSIS IS NEEDED
C     THE CORRECT BRANCH OF THE ODE WAS NOT USED.  THE PROPER BRANCH
C     HAS BEEN IDENTIFIED AND THE STEP WILL BE REPEATED.
C     (KOUNTR=-2 WILL CAUSE STEP TO BE REPEATED.)
C-----------------------------------------------------------------------
C
      GO TO (205,210,215),INDEX
C
  205 CONTINUE
C
C     BOUNCED ON AN ALPHA BOUNDARY
C
      PRINT 1523,T
 1523 FORMAT(//,' BOUNCED ON AN |ALPHA| = |ALPHAM| BOUNDARY AT T=',
     1          D15.7,//)
C
C     CHANGE BRANCH:
C
      IF (IZWEIG(1) .EQ. 1)   ISGN = +1
      IF (IZWEIG(1) .EQ. 2)   ISGN = -1
      IZWEIG(1) = IZWEIG(1) + ISGN
      GO TO 230
C
  210 CONTINUE
C
```

```
C     BOUNCED ON A TIME BOUNDARY
C
      PRINT 1522,T
 1522 FORMAT(//,' PROCESS BOUNCED ON A TIME BOUNDARY--IMPOSSIBLE',/,
     1       ' TERMINATING ERROR:   T = ',D15.7,//)
      STOP
C
  215 CONTINUE
C
C     BOUNCED ON A VELOCITY BOUNDARY
C
      PRINT 1524,T
 1524 FORMAT(//,' BOUNCED ON A VELOCITY BOUNDARY AT T=',
     1          D15.7,//)
C
      IF (IZWEIG(3) .EQ. 1)   ISGN = +1
      IF (IZWEIG(3) .EQ. 3)   ISGN = -1
      IF (IZWEIG(3) .EQ. 2  .AND. SLOPE .GT. 0)    ISGN = -1
      IF (IZWEIG(3) .EQ. 2  .AND. SLOPE .LT. 0)    ISGN = +1
C
      IZWEIG(3) = IZWEIG(3) + ISGN
C
  230 CONTINUE
      CALL KRODE(T,Y,YP)
      KF = KF - 1
      KOUNTR = -2
C-----------------------------------------------------------------------
C     FORMAT STATEMENTS:
C-----------------------------------------------------------------------
 1511 FORMAT(' AT UPDATE  T = ',D23.16,'  TIME   = ',D15.7)
 1512 FORMAT(' UPDATE COMPONENT 3:',/,' T = ',D15.7,2X,
     1       ' AT V = ',D15.7,'  IZWEIG(3) = ',I3)
 1515 FORMAT(' UPDATE COMPONENT 1:',/,' ALPHA = ',D15.7,3X,' ALPHAM = ',
     1        D15.7,3X,' IZWEIG(1) = ',I3)
      RETURN
      END
```

Subroutine WARN for printing warning messages if the wrong ODE branch is used:

```
C-----------------------------------------------------------------------
      SUBROUTINE  WARN(MODE,PHI,NPHI,T,Y,NEQN,INDEX)
C-----------------------------------------------------------------------
      IMPLICIT REAL*8 (A-H,O-Z)
C
      DIMENSION PHI(NPHI),Y(NEQN)
C
      COMMON/BRANCH/IZWEIG(3)
      COMMON/VBD/V,VLOWER,VUPPER
      COMMON/FINALT/TOUT
      COMMON/DUMMY/IWARN1,IWARN2,IWARN3,IWARN4
      COMMON/PHILIM/ALPHA,DALDT,ALPHAM,DALMDT
      DATA TF/2.7D0/
C-----------------------------------------------------------------------
C     PURPOSE:   TO PRINT WARNING MESSAGES IF INCORRECT BRANCHING IS
C                BEING USED.
C
C     MODES OF OPERATION:
```

