Institut für
Dynamik der Flugsysteme

Bl.:
IB.Nr.: 515-83/1

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

Freigabe:

Die Bearbeiter:
Dr. M.K. Horn

Unterschriften:
i.A. Well

Dr. K.H. Well
Der Abteilungsleiter

Klaus Heinrich Well

Der stellv. Institutsdirektor:

Dr.Ing.J.Ackermann
Der Institutsdirektor:

J. Ackermann

Dieser Bericht enthält:

126 Blatt davon
1 Bilder
97 ~~Diagramme~~ Bl.Rechnerprotokoll

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. (Current report)

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. (See reference [1].)

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].)

CONTENTS

Optimal Control Problems, Numerical Analysis, Linear Interpolation

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

Summary

The state/control-constraint optimal control problem is presented which extenxive use is made of linearly interpolated data. To permit numerical solutions of this type of problem, the integration package halts the solution automatically at the grid points of the tables, permitting the interpolation region of the associated differential equations to be changed. Thus, the differential equations are analyzed as truly piecewise continuous systems. An example of a realistic problem is given showing the structure of the differential equations and the related subroutines for adjusting subscripting values pertaining to the tables.

Optimale Steuerungsprobleme, numerische Mathematik, lineare Interpolation

Ein Fortran-Programm zur Lösung optimaler Steuerungsprobleme mit Zustands-/Steuerungsbeschränkungen und mit tabellarisch gegebenen Systemgleichungen

Übersicht

Ein zustands-/steuerungs-beschränktes optimales Steuerungsproblem wird präsentiert, in dem das Differentialgleichungssystem von vielen linear interpolierten Daten abhängig ist. Um die numerische Lösung eines solchen Problems zu ermöglichen, hält das Integrationsprogramm die Lösung an jedem Tabellenstützpunkt automatisch an, damit das Interpolationsgebiet gewechselt werden kann. Dadurch wird das Differentialgleichungssystem wie ein stückweise stetiges System behandelt. Die Struktur der Differentialgleichungen sowie die zugehörigen Unterprogramme zur Anpassung der Tabellenkoeffizienten wird anhand eines realistischen Beispiels aufgezeigt.

1. INTRODUCTION

In the state/control-constraint optimal control problem (OCP), one frequently encounters right hand sides having expressions involving the use of functions written in tabular form. In such problems the user is confronted with the necessity of interpolating data from these tables. Bilinear interpolation of the data defines a continuous tabular function with discontinuous slopes along lines corresponding to each grid entry in the table domain. The simplicity of the formulation makes the method appealing. Perhaps the most important advantage of the scheme, however, is the ability to interchange data sets with ease. In contrast, smooth curve fitting techniques require considerable, off-line computing time to generate representative curves, and new curves must be computed each time a new data set is given (or each time that a data set is altered).

Solving the system of ordinary differential equations (ODEs) for the OCP, in which the right hand sides contain expressions defined by linearly interpolated tabular data, requires that the integration procedure stop at each table grid entry so that the ODE may be reevaluated using the correct expressions for the "new" region of the table. In addition, the analysis at such points is particularly important if one uses adjoint variables in solving the OCP, since the adjoint differential equations themselves exhibit actual "jumps" at the grid points (rather than just slope changes) when linearly interpolated data is used. In the presence of such discontinuities (either in the ODE itself or in the higher derivatives), even a well written software package will waste computing time by reducing the step size severely in an attempt to isolate the points of discontinuity, i.e., several attempted steps are made before a sufficiently small step length is used. In addition, these points are not located within any specified tolerance introducing additional errors into the ODE solution. These added errors can cause convergence difficulties for the optimization process, greatly increasing the computation time. (See [1].) Thus, to use linear interpolation effectively, one must employ a software package designed to treat tabular functions truly as piecewise continuous functions.

The purpose of this research project is the creation of a computer program, designed to solve the state/control-constraint optimal control problem in which extensive use is made of linearly interpolated tabular functions. This problem requires the development (or adaptation) of several software packages. The essential package for solving the ODEs is the RKF45T program which halts the integration whenever any component of a user-supplied vector of stopping conditions, PHI(J), vanishes. Any stopping condition which can be written in terms of the independent variable, the dependent variables, and/or the first derivatives may be imposed. Thus, the table entry values may be specified as stopping points for the integration, giving the user the opportunity to update the ODE system as each grid value is reached. Through such analysis, the ODE may be treated as the actual piecewise continuous system that it is. Since the integration routine is not called directly by the user, the RKF45T package must be able to make all updates without returning to the driving program. This characteristic of the problem necessitates the development of a particular program structure for handling the updates of the tabular functions.

The OCP is treated as a parameter optimization problem in the example presented, using an adapted version of the TOMP (Trajectory Optimization by Mathematical Programming) software package [4] for evaluating the cost func-

tion and/or the gradient. The modified version, TROMPP, (zero TRapping capabilities added to TOMP with additional Partition stops), has been written to handle the linearly interpolated, tabular data as analyzed by the RKF45T system. Thus, the RKF45T and the TROMPP packages form a system for supplying the value of the cost function and/or the gradients for an optimization process. This TROMPP/RKF45T system may be used with any mathematical programming package. The particular package used in the example presented is the SLLSQP (Sequential Linear Least Squares Programming) [5] chosen because of its rapid convergence rate.

The current report is one of four volumes all related to the solution of the OCP for problems in which the state equations have piecewise continuous derivatives. The ordering of the reports has been chosen according to the motivation of the project, so that the reader may see both the forest and the trees. The current report, Volume 1, describes the OCP involving the use of linearly interpolated tabular data. The description of the bookkeeping involved in this problem is sufficient to fill the entire volume. Thus, for an indepth understanding of parts of Volume 1, the reader may be referred to later volumes in which descriptions treating simpler problems may be enlightening. To separate the various problems involved in analyzing the piecewise continuous equations with tabular data expressions, Volume 2 has been included, treating a different type of OCP problem, namely one in which the derivatives are piecewise continuous but do not depend upon tabular data [1]. Much of the discussion in [1] centers around the description of the stopping conditions and the treatment of discontinuities, which may prove useful in clarifying part of the analysis in the current volume. The problem in [1] also handles constraint equations described as a function of the dependent variables. Volume 3 [2] describes the use of the RKF45T package along with examples which go beyond the scope of either the current report or [1]. Volume 4 [3] gives a description of the structure of the tables used in the current report and the interpolation formulas for the tabular data as well as several useful subroutines for handling the data and for analyzing control parameters expressed as cubic splines. (Volume 4 could actually be considered as a large appendix to Volume 1.) In the current report, the reader should expect to get an overview of the solution of an OCP involving linearly interpolated tabular data, but may need to read parts of further volumes for details.

The use of the RKF45T/TROMPP system is illustrated in the solution of the minimum-time-to-climb problem in which drag and thrust characteristics are supplied through tabular data. The problem description has been written for easy adaptation to other problems involving tabular data. The application may be divided into two basic parts: 1) the table structure and its communication with the RKF45T/TROMPP packages, and 2) the integration stops and updates from the RKF45T/TROMPP packages. This report gives a description of features in TROMPP related to the integration stops (as well as a program listing) and provides a general description of the programs needed for use in the RKF45T package. The table structure is described briefly (with further details contained in Volume 4). The user should realize that this report is designed to give an overview of the problem. Details may often come from clarifications in further volumes or from descriptions later in the current report. The complexity of the bookkeeping necessitates such a structure. Otherwise, the reader could easily become lost in details and miss the general picture.

2. FEATURES OF TROMPP

The TROMPP package determines the value of the user-supplied cost function (subroutine COSTF) and/or the gradient values for the OCP. In order to determine these functions, a system of differential equations is solved over the normalized time interval 0 to 1. The control parameters used in evaluating the ODE system are described as a cubic splines. During the integration, stops are made at each grid point defining the cubic splines (as in TOMP [4]). TROMPP differs from the TOMP package in that the user may supply additional vectors of stopping values (independent variable stops) in both the forward and backward integrations as well as activate the RKF45T trapping option to halt the integration whenever any user-supplied stopping conditions, described as a function of t, y, and y', are isolated. The independent variable stops are imposed in the form of the vectors TPART1 (for the forward and backward integrations) and TPART2 (for the backward integration only). Each vector is dimensioned 100.

The idea behind the modification of the TOMP program is to enable the user to treat two types of problems efficiently. Both problems deal with halting the integration at points of discontinuity (or discontinuous slopes) so that the ODE system can be updated using the new information. The first type of integration stop involves table grids in which the domain parameter is expressed as a function of t (say as a cubic spline). The lift coefficient will be presented as such a parameter, appearing in the tables defining the drag. The lift coefficient grid may be analyzed before the integration begins to determine the values of t corresponding to points on the lift coefficient grid axis. These values will be imposed as stopping conditions on the integration. The second type of integration stop concerns the location of the t values corresponding to specified functions of t, y, and y' which appear as table domain parameters. The user defines the stopping points, e.g., specific values of the Mach number or altitude. The RKF45T package halts the integration at these points (through an internal iterative process) and permits the user to update the ODE system before continuing. Thus, TROMPP/RKF45T system allows the user to communicate with the ODE system during the integration so that the piecewise continuous nature of the tables is analyzed properly.

The TROMPP package is basically the TOMP program whose description is contained in [4]. The modifications are clearly marked. Two additional subroutines have been included in the TROMPP package to handle the user-supplied stopping vectors. GRPART and TRPART require no user changes. At each user-supplied stopping point (i.e., from TPART1 or TPART2) an update call is made from TRPART. This call references a user-supplied subroutine PART with parameters identifying the particular update point. Subroutine PART is the *only* major programming effort that the user must provide in conjunction with TROMPP. The calling sequence and defining parameters for PART are described in §4.3. An application of the PART updates is illustrated in §5.3 using lift coefficient grid stops. Flow charts for both GRPART and TRPART are given in Appendix B, with the listings given in Appendix A.

The use of the stopping partitions is illustrated in the minimum-time-to-climb problem described in §5. In this example, tabular data is used extensively in determining the right hand sides (RHS) of the ODE system. The integration is to be stopped as the domain parameters of the table (e.g., Mach number, lift coefficient, or altitude) cross the entries in the appropriate grids. At these "crossings" the limits identifying the table entries are shifted to designate the "new" region of the table. These shifts are pos-

sible during the updates provided in TROMPP (for TPART1 or TPART2 vectors) or in RKF45T as the user-supplied stopping conditions from subroutine SUBPHI are isolated. For the example presented in this report, SUBPHI isolates each Mach number and altitude entry in a total of four tables.

The user-imposed stopping conditions TPART1, i.e., the cubic spline analysis stops for the lift coefficient (described in [3]) are supplied before any forward integration. These conditions will also be imposed during the backward integration (unless the user deactivates the partition stop.) The values of t established during the forward integration as corresponding to Mach number or altitude grid stops (determined by the RKF45T trapping process) are imposed as TPART2 stops during the backward integration. The user has the opportunity to update the ODE system at each TPART1 and TPART2 stop as well as at any RKF45T update point.

3. TABLE DESCRIPTION

The description of the table structure is best presented by example. The tables defining the problem in §5 are expressed in general terms for easy adaptation to any ODE system requiring integration stops at grid values of user-supplied tables. The names used in the common blocks are deliberately "non-commital" so that the structure can be applied to any problem. The structure of the tables is described thoroughly in [3] (The general description of the tables is written for six different tables, with changes for fewer tables requiring simple deletions and with changes for more tables involving established patterns. The example in §5 uses only four tables.)

As far as the evaluation of the ODE system is concerned, the user is given the indices of the bracketing grid entries in the tables. These subscript values are held in common blocks, CON1, CON2, CON3, CON4, CON5 and CON6, (described below). The user makes *no* judgements as to the location of the domain parameters in relation to their respective grids. He merely uses the bounds defined by the indices given in the common blocks. The RKF45T package isolates the grid crossings first stepping over the boundary and then iterating until the table grid value is located accurately. Thus, the given bounding values in the table will define all interpolation, even outside the grid area until the bound is located. Then the grid bounds will be shifted and the new values will represent the function until another bound is isolated. (See Figure 1.) Such a representation of the tabular function gives a "fixed" description of the function and removes chattering around grid bounds as they are crossed. *To repeat*, the user makes *no* judgement about the indices to be used for interpolating. He uses those values provided through common blocks CON1,..., CON6.

The update procedure is "automatic" as far as the subroutines evaluating the ODE are concerned. Thus, the user considers the tables analysis as having two parts. One section defines the tables (reads in data, defines the table size, dimension, etc., and evaluates the function); the second identifies the stopping conditions and updates the boundary indices as the grids are crossed. Although the table structure is described thoroughly in [3], the basic table elements are described briefly below. The analysis of the stopping conditions requires user-supplied subroutines which are described for the minimum-time-to-climb problem in §§4.1, 4.2, and 4.3. Some of these subroutines are

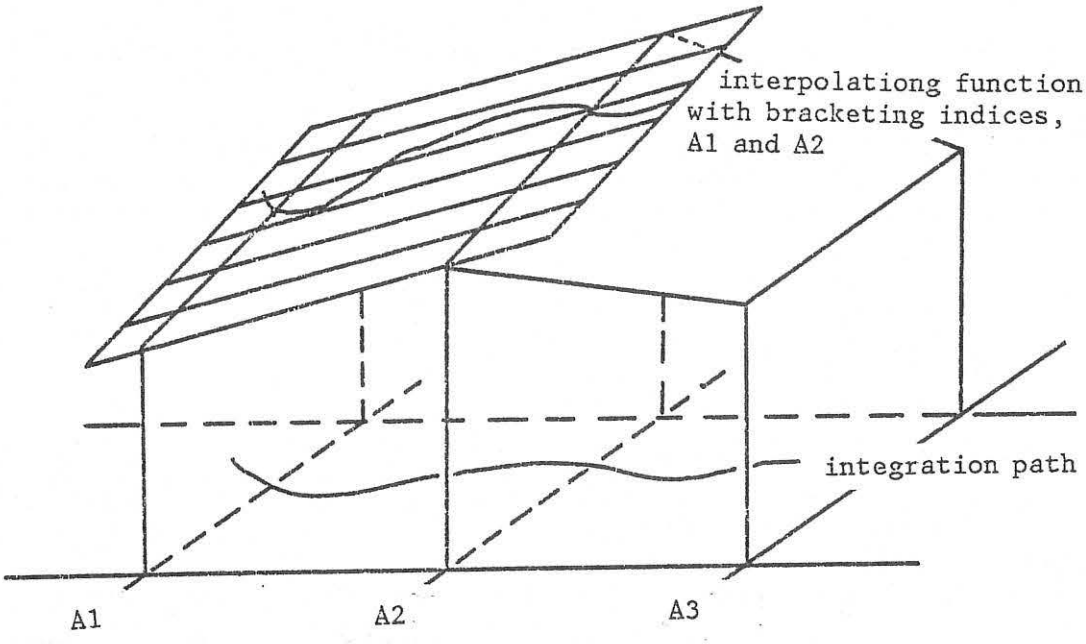


Figure 1. Interpolation limits: Limits remain (A1, A2) even though the integration steps into the (A2, A3) region during the isolation of the A2 bound. After A2 is isolated, the bounds are shifted to (A2, A3)

essentially model routines, requiring few user changes. Others require "user construction".

The most difficult task for the user concerning the tables is that of defining the elements and related parameters. While this job is merely one of bookkeeping, the user must follow a defined structure in order that the communication between RKT45T, TROMPP, and the tables is correct.

Each table is stored in a separate common block, named TABLE1, TABLE2, TABLE3, etc., with the corresponding interpolating indices stored in CON1, CON2, CON3, respectively. The additional common block for describing the tables is TLIMIT. TLIMIT defines parameters related to all tables while TABLE* and CON* refer to the *th table. For the example in §5, the common blocks appear as follows:

```
COMMON/TABLE1/T1P1(26),          TAB1(26)
COMMON/TABLE2/T2P1(24),T2P2(35),TAB2(24,35)
COMMON/TABLE3/T3P1(26),T3P2(11),TAB3(26,11)
COMMON/TABLE4/T4P1(26),T4P2(11),TAB4(26,11)

COMMON/TLIMIT/INDIC(4,2),NGE(4,2),NCOMP(4),NTABLE

COMMON/CON1/KL11,KU11
COMMON/CON2/KL21,KU21,KL22,KU22
COMMON/CON3/KL31,KU31,KL32,KU32
COMMON/CON4/KL41,KU41,KL42,KU42 .
```

The TABLE common blocks define the elements in the tables. The TP1 and TP2 vectors are referred to as domain parameters and may be "over-dimensioned" (where the designating table number is suppressed). Table 1 has only one

domain parameter (T1P1), tables 2, 3, and 4, have two such parameters (T2P1, T2P2 for table 2, etc.) Tables 5 and 6 do not exist. The tabular function, TAB, associates TP1 with the first subscript, TP2, with the second, and TP3, with the third (or as far as the dimensioning goes), for each table. The parameters in TLIMIT are described below. The actual number of TP elements is stored in NGE. (For example, the number of T2P1 elements is given by NGE(2,1)). The TP arrays may be over dimensioned (therefore the need for the NGE values). If the given tables are "full", the NGE values for the given tables would be:

```
NGE(1,1) = 26
NGE(2,1) = 24 ,     NGE(2,2)=35
NGE(3,1) = 26 ,     NGE(3,2)=11
NGE(4,1) = 26 ,     NGE(4,2)=11
```

INDIC identifes the TP parameters, in a general sense, e.g., as Mach number, altitude, or lift coefficient. This vector provides the communication between the tables and the RKF45T package. In the common blocks given (for the example in §5), the "non-committal" names, e.g., T2P1, represent specific parameters. The domain parameters for the example in §5 are listed below:

```
T1P1  is Mach number,
T2P1  is Mach number,      T2P2  is  lift coefficient
T3P1  is Mach number,      T3P2  is  altitude
T4P1  is Mach number,      T4P2  is  altitude .
```

These TP parameters must be associated with the trapping parameters, TRPR, analyzed in the RKF45T program. This association occurs through the vector INDIC. In this example,

```
Mach number       is associated with trapping parameter   1
altitude          is associated with trapping parameter   2
lift coefficient  is associated with trapping parameter   3
```

i.e., Mach number will be associated with TRPR(1), altitude, with TRPR(2), and lift coefficient with TRPR(3) (if the lift coefficient is handled as such a parameter). (The trapping parameters are defined fully in [3]. Here the reader only needs to know that the ordering is arbitrary and user-supplied, and that the trapping parameters serve as the connecting link between the tables and the RKF45T package through the indicator INDIC.)

```
If INDIC(*,**) = J,      then  table number *,  parameter number **
                         is associated with TRPR(J).
```

In the given example, each TP grid representing Mach number must be designated "1" altitude must be designated "2" lift coefficient must be designated "3". This requirement assigns the following values to INDIC:

```
INDIC(1,1) = 1
INDIC(2,1) = 1 ,    INDIC(2,2) = 3
INDIC(3,1) = 1 ,    INDIC(3,2) = 2
INDIC(4,1) = 1 ,    INDIC(4,2) = 2
```

Setting the INDIC parameters correctly is an essential part of the table definition.

NCOMP(*) is the dimension of the TAB* parameter, i.e., the number of domain vectors, not the length of the vectors. For example, NCOMP(3)=2, states that TAB3 is analyzed as a two dimensional table. For the above example,

```
NCOMP(1) = 1
NCOMP(2) = 2
NCOMP(3) = 2
NCOMP(4) = 2   .
```

NTABLE is simply the number of tables being analyzed. (For this example, NTABLE=4.)

The CON common blocks hold the indices of the table parameters to be used in interpolating the tabular functions, i.e., CON1 stores the current interpolating indices for table 1, CON2 for table 2, etc. The bounds used for interpolating the given tables are:

Domain parameters		Tabular function	
Lower bound	Upper bound	Bounding values	
Table 1:			
T1P1(KL11)	T1P1(KU11)	TAB1(KL11)	TAB1(KU11)
Table 2:			
T2P1(KL21)	T2P1(KU21)	TAB2(KL21,KL22)	TAB2(KU21,KL22)
T2P2(KL22)	T2P2(KU22)	TAB2(KL21,KU22)	TAB2(KU21,KU22)
Table 3:			
T3P1(KL31)	T3P1(KU31)	TAB3(KL31,KL32)	TAB3(KU31,KL32)
T3P2(KL32)	T3P2(KU32)	TAB3(KL31,KU32)	TAB3(KU31,KU32)
Table 4:			
T4P1(KL41)	T4P1(KU41)	TAB4(KL41,KL42)	TAB4(KU41,KL42)
T4P2(KL42)	T4P2(KU42)	TAB4(KL41,KU42)	TAB4(KU41,KU42) .