```
C
C     MODE = 1    INITIALIZES CONSTANTS USED IN MODE=2
C
C                 MODE 1 IS USED IN SUBPHI (KRODE) IN THE INITIALI-
C                 ZATION OF THE TRAPPING PROCEDURE (KOUNTR=0)
C
C     MODE = 2    TIME-LAG WARNING:  IF TRAPPING IS BEING USED, THE
C                 BRANCHING BOUNDARIES WILL BE VIOLATED ON ONE STEP
C                 BEFORE THE TRAPPING ITERATION IS ACTIVATED.  ONCE
C                 THE EXISTENCE OF THE ZERO IS DETECTED, INDEX IS NO
C                 LONGER ZERO AND CAN BE USED AS A FLAG TO INDICATE
C                 NO WARNING IS TO BE PRINTED.  THUS, MODE 2 HAS
C                 A TIME LAG OF ONE STEP BEFORE DECIDING IF WARNINGS
C                 ARE IN ORDER.
C
C                 MODE 2 IS USED IN SUBPHI (KRODE) WHENEVER PHI IS
C                 EVALUATED.
C
C
C     MODE = 4    NO TIME LAG WARNING:  AT UPDATE IN SUBPHI THE
C                 BRANCHING SHOULD BE SET PROPERLY.  MODE 4 PRINTS
C                 WARNING IS ANY BRANCH VIOLATION IS DETECTED.
C
C                 MODE 4 IS USED IN SUBPHI (KRODE) AT THE INITIALI-
C                 ZATION PROCESS (KOUNTR=0) AND AT UPDATE.
C-----------------------------------------------------------------------
      IF (MODE .EQ. 2)   GO TO 10
      IF (MODE .EQ. 4)   GO TO 40
C-----------------------------------------------------------------------
C     MODE 1:   SET WARNING FLAGS FOR FUTURE ANALYSIS (HELD IN DUMMY
C               COMMON FOR PROTECTION)
C-----------------------------------------------------------------------
      IWARN1=0
      IWARN2=0
      IWARN3=0
      RETURN
C
   10 CONTINUE
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
C     MODE 2:   CHECK FOR VIOLATIONS--ADJUSTMENTS FOR TRAPPING
C                                     ANALYSIS
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
C     VIOLATION CHECKS FOR IZWEIG(1)
C-----------------------------------------------------------------------
      IWARN1 = IWARN1 + 1
      IF (DABS(ALPHAM) .GT. DABS(ALPHA)  .AND. IZWEIG(1) .EQ. 1)
     1        IWARN1 = 0
      IF (DABS(ALPHAM) .LT. DABS(ALPHA)  .AND. IZWEIG(1) .EQ. 2)
     1        IWARN1 = 0
      IF (IWARN1 .EQ. 0)   GO TO 15
      IF (IWARN1 .GE. 2 .AND. INDEX .EQ. 0)  GO TO 14
      GO TO 15
   14 CONTINUE
C
C     DIFFICULTIES--WRONG BRANCH OF IZWEIG(1) IS USED
C
```

```
      PRINT 1501,T,IZWEIG(1),PHI(1),ALPHA,ALPHAM
 1501 FORMAT(//,' ALERT--WRONG BRANCH OF IZWEIG(1) IS BEING USED',/,
     1    ' AT T = ',D15.7,/,
     2    ' IZWEIG(1)=',I3,2X,'PHI(1)=',D15.7,/,
     3    2X,'ALPHA=',D15.7,2X,'ALPHAM=',D15.7,/)
C
   15 CONTINUE
C
C-----------------------------------------------------------------------
C     VIOLATION CHECKS FOR IZWEIG(2)
C-----------------------------------------------------------------------
C
      IWARN2 = IWARN2 + 1
      IF (T*TOUT .LT. TF    .AND. IZWEIG(2) .EQ. 1)  IWARN2= 0
      IF (T*TOUT .GT. TF    .AND. IZWEIG(2) .EQ. 2)  IWARN2= 0
      IF (IWARN2 .EQ. 0)   GO TO 25
      IF (IWARN2 .GE. 2 .AND. INDEX .EQ. 0)  GO TO 24
      GO TO 25
   24 CONTINUE
C
C     DIFFICULTIES--WRONG BRANCH OF IZWEIG(2) IS USED
C
      TSCALE = T*TOUT
      PRINT 1502,T,IZWEIG(2),PHI(2),TSCALE
 1502 FORMAT(//,' ALERT--WRONG BRANCH OF IZWEIG(2) IS BEING USED',/,
     1    ' AT T = ',D15.7,/,
     2    ' IZWEIG(2)=',I3,2X,'PHI(2)=',D15.7,/,' ACTUAL TIME = ',D15.7,
     3    /)
C
   25 CONTINUE
C
C-----------------------------------------------------------------------
C     VIOLATION CHECKS FOR IZWEIG(3)
C-----------------------------------------------------------------------
C
      IWARN3 = IWARN3 + 1
      IF (V .LT. 150.D0 .AND. IZWEIG(3) .EQ. 1)  IWARN3 = 0
      IF (V .LT. 536.36363636364D0  .AND.  V .GT. 150.0D0
     1              .AND.     IZWEIG(3) .EQ. 2)  IWARN3 = 0
      IF (V .GT. 536.36363636364D0  .AND.
     1                        IZWEIG(3) .EQ. 3)  IWARN3 = 0
      IF (IWARN3 .EQ. 0)   GO TO 35
      IF (IWARN3 .GE. 2 .AND. INDEX .EQ. 0)  GO TO 34
      GO TO 35
   34 CONTINUE
C
C     DIFFICULTIES--WRONG BRANCH OF IZWEIG(3) IS USED
C
      PRINT 1503,T,IZWEIG(3),PHI(3),V,VLOWER,VUPPER
 1503 FORMAT(//,' ALERT--WRONG BRANCH OF IZWEIG(3) IS BEING USED',/,
     1    ' AT T = ',D15.7,/,
     2    ' IZWEIG(3)=',I3,2X,'PHI(3)=',D15.7,/,
     3    2X,'V=',D15.7,2X,'VLOWER=',D15.7,2X,'VUPPER=',D15.7,/)
C
   35 CONTINUE
      RETURN
C
   40 CONTINUE
```

```
C-----------------------------------------------------------------------
C     MODE 4:   CHECK FOR VIOLATIONS--NO ADJUSTMENTS FOR TRAPPING
C                                     ANALYSIS
C               WARNING CONDITIONS ARE THE SAME AS IN MODE=3, BUT
C               NO TIME LAG IS PERMITTED.
C-----------------------------------------------------------------------
C
C-----------------------------------------------------------------------
C     VIOLATION CHECKS FOR IZWEIG(1)
C-----------------------------------------------------------------------
C
      EPS = 1.D-03
      IWARN = 1
      IF (DABS(ALPHAM) .GT. DABS(ALPHA)-EPS  .AND. IZWEIG(1) .EQ. 1)
     1        IWARN  = 0
      IF (DABS(ALPHAM) .LT. DABS(ALPHA)+EPS  .AND. IZWEIG(1) .EQ. 2)
     1        IWARN  = 0
      IF (IWARN  .EQ. 0)   GO TO 45
      PRINT 1501,T,IZWEIG(1),PHI(1),ALPHA,ALPHAM
   45 CONTINUE
C
C-----------------------------------------------------------------------
C     VIOLATION CHECKS FOR IZWEIG(3)
C-----------------------------------------------------------------------
C
      IWARN  = 1
      IF (T*TOUT .LT. TF + EPS  .AND. IZWEIG(2) .EQ. 1)  IWARN = 0
      IF (T*TOUT .GT. TF - EPS  .AND. IZWEIG(2) .EQ. 2)  IWARN = 0
      IF (IWARN  .EQ. 0)   GO TO 46
      TSCALE = T*TOUT
      PRINT 1502,T,IZWEIG(2),PHI(2),TSCALE
   46 CONTINUE
C
C-----------------------------------------------------------------------
C     VIOLATION CHECKS FOR IZWEIG(3)
C-----------------------------------------------------------------------
C
      EPS = 1.D-1
      IWARN  = 1
      IF (V - EPS .LT. 150.D0 .AND. IZWEIG(3) .EQ. 1) IWARN = 0
      IF (V - EPS .LT. 536.36363636364D0 .AND. V + EPS .GT. 150.0D0
     1              .AND.     IZWEIG(3) .EQ. 2)  IWARN  = 0
      IF (V - EPS .GT. 536.36363636364D0 .AND.
     1                        IZWEIG(3) .EQ. 3)  IWARN  = 0
      IF (IWARN  .EQ. 0)   GO TO 47
      PRINT 1503,T,IZWEIG(3),PHI(3),V,VLOWER,VUPPER
   47 CONTINUE
C
      RETURN
      END
```

Subroutine ZWEIGE for determing the initial branches of the ODE:

```
C-----------------------------------------------------------------------
      SUBROUTINE ZWEIGE(NEQN,T,Y)
C-----------------------------------------------------------------------
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION Y(NEQN),YP(6)
```

```
      COMMON /CTROMP/
     1  GRID(15,5),STIFF(15,5),U(15,5,5),A(15,5,5),B(15,5,5),C(15,5,5),
     2  UL(15,5,5),UU(15,5,5),UH(15,15,5,5),AH(15,15,5,5),BH(15,15,5,5),
     3  CH(15,15,5,5),P(10),ZO( 6),TOL,INTERV,KONTRL,KNOT(5),NP,NZ,
     4  MQP,IG,NPHI,LDUM(18)
C
      COMMON/BRANCH/IZWEIG(3)
      COMMON/PHILIM/ALPHA,DALDT,ALPHAM,DALMDT
      COMMON/FINALT/TFINAL
      LOGICAL LDUM
      DATA CALPHA/-2.33D-03/,KNOT1/15/
C-----------------------------------------------------------------------
C     SUBROUTINE FOR DETERMINING BRANCHES OF ODES AT INITIAL CONDITIONS
C-----------------------------------------------------------------------
      V = Y(1)
C-----------------------------------------------------------------------
C     INITIAL VALUE: T = 0
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
C     IZWEIG(1):
C-----------------------------------------------------------------------
C
C     COMPUTE AZ  FOR DETERMINING ALPHA:
C
      AZ    = SPLINT(KNOT1,GRID(1,1),U(1,1,1),STIFF(1,1),
     1              A(1,1,1),B(1,1,1),C(1,1,1),T)
C
C     COMPUTE AZM FOR DETERMINING ALPHAM:
C
      IF (V .LT.  150.0D0)                  AZM = 0.002D0 * V * V
      IF (V .LT. 536.3636363636365D0 .AND. V .GE. 150.D0)
     1                                      AZM = 45.0D0
     2                                          + 0.660D0 * (V - 150.0D0)
      IF (V .GE. 536.3636363636365D0)
     1                                      AZM = 300.0D0
      ALPHA = AZ/(CALPHA*V*V)
      ALPHAM = -AZM /(CALPHA * V * V)
C
      IZWEIG(1) = 2
      IF (DABS(ALPHA) .LT. ALPHAM)  IZWEIG(1) = 1
C-----------------------------------------------------------------------
C     IZWEIG(2):
C-----------------------------------------------------------------------
      IZWEIG(2) = 1
C-----------------------------------------------------------------------
C     IZWEIG(3):
C-----------------------------------------------------------------------
      IZWEIG(3) = 1
C
      RETURN
      END
```

APPENDIX C. OUTPUT FROM OCP EXAMPLE

(Comments added to output listing are in italics)

Initial Parameters:

```
 NUMBER OF INTERVALS FOR GRID =   1
 NUMBER OF CONTROL PARAMETERS =   1
 NUMBER OF KNOTS              =  15
 NUMBER OF DESIGN PARAMETERS  =   0
 NUMBER OF DEPENDENT VARIABLES=   6
 LOGICAL PARAMETERS--IMPULS =  F
                     STORE  =  T   STORB  =  F
                     LPLOT  =  F
                     LBOUND =  F   UBOUND =  F
                     LFCT   =  F
 INTEGRATION TOLERANCE        =   0.1000000D-04
 NUMBER OF CONSTRAINTS        =   4


                  Z0( 1) =   0.5860000D+02
                  Z0( 2) =   0.1570796D+01
                  Z0( 3) =   0.4598755D-14
                  Z0( 4) =  -0.8790000D+01
                  Z0( 5) =   0.0
                  Z0( 6) =   0.0


    STIFFNESS FACTOR( 1) =   0.0
    STIFFNESS FACTOR( 2) =   0.0
    STIFFNESS FACTOR( 3) =   0.0
    STIFFNESS FACTOR( 4) =   0.0
    STIFFNESS FACTOR( 5) =   0.0
    STIFFNESS FACTOR( 6) =   0.0
    STIFFNESS FACTOR( 7) =   0.0
    STIFFNESS FACTOR( 8) =   0.0
    STIFFNESS FACTOR( 9) =   0.0
    STIFFNESS FACTOR(10) =   0.0
    STIFFNESS FACTOR(11) =   0.0
    STIFFNESS FACTOR(12) =   0.0
    STIFFNESS FACTOR(13) =   0.0
    STIFFNESS FACTOR(14) =   0.0
    STIFFNESS FACTOR(15) =   0.0


                GRID( 1) =   0.0
                GRID( 2) =   0.3703704D-02
                GRID( 3) =   0.1851852D-01
                GRID( 4) =   0.3703704D-01
                GRID( 5) =   0.7407407D-01
                GRID( 6) =   0.1111111D+00
                GRID( 7) =   0.2222222D+00
                GRID( 8) =   0.3703704D+00
                GRID( 9) =   0.4629630D+00
                GRID(10) =   0.5555556D+00
```

```
                GRID(11) =   0.6666667D+00
                GRID(12) =   0.7777778D+00
                GRID(13) =   0.8518519D+00
                GRID(14) =   0.9259259D+00
                GRID(15) =   0.1000000D+01


  CONTROL PARAMETER( 1) =   0.0
  CONTROL PARAMETER( 2) =   0.1000000D+01
  CONTROL PARAMETER( 3) =   0.5000000D+01
  CONTROL PARAMETER( 4) =   0.1000000D+02
  CONTROL PARAMETER( 5) =   0.1200000D+02
  CONTROL PARAMETER( 6) =   0.1500000D+02
  CONTROL PARAMETER( 7) =   0.1000000D+02
  CONTROL PARAMETER( 8) =   0.8000000D+02
  CONTROL PARAMETER( 9) =   0.1000000D+03
  CONTROL PARAMETER(10) =   0.1200000D+03
  CONTROL PARAMETER(11) =   0.1800000D+03
  CONTROL PARAMETER(12) =   0.2200000D+03
  CONTROL PARAMETER(13) =   0.2300000D+03
  CONTROL PARAMETER(14) =   0.2400000D+03
  CONTROL PARAMETER(15) =   0.2500000D+03


   PARAMETERS FOR SLLSQP:
                                MODE  =    0
                               MAXIT  =   50
   NUMBER OF CONTROL GRID POINTS,  NX  =   15
   NUMBER OF EQUALITY CONSTRAINTS, ME  =    3
   LEADING DIMENSION OF DG MATRIX, LDG =   40

   BOUNDS ON CONTROL VALUES:

    XL(  1) =   0.0                 XU(  1) =   0.1000000D+00
    XL(  2) =   0.0                 XU(  2) =   0.1000000D+02
    XL(  3) =   0.0                 XU(  3) =   0.5000000D+02
    XL(  4) =   0.0                 XU(  4) =   0.9500000D+02
    XL(  5) =   0.0                 XU(  5) =   0.3000000D+03
    XL(  6) =   0.0                 XU(  6) =   0.3000000D+03
    XL(  7) =   0.0                 XU(  7) =   0.3000000D+03
    XL(  8) =   0.0                 XU(  8) =   0.3000000D+03
    XL(  9) =   0.0                 XU(  9) =   0.3000000D+03
    XL( 10) =   0.0                 XU( 10) =   0.3000000D+03
    XL( 11) =   0.0                 XU( 11) =   0.3000000D+03
    XL( 12) =   0.0                 XU( 12) =   0.3000000D+03
    XL( 13) =   0.0                 XU( 13) =   0.3000000D+03
    XL( 14) =  -0.3000000D+03       XU( 14) =   0.3000000D+03
    XL( 15) =  -0.3000000D+03       XU( 15) =   0.3000000D+03


 MINIMIZER: SLLSQP
          SEQUENTIAL LINEAR LEAST SQUARES PROGRAMMING


Initial integration (with trapping stops printed)
```

```
 OUTPUT: T=  0.0
  VELOCITY=  0.58600D+02   GAMMA= 0.15708D+01   X        = 0.45988D-14
  Z        = -0.87900D+01   IA  = 0.0          DGAMMA/DT= 0.0
 INITIAL BRANCHES:     1   1   1