Further details concerning the description of the tables may be found in [3].

4. SUBROUTINES FOR LOCATING GRID POINTS

User-supplied subroutines for isolating grid values of tabular data are also described by example. The routines described here are model routines, where SUBPHI and BOUNCE require few user changes, while PHIPAR and particularly PART will require more user attention. The purpose of each subroutine is described below:

PHIPAR — is structured by the user and gives the current values of the parameters used as stopping conditions (trapping parameters) during the integration.

SUBPHI — evaluates the PHI components after each successful step and references BOUNDS to update limits when a grid bound is isolated.

BOUNDS shifts the indices for the grid values in each table as the grid entries are reached.

PART is used to supply and update the TPART1 and TPART2 vectors for the user-supplied partition (independent variable) stops during the integration.

These subroutines are described in detail in the following subsections. Examples of the routines are listed in Appendix A with clarifying headings.

4.1 Subroutine PHIPAR

Subroutine PHIPAR evaluates the trapping parameters at the current value of t. The trapping parameters, TRPR(I), are the current values of the parameters being trapped. These terms appear in the domain of at least one table, and thus, are designated as TRPR(I) elements which the RKF45T package is to analyze. Each TRPR(I) component has two bracketing values from the table entries which are the current stopping points which the RKF45T package seeks. The user orders the TRPR elements arbitrarily when designing the PHIPAR subroutine. Once the order is established, the user is to convey that information to the tables through the vector, INDIC described in §3 and [3].

The standard calling sequence for PHIPAR is

```
SUBROUTINE PHIPAR(MODE,NEQN,T,Y,YP,NTRPR,TRPR,TRPRP)
```

whose parameters are identified in the program description in Appendix A. The basic structure of the subroutine is user-supplied. The user must be sure that the order selected for the TRPR(I) elements is properly related to the tables through the INDIC vector. The user must supply the current values of TRPR and TRPRP (the derivative of TRPR) when PHIPAR is referenced by the RKF45T package (MODE=1). If the user references PHIPAR separately, MODE may be used to designate the source. Clarifying remarks concerning the calling sequence and use of PHIPAR may be found in the program listing in Appendix A.

4.2 Subroutine SUBPHI

Subroutine SUBPHI evaluates the functions which define the grid crossings in the tables. Once a grid boundary has been isolated, SUBPHI (along with BOUNDS) updates the grid limits in the tables. SUBPHI is essential to the entire trapping procedure and merits a thorough description. The use of SUBPHI is described in detail in [1] and [2]. The main sections of the routine are described below.

SUBPHI has three basic sections: (1) the initialization block, (2) the PHI vector evaluation, and (3) the update portion. The basic structure of SUBPHI need not be changed regardless of the number of tables. Details related to specific problems enter through common blocks (which would have to be altered) and through PHIPAR. Otherwise, the structure of SUBPHI is standard. (The user may wish to modify SUBPHI, however, if additional stopping conditions are desired not related to the tables.)

The PHI vector is treated in each section of SUBPHI, requiring current values of the TRPR vector from PHIPAR. (PHIPAR is referenced immediately upon entry into SUBPHI.) A PHI vector must be chosen which changes sign as it passes through zero, i.e., the PHI function should not "bounce" on a zero. (For a description on "bouncing" see [1].) The function

$$PHI(1) = (ZU - Z)*(Z - ZL)$$

is positive whenever $ZL < Z < ZU$ and negative whenever $Z < ZL$ or $Z > ZU$. Thus, if ZL and ZU identify consecutive grid values, the PHI function suits our purpose quite well. The one difficulty which might occur concerns the "size" of the PHI function, e.g., the magnitude of the altitude is on the order of 1.D+03, while the magnitude of the Mach number is on the order of 1.0D+00 . Thus, a scaled PHI component is introduced

$$PHI(I) = (BOUNDU(I) - TRPR(I))*(TRPR(I) - BOUNDL(I))/ SCALE(I)$$

where $SCALE(I) = 0.5D0 * (BOUNDU(I)-BOUNDL(I)) * (DABS(BOUNDU) + DABS(BOUNDL(I))$.

Each trapping parameter has its own bounds, BOUNDL, BOUNDU, chosen from all of the grids in the tables associated with that parameter. (The greatest lower bound and least upper bound are selected from the associated grids.) The TRPRP vector, the derivatives of TRPR, is generated in PHIPAR (or set equal to zero) so that PHIP(I) terms (estimates, or dummy values) may be evaluated. Both PHI(I) and PHIP(I) are evaluated in DO loops so that no user changes are required to determine the values of these parameters.

4.2.1 The Initialization Block

The initialization is identified through the parameter, KOUNTR=0. In this block, the BOUNDL and BOUNDU values are determined by referencing subroutine INITBD. The structure of INITBD is standard with the only adaptations needed for other programs, being in the two common blocks (which may be copied from SUBPHI). Subroutine WARN is called to check that appropriate bounds have been set. (WARN, listed in Appendix A, prints warning messages if the given bounds are violated. The subroutine is not essential to the table analysis, but provides an important safety check.) The SCALE(I) values are also set in the initialization block using the BOUNDL and BOUNDU values just established. The initialization block, identified by KOUNTR=0, is referenced at the beginning of each iteration. Table limits should have already been established by the user in the TROMPP initialization call (either in subroutine ZWEIGE or in subroutine PART). (SUBPHI has already referenced PHIPAR so the current TRPR values are available.) No discussion of the calling sequence is needed since the communication between SUBPHI and INITBD is not changed by the user.

4.2.2 Evaluating the PHI Functions

The section evaluating PHI(I) and PHIP(I) is straightforward. The PHI functions are defined in function statements using the scaled expression in §4.2.

If the user wants to print the functions, the following information is of importance. SUBPHI is referenced after every integration step. If INDEX = 0, the trapping iteration is not active. If INDEX > 0, PHI(INDEX) is currently being analyzed in the RKF45T system. Thus, the user wanting information only during the trapping iteration (or never during the iteration) has a "flag" in the parameter INDEX.

4.2.3 The Update Section

Once a boundary has been isolated, SUBPHI is referenced in order that the user may make updates in the tables and ODE expressions. The current TRPR values are determined before entering the update section. INDEX indicates which trapping parameter (or actually which PHI component) is being updated. The "lower" or "upper" boundary is identified by checking the derivative of PHI(INDEX) with respect to TRPR(INDEX). (PHI(INDEX) is parabolic (nose up) in terms of TRPR(INDEX), so that if PHIP(INDEX) (with respect to TRPR(INDEX)) is positive, a lower boundary is being crossed, and if the value is negative, an upper boundary is being crossed.) The boundary information is needed in BOUNDS which updates the tables.

4.2.4 Subroutine BOUNDS

When a grid value has been isolated by the RKF45T package, the value must first be identified as an upper or lower boundary. With this information the user knows in which direction to shift the subscripts. The update of the table parameter subscripts, however, is complicated by the fact that several table parameter arrays may be associated with the trapping parameter being updated, e.g., four Mach number grids are used in defining the tables, not all elements being identical. Thus, a separate subroutine, BOUNDS, is referenced to sort through all tables analyzing any domain parameter grid which is being updated, and choosing the bounds from among all of these arrays.

Upon entry into BOUNDS, the update point has been identified as either an upper or lower boundary. BOUNDS compares INDIC(I,J) with INDEX, the index of the TRPR parameter being updated, for each table, I, and each domain parameter, J. If parameter J from table I is to be updated, BOUNDS shifts the bounds one unit in the correct direction. The lower and upper subscript values are stored in IBL(I,J) and IBU(I,J), respectively. TABBND is referenced to set the adjusted IBL(I,J) and IBU(I,J) equal to the corresponding KL and KU values from common block CON for table I, and to set GRIDL(I,J) and GRIDU(I,J) equal to the corresponding TP values from table I parameter grid J. Upon return to BOUNDS, GRIDL(I,J) or GRIDU(I,J) may be adjusted slightly if the isolated boundary has not yet been crossed, to ensure the correct sign of the PHI component at update. As each table is analyzed, the maximum of the adjusted GRIDL values is chosen as the new lower bound for TRPR(INDEX), with the minimum of the adjusted GRIDU values, as the new upper bound.

4.3 Subroutine PART

The user is *required* to supply the subroutine PART which *may* be used to supply a vector, TPART1, of stopping values (independent variable stops) for any forward integration and/or a vector, TPART2, of stopping values for the backward integration. (The user is not required to supply either TPART1 or TPART2.) Part is referenced before any integration is initialized. If a TPART1 vector is supplied, the stopping conditions will be imposed on the independent variable during the forward integration. At each stopping point, PART will be called in an update mode so that the user may update the ODE system if needed. The TPART1 vector will also be imposed on the backward integration (if backward differencing is used), although the user has the opportunity to deactivate its use before the backward integration begins. The user also has the opportunity to impose a second vector of stopping conditions TPART2, on the backward integration. If either TPART1 or TPART2 stops are active, PART will be called in an update mode as the stops are encountered.

The calling sequence for PART is:

```
   SUBROUTINE PART(MODE,TPART1,NTPRT1,ISTOP1,TPART2,NTPRT2,ISTOP2,
  1                FINTEG,NEQN,T,Y,YP)
```

If MODE=1, PART is being called in the initialization mode, i.e., before the integration begins.

If FINTEG = .TRUE., the integration is in the forward direction; if FINTEG = .FALSE., the integration is in the backward direction.

If the user is supplying a TPART1 vector, he must also declare the length of the vector, NTPRT1. Similarly, if the user is supplying a TPART2 vector, the length, NTPRT2, must also be given. In the input mode, ISTOP1 and ISTOP2 play no role. The ODE solution and the derivative evaluation are available, at T, with the variable dimension parameter, NEQN, giving the length of the vectors Y and YP.

If MODE=2, PART is being called in the update mode. If ISTOP1 is non-zero, TPART1(ISTOP1) is being updated at the current value of T, Y, and YP. If ISTOP2 is non-zero, TPART2(ISTOP2) is being updated. Both TPART1 and TPART2 components can correspond to the same value of T, so one may have two updates with one call.

The use of PART is described in the example given in this report. The subroutine is used to supply stops corresponding to lift coefficient grid elements. These stops (TPART1) are imposed in both forward and backward integrations. As the forward integration is performed, additional stops from the RKF45T package are stored to be used as a TPART2 array of stopping conditions in the backward integration. The example PART and related routines are discussed in §5.3.

5. THE MINIMUM-TIME-TO-CLIMB PROBLEM USING LINEARLY INTERPOLATED TABULAR DATA

The RKF45T system with a table structure defined briefly in §3 (and thoroughly in [3]) is used to solve the minimum-time-to-climb problem for the equations of motion given in §5.1.

5.1 State Equations

The equations of motion of an aircraft in the vertical plane in a flight path coordinate system are:

$$V' = g\,[\,(T - D)/W - \sin\gamma\,] \tag{5.1}$$

$$\gamma' = g\,[\,\rho\, S\, V^2\, c_L/(2W) - \cos\gamma\,]\,/\,V \tag{5.2}$$

$$h' = V \sin\gamma \tag{5.3}$$

$$W' = -\,SFC * T \tag{5.4}$$

with V, the velocity, γ, the flight path angle, h, the altitude, and W, the weight. SFC is the specific fuel consumption, supplied by tabular data. The remaining parameters are ρ, the air density, S, the planform area, T, the thrust, and D, the drag. The Mach number is defined as, M = V/a, with a, the speed of sound expressed as a function of h. The thrust is given by

$$T = T_{max}\, c_T\,(M,h)\,\delta$$

where the power setting, δ, will be held constant in the example presented.

With c and δ prescribed as functions of t, and with initial conditions V, γ, h, and W given at t=0, (5.1) - (5.4) may be integrated.

The drag coefficient may be written, $c_D = c_{D_0} + \Delta c_D$ giving

$$D = [c_{D_0} + \Delta c_D]\ \rho\, S\, V^2 / 2$$

where $c_{D_0} = c_{D_0}(M, h)$ and $\Delta c_D = \Delta c_D(M, c_L)$

are both given in terms of tabular data which is to be interpolated linearly (using the bilinear interpolation formulas given in [3]). The additional parameters, $c_T = c_T\,(M,h)$ and SFC = SFC(M,h), are also expressed in terms of tabular data which is to be interpolated similarly.

The $c_L\,(t)$ (and $\delta(t)$) controls are to be determined so that the final conditions: V_f, γ_f, and h_f are satisfied with

(5.5) $t_f \equiv$ minimum (minimum time)

or

(5.6) $W(t_f) \equiv$ maximum (minimum fuel consumption)

Optimal control theory gives the conditions under which the cost function (5.5) or (5.6) reaches an extremum. The Hamiltonian function is:

(5.7) $H \equiv -\lambda_V V' - \lambda_\gamma \gamma' - \lambda_h h' - \lambda_W W'$

The Lagrangian multipliers, determined from the Euler equations:

(5.8) $\lambda_V' = -\lambda_V g [\partial T/\partial V - \partial D/\partial V]/W - \lambda_\gamma g [\rho S c_L/(2W) + (\cos \gamma)/V^2]$

$- \lambda_h \sin \gamma + \lambda_W [T \partial SFC/\partial V + SFC \partial T/\partial V]$

(5.9) $\lambda_\gamma' = \lambda_V g \cos \gamma - \lambda_\gamma g (\sin \gamma)/V - \lambda_h V \cos \gamma$

(5.10) $\lambda_h' = -\lambda_V g [\partial T/\partial h - \partial D/\partial h]/W - \lambda g V S c_L (\partial \rho/\partial h)/(2 W)$

$+ \lambda_W [T \partial SFC/\partial h + SFC \partial T/\partial h]$

(5.11) $\lambda_W' = \lambda_V g (T - D)/W^2 + \lambda_\gamma g \rho S V c_L/(2 W^2)$

The partial derivatives with respect to V and h are:

$\partial T/\partial V = T_{max} (\partial c_T/\partial V) \delta$, where $\partial c_T/\partial V = (\partial c_T/\partial M) (\partial M/\partial V)$

$\partial D/\partial V = \rho S V c_D + \rho S V^2 (\partial c_D/\partial V) / 2$, where $\partial c_D/\partial V = (\partial c_D/\partial M) (\partial M/\partial V)$

$\partial SFC/\partial V = (\partial SFC/\partial M) (\partial M/\partial V)$

$\partial T/\partial h = T_{max} [\partial c_T/\partial h + (\partial c_T/\partial M) (\partial M/\partial a) (\partial a/\partial h)] \delta$

$\partial D/\partial h = [(\partial \rho/\partial h) V^2 S c_D + \rho V^2 S (\partial c_D/\partial M) (\partial M/\partial a) (\partial a/\partial h)] / 2$

$\partial SFC/\partial h = \partial SFC/\partial h + (\partial SFC/\partial M) (\partial M/\partial a) (\partial a/\partial h)$

In the case of linearly interpolated data all the above expressions have jumps at each of the corresponding table grid entries. Data for the given problem has been taken from [6], describing the F-4 Phantom.

5.2 Defining the Right-Hand-Sides

The right hand sides of the ordinary differential equations contain discontinuities if the tables used in determining the drag coefficient, thrust coefficient, and specific fuel consumption are interpolated linearly. The RKF45T package is used to isolate each of the discontinuities as they are encountered. The discontinuities correspond to the grid entries of the domain vectors defining the tables. Thus, these points are imposed as stopping conditions for the RKF45T system. More specifically, the trapping parameters, (Mach number, lift coefficient, and altitude) are located in each of their respective table grids with the current bracketing indices from these grids used as stopping values. When a grid value is isolated, the bracketing indices are shifted and "new" limits are imposed. The ODE is then reevaluated always using the current indices.

When the user programs the right hand sides of the ODEs, he makes no judgements as to the location of the trapping parameters. He is supplied with the current bracketing indices for each trapping parameter in each table through the common blocks CON1, CON2, CON3 and CON4 with the values of the domain grids and tabular functions in TABLE1, TABLE2, TABLE3, and TABLE4. (See §3.) The ODEs are to be written in terms of this information. When a shift in indices occurs, the shift takes place outside the RHS system of subroutines. All information in RHS is current, corresponding to conditions at T.

The subroutines describing the RHS system for the equations in §5.1 are listed in Appendix A. This system includes: RHS, FDRAG, FTRIEB, FATM, LIFTC, PLIFTC, MATRX1, and MATRX2 . The "main" program of this system is RHS which "organizes" the YP values. The additional subroutines supply the following terms for RHS:

Subroutine	Values supplied
FDRAG	determines drag and partial derivatives of drag with respect to velocity and altitude
FTRIEB	determines thrust and partial derivatives of thrust with respect to velocity and altitude
FATM	determines density and air speed and derivatives with respect to altitude
LIFTC	determines lift coefficient and derivative of lift coefficient with respect to T
PLIFTC	determines lift coefficient and derivative of lift coefficient with respect to T, using perturbed coefficients from the TROMPP system (used in generating the gradient evaluations)
MATRX1	performs one dimensional matrix multiplication (for forming the gradient evaluations)

MATRX2 performs two dimensional matrix multiplication subroutine (for forming the gradient evaluations)

The FDRAG and FTRIEB subroutines need further clarification since these two routines handle the tabular data. The other routines are documented sufficiently in the listing in Appendix A.

5.2.1 Subroutine FDRAG

In determining the drag, subroutine FDRAG requires tabular data from TABLE1 and TABLE2 and corresponding indices from CON1 and CON2. These values will have already been set each time that RHS is referenced.

Since FDRAG is a particular application, the parameters in the common blocks have been renamed to make identification easier. The common blocks, as they appear in FDRAG, are:

```
COMMON/TABLE1/ AM1(26),           TAB1(26)
COMMON/TABLE2/ AM2(24), CA2(35), TAB2(24,35)
COMMON/CON1/   KML1,KMU1
COMMON/CON2/   KML2,KMU2,KCAL2,KCAU2
```

The bracketing grid values of c_0 are:

AM1(KML1) giving the current lower bracketing index on Mach number
AM1(KMU1) giving the current upper bracketing index on Mach number

Table 2 gives the Δc_0 data as a function of Mach number and lift coefficient. The bracketing grid values are:

AM2(KML2) giving the current lower bracketing index on Mach number
AM2(KMU2) giving the current upper bracketing index on Mach number

CA2(KCAL2) giving the current lower bracketing index on lift coefficient
CA2(KCAU2) giving the current upper bracketing index on lift coefficient .

The bilinear interpolation formulas along with the derivative formulas are given as function statements (See [3].). Thus, when an interpolated value is generated, the user need only be concerned with typing the subscripts correctly.

The FDRAG subroutine may be used in two modes. Mode 1 determines only the drag. Mode 2 determines drag and its partial derivatives with respect to velocity and altitude (which are needed if backward differencing is used to generate gradients in TROMPP).

5.2.2 Subroutine FTRIEB

In determining thrust, subroutine FTRIEB requires tabular data from TABLE3 and TABLE4 and corresponding indices from CON3 and CON4. These values will have already been set each time that RHS is referenced.

Since FTRIEB is a particular application, the parameters in the common blocks have been renamed to make identification easier. The common blocks, as they appear in FTRIEB, are:

```
COMMON/TABLE3/ AM3(26),  H3(11), TAB3(26,11)
COMMON/TABLE4/ AM4(26),  H4(11), TAB4(26,11)
COMMON/CON3/   KML3,KMU3,KHL3,KHU3
COMMON/CON4/   KML4,KMU4,KHL4,KHU4
```

Table 3 gives the Csi data as a function Mach number and altitude. The bracketing grid values are:

AM3(KML3)	giving the current lower bracketing index on Mach number
AM4(KMU4)	giving the current upper bracketing index on Mach number
H3(KHL3)	giving the current lower bracketing index on altitude
H3(KHU3)	giving the current upper bracketing index on altitude.

Table 3 gives the SFC data as a function Mach number and altitude. The bracketing grid values are:

AM4(KML4)	giving the current lower bracketing index on Mach number
AM4(KMU4)	giving the current upper bracketing index on Mach number
H4(KHL4)	giving the current lower bracketing index on altitude
H4(KHU4)	giving the current upper bracketing index on altitude.

The bilinear interpolation formulas along with the derivative formulas are given as function statements. Thus, when an interpolated value is generated, the user need only be concerned with typing the subscripts correctly.

The FTRIEB subroutine may be used in two modes. Mode 1 determines only the thrust. Mode 2 determines thrust and its partial derivatives with respect to velocity and altitude (which are needed if backward differencing is used to generate gradients in TROMPP).

5.2.3 Setting-up the Trapping System

The GDE expression, described in §5.1, requires no knowledge of the trapping system, merely the use of the indices supplied as a result of the trapping procedure. Now the user must forget the RHS system and concern himself with setting the traps for the indices. The communication between these two systems occurs through the common blocks TABLE* and CON*, where, for this example, * = 1,2,3, or 4.

The table structure is thoroughly described in [3], including the definitions of the related common blocks. That report also gives an example of subroutines for initializing and later adjusting the bracketing values of the TRPR values for linear interpolation. Once the table structure is understood, the trapping analysis centers on the generation of the stopping vectors, TPART1 and TPART2, and on the use of SUBPHI to stop the integration at the grid boundaries not handled in TPART1. The TPART1 vector is established before each integration in subroutine PART. PART is also responsible for the updates of the TPART1 vector. TPART2 is established through the SUBPHI updates as the

forward integration progresses. In the backward integration, the SUBPHI analysis is suspended, and PART becomes responsible for updating conditions at each TPART2 stop.

5.3 Subroutine PART

The general description of PART (§4.3) needs to be extended, since, for this example, PART supplies the stopping conditions for the lift coefficient in the forward integration and all stopping conditions in any backward integration. While many of the remarks in this section pertain to the current example, the pattern may be extended to the general problem. The PART routine for the current example is listed in Appendix A.

5.3.1 Subroutine PART, Mode 1, Forward Integration

Mode 1 initializes conditions for the forward integration. If TROMP is referenced in the forward differencing mode (IMPULS=.FALSE.), the TPART1 vector is determined but no TPART2 analysis is activated. If TROMP is referenced in the backward differencing mode (IMPULS=.TRUE.), the TPART1 analysis is again active. If the gradient is being generated (IG=2), the TPART2 vector is also initialized.

5.3.1.1 Subroutine PART, Mode 1, Forward Integration, Forward Differencing

In mode 1 during the forward integration, PART is asked to supply the TPART1 vector (and define the length, NTPRT1). (NTPRT1=0 upon entry into PART. Failure to reset NTPRT1 will activate no TPART1 vector stops.) PART may also be used for the general initialization of the particular integration.

The example presented uses PART to set the table indices before the integration begins, by referencing TABLIM. (See [3].) Thus, the RHS system is given the proper information at T=0, before being referenced. PART then proceeds to locate T values which correspond to the lift coefficient grid entries in Table 2, since the lift coefficient, expressed as a cubic spline, can be analyzed analytically. The T values corresponding to the lift coefficient stops will be imposed as the TPART1 vector. PART references PARTCA to identify these T values. Transfering the analysis to PARTCA (and further to SHIFT and QROOT) is designed to streamline the logic in PART. (PARTCA, SHIFT, and QROOT are described in [3].) The TPART1 vector is returned to PART from PARTCA and the initial indices are stored in common block CON2, corresponding to the second domain parameter (which is the lift coefficient). Thus, the lift coefficient grid stops are completely identified before the integration begins.

5.3.1.2 Subroutine PART, Mode 1, Forward Integration, Backward Differencing

In mode 1 (during the forward integration) with backward differencing being used, PART establishes the TPART1 vector as in the forward differencing analysis (§5.1.1.1). If a gradient is being evaluated, a backward integration will be used in which the stopping points located by the SUBPHI program will be imposed as the TPART2 vector. The initial point in the TPART2 vector (T=0) along with the associated parameters (in this example, the LPOINT vector and NPOINT) must be set. The parameters associated with the TPART2 stops are described in §5.4.1.)

5.3.2 Subroutine PART, Mode 2, the Update Mode

Mode 2 of PART deals with the update of a TPART1 or a TPART2 vector element. The analysis involves the shifting of indices and reevaluation of the derivative to correspond to the adjusted conditions. If forward differencing is being used, only a TPART1 vector stop is possible. If backward differencing is being used, then the TPART1 vector stops are updated in the forward integration in the same manner used in forward differencing. In the backward integration, both the TPART1 and TPART2 vector stops are imposed, with the type of update identified through ISTOP1 and ISTOP2. Once the "type of stop" is identified, the update again consists of shifting indices and reevaluating the ODE.

5.3.2.1 Subroutine PART, Mode 2, Forward Integration, Forward or Backward Differencing

Mode 2 during the forward integration identifies the update of a TPART1 stop. (Only the TPART1 stops are active in PART during the forward integration, so no update confusion occurs between TPART1 and TPART2.) ISTOP1 is supplied to identify the component of TPART1, i.e., T=TPART1(ISTOP1). In the initialization phase, PARTCA delivered not only TPART1(J) values but also corresponding IPARTL(J) and IPARTU(J) values to identify the bracketing indices of the T2P2 (lift coefficient) grid. (The parameters are identified as TVECT, IVECTL, and IVECTU in PARTCA.) The IBL(2,2) and KL22 values are set equal to the IPARTL(ISTOP1) value, while the IBU(2,2) and KU22 values are set equal to the IPARTU(ISTOP1) value. The GRIDL(2,2) and GRIDU(2,2) values are then set equal to the T2P2(KL22) and T2P2(KU22) values, respectively. Once the bounds have been shifted, the user must evaluate the derivative since the slopes will change after update. WARN is also referenced to ensure that the bounds are set correctly. Thus, once the TPART1 vector is established in the initialization mode of PART, the analysis of the TPART1 vector consists of merely shifting the subscripts and reevaluating the derivative at update.

5.3.2.2 Subroutine PART, Mode 2, Backward Integration, (Backward Differencing)

Mode 2 during the backward integration updates either a TPART1 or a TPART2 stop. The type of stop is identified through the parameters ISTOP1 and ISTOP2. (ISTOP1=JJ, JJ non-zero, identifies the stop as TPART1(JJ); ISTOP2=KK, KK non-zero, identifies the stop as TPART2(KK), where both a TPART1 and TPART2 stop may be active at the same update point.) The TPART1

updates are similar to those in the forward integration, the difference being that the direction of integration is reversed. The TPART2 updates involve the analysis of the three-dimensional vector LPOINT, (described in detail in §5.4.1). LPOINT(I,J,ISTOP2)=LL, LL non-zero, indicates the |LL|th point in table I, table parameter J. ISTOP2 is the index of the TPART2 element used as the stopping condition. LL is used to determine the new lower bound on the parameter being updated. An example of the index adjustment is given in §5.4.1. Once the IBL and IBU bounds are established, the update procedure is similar to that followed in the lift coefficient update.

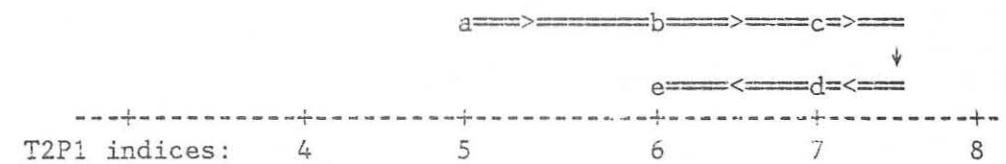
5.4 SUBPHI Stops for Establishing the TPART2 Vector

Section 4.2 describes the use of SUBPHI for stopping the integration whenever a component of PHI vanishes (i.e., in this example, whenever a Mach number or altitude grid entry is crossed). During the boundary shifting analysis in the forward integration, the user may store information about the current stopping values so that they may be used as the TPART2 stopping conditions during the backward integrations (avoiding the trapping analysis a second time). This analysis is performed in BOUNDS if the forward integration is active and a gradient evaluation (IG=2) is being generated in TROMPP. The TPART2 vector is named PVECT with the three dimensional vector LPOINT identifying the table, table parameter, and PVECT point and with NPOINT identifying the number of points currently in PVECT. LPOINT(I,J,K) = LL, gives information about table I, table parameter J, for the current point, TPART2=PVECT(K). If LL=0, no update is being made concerning this table parameter. If the update occurs at the current lower bound (the new upper bound), LL < 0. If the update occurs at the current upper bound (the new lower bound), LL > 0. In either case |LL| is the value of IBL(I,J) AFTER the bounds are updated.

5.4.1 Clarification of LPOINT values at Update

The following example is given to clarify the update procedure for the TPART2 vector elements using the appropriate LPOINT values.

Assume that the following grid is being analyzed, the T2P1 grid with the integration stops during the forward integration being labeled "a,b,c,d, and e".



Integration stop	New Bounds IBL	IBU	LPOINT(2,1,*)
a	5	6	+ 5
b	6	7	+ 6
c	7	8	+ 7
d	6	7	- 6
e	5	6	- 5

For example, at update of "a", the new bounds will be IBL(2,1)=5 and IBU(2,1)=6. The update occurs at the new lower bound (the previous upper bound), and therefore, LPOINT(2,1,a) is given a positive sign, the magnitude being IBL(2,1) after update. At update of "d", the new bounds are IBL(2,1)=6, IBU(2,1)=7, where "d" is at the new "upper" boundary (the previous lower bound). Thus, LPOINT has a "negative" sign, the magnitude being the new IBL(2,1).

In integrating backwards, the process starts at "e" and moves toward "d". The update formula for IBL and IBU at "d" when integrating from T=1 to T=0 is:

$$IBL(2,1)_{new} = |LPOINT(2,1,d)| - ISIGN(1,LPOINT(2,1,d))$$
$$= 6 - (-1) = 7$$

$$IBU(2,1)_{new} = IBL(2,1)_{new} + 1 = 8$$

which does indeed represent the conditions for integrating from "d" towards "c". To continue the example (with LPOINT > 0), as the integration reaches "c" (from the "d" direction),

$$IBL(2,1)_{new} = |LPOINT(2,1,c)| - ISIGN(1,LPOINT(2,1,c))$$
$$= 7 - (+1) = 6$$

$$IBU(2,1)_{new} = IBL(2,1)_{new} + 1 = 7$$

which sets the bounds correctly for integrating from "c" towards "b". The general formula for updating the IBL coefficient for table *, parameter **, at TPART2(ISTOP2) is

$$IBL(*,**)_{new} = |LPOINT(*,**,ISTOP2)| - ISIGN(1,LPOINT(*,**,ISTOP2))$$

$$IBU(*,**)_{new} = IBL(*,**)_{new} + 1 \ ,$$

where LPOINT(*,**,ISTOP2) is non-zero. If LPOINT(*,**,ISTOP2)=0, the integration stop does not correspond to the table *, parameter ** grid and no update should be made.

6. SUBROUTINES FOR TREATING TABULAR DATA

Due to the large number of subroutines used in solving the problem in §5, an alphabetical listing of the subroutine names and their main role in the solution of the problem are given. A second list is given providing the subroutines used in the RKF45T system. The subroutines are listed in Appendix A, with clarifying remarks in the comment cards.

6.1 Subroutines Associated with the Minimum-Time-to-Climb Problem

In solving the minimum-time-to-climb problem using tabular data with integration stops at the domain parameter entries, the driving program, PHANT, uses the following subroutines:

BOUNDS — used in conjunction with SUBPHI to shift bounding limits for PHI components

COSTF — referenced by TROMPP to evaluate the cost function (and partial derivatives of constraint functions if a gradient evaluation is being made)

EX — EX1DT1 and EX2DT2 referenced by TABLES to extend the Mach number grid at the lower limit on tables 1 and 2

FATM — referenced by RHS to determine the density of the air and the speed of sound along with the derivatives with respect to altitude

FDRAG — referenced by RHS to determine the drag (and the partial derivatives if a gradient evaluation is being determined by backward differencing)

FTRIEB — referenced by RHS to determine the thrust (and the partial derivatives if a gradient evaluation is being determined by backward differencing)

GRPART — referenced by TROMPP to establish the current integration limit (either a cubic spline knot or a TPART1 or TPART2 value) (Flow chart is given in Appendix B.)

INITBD — used in conjunction with SUBPHI to set initial bounding limits for PHI components

INSERT — referenced by QROOT to insert the current roots into a vector of such roots so that the vector elements are monotonic increasing

LIFTC — referenced by RHS to compute lift coefficient (and possibly derivative of lift coefficient with respect to t) using the assumed control function (cubic spline) (See PLIFTC)

LOCATE — referenced by TABLIM to locate the given TRPR value within a specified array and to return the bracketing indices from that array along with the bracketing array elements to be used as the current bounds for that table domain member

MATRX1 — referenced by RHS to evaluate the dot product, B •Y, two vectors of length 4

MATRX2 — referenced by RHS to perform matrix multiplication between a 4x4 matrix and a vector

PART — referenced by TRPART (TROMPP) to establish or update the TPART1 or TPART2 vector components

PARTCA	referenced by PART to set the TPART1 vector of lift coefficient stops
PHIPAR	referenced by SUBPHI and PART to determine the trapping parameters, TRPR and TRPRP
PLIFTC	referenced by RHS to compute lift coefficient (and possibly derivative of lift coefficient with respect to t) using the perturbed control function (cubic spline) for the forward differencing integrations
PR	PR1DT1, PR2DT2, PR2DT3, PR2DT4 referenced by TABLES to print the tables for verification
QROOT	referenced by PARTCA to solve for the roots of a cubic equation
QMXMN	referenced by QROOT to establish maximum and minimum bounds on a given cubic equation
RD	RD2DT2 referenced by TABLES to reduce the number of lift coefficient entries in Table 2
REATAB	an "inherited" subroutine which reads the data from files
RHS	referenced to determine the derivative evaluations
SHIFT	referenced by PARTCA to adjust cubic coefficients from the spline package to suit representation in QROOT
SLLSQP	(Sequential Linear Least Squares Program) due to K. Schittkowski and D. Kraft used as the optimization package [5]
SUBPHI	referenced by the RKF45T system to supply the values of the PHI and PHIP functions or to update a located zero of PHI
TABLE	referenced by the driving program to set data into the table common blocks and to set parameters defining the table dimensions
TABLIM	referenced by PART to establish the bracketing indices on the trapping parameter value in each table grid
TABBND	referenced by BOUNDS during each update to shift the indices on table parameter values being updated
TROMPP	Trajectory Optimization by Mathematical Programming with zero-trapping capabilities and added stopping partititions (modified version of TOMP [4]) used to determine cost function and gradient evaluations
TRPART	referenced by TROMPP to obtain TRPART1 or TPART2 vectors or to update integration stops corresponding to an element in one of the two arrays (Flow chart is given in Appendix B.)
WARN	designed for checking to see if current table bounds bracket the associated TRPR elements. A "time-lag" is built in to discontinue the warning analysis if the trapping iteration is in effect.

ZWEIGE is a dummy routine in this application. ZWEIGE is generally used to initialize branching indicators when a small number of discontinuities is being analyzed (See [1].)

6.2 Subroutines Associated with RKF45T System

The RKF45T system is documented in [2]. The subroutines of major importance in the RKF45T package are:

RKF45T which is the driving program for the package which acts as an interface between the user and the internal workings of the integrator,

RKFST which is the decision making subroutine for the integration,

SETRAP which serves as an interfacing routine between RKFST and TRAPPD monitoring the PHI elements to see if the trapping procedure should be activated to isolate a zero of any PHI component,

TRAPPD which uses an iterative procedure to isolate the zeros of the PHI components which have changed sign over a given integration step,

SCALED which evaluates the solution at a point within a given integration step during the trapping procedure, and

VANISH which checks to see if the solution has vanished throughout the step (referenced by TRAPPD at update).

Subroutines associated with the RKF45T system (of lesser importance) are:

BOUNCD which analyzes the PHI components to see if any have "bounced" on a zero (In general, the RKF45T system does not locate bouncing zeros, but if one is detected, it is treated in a special manner.),

FLAGCK which adjusts IFLAG if the trapping option is being used,

OUTFLG which prints warning messages if IFLAG indicates difficulties,

PANIC which serves as an emergency option which prints dense output throughout a step where difficulties are about to terminate the trapping iteration,

SHIFTI which checks to see if INDEX (the parameter indicating the component of PHI currently being trapped) should be shifted, and

TSTAR which estimates the value of T that will cause PHI(INDEX) to vanish. (The value of TSTAR establishes the next step size to be taken by TRAPPD.)

Common blocks in RKF45T are to be avoided. To aid in debugging, however, one form of the package is available with a block data subprogram for specifying the printing indices for TRAPPD and related subroutines. Sufficient comment cards are included so that the user knows which constants activate printing in which subroutines.

7. COMMON STATEMENTS

A great deal of information is passed through the presented example in the form of common blocks. In order to help the user keep track of the elements, the following list is presented giving the common block name and the subroutines in which the blocks are located. This information is particularly important if the user needs to change the length of any arrays in the blocks. A few of the common blocks are inserted for debugging during the development of the program and may be deleted. These blocks are denoted with an * before their listing.

COMMON block	Subroutine
ATHSF3:	TABLE, REATAB
ACWOCA:	TABLE, REATAB
BNDVAL:	DRIVER, COSTF
CHCKCA:	RHS, WARN
CON1:	FDRAG, PART, TABLIM, TABBND, WARN
CON2:	FDRAG, PART, TABLIM, TABBND, WARN
CON3:	FTRIEB, PART, TABLIM, TABBND, WARN
CON4:	FTRIEB, PART, TABLIM, TABBND, WARN
*CRKF45:	BOUNDS, DRIVER, SUBPHI
CTRMPP:	DRIVER, COSTF, BOUNDS, LIFTC, PLIFTC, PARTCA, RHS, TROMPP
DUMMY:	WARN
DTABL2:	TABLE, R2DT2
*EMERG:	FDRAG, FTRIEB
FKOUNT:	DRIVER, RHS,
*FSTEP:	DRIVER, FDRAG, FTRIEB, SUBPHI, RHS,
GRIDBD:	BOUNDS, INITBD, PART, SUBPHI, TABLIM, TABBND, WARN
IDENT:	DRIVER, SUBPHI, TROMPP, WARN
PARTV:	PART, BOUNDS
TABLE1:	FDRAG, PART, TABLE, TABLIM, TABBND, WARN
TABLE2:	FDRAG, PART, PARTCA, TABLE, TABLIM, TABBND, WARN
TABLE3:	FTRIEB, PART, TABLE, TABLIM, TABBND, WARN
TABLE4:	FTRIEB, PART, TABLE, TABLIM, TABBND, WARN

TLIMIT: BOUNDS, FDRAG, FTRIEB, INITBD, PART, PARTCA, SUBPHI, TABLE, TABLIM, TABBND, WARN

* denotes common blocks for checking difficulties which are not essential to the table analysis

8. COMPUTATION: INITIALIZATION AND RESULTS

With the table structure (defined in [3]) used to construct the right hand sides as given in §§5.1 and 5.2, and with the update procedure as defined in §§5.3 and 5.4, one may apply an optimization package to the equations of motion described in §5.1 and be assured that the tabular data representation will remain fixed from iteration to iteration. (That is, with each grid boundary crossing as an integration stopping point, and with updates in the ODE as the grids are crossed, the tabular functions have a definite structure which will not change regardless of the control parameter description or of the integration path through the table domain.) With the set-up procedure described in the preceding sections, the user may now apply any mathematical programming package to optimize the solution. As long as the output is "warning free", the trapping procedure is proceeding smoothly.

The SLLSQP (Sequential Linear Least Squares Program) is applied to the problem described in §5.0, with TROMPP used to generate the cost function and gradient evaluations. The lift coefficient serves as the only control function, being described as a cubic spline. The initialization parameters are listed in the driving program, PHANT, in Appendix A, and are printed in the resulting output in Appendix C. (This report assumes that the user is familiar with the TROMPP (or at least TOMP [4]) software package, so that the meaning of the input parameters is clear.

The initial estimates for the control function data (lift coefficient to be expressed as a cubic spline) is given in a data statement in the driving program. Control bounds, initial conditions, and other pertinent information appear in both the driving program (listed in Appendix A) and in the computational results.