 OUTPUT: T=  0.370370D-02
  VELOCITY=  0.61180D+02   GAMMA= 0.15682D+01   X        = 0.64318D-03
  Z        = -0.94998D+01   IA  = 0.0          DGAMMA/DT=-0.42520D+00

 OUTPUT: T=  0.185185D-01
  VELOCITY=  0.71132D+02   GAMMA= 0.15227D+01   X        = 0.70860D-01
  Z        = -0.12638D+02   IA  = 0.0          DGAMMA/DT=-0.13498D+01

 OUTPUT: T=  0.370370D-01
  VELOCITY=  0.82203D+02   GAMMA= 0.14276D+01   X        = 0.50187D+00
  Z        = -0.17167D+02   IA  = 0.0          DGAMMA/DT=-0.17338D+01

 OUTPUT: T=  0.740741D-01
  VELOCITY=  0.10447D+03   GAMMA= 0.12541D+01   X        = 0.31192D+01
  Z        = -0.27860D+02   IA  = 0.0          DGAMMA/DT=-0.11297D+01

 OUTPUT: T=  0.111111D+00
  VELOCITY=  0.12981D+03   GAMMA= 0.11404D+01   X        = 0.82395D+01
  Z        = -0.40736D+02   IA  = 0.0          DGAMMA/DT=-0.81293D+00

C-------------------------------------------------------------------------

 UPDATE COMPONENT 3-4:
 T =   0.1383976D+00   AT V =   0.1500000D+03  IZWEIG(3) =   2

C-------------------------------------------------------------------------

 OUTPUT: T=  0.222222D+00
  VELOCITY=  0.21979D+03   GAMMA= 0.10113D+01   X        = 0.38863D+02
  Z        = -0.94147D+02   IA  = 0.0          DGAMMA/DT=-0.15685D+00

 OUTPUT: T=  0.370370D+00
  VELOCITY=  0.35303D+03   GAMMA= 0.85030D+00   X        = 0.11899D+03
  Z        = -0.20340D+03   IA  = 0.0          DGAMMA/DT=-0.46422D+00

 OUTPUT: T=  0.462963D+00
  VELOCITY=  0.44220D+03   GAMMA= 0.72101D+00   X        = 0.20240D+03
  Z        = -0.28613D+03   IA  = 0.0          DGAMMA/DT=-0.39304D+00

C-------------------------------------------------------------------------

 UPDATE COMPONENT 3-4:
 T =   0.5500937D+00   AT V =   0.5363636D+03  IZWEIG(3) =   3

C-------------------------------------------------------------------------

 OUTPUT: T=  0.555556D+00
  VELOCITY=  0.54257D+03   GAMMA= 0.61467D+00   X        = 0.31697D+03
  Z        = -0.37592D+03   IA  = 0.0          DGAMMA/DT=-0.34060D+00

 OUTPUT: T=  0.666667D+00
  VELOCITY=  0.67486D+03   GAMMA= 0.48871D+00   X        = 0.50103D+03
  Z        = -0.48886D+03   IA  = 0.0          DGAMMA/DT=-0.36924D+00
```

```
 OUTPUT: T=  0.777778D+00
  VELOCITY=  0.82101D+03   GAMMA= 0.35973D+00   X        = 0.74306D+03
  Z        = -0.59735D+03    IA  = 0.0          DGAMMA/DT=-0.34569D+00

C-------------------------------------------------------------------------

 AT UPDATE  T =  0.84375D+00  TIME   =   0.2700000D+01  IZWEIG(2) =   2

C-------------------------------------------------------------------------

 OUTPUT: T=  0.851852D+00
  VELOCITY=  0.91488D+03   GAMMA= 0.28400D+00   X        = 0.93960D+03
  Z        = -0.66239D+03    IA  = 0.0          DGAMMA/DT=-0.31360D+00

 OUTPUT: T=  0.925926D+00
  VELOCITY=  0.89551D+03   GAMMA= 0.22467D+00   X        = 0.11472D+04
  Z        = -0.71645D+03    IA  = 0.0          DGAMMA/DT=-0.32924D+00