The OCP consists of finding the minimum time required to reach a given altitude with specified final velocity and flight path angle. This problem has been run using both forward and backward differencing in generating the partial derivatives. As long as no warning messages are printed, the RKF45T/TROMPP analysis is stopping the integration as each grid entry is encountered permitting the updating of the ODE system. The t values corresponding to the grid partition stops are given in Appendix C as well as those for the optimal solution. In the backward mode, 30943 derivative evaluations were required to solve the problem of which 2650 were used in generating the grid stops (less than 9%). The results using forward differencing lead to essentially the same optimal solution. A comparison between the derivative counts using forward and backward differencing methods could be misleading since the adjoint system used in the backward differencing increases the size of the ODE (although the use of adjoint system decreases the total derivative count. (The actual function count needed for forward differencing is 113635 of which 13334 are used for the trapping analysis.) Both forward and backward differencing schemes converged in 21 iterations. A comparison of CPU

time shows that 18.05 sec are needed to solve the problem using backward differencing while 35.61 sec are required using forward differencing. If trapping is used in both the forward and backward integrations, the CPU time increases to 19.33, an increase of some 7% over the time required if the stopping points are stored during the forward integration. (Treating the problem with forward and backward trapping,however, avoids a great deal of bookkeeping work for the user.)

9. CONCLUSIONS

The use of linearly interpolated tabular data in system equations greatly simplifies the programming efforts in flight dynamics problems. Such a model, permits an easy interchange of data as new aircraft are tested on the computer. In contrast, using smooth curve fitting techniques to represent tabular data, while mathematically pleasing, requires a great deal of off-line computing time to generate the curve fits, and new data or changes in existing tables require repeating the curve fitting process from the beginning. The use of linearly interpolated data to represent tabular information, however, introduces discontinuities in the higher order derivatives in the ODE system. The integration errors introduced by failing the isolate these points of discontinuity precisely affect the individual integrations to some degree and are particularly detrimental to the convergence rate in the optimization process. More specifically, more time is wasted if the integrator attempts to locate the points of discontinuity by itself rather than by using an efficient (and more accurate) convergence scheme, and the errors from the individual integrations "confuse" the optimization process in that the method can not distinguish between perturbations in the optimal estimates and the integration "noise" from errors caused by the errors due to the discontinuities.

In this report, a table structure is defined which permits the use of numerous tables which may have the domain functions but whose grids need not be identical. The communication between the tables and the RKF45T package is defined so that the RKF45T/TROMPP package may isolate the grid crossings in the various tables, as each is encountered. Updates in the ODE system are then made so that the ODE expressions reflect the current conditions in the tables. The ODE evaluation subroutines themselves are given the current bracketing values for the linear interpolation. These subroutines make no decisions as to the locations of the current parameters in the table domains. All decisions concerning the changing of these bracketing indices occur in the RKF45T system. Such a structure simplifies the programming in the ODE expressions. The RKF45T subroutines for analyzing the tables are generally model subroutines whose extensions require basic pattern recognition. Thus, the given example can be extended readily to different types of problems in which linearly interpolated tabular data is used. Increases or decreases in table number and/or size pose no difficulties. The system passes through a large region of the table domain in the given example without encountering any difficulties in the table grid locations or in updating.

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. 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.

10. REFERENCES

[1] Horn, M.K. — A Numerical Solution of State/Control-Constraint Optimal Control Problems with Piecewise Continuous Derivatives Using RKF45T
DFVLR-IB 515-83/2

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

[3] Horn, M.K. — Subroutines for Handling Tabular Data Used in System Equations
DFVLR-IB 515-82/16

[4] Kraft, D. — FORTRAN-Programme zur numerischen Lösung optimaler Steuerungsprobleme
DFVLR-Mitteilung 552-80/3

[5] Kraft, D.
Schittkowski, K. — Theorie und Anwendung der sequentiellen quadratischen Programmierung in Steuerungs- und Regelungsaufgaben
DFVLR-Forschungsbericht (in Bearbeitung)

[6] Well, K.H.
Kraft, D.
Berger, E. — Zwischenbericht über die Entwicklung eines schnellen Optimierungsverfahrens zur Berechnung ebener, zeit- und verbrauchsoptimaler Bahnen von Kampfflugzeugen
DFVLR-A 552-76/8.

APPENDIX A. LISTING

1. SUBROUTINES USED IN SOLVING THE MINIMUM-TIME-TO-CLIMB PROBLEM

Subroutine BOUNDS

```
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
      SUBROUTINE BOUNDS(INDEX,PARAM,T,TOLER,BNDRYL,BNDRYU,UPPER,
     1                  NTRPR,SCALE)
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
C     PURPOSE:
C         TRAPPING ANALYSIS HAS JUST ISOLATED A ZERO WITH TRAPPING
C         PARAMETER VALUE "PARAM", TRAPPING PARAMETER NUMBER=INDEX.
C         BOUNDS SHIFTS TABLE BRACKETING INDICES WHICH HAVE THIS
C         PARAMETER AS A DOMAIN MEMBER.
C
C         PARTITION FOR INTEGRATION STOPS SET IF BACKWARD DIFFERENCING
C         SCHEME IS USED.
C
C     STRUCTURE:      STANDARD (SEE USER CHANGES)
C
C     INPUT PARAMETERS:
C         INDEX       TRAPPING PARAMTER INDICATOR
C         PARAM       VALUE OF TRAPPING PARAMTER AT ZERO PHI(INDEX)
C                     POINT
C         T           VALUE OF INDEPENDENT VARIABLE
C         TOLER       CONVERGENCE TOLERANCE
C         FORWRD      LOGICAL PARAMETER,
C                     FORWRD=.TRUE.  ==>  FORWARD DIFFERENCING SCHEME
C                     FORWRD=.FALSE. ==>  BACKWARD DIFFERENCING SCHEME
C
C     OUTPUT PARAMETERS:
C         BOUNDL      MAXIMUM LOWER BOUND FROM ALL ANALYZED GRIDS
C         BOUNDU      MINIMUM UPPER BOUND FROM ALL ANALYZED GRIDS
C
C         GRID BOUNDS AND INDEX LIMITS ALSO ADJUSTED BY CALL TO
C         TABBND (HELD IN COMMON BLOCK GRIDBD)
C
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
C  USER CHANGES (MINIMAL):
C
C  1) USER MUST SUPPLY PROPER DIMENSIONING VALUES IN COMMON BLOCKS.
C
C     A) SEE SUBROUTINES TABLIM FOR DIMENSION OF COMMON BLOCKS TLIMIT
C           AND GRIDBD
C
C     B) DIMENSIONING OF PVECT AND LPOINT:
C           LIMP = DIMENSION OF PVECT IS SET AT 30. IF USER INCREASES
C                  THE SIZE OF PVECT, HE MUST MAKE THE SAME CHANGES
C                  IN COMMON BLOCK  PARTV IN SUBPHI.
```

```
C
C            COMPARE LPOINT DIMENSION WITH IBL(II,JJ)
C
C            DIMENSION PVECT(LIMP), LPOINT(II,JJ,LIMP)
C
C
C  2) FCT AND FCTP MUST HAVE THE SAME EXPRESSIONS AS IN SUBPHI
C                  ====
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
C
      IMPLICIT REAL*8 (A-H,O-Z)
C
      DIMENSION SCALE(NTRPR)
C
      COMMON/TLIMIT/INDIC(4,2),NGE(4,2),NCOMP(4),NTABLE
      COMMON/GRIDBD/GRIDL( 4, 2),GRIDU( 4, 2),IBL( 4, 2),IBU( 4, 2)
      COMMON/PARTV/PVECT( 30),LPOINT( 4, 2, 30),NPOINT
      DATA          LIMP/ 30/
C
      COMMON/CTRMPP/GRID(15,5),STIFF(15,5),U(15,5,5),A(15,5,5),
     1              B(15,5,5),C(15,5,5),UL(15,5,5),UU(15,5,5),
     2              UH(15,15,5,5),
     3              AH(15,15,5,5),BH(15,15,5,5),CH(15,15,5,5),
     4              P(10),Y(70),TOL,INTERV,KONTRL,KNOT(5),NP,NY,
     5              MQP,IG,NPHI,
     6              IMPULS,FINTEG,LDUM(16),TRAPB,LDUM2(2)
      DATA  IPRINT/0/
      COMMON/CRKF45/IOPT,IDUM45(4)
C
      LOGICAL UPPER,IMPULS,LDUM,FINTEG,GRADI,TRAPB,LDUM2
C
C-----------------------------------------------------------------------
C     FUNCTION STATEMENTS--AS IN SUBPHI   (DERIVATIVE IS MISSING DX/DT,
C                                          SINCE DERIVATIVE  IS WRT X)
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
      FCT(X,XU,XL,XSCALE)= (XU - X)*(X-XL)*XSCALE
      DFCTDX(X,XU,XL,XSCALE)=    (-2.D0*X + XU+XL)*XSCALE
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
C     SET GRADI = .TRUE.    IF GRADIENTS ARE BEING FORMED AND A  TPART2
C                           VECTOR IS BEING USED
C     SET GRADI = .FALSE.   OTHERWISE
C-----------------------------------------------------------------------
      GRADI = .TRUE.
      IF (.NOT. IMPULS)  GRADI = .FALSE.
      IF (TRAPB)         GRADI = .FALSE.
      IF (.NOT. FINTEG)  GRADI = .FALSE.
      IF (IG .NE. 2)     GRADI = .FALSE.
      IF (.NOT. GRADI)   GO TO 6
      NPOINT = NPOINT + 1
      IF (NPOINT .LE. LIMP)    GO TO 4
      PRINT 1500,NPOINT,LIMP
      STOP
    4 CONTINUE
      PVECT(NPOINT) = T
      DO 5 ITABLE = 1,NTABLE
```

```
      LIMIT = NCOMP(ITABLE)
      DO 5 ICOMP  = 1,LIMIT
    5 LPOINT(ITABLE,ICOMP,NPOINT) = 0
C
    6 CONTINUE
C-----------------------------------------------------------------------
      IBOUND = 0
C
      IF (IOPT .EQ. 1)  IPRINT = 1
C
C-----------------------------------------------------------------------
C     STUDY EACH TABLE AND EACH COMPONENT OF THAT TABLE
C-----------------------------------------------------------------------
      DO 50 ITABLE=1,NTABLE
      LIMIT = NCOMP(ITABLE)
      DO 50 ICOMP = 1,LIMIT
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
      IF (INDEX .NE. INDIC(ITABLE,ICOMP))  GO TO 50
C-----------------------------------------------------------------------
C     UPDATE OF COMPONENT "ICOMP".  SEE IF PHI(INDEX) VANISHES USING
C        CURRENT GRID BOUNDS FROM TABLE--ITABLE  COMPONENT--ICOMP
C-----------------------------------------------------------------------
      IF (UPPER)  GO TO 7
C     CONVERGENCE TO LOWER BOUND--USE BNDRYU AS ONE LIMIT
      PHI = FCT(PARAM,BNDRYU,GRIDL(ITABLE,ICOMP),SCALE(INDEX))
      GO TO 8
    7 CONTINUE
C     CONVERGENCE TO UPPER BOUND--USE BNDRYU AS ONE LIMIT
      PHI = FCT(PARAM,GRIDU(ITABLE,ICOMP),BNDRYL,SCALE(INDEX))
    8 CONTINUE
C
      IF (IPRINT .EQ. 1)  PRINT 999,ITABLE,ICOMP,PHI,TOLER
  999 FORMAT(' TABLE=',I2,'  COMP=',I2,2X,'PHI = ',D15.7,2X,'TOLER=',
     1    D15.7)
C
      IF (DABS(PHI) .GT. TOLER)  GO TO 50
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
C     CONVERGENCE TO BOUNDARY IN TABLE--ITABLE FOR COMPONENT--ICOMP
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
C
      IF (UPPER)  GO TO 10
C
C     CONVERGENCE TO LOWER BOUNDARY
C
      IF (IPRINT .EQ. 1)  PRINT 1501,INDEX,PARAM,GRIDL(ITABLE,ICOMP),
     1           GRIDU(ITABLE,ICOMP)
 1501 FORMAT(' CHECK CONVERGENCE--LOWER BOUND',/,2X,I3,3(2X,D15.7))
C
      IBL(ITABLE,ICOMP) = IBL(ITABLE,ICOMP) - 1
      IBU(ITABLE,ICOMP) = IBU(ITABLE,ICOMP) - 1
C
C     SET TPART2 STOPPING LIMITS IF GRADI IS BEING EVALUATED
C
      IF (GRADI)   LPOINT(ITABLE,ICOMP,NPOINT) = -IBL(ITABLE,ICOMP)
C
```

```
C     CALL TABBND TO SHIFT SUBSCRIPTS AND AND GRIDL, GRIDU VALUES
C
      CALL TABBND(ITABLE,ICOMP)
      IF (PARAM .GT. GRIDU(ITABLE,ICOMP))  GRIDU(ITABLE,ICOMP) = PARAM
      GO TO 12
C-----------------------------------------------------------------------
C     CONVERGENCE TO THE UPPER BOUNDARY
C-----------------------------------------------------------------------
   10 CONTINUE
      IBL(ITABLE,ICOMP) = IBL(ITABLE,ICOMP) + 1
      IBU(ITABLE,ICOMP) = IBU(ITABLE,ICOMP) + 1
C
C     SET TPART2 STOPPING LIMITS IF GRADI IS BEING EVALUATED
C
      IF (GRADI)   LPOINT(ITABLE,ICOMP,NPOINT) = +IBL(ITABLE,ICOMP)
C
C
      IF (IPRINT .EQ. 1)  PRINT 1502,INDEX,PARAM,GRIDL(ITABLE,ICOMP),
     1           GRIDU(ITABLE,ICOMP)
 1502 FORMAT(' CHECK CONVERGENCE--UPPER BOUND',/,2X,I3,3(2X,D15.7))
C
C     CALL TABBND TO SHIFT SUBSCRIPTS AND AND GRIDL, GRIDU VALUES
      CALL TABBND(ITABLE,ICOMP)
C
C     IF PARAM IS NOT ACROSS THE BOUNDARY SET LOWER BOUND = PARAM
C
      IF (PARAM .LT. GRIDL(ITABLE,ICOMP))  GRIDL(ITABLE,ICOMP) = PARAM
   12 CONTINUE
C
C-----------------------------------------------------------------------
C     SET MAXIMUM LOWER BOUND AND MINIMUM UPPER BOUND FROM  GRIDL
C         AND GRIDU VALUES GENERATED IN THE CURRENT SEARCH
C-----------------------------------------------------------------------
      IF (IBOUND .EQ. 1)  GO TO 11
      BOUNDL = GRIDL(ITABLE,ICOMP)
      BOUNDU = GRIDU(ITABLE,ICOMP)
   11 CONTINUE
      IBOUND = 1
      BOUNDU = DMIN1(GRIDU(ITABLE,ICOMP),BOUNDU)
      BOUNDL = DMAX1(GRIDL(ITABLE,ICOMP),BOUNDL)
C
   50 CONTINUE
C
      BNDRYL = BOUNDL
      BNDRYU = BOUNDU
C
      IF (.NOT. GRADI)    RETURN
      IF (IPRINT .EQ. 0)  RETURN
C
      DO 62 J = 1,NPOINT
      PRINT 1509,J,LPOINT(1,1,J),LPOINT(2,1,J),LPOINT(3,1,J),
     1           LPOINT(3,2,J),LPOINT(4,1,J),LPOINT(4,2,J)
   62 CONTINUE
 1509 FORMAT(' LPOINT VALUES:',/, ' J=',I2,3X,6(2X,I4))
C
 1500 FORMAT(//,' NUMBER OF STOPPING POINTS STORED FOR BACKWARD '
     1    ,'INTEGRATION = ',I6,/,' THIS EXCEEDS THE LIMIT SET = ',I2,/,
     2     ' THE USER MUST INCREASE THE VALUE OF LIMP AS WELL AS THE '
```

```
     3   ,/,'DIMENSIONING IN PVECT AND LPOINT IN COMMON BLOCK PARTV'
     4     ,' WHICH APPEARS IN BOUNDS AND SUBPHI ',//,
     5      ' TERMINAL ERROR',//)
      RETURN
      END
```

Subroutine COSTF for evaluating the cost function and derivatives

```
C-----------------------------------------------------------------------
      SUBROUTINE COSTF(P,X,Y,F,G,FP,FY,GP,GY,IFLAG,TO,WORK)
C-----------------------------------------------------------------------
C
      IMPLICIT REAL*8 (A-H,O-Z)
C
      DIMENSION G(1),P(1),X(1),Y(1),FP(1),FY(1),GP(25,25),GY(25,25)
      DIMENSION WORK(1)
      COMMON/CTRMPP/GRID(15,5),STIFF(15,5),U(15,5,5),A(15,5,5),
     1              B(15,5,5),C(15,5,5),UL(15,5,5),UU(15,5,5),
     2              UH(15,15,5,5),
     3              AH(15,15,5,5),BH(15,15,5,5),CH(15,15,5,5),
     4              DESP(10),YDUM(70),TOL,INTERV,KONTRL,KNOT(5),NP,NY,
     5              IDUM(3),
     6              LDUM(21)
      COMMON/BNDVAL/VF,GAMMAF,HF
C-----------------------------------------------------------------------
C     NOTE:  THE LEADING DIMENSION ON GP AND GY MUST MATCH THAT IN
C            THE DRIVING PROGRAM FOR DG--HERE 38
C-----------------------------------------------------------------------
      LOGICAL LDUM
C
      IF (IFLAG .EQ. 1)  GO TO 20
C
C     COMPUTE THE COST FUNCTION
C
      F = DESP(1)
C
C     COMPUTE CONSTRAING VECTORS:
C     PSI1=V-VF,PSI2=GAMMA-GAMMAF,PSI3=H-HF
C
      G(1) = (Y(1)-VF)/VF
      G(2) = Y(2)-GAMMAF
      G(3) = (Y(3)-HF)/HF
C
C
      GO TO 30
C
C     IFLAG = 1 PREPARE INITIAL CONDITIONS OF ADJOINT VARIABLES
C
   20 CONTINUE
C     FORM FY (DFDZ)--ALL SHOULD BE ZERO--THESE ARE SET IN TROMPP
C
C     FORM GY (DGDZ)--SUPPLY ONLY NON-ZERO VALUES
C
      GY(1,1) = 1.0D0/VF
```

```
      GY(2,2) = 1.0D0
      GY(3,3) = 1.0D0/HF
C
C     FORM FP (DFDP) DERIVATIVE OF F WRT DESIGN PARAMETER
C
      FP(1) = 1.0D0
C
C     FORM GP (DGDP) DERIVATIVE OF G WRT DESIGN PARAMETER
C     ALL ARE ZERO
C
   30 CONTINUE
      RETURN
      END
```

Subroutines EX1DT1 and EX2DT2, which are used to adjust the table size, are given in [3]

Subroutines EX1DT1 and EX2DT2, which are used to adjust the table size, are given in [3]

Subroutines FATM for determining the density and airspeed:

```
C
C-----------------------------------------------------------------------
      SUBRCUTINE FATM(H,RHO,ASP,DRHODH,DASPDH)
C-----------------------------------------------------------------------
C
C     PURPOSE:    COMPUTES DENSITY AND AIRSPEED AND THEIR DERIVA-
C                 TIVES FOR USE IN FTRIEB AND FCW.
C
C     STRUCTURE:  USER SUPPLIED
C
C     CALLING PARAMETERS:
C                 H           ALTITUDE
C
C     RETURNED PARAMETERS:
C                 RHO         DENSITY
C                 ASP         SPEED OF SOUND
C                 DRHODH      DERIVATIVE OF DENSITY WRT ALTITUDE
C                 DASPDH      DERIVATIVE OF SPEED OF SOUND WRT ALTITUDE
C
C     COMMENTS:   MODIFIED FROM PROGRAM BY K.H. WELL
C                 MODIFICATIONS 6.11.81
C                       COMMONS DELETED, ATMOSPHERE SELECTION INTERNAL,
C                       CALLING SEQUENCE ADDED, PRINT STATEMENT INTERNAL
C     ADAPTATIONS: M.K. HORN, MAY, 1982
```

```
C-----------------------------------------------------------------------
C
      IMPLICIT REAL*8 (A-H,O-Z)
C
      DATA RHOOA,FR2,FR5,FREXP/
     *.124921D0,.22559307019D-04,.34025855254D+03,4.25612D0/
      DATA RHOO,CK1,CK2,CK3,CK4,CK5,CK6/
     *1.249152361D-01,1.0228055OD0  ,1.21226930D-01,-3.48643241D-02,
     *3.50991865D-03,-8.33000535D-05,1.15219733D-06/
C
      DATA IATMO/2/
C
C
C  ATMOSPHAERE WIRD IN JEDEM FALL ANALYTISCH BERECHNET
C  ANALYTISCHE AUSDRUECKE :
C
      GO TO (10,20),IATMO
C
   10 TR=1.D0-FR2*H
      RHO=RHOOA*TR**FREXP
      SQTR=DSQRT(TR)
      ASP=FR5*SQTR
C
      DTRDH=-FR2
      DRHODH=FREXP*RHO/TR*DTRDH
      DASPDH=FR5*.5D0/SQTR*DTRDH
      RETURN
C
   20 CONTINUE
      T=((3.72D-12*H+.1933D-6)*H-8.877D-3)*H+292.1D0
      IF (T.LT.0.D0) PRINT 1500
      SQT=DSQRT(T)
      ASP=20.0468D0*SQT
      HKM=H*1.0D-03
      POL=HKM*(CK3+HKM*(CK4+HKM*(CK5+HKM*CK6)))
      PHI=CK1*DEXP(-POL)
      RHO=RHOO*DEXP(-(CK1+CK2*HKM-PHI))
C
      DTDH=(2.D0*3.72D-12*H+2.D0*.1933D-6)*H-8.877D-3
      DASPDH=DTDH*10.0234D0/SQT
      DPHIDH=-PHI*(CK3+HKM*(2.D0*CK4+HKM*(3.D0*CK5+HKM*4.D0*CK6)))
      DRHODH=-RHO*(CK2-DPHIDH)*1.0D-03
C
 1500 FORMAT(//,' ERROR IN FATM--T .LT. 0',/)
C
      RETURN
      END
```

Subroutines FDRAG for determinin the drag and partial derivatives

```
C
C-----------------------------------------------------------------------
      SUBROUTINE FDRAG(IPD,AM,CA,V,H,S,RHO,DRHODH,A,DADH,
     1                 DRAG,DDRDV,DDRDH,DDRDCA)
```

```
C-----------------------------------------------------------------------
C
C   PURPOSE:
C
C       MODE 1:
C       (IPD=0) COMPUTATION OF DRAG USING LINEARLY INTERPOLATED
C       VALUES FROM TABLES SET IN SUBROUTINE TABLE.
C       MODE 2:
C       (IPD=1) COMPUTATION OF DRAG AND ITS PARTIAL DERIVATIVES
C       WITH RESPECT TO VELOCITY, ALTITUDE, AND LIFT COEFFICIENT.
C
C
C   NOTATION:
C
C       SOME OF THE NOTATION IN THE PROGRAM IS IN GERMAN (DUE TO
C       HISTORICAL REASONS IN DEVELOPING THE SOFTWARE).  THE FOLLOWING
C       CONVERSION TABLE IS GIVEN:
C
C       ENGLISH NOTATION                    GERMAN NOTATION
C            CL                                  CA
C            CD                                  CW
C            CD0                                 CW0
C            DCD  (DELTA CD)                     DCW
C
C
C   STRUCTURE:    USER SUPPLIED
C
C   INPUT  PARAMETERS:
C        IPD          MODE OF OPERATION
C                     IPD = 0,  DRAG IS COMPUTED
C                     IPD = 1,  DRAG AND PARTIAL DERIVATIVES ARE
C                               COMPUTED
C        AM           MACH NUMBER = V/A
C        CA           LIFT COEFFICIENT
C        V            VELOCITY
C        H            ALTITUDE
C        S            PLANFORM AREA
C        RHO          AIR DENSITY
C        DRHODH       DERIVATIVE OF DENSITY WRT ALTITUDE
C        A            SPEED OF SOUND
C        DADH         DERIVATIVE OF SPEED OF SOUND WRT ALTITUDE
C
C   OUTPUT PRARMETERS:
C        DRAG         VALUE OF DRAG
C        DDRDV        DERIVATIVE OF DRAG WRT VELOCITY
C        DDRDH        DERIVATIVE OF DRAG WRT ALTITUDE
C        DDRDCA       DERIVATIVE OF DRAG WRT LIFT COEFFICIENT
C
C
C   COMMENTS:
C      TWO TABLES ARE USED:  TABLE1  GIVES  CW0 IN TERMS OF AM (AND H)
C                                           (ACTUALLY CONSTANT IN H)
C                            TABLE2  GIVES  DCW IN TERMS OF AM AND CA
C
C   PROGRAMMER:  M.K. HORN, MAY 1982
C-----------------------------------------------------------------------
C
      IMPLICIT REAL*8 (A-H,O-Z)
```

```
C
C
      COMMON/TABLE1/ AM1(26),         TAB1(26)
      COMMON/TABLE2/ AM2(24), CA2(35),TAB2(24,35)
      COMMON/TLIMIT/INDIC(4,2),NTE(4,2),NCOMP(4),NTABLE
C
      COMMON/CON1/KML1,KMU1
      COMMON/CON2/KML2,KMU2,KCAL2,KCAU2
C
      COMMON/FSTEP/ITOPH
      COMMON/EMERG/KPRINT
C
      DATA JPRINT/0/
C
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
C     FUNCTION STATEMENTS
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
      VAL2D1(P,Q,F00,F10,F01,F11) = (1.0D0-P)*(1.0D0-Q)*F00 + P*(1.0D0
     1   -Q)*F10 + Q*(1.0D0-P)*F01 + P*Q*F11
      PART11(F1,F0,X1,X0) = (F1 - F0)/(X1-X0)
      RATIO(AX,A1,A2) = (AX-A1)/(A2-A1)
C
      PARTV1(Q,DELTA1,F00,F10,F01,F11) = ((F10-F00)*(1.D0 - Q)
     1                                  + (F11-F01)*Q) / DELTA1
C
      PARTV2(P,DELTA2,F00,F10,F01,F11) = ((F01-F00)*(1.D0 - P)
     1                                  + (F11-F10)*P) / DELTA2
C
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
C
      IPRINT = 0
      IF (JPRINT .EQ. 1)       IPRINT = ITOPH
C
   20 CONTINUE
C-----------------------------------------------------------------------
C     CW0 DOES NOT CHANGE WITH ALTITUDE--FUNCTION OF MACH NUMBER ONLY
C-----------------------------------------------------------------------
C
      DCW0DM = PART11(TAB1(KMU1),TAB1(KML1),AM1(KMU1),AM1(KML1))
      CW0 = TAB1(KML1) + DCW0DM*(AM - AM1(KML1))
C
C-----------------------------------------------------------------------
C     DETERMINE DCW = DCW(M,CA)
C-----------------------------------------------------------------------
C
      P = RATIO(AM,  AM2(KML2), AM2(KMU2))
      Q = RATIO(CA,CA2(KCAL2),CA2(KCAU2))
      DCW = VAL2D1(P,Q,TAB2(KML2,KCAL2),TAB2(KMU2,KCAL2),
     1             TAB2(KML2,KCAU2),TAB2(KMU2,KCAU2))
C-----------------------------------------------------------------------
C     EVALUATE CW
C-----------------------------------------------------------------------
      CW = CW0 + DCW
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
```

```
      DRAG = 0.50D0 * RHO * V*V * S * CW
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
C
      IF (IPD .EQ. 0)  GO TO 30
C
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
C     PARTIAL DERIVATIVES
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
C     THESE PARTIAL DERIVATIVES ARE NEEDED ONLY IN THE BACKWARD
C     INTEGRATION--RETURN IF IPD = 0
C
C     NOTE:  D(CW)/DCA = D(DCW)/DCA SINCE CW0 IS INDEPENDENT OF CA.
C-----------------------------------------------------------------------
      DELTM  = AM2(KMU2) - AM2(KML2)
      DELTCA = CA2(KCAU2) - CA2(KCAL2)
      DDCWDM =PARTV1(Q,DELTM, TAB2(KML2,KCAL2),TAB2(KMU2,KCAL2),
     1                        TAB2(KML2,KCAU2),TAB2(KMU2,KCAU2))
      DCWDCA =PARTV2(P,DELTCA,TAB2(KML2,KCAL2),TAB2(KMU2,KCAL2),
     1                        TAB2(KML2,KCAU2),TAB2(KMU2,KCAU2))
      DCWDM = DCWODM + DDCWDM
      DMDV = 1.0D0/A
      DMDA = -V/(A*A)
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
      DDRDCA = 0.50D0 * RHO * V * V * S * DCWDCA
      DDRDV = RHO*S*V*CW + 0.50D0*RHO*S * V*V *DCWDM * DMDV
      DDRDH = 0.50D0 * DRHODH * V*V * S *CW
     1               + 0.50D0 * RHO * V*V * S * DCWDM*DMDA*DADH
      KPRINT = 0
      IF (KPRINT .EQ. 1)     PRINT 1300,DDRDCA,DDRDV,DDRDH
 1300 FORMAT(' FROM FDRAG:',/,' DDRDCA=',D15.7,2X,'DDRDV=',D15.7,2X,
     1       'DDRDH=',D15.7)
      DWRTM1=(TAB2(KMU2,KCAU2)-TAB2(KML2,KCAU2))/DELTM
      DWRTM0=(TAB2(KMU2,KCAL2)-TAB2(KML2,KCAL2))/DELTM
      DWRTC1=(TAB2(KMU2,KCAU2)-TAB2(KMU2,KCAL2))/DELTCA
      DWRTC0=(TAB2(KML2,KCAU2)-TAB2(KML2,KCAL2))/DELTCA
      IF (KPRINT .EQ. 1)  PRINT 1302,DWRTM1,DDCWDM,DWRTM0
      IF (KPRINT .EQ. 1)  PRINT 1303,DWRTC1,DCWDCA,DWRTC0
 1302 FORMAT(' CHECK DERIVATIVES:',/,' DWRTM1 = ',D15.7,2X,' DDCWDM = ',
     1     D15.7,2X,' DWRTM0 = ',D15.7)
 1303 FORMAT(' CHECK DERIVATIVES:',/,' DWRTC1 = ',D15.7,2X,' DCWDCA = ',
     1     D15.7,2X,' DWRTC0 = ',D15.7)
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
   30 CONTINUE
C
      IF (IPRINT .EQ. 0)  RETURN
C-----------------------------------------------------------------------
C     PRINT OPTIONS FOR SAFETY CHECKS
C-----------------------------------------------------------------------
      PRINT 510,AM1(KML1),TAB1(KML1),AM,CW0,AM1(KMU1),TAB1(KMU1)
      PRINT 512,AM2(KML2),CA2(KCAL2),AM,CA,AM2(KMU2),CA2(KCAU2)
      PRINT 513,TAB2(KML2,KCAL2),TAB2(KML2,KCAU2),
     1      DCW,TAB2(KMU2,KCAL2),TAB2(KMU2,KCAU2)
  510 FORMAT(/,' IN FDRAG--TABLE 1 VALUES (CW0 = F(M))',/,
```

```
     1  ' LOWER BOUND--M = ',D23.16,2X, 'CW0 = ',D23.16,/,
     2  '               M = ',D23.16,2X, 'CW0 = ',D23.16,/,
     3  ' UPPER BOUND--M = ',D23.16,2X, 'CW0 = ',D23.16,//)
C
  512 FORMAT(/,' IN FDRAG--TABLE 2 GRID VALUES        ',/,
     1  ' LOWER BOUND--M = ',D23.16,2X, ' CA = ',D23.16,/,
     2  '               M = ',D23.16,2X, ' CA = ',D23.16,/,
     3  ' UPPER BOUND--M = ',D23.16,2X, ' CA = ',D23.16,//)
C
  513 FORMAT(/,' IN FDRAG--TABLE 2 VALUES (DCW = F(M,CA))',/,
     1         ' LOWER  = ',D23.16,5X,' UPPER = ',D23.16,/,
     2         ' DCW    = ',10X,D23.16,/,
     3         ' UPPER  = ',D23.16,5X,' UPPER = ',D23.16,/)
C
      RETURN
      END
```

Subroutines FTRIEB for determinin the thrust and partial derivatives

```
C-----------------------------------------------------------------------
      SUBROUTINE FTRIEB(IPD,AM,H,V,A,DADH,DELTA,TMAX,
     1                  SFC,THRUST,DSFCDV,DSFCDH,DTHRDV,DTHRDH)
C-----------------------------------------------------------------------
C
C     PURPOSE:
C         MODE 1: (IPD=0)
C         FTRIEB COMPUTES SFC AND THRUST USING LINEARLY INTERPOLATED
C         VALUES FROM TABLES SET IN SUBROUTINE TABLES.
C
C         MODE 2: (IPD=1)
C         FTRIEB COMPUTES SFC AND THRUST, AND THEIR PARTIAL DERIVA-
C         TIVES USING LINEARLY INTERPOLATED VALUES FROM TABLES SET
C         IN SUBROUTINE TABLE.
C
C   NOTATION:
C
C       SOME OF THE NOTATION IN THE PROGRAM IS IN GERMAN (DUE TO
C       HISTORICAL REASONS IN DEVELOPING THE SOFTWARE).  THE FOLLOWING
C       CONVERSION TABLE IS GIVEN:
C
C       ENGLISH NOTATION                    GERMAN NOTATION
C            CTI                                 CSI
C
C
C     STRUCTURE:          USER SUPPLIED
C
C     INPUT PARAMETERS:
C
C         IPD             MODE OF OPERATION
C                         IPD=0,  SFC AND THRUST COMPUTED
C                         IPD=1,  SFC, THRUST AND PARTIAL DERIVA-
C                                 TIVES COMPUTED
C         AM              MACH NUMBER = V/A
C         H               ALTITUDE
```

```
C          V              VELOCITY
C          A              SPEED OF SOUND
C          DADH           DERIVATIVE OF SPEED OF SOUND WRT ALTITUDE
C          DELTA          POWER SETTING (CONSTANT)
C          TMAX           CONSTANT (MAXIMUM THRUST)
C
C      OUTPUT PARAMETERS:
C
C          SFC            SPECIFIC FUEL CONSUMPTION
C          THRUST         THRUST
C          DSFCDV         DERIVATIVE OF SFC WRT VELOCITY
C          DSFCDH         DERIVATIVE OF SFC WRT ALTITUDE
C          DTHRDV         DERIVATIVE OF THRUST WRT VELOCITY
C          DTHRDH         DERIVATIVE OF THRUST WRT ALTITUDE
C
C       TWO TABLES:  TABLE3  GIVES  SFC = TAB3(AM,H)
C                    TABLE4  GIVES  CSI = TAB4(AM,H)
C                    THRUST = TMAX*CSI*DELTA,  DELTA = POWER SETTING
C                                                      (CONSTANT)
C
C      PROGRAMMER:  M.K. HORN, MAY, 1982
C                   MAJOR SIMPLIFICATIONS FOR TOMP (6.11.81)
C-----------------------------------------------------------------------
       IMPLICIT REAL*8 (A-H,O-Z)
C
C
       COMMON/TABLE3/AM3 (26),  H3(11),TAB3(26,11)
       COMMON/TABLE4/ AM4(26),  H4(11),TAB4(26,11)
       COMMON/TLIMIT/INDIC(4,2),NGE(4,2),NCOMP(4),NTABLE
C
       COMMON/CON3/KML3,KMU3,KHL3,KHU3
       COMMON/CON4/KML4,KMU4,KHL4,KHU4
C
       COMMON/FSTEP/ITOPH
       COMMON/EMERG/KPRINT
       DATA JPRINT/0/
C
C-----------------------------------------------------------------------
C              SFC     = TAB4(MACH,ALT)
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
C      FUNCTION STATEMENTS
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
       VAL2D1(P,Q,F00,F10,F01,F11) = (1.0D0-P)*(1.0D0-Q)*F00 + P*(1.0D0
      1   -Q)*F10 + Q*(1.0D0-P)*F01 + P*Q*F11
       PART11(F1,F0,X1,X0) = (F1-F0)/(X1-X0)
       RATIO(AX,A1,A2) = (AX-A1)/(A2-A1)
C
       PARTV1(Q,DELTA1,F00,F10,F01,F11) = ((F10-F00)*(1.D0 - Q)
      1                                  + (F11-F01)*Q) / DELTA1
C
       PARTV2(P,DELTA2,F00,F10,F01,F11) = ((F01-F00)*(1.D0 - P)
      1                                  + (F11-F10)*P) / DELTA2
C
C-----------------------------------------------------------------------
C
       IPRINT = 0
```

```
      IF (JPRINT .EQ. 1)       IPRINT = ITOPH
C
C-----------------------------------------------------------------------
C     TABLE 3:
C-----------------------------------------------------------------------
      PCSI = RATIO(AM,AM3(KML3),AM3(KMU3))
      QCSI = RATIO( H, H3(KHL3), H3(KHU3))
      CSI   = VAL2D1(PCSI,QCSI,TAB3(KML3,KHL3),TAB3(KMU3,KHL3),
     1                  TAB3(KML3,KHU3),TAB3(KMU3,KHU3))
      THRUST = TMAX * CSI * DELTA
C-----------------------------------------------------------------------
C     TABLE 4:
C-----------------------------------------------------------------------
      PSFC = RATIO(AM,AM4(KML4),AM4(KMU4))
      QSFC = RATIO( H, H4(KHL4), H4(KHU4))
      SFC   = VAL2D1(PSFC,QSFC,TAB4(KML4,KHL4),TAB4(KMU4,KHL4),
     1                  TAB4(KML4,KHU4),TAB4(KMU4,KHU4))
C-----------------------------------------------------------------------
      IF (IPD .EQ. 0)  GO TO 50
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
C    PARTIAL DERIVATIVES
C-----------------------------------------------------------------------
      DELTM =  AM3(KMU3) - AM3(KML3)
      DELTH =   H3(KHU3) -  H3(KHL3)
      DCSIDM = PARTV1(QCSI,DELTM   ,TAB3(KML3,KHL3),TAB3(KMU3,KHL3),
     1                              TAB3(KML3,KHU3),TAB3(KMU3,KHU3))
      DCSIDH = PARTV2(PCSI,DELTH   ,TAB3(KML3,KHL3),TAB3(KMU3,KHL3),
     1                              TAB3(KML3,KHU3),TAB3(KMU3,KHU3))
      DELTM =  AM4(KMU4) - AM4(KML4)
      DELTH =   H4(KHU3) -  H4(KHL3)
      DSFCDM = PARTV1(QSFC,DELTM   ,TAB4(KML4,KHL4),TAB4(KMU4,KHL4),
     1                              TAB4(KML4,KHU4),TAB4(KMU4,KHU4))
      DSFCDH = PARTV2(PSFC,DELTH   ,TAB4(KML4,KHL4),TAB4(KMU4,KHL4),
     1                              TAB4(KML4,KHU4),TAB4(KMU4,KHU4))
C
      DELTM =  AM3(KMU3) - AM3(KML3)
      DELTH =   H3(KHU3) -  H3(KHL3)
      DWRTM1=(TAB3(KMU3,KHU3)-TAB3(KML3,KHU3))/DELTM
      DWRTM0=(TAB3(KMU3,KHL3)-TAB3(KML3,KHL3))/DELTM
      DWRTH1=(TAB3(KMU3,KHU3)-TAB3(KMU3,KHL3))/DELTH
      DWRTH0=(TAB3(KML3,KHU3)-TAB3(KML3,KHL3))/DELTH
      KPRINT = 0
      IF (KPRINT .EQ. 1)  PRINT 1302,DWRTM1,DCSIDM,DWRTM0
      IF (KPRINT .EQ. 1)  PRINT 1303,DWRTH1,DCSIDH,DWRTH0
C
      DELTM =  AM4(KMU4) - AM4(KML4)
      DELTH =   H4(KHU4) -  H4(KHL4)
      DWRTM1=(TAB4(KMU4,KHU4)-TAB4(KML4,KHU4))/DELTM
      DWRTM0=(TAB4(KMU4,KHL4)-TAB4(KML4,KHL4))/DELTM
      DWRTH1=(TAB4(KMU4,KHU4)-TAB4(KMU4,KHL4))/DELTH
      DWRTH0=(TAB4(KML4,KHU4)-TAB4(KML4,KHL4))/DELTH
      IF (KPRINT .EQ. 1)  PRINT 1304,DWRTM1,DSFCDM,DWRTM0
      IF (KPRINT .EQ. 1)  PRINT 1305,DWRTH1,DSFCDH,DWRTH0
 1302 FORMAT(' CHECK DERIVATIVES:',/,' DWRTM1 = ',D15.7,2X,' DCSIDM = ',
     1     D15.7,2X,' DWRTM0 = ',D15.7)
 1303 FORMAT(' CHECK DERIVATIVES:',/,' DWRTH1 = ',D15.7,2X,' DCSIDH = ',
     1     D15.7,2X,' DWRTH0 = ',D15.7)
```

```
C
 1304 FORMAT(' CHECK DERIVATIVES:',/,' DWRTM1 = ',D15.7,2X,' DSFCDM = ',
     1     D15.7,2X,' DWRTM0 = ',D15.7)
 1305 FORMAT(' CHECK DERIVATIVES:',/,' DWRTH1 = ',D15.7,2X,' DSFCDH = ',
     1     D15.7,2X,' DWRTH0 = ',D15.7)
C
      DMDV = 1.0D0/A
      DMDA = -V/(A*A)
C
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
      DSFCDV = DSFCDM * DMDV
      DSFCDH = DSFCDH + DSFCDM*DMDA*DADH
C
      DTHRDV = TMAX * DCSIDM * DMDV * DELTA
      DTHRDH = TMAX * (DCSIDH + DCSIDM*DMDA*DADH) * DELTA
      IF (KPRINT .EQ. 1)  PRINT 1300,DTHRDV,DTHRDH
      IF (KPRINT .EQ. 1)  PRINT 1301,DSFCDV,DSFCDH
 1300 FORMAT(' IN FTRIEB:',/,' DSFCDV=',D15.7,2X,'DSFCDH=',D15.7)
 1301 FORMAT(' IN FTRIEB:',/,' DTHRDV=',D15.7,2X,'DTHRDH=',D15.7)
C-----------------------------------------------------------------------
   50 CONTINUE
C-----------------------------------------------------------------------
      IF (IPRINT .EQ. 0)  RETURN
C-----------------------------------------------------------------------
C     PRINT OPTIONS FOR SAFETY CHECKS
C-----------------------------------------------------------------------
      PRINT 510,AM3(KML3), H3(KHL3),AM, H,AM3(KMU3), H3(KHU3)
      PRINT 511,AM4(KML4), H3(KHL4),AM, H,AM4(KMU4), H4(KHU4)
      PRINT 512,TAB3(KML3,KHL3),TAB3(KML3,KHU3),
     1      CSI,TAB3(KMU3,KHL3),TAB3(KMU3,KHU3)
      PRINT 513,TAB4(KML4,KHL4),TAB4(KML4,KHU4),
     1      SFC,TAB4(KMU4,KHL4),TAB4(KMU4,KHU4)
C
  510 FORMAT(/,' IN FTRIEB--TABLE 3 GRID VALUES        ',/,
     1  ' LOWER BOUND--M = ',D23.16,2X, '  H = ',D23.16,/,
     2  '                M = ',D23.16,2X, '  H = ',D23.16,/,
     3  ' UPPER BOUND--M = ',D23.16,2X, '  H = ',D23.16,//)
C
  511 FORMAT(/,' IN FTRIEB--TABLE 4 GRID VALUES        ',/,
     1  ' LOWER BOUND--M = ',D23.16,2X, '  H = ',D23.16,/,
     2  '                M = ',D23.16,2X, '  H = ',D23.16,/,
     3  ' UPPER BOUND--M = ',D23.16,2X, '  H = ',D23.16,//)
C
  512 FORMAT(/,' IN FTRIEB--TABLE 3 VALUES (CSI = F(M, H))',/,
     1         ' LOWER  = ',D23.16,5X,' UPPER = ',D23.16,/,
     2         ' CSI      ',10X,D23.16,/,
     3         ' UPPER  = ',D23.16,5X,' UPPER = ',D23.16,/)
C
  513 FORMAT(/,' IN FTRIEB--TABLE 2 VALUES (SCF = F(M, H))',/,
     1         ' LOWER  = ',D23.16,5X,' UPPER = ',D23.16,/,
     2         ' SFC      ',10X,D23.16,/,
     3         ' UPPER  = ',D23.16,5X,' UPPER = ',D23.16,/)
C
      RETURN
      END
```