 OUTPUT: T=  0.100000D+01
  VELOCITY=  0.87568D+03   GAMMA= 0.16150D+00   X        = 0.13532D+04
  Z        = -0.75682D+03    IA  = 0.0          DGAMMA/DT=-0.34561D+00
```

Results from each iteration

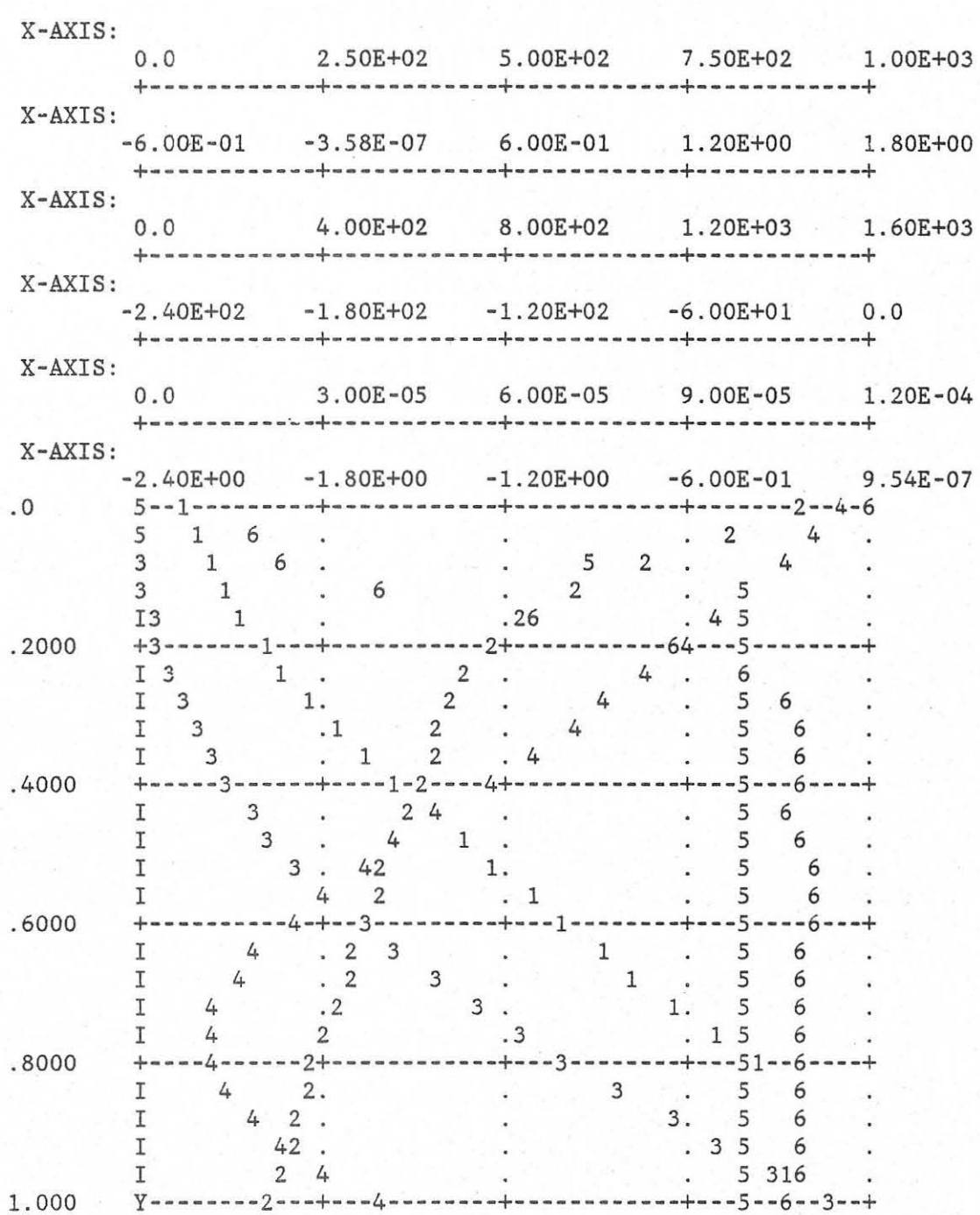
```
 ITERATION =   1   F       = -0.109460D+01
                   G(  1 ) =  -0.1925268D+01   G(  2) =   0.3730022D+01
                   G(  3 ) =  -0.9786908D-01   G(  4) =   0.1000000D-03

 ITERATION =   2   F       = -0.108795D+01
                   G(  1 ) =  -0.1695904D+01   G(  2) =   0.3295920D+01
                   G(  3 ) =  -0.9064575D-01   G(  4) =   0.1000000D-03

 ITERATION =   3   F       = -0.108348D+01
                   G(  1 ) =  -0.1506637D+01   G(  2) =   0.2933597D+01
                   G(  3 ) =  -0.8308310D-01   G(  4) =   0.9304128D-04

 ITERATION =   4   F       = -0.108038D+01
                   G(  1 ) =  -0.1327748D+01   G(  2) =   0.2595101D+01
                   G(  3 ) =  -0.7542239D-01   G(  4) =  -0.3809821D-04

 ITERATION =   5   F       = -0.107829D+01
                   G(  1 ) =  -0.1172664D+01   G(  2) =   0.2300036D+01
                   G(  3 ) =  -0.6837602D-01   G(  4) =  -0.1787329D-03

 ITERATION =   6   F       = -0.107770D+01
                   G(  1 ) =  -0.1047959D+01   G(  2) =   0.2057852D+01
                   G(  3 ) =  -0.6185829D-01   G(  4) =  -0.1799745D-03

 ITERATION =   7   F       = -0.107714D+01
                   G(  1 ) =  -0.8663255D+00   G(  2) =   0.1708945D+01
                   G(  3 ) =  -0.5324333D-01   G(  4) =  -0.1713003D-03
```

```
ITERATION =   8   F       = -0.107748D+01
                  G(  1 ) =  -0.5704772D+00    G(  2) =   0.1141819D+01
                  G(  3 ) =  -0.4056563D-01    G(  4) =  -0.1851074D-03

ITERATION =   9   F       = -0.108004D+01
                  G(  1 ) =  -0.3986150D+00    G(  2) =   0.8091109D+00
                  G(  3 ) =  -0.3173114D-01    G(  4) =  -0.1621028D-03