Subroutine INITBD sets initial bounding values for the SUBPHI subroutine

```
C-----------------------------------------------------------------------
      SUBROUTINE INITBD(NTRPR,BOUNDL,BOUNDU)
C-----------------------------------------------------------------------
C     PURPOSE:  INITBD SETS THE BOUNDING VALUES FOR THE TRAPPING
C               PARAMETERS BEFORE THE INTEGRATION BEGINS.
C
C     STRUCTURE:  STANDARD (USER-SUPPLIED DIMENSIONS IN COMMON BLOCKS.
C                           SEE PHI DESCRIPTION.)
C
C     INPUT  PARAMETERS:
C       NTRPR         DIMENSION OF BOUNDL AND BOUNDU VECTORS
C
C     OUTPUT PARAMETERS:
C       BOUNDL(J)     LOWER BOUND ON TRAPPING PARAMETER NUMBER J
C       BOUNDU(J)     UPPER BOUND ON TRAPPING PARAMETER NUMBER J
C
C
C     PROGRAMMER:  M.K. HORN, DFVLR-OBERPFAFFENHOFEN, AUGUST, 1982
C
C-----------------------------------------------------------------------
      IMPLICIT REAL*8 (A-H,O-Z)
C
      DIMENSION BOUNDL(NTRPR),BOUNDU(NTRPR)
C-----------------------------------------------------------------------
      COMMON/TLIMIT/INDIC(4,2),NGE(4,2),NCOMP(4),NTABLE
      COMMON/GRIDBD/GRIDL( 4, 2),GRIDU( 4, 2),IBL( 4, 2),IBU( 4, 2)
      DATA IPRINT/0/
C-----------------------------------------------------------------------
C     INITIALIZATION BLOCK:
C-----------------------------------------------------------------------
C
C     BOUNDS GRIDL AND GRIDU ARE SET IN TABLIM (ALREADY REFERENCED)
C     BEFORE EACH INTEGRATION BEGINS.  BOUNDU(I) AND BOUNDL(I),
C     I=1,...,NTRPR, MUST BE SELECTED FROM THESE VALUES.
C
      DO 10 II = 1,NTRPR
C
      ISET = 0
      DO 10 ITABLE = 1,NTABLE
      LIMIT = NCOMP(ITABLE)
      DO 10 ICOMP = 1,LIMIT
C
      IF (II .NE. INDIC(ITABLE,ICOMP) )   GO TO 10
      IF (ISET .GT. 0)   GO TO 8
C
C     NO BOUNDS HAVE BEEN SET YET FOR TRAPPING PARAMETER "II"
C
      BOUNDL(II) =  GRIDL(ITABLE,ICOMP)
      BOUNDU(II) =  GRIDU(ITABLE,ICOMP)
      ISET = 1
      GO TO 10
C
    8 CONTINUE
C
```

```
C     BOUNDS HAVE BEEN SET FOR TRAPPING PARAMETER "II"--COMPARE THESE
C       WITH THOSE FOR TABLE "ITABLE"  COMPONENT "ICOMP"
C
      IF (GRIDL(ITABLE,ICOMP) .GT. BOUNDL(II))     BOUNDL(II) =
     1                                             GRIDL(ITABLE,ICOMP)
      IF (GRIDU(ITABLE,ICOMP) .LT. BOUNDU(II))     BOUNDU(II) =
     1                                             GRIDU(ITABLE,ICOMP)
   10 CONTINUE
C
      IF (IPRINT .EQ. 0)  GO TO 14
      PRINT 1588,(J,BOUNDL(J),J,BOUNDU(J),J=1,NTRPR)
 1588 FORMAT(' INITIAL BOUNDS BEFORE INGRATION BEGINS:',/,
     1      (5X,'BOUNDL',I2,') = ',D15.7,2X,'BOUNDU',I2,') = ',D15.8))
   14 CONTINUE
C
      RETURN
      END
```

Subroutine INSERT, in conjunction with QROOT, is given in [3]

Subroutine LIFTC, determines the value of the lift coefficient

```
C-----------------------------------------------------------------------
C     LIFT COEFFICIENT GENERATOR USING CUBIC SPLINE COEFFICIENTS FROM
C          TOMP
      SUBROUTINE LIFTC(IDERIV,T,CA,DCADT,NODE)
C-----------------------------------------------------------------------
      IMPLICIT REAL*8  (A-H,O-Z)
      COMMON/CTRMPP/GRID(15,5),STIFF(15,5),U(15,5,5),A(15,5,5),
     1              B(15,5,5),C(15,5,5),UL(15,5,5),UU(15,5,5),
     2              UH(15,15,5,5),
     3              AH(15,15,5,5),BH(15,15,5,5),CH(15,15,5,5),
     4              P(10),Y(70),TOL,INTERV,KONTRL,KNOT(5),NP,NY,
     5              IDUM(3),
     6              LDUM(21)
C
      LOGICAL LDUM
C-----------------------------------------------------------------------
C   NOTATION:
C
C       SOME OF THE NOTATION IN THE PROGRAM IS IN GERMAN (DUE TO
C       HISTORICAL REASONS IN DEVELOPING THE SOFTWARE).  THE FOLLOWING
C       CONVERSION TABLE IS GIVEN:
C
C       ENGLISH NOTATION                    GERMAN NOTATION
C            CL                                  CA
C
C-----------------------------------------------------------------------
C     THERE IS ONLY ONE INTERVAL.  THUS, T DOES NOT HAVE TO BE LOCATED
C     WITHIN THE INTERVAL MESHES.  EACH INTERVAL "I" HAS KNOT(I)NODES.
```

```
C-----------------------------------------------------------------------
C     COMPUTE LIFT COEFFICIENT (WITHOUT PARAMETER DISTURBANCE)
C-----------------------------------------------------------------------
C
      CA    = SPLINT(KNOT(1),GRID(1,1),U(1,1,1),STIFF(1,1),
     1           A(1,1,1),B(1,1,1),C(1,1,1),T)
      IF (IDERIV .EQ. 0)  RETURN
      DCADT = DSPLNT(KNOT(1),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     DCADT (WRT TIME) MUST BE SCALED BY 1/TFINAL = 1/P(1)
C-----------------------------------------------------------------------
      DCADT = DCADT/P(1)
C
      RETURN
      END
```

Subroutine LOCATE, determines the bracketing indices for locating a parameter in a given vector array

```
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
      SUBROUTINE LOCATE(NPTS,AV,VECTOR,MPT,NPT,BNDL,BNDU,ITABLE,ICOMP)
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
C     PURPOSE:
C         SUBROUTINE LOCATE CHOSES MPT AND NPT SO THAT AV IS
C         BRACKETED BY VECTOR(MPT) AND VECTOR(NPT), MPT = NPT - 1 .
C         BOUNDS: BNDL= VECTOR(MPT), BNDU=VECTOR(NPT) ARE
C         ARE RETURNED.
C
C     STRUCTURE:            STANDARD (NO USER CHANGES)
C
C     INPUT PARAMETERS:
C            NPTS           DIMENSION OF ARRAY VECTOR
C            VECTOR         VECTOR ARRAY, DIMENSIONED NPTS
C            AV             THE VALUE TO BE LOCATED IN ARRAY VECTOR
C            ITABLE         TABLE BEING ANALYZED (FOR PRINT STATEMENTS)
C            ICOMP          COMPONENT OF TABLE #ITABLE BEING ANALYZED
C
C
C     OUTPUT PARAMETERS:
C            MPT            INDEX VALUE BRACKETING AV
C            NPT            INDEX VALUE BRACKETING AV, NPT > MPT
C            BOUNDL         VECTOR(MPT)
C            BOUNDU         VECTOR(NPT)
C
C     LABELING:
C                                        **********************0
C           ----------+---------------+------------------+----------
C              VECTOR(1)        VECTOR(2)           VECTOR(3)
C
C      ALL POINTS "*" HAVE NPT=3, MPT=2 .
C      POINT "0" HAS NPT=4, MPT=3 .
```

```
C
C
C
C     REQUIREMENTS:
C     VECTOR ENTRIES SHOULD BE DISTINCT AND SHOULD BE IN ASCENDING
C     ORDER.
C
C     COMMENTS:
C     IF AV LIES OUTSIDE OF THE GRID, A WARNING WILL BE PRINTED,
C     AND THE PROGRAM WILL TERMINATE.  USER MUST THEN EXTEND THE
C     TABLES (USING, E.G.,  EXT1D, EXT2D, OR EXT3D)  TO CONTINUE.
C
C     USER CHANGES: NONE
C
C     PROGRAMMER: M.K. HORN, DFVLR-OBERPFAFFENHOFEN, MAY, 1982
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION VECTOR(NPTS)
C
      IF (AV .LT. VECTOR(1) .OR. AV .GT. VECTOR(NPTS))  GO TO 100
      DO 10 I = 2,NPTS
      IF (AV .LT. VECTOR(I))  GO TO 30
   10 CONTINUE
C-----------------------------------------------------------------------
C     SET MPT, NPT
C-----------------------------------------------------------------------
   30 CONTINUE
      NPT = I
      MPT = I-1
      BNDL = VECTOR(MPT)
      BNDU = VECTOR(NPT)
      RETURN
C
  100 CONTINUE
C
      PRINT 1500
      PRINT 1501,ITABLE,ICOMP,AV,VECTOR(1),NPTS,VECTOR(NPTS)
 1500 FORMAT(//,' DIFFICULTIES IN TABLE ',I3,3X,'WITH COMPONENT ',I3,
     1  /,' VALUES ARE OUTSIDE OF THE TABLE GRID.',//,
     2    ' THE PROGRAM WILL TERMINATE.',//,' THE USER SHOULD CALL',
     3    ' TABEXT TO EXTEND THE TABLE,'/,' UNLESS THE LIMITS ARE',
     4    ' ALREADY REDICULOUSLY LARGE.',//)
 1501 FORMAT(/,' TABLE= ',I2,2X,'ICOMP= ',I2,' PARAMETER=',D23.16,/,
     1          ' VECTOR(1) = ',D23.16,2X,'VECTOR(',I3,') = ',
     2         D23.16,///,' TERMINAL ERROR',//)
      STOP
      END
```

Subroutine MATRX1, MATRX2 used in conjunction with RHS evaluation to multiply matrices

```
C-----------------------------------------------------------------------
      SUBROUTINE MATRX1(NEQN,Y,SCPROD,ISHIFT,B)
```

```
C-----------------------------------------------------------------------
C
C     PURPOSE:  FORMS SCALAR PRODUCT
C
C     DIMENSIONS:  B  = 4 X 1 ,  Y COMPONENTS USED 1,...,4
C
C                           T
C     SYSTEM:     SPROD = B  Y  IS FORMED
C
C     NOTE:
C
C     SUBSCRIPTS ARE SHIFTED BY THE AMOUNT "ISHIFT" IN RHS
C     I.E.,  Y(ISHIFT+1),...,Y(ISHIFT+4) ARE USED IN GENERATING SCPROD
C
C     PROGRAMMER:  M.K. HORN,  JUNE, 1982
C-----------------------------------------------------------------------
C
      IMPLICIT REAL*8 (A-H,O-Z)
C
      DIMENSION Y(NEQN)
      DIMENSION B(4)
      DATA LIMIT/4/
C
      SCPROD = 0.0D0
      DO 55 J = 1,LIMIT
   55 SCPROD = SCPROD + B(J)*Y(J+ISHIFT)
C
      RETURN
      END
C-----------------------------------------------------------------------
      SUBROUTINE MATRX2(NEQN,Y,YP,ISHIFT,A)
C-----------------------------------------------------------------------
C
C     PURPOSE:  PERFORMS MATRIX MULTIPLICATION
C
C     DIMENSIONS:  A  = 4 X 4 ,  Y COMPONENTS USED 1,...,4
C
C     SYSTEM:      YP = A Y  IS FORMED
C
C     NOTE:
C
C     SUBSCRIPTS ARE SHIFTED BY THE AMOUNT "ISHIFT"
C     I.E.,  YP(ISHIFT+1),...,YP(ISHIFT+4) ARE GENERATED USING
C             Y(ISHIFT+1),  , Y(ISHIFT+4)
C
C     PROGRAMMER:  M.K. HORN,  JUNE, 1982
C-----------------------------------------------------------------------
C
      IMPLICIT REAL*8 (A-H,O-Z)
C
      DIMENSION Y(NEQN),YP(NEQN)
      DIMENSION A(4,4)
      DATA LIMIT/4/
C
      DO 12 I = 1,LIMIT
      YP(I+ISHIFT) = 0.0D0
      DO 12 J = 1,LIMIT
   12 YP(I+ISHIFT) = YP(I+ISHIFT) + A(I,J)* Y(J+ ISHIFT)
```

```
C
      RETURN
      END
```

Subroutine PART, for establishing and updating independent variable stops

```
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
      SUBROUTINE PART(MODE,TPART1,NTPRT1,ISTOP1,TPART2,NTPRT2,ISTOP2,
     1                FINTEG,NEQN,T,Y,YP)
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION Y(NEQN),YP(NEQN)
      DIMENSION TPART1(100),TPART2(100)
      DIMENSION IPARTL(100),IPARTU(100)
C
      DIMENSION TRPR(2),TRPRP(2)
      DATA NTRPR/2/
      DATA IPRINT/0/
      DATA IZERO/1/,MODE2/2/,MODE3/3/,MODE4/4/
C
      COMMON/PRINTR/IPR
      COMMON/CTRMPP/GRID(15,5),STIFF(15,5),U(15,5,5),A(15,5,5),
     1              B(15,5,5),C(15,5,5),UL(15,5,5),UU(15,5,5),
     2              UH(15,15,5,5),
     3              AH(15,15,5,5),BH(15,15,5,5),CH(15,15,5,5),
     4              P(10),YDUM(70),TOL,INTERV,KONTRL,KNOT(5),NP,NY,
     5              MQP,IG,NPHI,
     6              IMPULS,LDUM(20)
      LOGICAL IMPULS,LDUM
C
C
      COMMON/TLIMIT/INDIC(4,2),NGE(4,2),NCOMP(4),NTABLE
      COMMON/GRIDBD/GRIDL( 4, 2),GRIDU( 4, 2),IBL( 4, 2),IBU( 4, 2)
      COMMON/PARTV/PVECT(30),LPOINT( 4, 2, 30),NPOINT
C
C
      COMMON/CHCKCA/CLIFT
C
      COMMON/TABLE1/T1P1(26),          TAB1(26)
      COMMON/TABLE2/T2P1(24),T2P2(35),TAB2(24,35)
      COMMON/TABLE3/T3P1(26),T3P2(11),TAB3(26,11)
      COMMON/TABLE4/T4P1(26),T4P2(11),TAB4(26,11)
C
      COMMON/CON1/KL11,KU11
      COMMON/CON2/KL21,KU21,KL22,KU22
      COMMON/CON3/KL31,KU31,KL32,KU32
      COMMON/CON4/KL41,KU41,KL42,KU42
      LOGICAL FINTEG
C
      IF (MODE .EQ. 2)   GO TO 50
C
C-----------------------------------------------------------------------
```

```
C     MODE 1:    CALL IS MADE BEFORE INTEGRATION BEGINS.
C                MAKE ANY NEEDED INITIALIZATION.
C-----------------------------------------------------------------------
C     INITIALIZATION:   TABLE LIMITS AT INITIAL CONDITIONS MUST BE
C                       SET (FOR WHICH TRPR VALUES ARE NEEDED)
C
C
      CALL PHIPAR(MODE2,NEQN,T,Y,YP,NTRPR,TRPR,TRPRP)
C
      IF (.NOT. FINTEG)   GO TO 45
C-----------------------------------------------------------------------
C     INITIALIZE FORWARD INTEGRATION:
C     CALL TABLIM TO INITIALIZE INDICES FOR TABLE ANALYSIS
C-----------------------------------------------------------------------
      CLIFT = U(1,1,1)
      CALL TABLIM(TRPR,NTRPR)
C-----------------------------------------------------------------------
C     CALL PARTCA TO ESTABLISH LIFT COEFFICIENT STOPS AND SET INDICES
C-----------------------------------------------------------------------
      CALL PARTCA(TPART1,IPARTL,IPARTU,NTPRT1)
      KL22 = IPARTL(1)
      KU22 = IPARTU(1)
      IF (IPR .NE. -1)   GO TO 777
      PRINT 775,(J,TPART1(J),J=1,NTPRT1)
 775  FORMAT(' TPART1(',I3,') = ',D15.7)
 777  CONTINUE
C
      IF (.NOT. IMPULS)  GO TO 15
      IF (IG .NE. 2)     GO TO 15
C-----------------------------------------------------------------------
C     BACKWARD DIFFERENCING WILL BE USED TO FORM GRADIENTS.  INITIAL-
C     IZE TPART2 VECTOR (PVECT AND RELATED PARAMETERS).
C
C     IMPULSE=.T. ==> BACKWARD DIFFERENCING IS USED.
C     IG  =  2    ==> GRADIENTS ARE TO BE FORMED.
C-----------------------------------------------------------------------
      PVECT(1) = T
C
C     LPOINT(TABLE ,COMP ,1) = KL"TABLE""COMP"
C
      LPOINT(1,     1,    1) = KL11
      LPOINT(2,     1,    1) = KL21
      LPOINT(2,     2,    1) = KL22
      LPOINT(3,     1,    1) = KL31
      LPOINT(3,     2,    1) = KL32
      LPOINT(4,     1,    1) = KL41
      LPOINT(4,     2,    1) = KL42
C
      NPOINT = 1
C
   15 CONTINUE
C
C-----------------------------------------------------------------------
C     CALL WARN (IN MODE4 TO SEE IF INDICES HAVE BEEN PROPERLY SET
C-----------------------------------------------------------------------
      CALL WARN(MODE4,TRPR,NTRPR,T,Y,NEQN,IZERO)
C
      IF (IPRINT .EQ. 0)  RETURN
```

```
      PRINT 1503,(J,TPART1(J),J=1,NTPRT1)
 1503 FORMAT(' TIME(',I3,'= ',D15.7)
C
      RETURN
C
   45 CONTINUE
C-----------------------------------------------------------------------
C     INITIALIZE BACKWARD INTEGRATION:  SET TPART2 VECTOR
C-----------------------------------------------------------------------
      NTPRT2 = NPOINT
      IF (NTPRT2 .LE. 0)  GO TO 48
      DO 46 J = 1,NPOINT
   46 TPART2(J) = PVECT(J)
C
   48 CONTINUE
C
C-----------------------------------------------------------------------
C     CALL WARN TO SEE IF INDICES ARE PROPERLY SET
C     (INDICES ARE LEFT OVER FROM FORWARD INTEGRATION)
C-----------------------------------------------------------------------
      CALL WARN(MODE4,TRPR,NTRPR,T,Y,NEQN,IZERO)
C
      IF (IPRINT .EQ. 0)  RETURN
      PRINT 1504,(J,TPART2(J),J=1,NPOINT)
 1504 FORMAT(' TPART2(',I3,') = ',D15.7)
      RETURN
C
   50 CONTINUE
C
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
C     MODE 2:    UPDATE MODE.  A TPART1 IS A STOPPING CONDITION.
C                MAKE ANY UPDATE NEEDED.
C                ISTOP1 .NE. 0 ===> TPART1(ISTOP1) IS STOPPING VALUE
C
C                IN FORWARD DIFFERENCING MODE OF TROMPP, NO TPART2
C                VECTOR WILL BE IMPOSED.
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
C
      CALL PHIPAR(MODE2,NEQN,T,Y,YP,NTRPR,TRPR,TRPRP)
C
      IF (.NOT. FINTEG)  GO TO 60
C-----------------------------------------------------------------------
C     FORWARD INTEGRATION--LIFT COEFFICIENT UPDATE:
C-----------------------------------------------------------------------
C
      KL22 = IPARTL(ISTOP1)
      KU22 = IPARTU(ISTOP1)
      IBL(2,2) = KL22
      IBU(2,2) = KU22
      GRIDL(2,2) = T2P2(KL22)
      GRIDU(2,2) = T2P2(KU22)
C
      CALL WARN(MODE3,TRPR,NTRPR,T,Y,NEQN,IZERO)
C
C
      RETURN
```

```
C
   60 CONTINUE
C
C-----------------------------------------------------------------------
C     BACKWARD INTEGRATION--NO UPDATES AT T=0
C-----------------------------------------------------------------------
      IF (T .LT. 1.D-12)  RETURN
C-----------------------------------------------------------------------
C     BACKWARD INTEGRATION--DETERMINE IF TPART1 OR TPART2 STOP
C-----------------------------------------------------------------------
C
      IF (ISTOP1 .EQ. 0)   GO TO 70
C-----------------------------------------------------------------------
C     STOPPING ON A TPART1 VECTOR ELEMENT (LIFT COEFFICIENT)
C-----------------------------------------------------------------------
      IF (ISTOP1 .EQ. 1)  GO TO 70
      KL22 = IPARTL(ISTOP1-1)
      KU22 = IPARTU(ISTOP1-1)
      IBL(2,2) = KL22
      IBU(2,2) = KU22
      GRIDL(2,2) = T2P2(KL22)
      GRIDU(2,2) = T2P2(KU22)
      INDEX = 6
   70 CONTINUE
      IF (ISTOP2 .EQ. 0)  GO TO 100
C-----------------------------------------------------------------------
C     STOPPING ON A TPART2 VECTOR ELEMENT (GENERATED DURING FORWARD
C          INTEGRATION
C-----------------------------------------------------------------------
      DO 75 ITABLE = 1,NTABLE
      LIMIT = NCOMP(ITABLE)
      DO 75 ICOMP  = 1,LIMIT
      INDEX = LPOINT(ITABLE,ICOMP,ISTOP2)
C
      IF (IABS(INDEX) .EQ. 0)  GO TO 75
      IBL(ITABLE,ICOMP) = IABS(INDEX) - ISIGN(1,INDEX)
      IF (IBL(ITABLE,ICOMP) .LE. 0)  IBL(ITABLE,ICOMP) = 1
      IBU(ITABLE,ICOMP) = IBL(ITABLE,ICOMP) + 1
C
      CALL TABBND(ITABLE,ICOMP)
   75 CONTINUE
C
  100 CONTINUE
C
C-----------------------------------------------------------------------
C     CALL "WARN" TO SEE IF BOUNDS ARE VIOLATED:
C-----------------------------------------------------------------------
      CALL WARN(MODE4,TRPR,NTRPR,T,Y,NEQN,IZERO)
C-----------------------------------------------------------------------
C     UPDATE ODE:
C-----------------------------------------------------------------------
      CALL RHS(T,Y,YP)
C-----------------------------------------------------------------------
C
      RETURN
      END
```

Subroutine PARTCA, for determining lift coefficient stops

```
C-----------------------------------------------------------------------
      SUBROUTINE PARTCA(TVECT,IVECTL,IVECTU,NTVECT)
C-----------------------------------------------------------------------
C
C     PURPOSE:
C
C        PARTCA (ALONG WITH QROOT) LOCATES ALL T VALUES
C        CORRESPONDING TO CA(I) INTERSECTIONS WITH THE CUBIC
C        CURVES.  THESE T VALUES ARE STORED IN TVECT AS STOPPING
C        CONDITIONS FOR THE INTEGRATION.  IVECTL AND IVECTU
C        STORE THE BRACKETING INDICIES CORRESPONDING TO THE
C        CA COMPONENTS.
C
C        PARTCA REFERENCES "QROOT" WITH A SET OF CUBIC COEFFICIENTS AND
C        THE VECTOR "CA" TO LOCATE THE VALUES OF T FOR WHICH THE
C        CA VALUES INTERSECT THE CUBIC. PARTCA ADDS THESE POINTS
C        TO VECTOR TVECT (FORMED IN INCREASING ORDER) AND STORES
C        THE BRACKETING INDICIES OF CA CORRESPONDING TO THE TVECT(J)
C        POINT IN IVECTL(J) AND IVECTU(J).
C
C
C     STRUCTURE:   USER SUPPLIED
C
C
C     INPUT  PARAMETERS:  CUBIC COEFFICIENTS (STORED IN COMMON CTRMPP)
C                         CA VECTOR          (STORED IN COMMON TABLE2)
C
C
C     OUTPUT PARAMETERS:  TVECT      VECTOR OF T VALUES FOR STOPPING
C                                    THE INTEGRATION (DIMENSION 100)
C
C
C                         IVECTL(J)  LOWER BRACKETING CA INDEX FOR THE
C                                    INTERVAL TVECT(J) TO TVECT(J+1)
C                                    (DIMENSIONED 100)
C
C                         IVECTU(J)  UPPER BRACKETING CA INDEX FOR THE
C                                    INTERVAL TVECT(J) TO TVECT(J+1)
C                                    (DIMENSIONED 100)
C
C
C     PROGRAMMER:  M.K. HORN, DFVLR-OBERPFAFFENHOFEN, JUNE, 1982
C-----------------------------------------------------------------------
      IMPLICIT REAL*8 (A-H,O-Z)
C
      COMMON/CTRMPP/GRID(15,5),STIFF(15,5),U(15,5,5),A(15,5,5),
     1              B(15,5,5),C(15,5,5),UL(15,5,5),UU(15,5,5),
     2              UH(15,15,5,5),
     3              AH(15,15,5,5),BH(15,15,5,5),CH(15,15,5,5),
     4              P(10),Y(70),TOL,INTERV,KONTRL,KNOT(5),NP,NY,
     5              IDUM(3),
     6              LDUM(21)
      COMMON/TABLE2/T2P1(24),  CA(35),TAB2(24,35)
      COMMON/TLIMIT/INDIC(4,2),NGE(4,2),NCOMP(4),NTABLE
```

```
      DATA ZAPP/1.D-10/
C
      DIMENSION TVECT(100),IVECTL(100),IVECTU(100)
      DIMENSION TDUM(100), ITDUM(100)
      DIMENSION IV(100)
      DIMENSION CACA(35)
      DATA NTDUM/100/,IPRINT/0/,EPS/1.D-10/
      LOGICAL LOGDUM
C
C-----------------------------------------------------------------------
C
C-----------------------------------------------------------------------
      NCA = NGE(2,2)
      DO 5 J = 1,NCA
   5  CACA(J) = CA(J)
C
C-----------------------------------------------------------------------
C     T = 0   IS THE FIRST TVECT POINT--LOCATE BRACKETING INDICES
C             (U(1,1,1) IS THE LIFT COEFFICIENT AT T=0.
C-----------------------------------------------------------------------
C
      TVECT(1) = 0.0D0
      IF (IPRINT .EQ. 1)       PRINT 1555,U(1,1,1)
 1555 FORMAT(' U AT T0 = ',D15.7)
      CALL LOCATE(NCA,U(1,1,1),CACA,MPT,NPT,BNDL,BNDU,0,0)
C
C
      IF (DABS(U(1,1,1)-CA(MPT)) .GT. ZAPP) GO TO 8
C
C     INITIAL VALUE OF LIFT COEFFICIENT LIES ON A CA GRID VALUE
C         LOCATE THE CA VALUE AT T+EPSILON FOR PROPER MPT,NPT LABELING
C
      CALL LIFTC(0,EPS,CAEPS,DUMMY,0)
      CALL LOCATE(NCA,CAEPS,   CACA,MPT,NPT,BNDL,BNDU,0,0)
C
C
    8 CONTINUE
C
      IVECTL(1) = MPT
      IVECTU(1) = NPT
      NTVECT = 1
C-----------------------------------------------------------------------
C     A SET OF CUBIC COEFFICIENTS IS ASSOCIATED WITH EACH SUB-INTERVAL
C        (GRID(I,1), GRID(I+1,1).  INTERSECT CA LINES WITH EACH
C        OF THESE CUBICS AND THROW OUT ANY T VALUES NOT IN THE
C        GRID SUB-INTERVAL  (DONE IN QROOT).  THE T VALUES WITHIN
C        THE GRID SUB-INTERVAL BECOME STOPPING CONDITIONS FOR THE
C        INTEGRATION--WITH POINTS STORED IN TVECT, INDICES IN IVECTL
C        AND IVECTU.
C-----------------------------------------------------------------------
      LIMIT = KNOT(1) - 1
C-----------------------------------------------------------------------
C     LIMIT IS THE NUMBER OF CUBIC EQUATIONS BEING ANALYZED
C         ONE FOR EACH INTERVAL (GRID(I,1), GRID(I+1,1)
C-----------------------------------------------------------------------
      DO 20 I = 1,LIMIT
C-----------------------------------------------------------------------
C     SET CUBIC COEFFICIENTS FROM CTROMPP-FOR THE ITH CUBIC
```

```
C        COEFFICIENT ORDERING IS SOMEWHAT SWITCHED
C-----------------------------------------------------------------------
      AA = C( I, 1, 1)
      BB = B( I, 1, 1)
      CC = A( I, 1, 1)
      DD = U( I, 1, 1)
      TLOWER = GRID(  I,1)
      TUPPER = GRID(I+1,1)
C-----------------------------------------------------------------------
C     COEFFICIENTS FROM SPLINE ARE SET ON THE INTERVAL (0, T).
C     CALL "SHIFT" TO SHIFT THE T SCALE FROM 0 TO GRID(I,1)
C     (THIS CHANGES THE AA,BB,CC,DD VALUES).
C-----------------------------------------------------------------------
      CALL SHIFT(TLOWER,AA,BB,CC,DD)
C-----------------------------------------------------------------------
C     CALL QROOT TO INTERSECT THE CA VALUES WITH THE CURRENT CUBIC
C-----------------------------------------------------------------------
      CALL QROOT(AA,BB,CC,DD,TLOWER,TUPPER,NCA,CACA,
     1           NTDUM,TDUM,ITDUM,NDUM)
C
      IF (NDUM .EQ. 0)  GO TO 20
C
C-----------------------------------------------------------------------
C     NDUM POINTS HAVE BEEN RETURNED FROM QROOT.  ADD THESE ON TO
C     THE TVECT VECTOR AND STORE THE BRACKETING VALUES IN IVECTL
C     AND IVECTU SLOTS CORRESPONDING TO THE TVECT VALUE.
C
C     MAKE SURE THAT THE FIRST TDUM VALUE IS NOT THE LAST TVECT VALUE.
C-----------------------------------------------------------------------
C
      IF (DABS(TVECT(NTVECT)-TDUM(1)) .GT. ZAPP)   GO TO 10
C-----------------------------------------------------------------------
C     TDUM(1) REPEATS THE LAST TVECT VALUE--REMOVE FROM VECTOR
C-----------------------------------------------------------------------
      NDUM = NDUM - 1
      IF (NDUM .EQ. 0)  GO TO 20
      DO 9 J = 1,NDUM
      TDUM(J) = TDUM(J+1)
    9 ITDUM(J) = ITDUM(J+1)
C-----------------------------------------------------------------------
C     ALL TDUM POINTS ARE DISTINCT FROM EXISTING TVECT VALUES
C     ADD TDUM ON TO THE END OF TVECT
C-----------------------------------------------------------------------
   10 CONTINUE
C
      DO 15 J = 1,NDUM
      TVECT(NTVECT+J) = TDUM(J)
C-----------------------------------------------------------------------
C     ITDUM(J) = KK CORRESPONDS TO THE VECTOR(KK) VALUE.  DETERMINE IF
C        THIS IS THE UPPER OR LOWER BRACKETING VALUE AND SET IVECTL
C        AND IVECTU VALUES.
C-----------------------------------------------------------------------
      T = TDUM(J)
      DERIV = (3.D0*AA*T + 2.D0*BB)*T + CC
      IF (DABS(DERIV) .GT. ZAPP)  GO TO 11
C
C     LOCAL MINIMUM OR MAXIMUM
C
```

```
      CURVE = 6.D0*AA*T + 2.D0*BB
      IVECTL(NTVECT+J) = ITDUM(J)
      IF (CURVE .LT. 0.0D0)  IVECTL(NTVECT+J) = ITDUM(J) - 1
      IVECTU(NTVECT+J) = IVECTL(NTVECT+J) + 1
      IF (IPRINT .EQ. 1)       PRINT 1669,TDUM(J)
 1669 FORMAT(//,' CA BOUNCES ON A CACA GRID AT T = ',D15.7,//)
      GO TO 15
C
   11 CONTINUE
      IF (DERIV .GT. 0.0D0)  GO TO 12
C
C     DECREASING FUNCTION
C
      IVECTL(NTVECT+J) = ITDUM(J) - 1
      IVECTU(NTVECT+J) = ITDUM(J)
      GO TO 15
   12 CONTINUE
C
C      INCREASING FUNCTION
C
      IVECTL(NTVECT+J) = ITDUM(J)
      IVECTU(NTVECT+J) = ITDUM(J) + 1
   15 CONTINUE
      NTVECT = NTVECT + NDUM
C
   20 CONTINUE
C
      IF (IPRINT .EQ. 0) RETURN
C-----------------------------------------------------------------------
C     PRINT TVECT VALUES AND BRACKETING INDICES AS A SAFETY CHECK.
C-----------------------------------------------------------------------
C
      DO 29 J = 1,NTVECT
      PRINT 1599,J,TVECT(J),IVECTL(J),IVECTU(J)
 1599 FORMAT(' TVECT(',I3,') = ',D15.7,2(2X,I3))
   29 CONTINUE
C
C
      LIMIT = NTVECT - 1
      DO 30 J = 1,LIMIT
      TAVG = 0.5D0*(TVECT(J) + TVECT(J+1))
      CCA = SPLINT(KNOT(1),GRID(1,1),U(1,1,1),STIFF(1,1),
     1             A(1,1,1),B(1,1,1),C(1,1,1),TAVG)
      IVL = IVECTL(J)
      IVU = IVECTU(J)
      PRINT 1500,TAVG,CA(IVL),CCA,CA(IVU)
 1500 FORMAT(' TAVG = ',D15.7,2X,' CA LOWER = ',D15.7,' CA = ',D15.7,
     1       ' CA UPPER = ',D15.7)
   30 CONTINUE
C
      RETURN
      END
```

Subroutine PHIPAR, for evaluating the trapping parameters

```
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
      SUBROUTINE PHIPAR(MODE,NEQN,T,Y,YP,NTRPR,TRPR,TRPRP)
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
C     PURPOSE:  PHIPAR EVALUATES THE TRAPPING PARAMETERS, TRPR, AND
C               THEIR DERIVATIVES FOR USE IN SUBPHI.
C
C     STRUCTURE: USER SUPPLIED (CALLING SEQUENCE STANDARD)
C
C     INPUT  PARAMETERS:
C            MODE             USER IDENTIFICATION PARAMETER
C
C                             MODE=1: PHIPAR CALLED FROM SUBPHI.  TRPR
C                                     AND TRPRP ARE TO BE COMPUTED
C
C                             MODE=2: PHIPAR CALLED FROM PART. YP MAY NOT
C                                     BE AVAILABLE.  ONLY TRPR VALUES ARE
C                                     TO BE COMPUTED.
C
C            NEQN             DIMENSION OF ODE SYSTEM
C            T                INDEPENDENT VARIABLE
C            Y                DEPENDENT VARIABLES
C            YP               DERIVATIVE OF Y
C            NTRPR            NUMBER OF TRAPPING PARAMETERS
C
C     OUTPUT PARAMETERS:
C            TRPR             TRAPPING PARAMETER VECTOR
C            TRPRP            DERIVATIVES OF TRAPPING PARAMETER
C
C     PROGRAMMER:  M.K. HORN, DFVLR-OBERPFAFFENHOFEN, JUNE, 1982.
C
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
C     COMMENTS:  FOR CURRENT APPLICATION--
C                TRAPPING PARAMETERS ARE NEEDED BEFORE THE INTEGRATION
C                BEGINS IN ORDER TO SET TABLE LIMITS.  DERIVATIVES
C                ARE NOT YET AVAILABLE.  PHIPAR IS CALLED FROM PART
C                WITH MODE = 2.  TRPR ARE TO BE COMPUTED.  TRPRP ARE
C                NOT TO BE COMPUTED.
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION Y(NEQN),YP(NEQN),TRPR(NTRPR),TRPRP(NTRPR)
C
C-----------------------------------------------------------------------
C     DEFINE THE PARAMETERS USED IN THE PHI VECTOR
C-----------------------------------------------------------------------
C
C     COMPONENT 1:  MACH NUMBER = VELOCITY / SPEED OF SOUND
C     COMPONENT 2:  ALTITUDE    = H
C
C     COMMENTS: (1) VELOCITY = Y(1)
C                   SPEED OF SOUND IS OBTAINED FROM SUBROUTINE FATM
C                   DERIVATIVE INFORMATION ALSO COMES FROM FATM
C
C               (2) ALTITUDE = H = Y(3)
C
```

```
C-----------------------------------------------------------------------
C     USER SETS DIMENSION OF TRAPPING PARAMETERS, NTRPR, IN CALLING
C     PROGRAM
C-----------------------------------------------------------------------
C
C-----------------------------------------------------------------------
C     TERMS NEEDED TO EVALUATE TRPR AND TRPRP
C-----------------------------------------------------------------------
C
      V = Y(1)
      H = Y(3)
      CALL FAT1( H,RHO,A,DRHODH,DADH)
      AM = V / A
C
C     IF LIFT COEFFICIENT IS A TRAPPING PARAMETER, ACTIVATE THE
C     FOLLOWING CALL STATEMENT (WHICH IS PRESENTLY A COMMENT CARD).
C
C     IZERO = 0
C     CALL LIFTC(IZERO,T,CLIFT,DCADT,IZERO)
C
      IF (MODE .EQ. 2)  GO TO 10
C
C      DERIVITIVES MUST BE WRT NORMALIZED TIME
C
      DVDT = YP(1)
      DHDT = YP(3)
      DMDT = -V/(A*A) * DADH * DHDT + DVDT/A
C
   10 CONTINUE
C
C-----------------------------------------------------------------------
C     TRPR AND TRPRP VALUES
C-----------------------------------------------------------------------
C
      TRPR(1) = AM
      TRPR(2) = H
C
      IF (MODE .EQ. 2)  RETURN
C
C
C      DERIVITIVES MUST BE WRT NORMALIZED TIME
C
      TRPRP(1) = DMDT
      TRPRP(2) = DHDT
C
      RETURN
      END
```