ITERATION =  10   F       = -0.108137D+01
                  G(  1 ) =  -0.3531334D+00    G(  2) =   0.7194758D+00
                  G(  3 ) =  -0.2878550D-01    G(  4) =  -0.1535623D-03

ITERATION =  11   F       = -0.108547D+01
                  G(  1 ) =  -0.1628500D+00    G(  2) =   0.3531537D+00
                  G(  3 ) =  -0.1943426D-01    G(  4) =  -0.1233577D-03

ITERATION =  12   F       = -0.109095D+01
                  G(  1 ) =   0.9998761D-01    G(  2) =  -0.1500635D+00
                  G(  3 ) =  -0.8464951D-02    G(  4) =  -0.2961854D-04

ITERATION =  13   F       = -0.109338D+01
                  G(  1 ) =   0.2862318D-01    G(  2) =  -0.3827715D-01
                  G(  3 ) =  -0.3739060D-02    G(  4) =   0.1335459D-04

ITERATION =  14   F       = -0.109467D+01
                  G(  1 ) =   0.2964904D-01    G(  2) =  -0.4689855D-01
                  G(  3 ) =  -0.6084254D-03    G(  4) =  -0.2865439D-04

ITERATION =  15   F       = -0.109522D+01
                  G(  1 ) =   0.3101787D-03    G(  2) =  -0.4316430D-03
                  G(  3 ) =  -0.3017842D-04    G(  4) =  -0.4733124D-05

MODE =    0

AFTER RETURN FROM SLLSQP:
X(  1) =   0.1000000D+00
X(  2) =   0.2942806D+01
X(  3) =   0.6388505D+01
X(  4) =   0.1149777D+02
X(  5) =   0.1936999D+02
X(  6) =   0.2470492D+02
X(  7) =   0.2340847D+02
X(  8) =   0.3929148D+02
X(  9) =   0.6919008D+02
X( 10) =   0.7635476D+02
X( 11) =   0.1272889D+03
X( 12) =   0.1684633D+03
X( 13) =   0.2027118D+03
X( 14) =   0.2005819D+03
X( 15) =   0.2351272D+03


F =  -0.1095228D+01
```

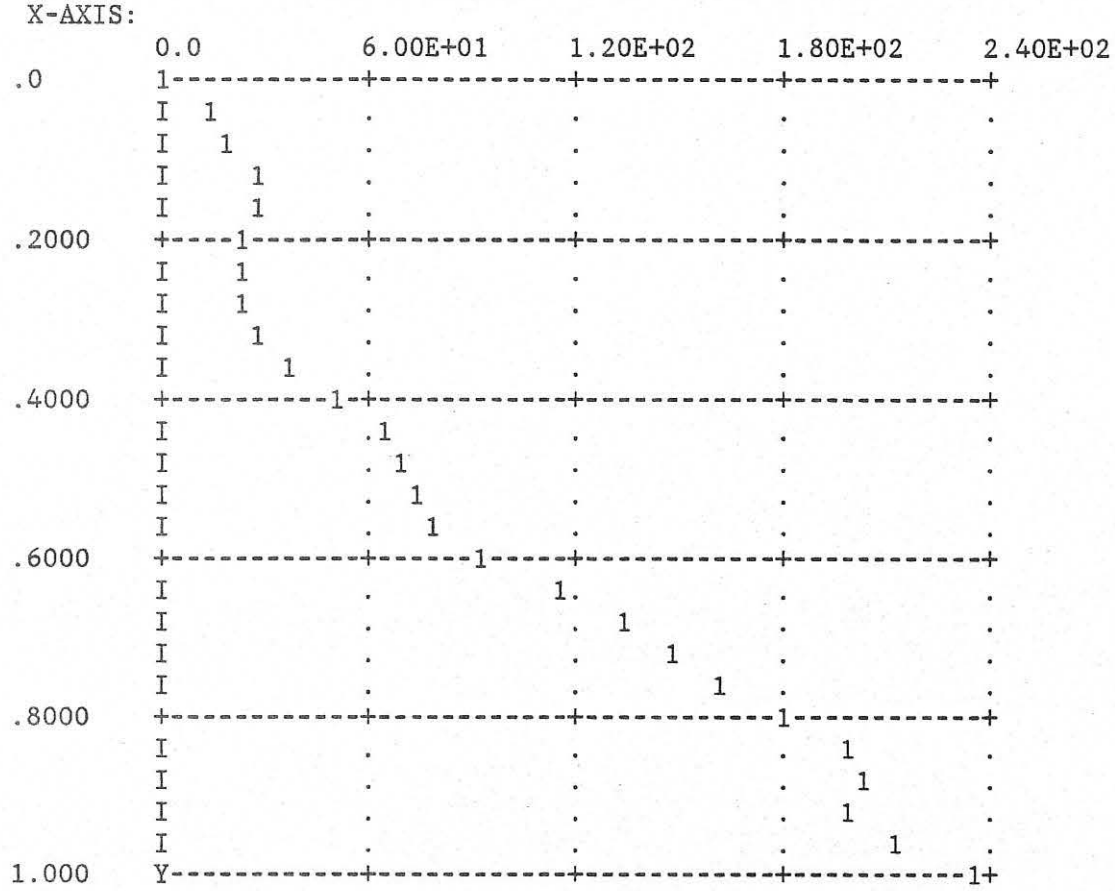
```
 NUMBER OF DERIVATIVE EVALUATIONS: TOTAL NFE =  67519
(including extra evaluations for
TROMP)


 NUMBER OF DERIVATIVE EVALUATIONS: TOTAL NFE =  66159
                                   NEXTRA    =   4357
(required for the integration)

Plotted results using the Optimal Trajectory


 FORWARD INTEGRATION BEFORE PLOTTING USING OPTIMAL TRAJECTORY:


 OUTPUT: T=  0.0
  VELOCITY=  0.58600D+02   GAMMA= 0.15708D+01   X        = 0.45988D-14
  Z        = -0.87900D+01    IA  = 0.0          DGAMMA/DT= 0.0

 OUTPUT: T=  0.370370D-02
  VELOCITY=  0.61126D+02   GAMMA= 0.15626D+01   X        = 0.21071D-02
  Z        = -0.94996D+01    IA  = 0.0          DGAMMA/DT=-0.11835D+01

 OUTPUT: T=  0.185185D-01
  VELOCITY=  0.70027D+02   GAMMA= 0.14799D+01   X        = 0.15520D+00
  Z        = -0.12607D+02    IA  = 0.0          DGAMMA/DT=-0.17122D+01

 OUTPUT: T=  0.370370D-01
  VELOCITY=  0.80279D+02   GAMMA= 0.13682D+01   X        = 0.80555D+00
  Z        = -0.17020D+02    IA  = 0.0          DGAMMA/DT=-0.20111D+01

 OUTPUT: T=  0.740741D-01
  VELOCITY=  0.96655D+02   GAMMA= 0.11199D+01   X        = 0.42143D+01
  Z        = -0.26931D+02    IA  = 0.55860D-04  DGAMMA/DT=-0.19723D+01

 OUTPUT: T=  0.111111D+00
  VELOCITY=  0.11301D+03   GAMMA= 0.89533D+00   X        = 0.10883D+02
  Z        = -0.37365D+02    IA  = 0.10023D-03  DGAMMA/DT=-0.17009D+01

 OUTPUT: T=  0.222222D+00
  VELOCITY=  0.18685D+03   GAMMA= 0.50442D+00   X        = 0.52217D+02
  Z        = -0.68508D+02    IA  = 0.10023D-03  DGAMMA/DT=-0.49206D+00

 OUTPUT: T=  0.370370D+00
  VELOCITY=  0.32356D+03   GAMMA= 0.34760D+00   X        = 0.16220D+03
  Z        = -0.11634D+03    IA  = 0.10023D-03  DGAMMA/DT=-0.23098D+00

 OUTPUT: T=  0.462963D+00
  VELOCITY=  0.41886D+03   GAMMA= 0.25819D+00   X        = 0.26702D+03
  Z        = -0.14896D+03    IA  = 0.10023D-03  DGAMMA/DT=-0.25608D+00

 OUTPUT: T=  0.555556D+00
  VELOCITY=  0.52370D+03   GAMMA= 0.18012D+00   X        = 0.40311D+03
  Z        = -0.17879D+03    IA  = 0.10023D-03  DGAMMA/DT=-0.18418D+00

 OUTPUT: T=  0.666667D+00
  VELOCITY=  0.66241D+03   GAMMA= 0.91385D-01   X        = 0.61162D+03
  Z        = -0.20719D+03    IA  = 0.10023D-03  DGAMMA/DT=-0.22260D+00
```

```
OUTPUT: T=  0.777778D+00
 VELOCITY=  0.81519D+03   GAMMA=-0.43562D-02   X        = 0.87350D+03
 Z        = -0.21811D+03    IA  = 0.10023D-03   DGAMMA/DT=-0.22045D+00

OUTPUT: T=  0.851852D+00
 VELOCITY=  0.91160D+03   GAMMA=-0.67782D-01   X        = 0.10794D+04
 Z        = -0.21043D+03    IA  = 0.10023D-03   DGAMMA/DT=-0.22986D+00

OUTPUT: T=  0.925926D+00
 VELOCITY=  0.89414D+03   GAMMA=-0.11903D+00   X        = 0.12924D+04
 Z        = -0.19049D+03    IA  = 0.10023D-03   DGAMMA/DT=-0.23168D+00

OUTPUT: T=  0.100000D+01
 VELOCITY=  0.87618D+03   GAMMA=-0.17454D+00   X        = 0.15000D+04
 Z        = -0.16000D+03    IA  = 0.10023D-03   DGAMMA/DT=-0.27338D+00

OUTPUT: T=  0.0
 VELOCITY=  0.58600D+02   GAMMA= 0.15708D+01   X        = 0.45988D-14
 Z        = -0.87900D+01    IA  = 0.0          DGAMMA/DT= 0.0
```

Parameter 1 is Velocity *Parameter 4 is z coordinate*
Parameter 2 is flight path angle, γ *Parameter 5 is I_α*
Parameter 3 is x coordinate *Parameter 6 is $d\gamma/dt$*



Parameter 1 is lift coefficient (control parameter)



Parameter 1 is α
Parameter 2 is α_m