Subroutine PLIFTC for determining the lift coefficient for the perturbed cubic coefficients

```
C-----------------------------------------------------------------------
C     PERTURBED COEFFICIENTS
C     LIFT COEFFICIENT GENERATOR USING CUBIC SPLINE COEFFICIENTS FROM
```

```
C          TOMP
      SUBROUTINE PLIFTC(IDERIV,T,CA,DCADT,NODE)
C-----------------------------------------------------------------------
      IMPLICIT REAL*8  (A-H,O-Z)
      COMMON/CTRMPP/GRID(15,5),STIFF(15,5),U(15,5,5),A(15,5,5),
     1             B(15,5,5),C(15,5,5),UL(15,5,5),UU(15,5,5),
     2             UH(15,15,5,5),
     3             AH(15,15,5,5),BH(15,15,5,5),CH(15,15,5,5),
     4             P(10),Y(70),TOL,INTERV,KONTRL,KNOT(5),NP,NY,
     5             LDUM(21),IDUM(3)
C
C
      LOGICAL LDUM
C-----------------------------------------------------------------------
C   NOTATION:
C
C       SOME OF THE NOTATION IN THE PROGRAM IS IN GERMAN (DUE TO
C       HISTORICAL REASONS IN DEVELOPING THE SOFTWARE).  THE FOLLOWING
C       CONVERSION TABLE IS GIVEN:
C
C       ENGLISH NOTATION                    GERMAN NOTATION
C            CL                                  CA
C
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
C     THERE IS ONLY ONE INTERVAL.  THUS, T DOES NOT HAVE TO BE LOCATED
C     WITHIN THE INTERVAL MESHES.  EACH INTERVAL "I" HAS KNOT(I)NODES.
C-----------------------------------------------------------------------
C
C-----------------------------------------------------------------------
C     COMPUTE LIFT COEFFICIENT WITH PARAMETER DISTURBANCE
C-----------------------------------------------------------------------
C
      CA = SPLINT(KNOT(1),GRID(1,1),UH(1,NODE,1,1),STIFF(1,1),
     1            AH(1,NODE,1,1),BH(1,NODE,1,1),CH(1,NODE,1,1),T)
C
      IF (IDERIV .EQ. 0)  RETURN
      DCADT = DSPLNT(KNOT(1),GRID(1,1),
     1                   UH(1,NODE,1,1),STIFF(1,1),AH(1,NODE,1,1),
     2                   BH(1,NODE,1,1),CH(1,NODE,1,1),T)
C-----------------------------------------------------------------------
C     DCADT (WRT TIME) MUST BE SCALED BY 1/TFINAL = 1/P(1)
C-----------------------------------------------------------------------
      DCADT = DCADT/P(1)
C
      RETURN
      END
```

Subroutine PR1DT1, PR2DT2, PR2DT3, PR2DT4 are all copies of model subroutine PR1DT or PR2DT given in [3]

Subroutine QROOT, for determining the zeros of a cubic equation is given in [3]

Subroutine QMXMN, used in conjunction with QROOT is given in [3]

Subroutine RD2DT2, is a copy of model subroutine RD2DT which is given in [3]

Subroutine RHS, for evaluating the differential equations,

```
C-------------------------------------------------------------------------
C-------------------------------------------------------------------------
      SUBROUTINE RHS(T,Y,YP)
C-------------------------------------------------------------------------
C-------------------------------------------------------------------------
C     PURPOSE:
C           RHS SUPPLIES THE RIGHT HAND SIDES OF THE DIFFERENTIAL
C           EQUATIONS (AND OF THE ADJOINT SYSTEM  IF FORWRD=TRUE)
C
C     STRUCTURE:  USER SUPPLIED
C
C     INPUT  PARAMETERS:
C           T       VALUE OF INDEPENDENT VARIABLE
C           Y       VALUE OF DEPENDENT VARIABLE
C
C     OUTPUT PARAMETERS:
C           YP      VALUE OF DERIVATIVE OF Y
C
C     PROGRAMMER:  M.K. HORN, DFVLR-OBERPFAFFENHOFEN, MAY, 1982
C-------------------------------------------------------------------------
      IMPLICIT REAL*8 (A-H,O-Z)
C
      DIMENSION Y(1),YP(1)
      DIMENSION PSTATE(4,4),PCA(4),PTF(4)
      DIMENSION DCA(11)
C
      DIMENSION YADJ(60),YPADJ(60)
      DATA NADJ/60/
C
C
      COMMON/CTRMPP/GRID(15,5),STIFF(15,5),U(15,5,5),ADUM(15,5,5),
     1              B(15,5,5),C(15,5,5),UL(15,5,5),UU(15,5,5),
     2              UH(15,15,5,5),
     3              AH(15,15,5,5),BH(15,15,5,5),CH(15,15,5,5),
     4              P(10),YDUM(70),TOL,INTERV,KONTRL,KNOT(5),NP,NY,
     5              IDUM(3),
     6              IMPULS,FINTEG,LDUM(19)
```

```
      COMMON/FSTEP/ITOPH
      COMMON/CHCKCA/CLIFT
      COMMON/FKOUNT/KOUNTF,KNTFIT
C
      DATA JPRINT/0/
      DATA IZERO/0/
C
C     THRUST-MAX IS FOR TWO ENGINES
      DATA TMAX,DELTA,S,G/8527.68D0,1.0D0,49.246D0,9.80665D0/
C
      LOGICAL FINTEG,LDUM,IMPULS
C
C
C     CONSTANTS NEEDED: TMAX, G, DELTA, S    (TMAX AND DELTA ARE NEEDED
C                                             FOR FTRIEB)
C
C     STATE VECTOR, Y:
C
C     Y(1) = V
C     Y(2) = GAMMA
C     Y(3) = H
C     Y(4) = WEIGHT
C
C     WITH MACH NUMBER DEFINED BY   AM = V/A, WHERE A = A(H)
C
C     IPD=0    PARTIAL DERIVATIVES ARE NOT COMPUTED
C     IPD=1    PARTIAL DERIVATIVES ARE COMPUTED IN FDRAG AND FTRIEB
C
C
      IPD = 1
      IF (FINTEG)  IPD = 0
C
      KOUNTF = KOUNTF + 1
      KNTFIT = KNTFIT + 1
C
      IPRINT = 0
      IF (JPRINT .EQ. 1)        IPRINT = ITOPH
C
      V = Y(1)
      GAMMA = Y(2)
      H = Y(3)
      WEIGHT = Y(4)
C
      SING = DSIN(GAMMA)
      COSG = DCOS(GAMMA)
C
      CALL FATM(H,RHO,A,DRHODH,DADH)
C
      CALL LIFTC(IZERO,T,CA,DCADT,IZERO)
      CLIFT = CA
      AM = V/A
C
      IF (IPRINT .NE. 0)   PRINT 1566,T,V,GAMMA,H,WEIGHT,AM,CA
 1566 FORMAT(/,' IN RHS--AT T =  ',D15.7,/,
     1       '      VELOCITY    = ',D15.7,' GAMMA    =  ',D15.7,/,
     1       '      ALTITUDE    = ',D15.7,' WEIGHT   =  ',D15.7,/,
     1       '      MACH NUMBER = ',D15.7,' LIFT CF. =  ',D15.7,//)
C
```

```
      CALL FDRAG(IPD,AM,CA,V,H,S,RHO,DRHODH,A,DADH,
     1          DRAG,DDRDV,DDRDH,DDRDCA)
      CALL FTRIEB(IPD,AM,H,V,A,DADH,DELTA,TMAX,
     1            SFC,THRUST,DSFCDV,DSFCDH,DTHRDV,DTHRDH)
C
      IF (IPRINT .NE. 0)    PRINT 1567,THRUST,DRAG
 1567 FORMAT(' THRUST = ',D15.7,2X,'DRAG = ',D15.7,/)
C
      YP(1) = G*((THRUST - DRAG)/WEIGHT - SING)
      YP(2) = G/V  * (0.5D0 * RHO * V*V * S *CA / WEIGHT - COSG )
      YP(3) = V * SING
      YP(4) = - SFC * THRUST / 3600.0D0
C
      TF = P(1)
C
      IF (IPRINT .NE. 0)    PRINT 1568,TF
 1568 FORMAT(' USING ',D15.7,'  AS TF TO SCALE THE DERIVATIVES',/)
C
C
      IF (.NOT. FINTEG)   GO TO 15
C
C-----------------------------------------------------------------------
C     DERIVATIVES ARE WRITTEN WRT TIME--FOR NORMALIZED DERIVATIVE
C     D( )/DTAU = D( )/DTIME * TF
C-----------------------------------------------------------------------
C
      DO 10 J = 1,4
   10 YP(J) = YP(J) * TF
      IF (IPRINT .EQ. 1)   PRINT 1570,TF,YP(1),YP(2),YP(3),YP(4)
C
 1570 FORMAT(' AFTER  SCALING WITH TF = ',D15.7,3X,' DERIVATIVES ARE:',
     1       /,4(2X,D15.7))
C
      RETURN
C
   15 CONTINUE
C
C-----------------------------------------------------------------------
C     FORM ADJOINT SYSTEM
C-----------------------------------------------------------------------
C     LV = Y(5)
C     LG = Y(6)
C     LH = Y(7)
C     LW = Y(8)
C-----------------------------------------------------------------------
C     DUE TO SUBSCRIPTING DIFFICULTIES--LABEL Y, YP FOR ADJOINT SYSTEM
C           YADJ AND YPADJ AND COPY INTO Y, YP BEFORE RETURNING
C-----------------------------------------------------------------------
C
      DO 18 J = 1,NADJ
      YADJ(J) = Y(J)
   18 CONTINUE
C
      DO 20 J = 1,NY
      YPADJ(J) = YP(J)
   20 CONTINUE
C
C-----------------------------------------------------------------------
```

```
C      CONSTRUCT COEFFICIENTS FOR THE DERIVATIVE OF THE HAMILTONIAN
C         WRT THE STATE VECTOR (PSTATE(4,4))
C      CONSTRUCT COEFFICIENTS FOR THE DERIVATIVE OF THE HAMILTONIAN
C         WRT LIFT COEFFICIENT (DHDCA(4))
C      CONSTRUCT COEFFICIENTS FOR THE DERIVATIVE OF THE HAMILTONIAN
C         WRT FINAL TIME       (DHDTF(4))
C-----------------------------------------------------------------------
C
      PSTATE(1,1) = -G/WEIGHT *(DTHRDV - DDRDV)
      PSTATE(1,2) = -G*(0.5D0*RHO*S*CA/WEIGHT + COSG/(V*V) )
      PSTATE(1,3) = -SING
      PSTATE(1,4) = (DSFCDV*THRUST + SFC*DTHRDV)/3600.0D0
C
      PSTATE(2,1) = G*COSG
      PSTATE(2,2) = -G*SING/V
      PSTATE(2,3) = -V*COSG
      PSTATE(2,4) = 0.0D0
C
C
      PSTATE(3,1) = -G/WEIGHT * (DTHRDH - DDRDH)
      PSTATE(3,2) = -G* 0.50D0 * V * S * CA * DRHODH / WEIGHT
      PSTATE(3,3) = 0.0D0
      PSTATE(3,4) = (DSFCDH * THRUST + SFC * DTHRDH) / 3600.0D0
C
      PSTATE(4,1) = G/(WEIGHT*WEIGHT) * (THRUST - DRAG)
      PSTATE(4,2) = G * 0.5D0 * RHO * V *S *CA / (WEIGHT*WEIGHT)
      PSTATE(4,3) = 0.0D0
      PSTATE(4,4) = 0.0D0
C
      PCA(1) = -G/WEIGHT * DDRDCA
      PCA(2) =  G*(0.5D0 * RHO * V * S / WEIGHT)
      PCA(3) =  0.0D0
      PCA(4) =  0.0D0
C
C
      PTF(1) = YP(1)/TF
      PTF(2) = YP(2)/TF
      PTF(3) = YP(3)/TF
      PTF(4) = YP(4)/TF
C
C-----------------------------------------------------------------------
C     1ST SET FOR GRADIENT OF COST FUNCTION
C-----------------------------------------------------------------------
C
      CALL MATRX2(NADJ,YADJ,YPADJ,4,PSTATE)
      CALL MATRX1(NADJ,YADJ,DHDCA1,4,PCA)
      CALL MATRX1(NADJ,YADJ,DHDTF1,4,PTF)
C
C-----------------------------------------------------------------------
C     2ND, 3RD, AND 4TH SET FOR GRADIENT OF COST FUNCTION
C-----------------------------------------------------------------------
C
      CALL MATRX2(NADJ,YADJ,YPADJ, 8,PSTATE)
      CALL MATRX1(NADJ,YADJ,DHDCA2,8,PCA)
      CALL MATRX1(NADJ,YADJ,DHDTF2,8,PTF)
C
      CALL MATRX2(NADJ,YADJ,YPADJ,12,PSTATE)
      CALL MATRX1(NADJ,YADJ,DHDCA3,12,PCA)
```

```
      CALL MATRX1(NADJ,YADJ,DHDTF3,12,PTF)
C
      CALL MATRX2(NADJ,YADJ,YPADJ,16,PSTATE)
      CALL MATRX1(NADJ,YADJ,DHDCA4,16,PCA)
      CALL MATRX1(NADJ,YADJ,DHDTF4,16,PTF)
C
      KOUNT = 20
C
C-----------------------------------------------------------------------
C     CONTROL DISTURBANCE DUE TO PARAMETER DISTURBANCE
C         FORM QUADRATURE EQUATIONS
C-----------------------------------------------------------------------
C
C     CONTROL PARAMETER AT CURRENT VALUE OF T
C
      KNOTK = KNOT(1)
C
      DO 80 I = 1,KNOTK
C
      CALL PLIFTC(IZERO,T,CA2,DERIV,I)
   80 DCA(I) = CA2 - CA
C
C     INTEGRAND FOR "QUADRATURES"
C
      DO 81  J = 1,KNOTK
      KOUNT = KOUNT + 1
      YPADJ(KOUNT) = DHDCA1*DCA(J)
   81 CONTINUE
C
      KOUNT = KOUNT + 1
      YPADJ(KOUNT) = DHDTF1
C
      DO 82  J = 1,KNOTK
      KOUNT = KOUNT + 1
      YPADJ(KOUNT) = DHDCA2*DCA(J)
   82 CONTINUE
C
      KOUNT = KOUNT + 1
      YPADJ(KOUNT) = DHDTF2
C
      DO 83  J = 1,KNOTK
      KOUNT = KOUNT + 1
      YPADJ(KOUNT) = DHDCA3*DCA(J)
   83 CONTINUE
C
      KOUNT = KOUNT + 1
      YPADJ(KOUNT) = DHDTF3
C
      DO 84  J = 1,KNOTK
      KOUNT = KOUNT + 1
      YPADJ(KOUNT) = DHDCA4*DCA(J)
   84 CONTINUE
C
      KOUNT = KOUNT + 1
      YPADJ(KOUNT) = DHDTF4
C
      DO 90 I = 1,KOUNT
   90 YP(I) = YPADJ(I)*TF
```

```
C
      RETURN
      END
```

Subroutine SHIFT, for adjusting cubic coefficients for QROOT and PARTCA is given in [3]

Subroutine SLLSQP, the optimization routine is given in ROOT and PARTCA, is given in [_]

Subroutine SUBPHI, for determining the PHI values for the RKF45T stops

```
C
C     MODEL PHI BLOCK FOR:   FORWARD DIFFERENCING OR
C                            BACKWARD DIFFERENCING
C                            PHI PARAMETERS: MACH NO. AND ALTITUDE
C                            LIFTC COEFF. STOPS ARE USER-SUPPLIED
C                            STRUCTURE OF PHI COMPONENTS
C                                      PHI(I)   = (XU-X)*(X-XL)
C
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
      SUBROUTINE SUBPHI(NPHI,INDEX,NEQN,T,Y,YP,PHI,PHIP,KOUNTR,UPDATE,
     1               IVAN,BOUNCE,ABSER)
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
C
C     PURPOSE:   SUBPHI OPERATES IN TWO MODES.
C                MODE 1:  (UPDATE = .FALSE.)
C                         THE USER MUST SUPPLY THE VALUES OF PHI,
C                         THE STOPPING CONDITIONS, AND OF PHIP, THE
C                         DERIVATIVES OF PHI WRT T, GIVEN THE
C                         VALUES OF T, Y, AND YP.
C
C                MODE 2:  (UPDATE = .TRUE.)
C                         THE USER IS INFORMED THAT PHI(INDEX) HAS
C                         VANISHED AT T, AND HE SHOULD MAKE ANY
C                         UPDATES NEEDED.
C
C
C     STRUCTURE:  STANDARD--WITH MINOR USER CHANGES (SEE BELOW.)
C
C     INPUT  PARAMETERS:
C
C       NPHI          NUMBER OF STOPPING CONDITIONS,
C                     DIMENSION OF PHI AND PHIP (2 X NO. OF TRAPPING
```

```
C                    PARAMETERS).
C       INDEX        THE PHI COMPONENT CURRENTLY BEING
C                    ANALYZED OR UPDATED.  (INDEX=0 IMPLIES
C                    INTEGRATION HAS NOT DETECTED THE EXISTENCE
C                    OF A NEW ZERO.)
C       NEQN         DIMENSION OF ODE SYSTEM (Y AND YP)
C       T            VALUE OF INDEPENDENT VARIABLE
C       Y            VALUE OF DEPENDENT VARIABLE, DIMENSIONED NEQN
C       YP           DERIVATIVE OF Y, DIMENSIONED NEQN
C       KOUNTR       COUNTING PARAMETER FOR USER'S BENEFIT.
C                    KOUNTR=0 INDICATES INITIALIZATION PHASE.
C                    KOUNTR IS CHANGED TO 1 AFTER THE INITIAL-
C                    ZATION CALL TO SUBPHI AND IS INCREMENTED
C                    BY UNITY AT EACH UPDATE CALL.
C
C       UPDATE       LOGICAL PARAMETER INDICATING MODE
C                    UPDATE=.FALSE. ===> USER MUST SUPPLY PHI AND PHIP
C                    UPDATE=.TRUE.  ===> PHI(INDEX) HAS VANISHED AT T.
C                                        MAKE ANY NEEDED CHANGES.
C       IVAN         LOGICAL PARAMETER, USED WHEN UPDATE=.TRUE.
C                    IVAN = .TRUE.  ===> PHI(INDEX) HAS VANISHED
C                                        THROUGHOUT THE INTEGRATION
C                                        STEP.
C                    IVAN = .FALSE. ===> PHI(INDEX) DID NOT VANISH
C                                        THROUGHOUT THE STEP
C
C       BOUNCE       LOGICAL PARAMETER, USED WHEN UPDATE=.TRUE.
C                                       IVAN=.FALSE.
C                    BOUNCE=.TRUE.  ===> PHI(INDEX) HAS BOUNCED ON A
C                                        ZERO. THIS POINT WAS UPDATED
C                                        ON THE PREVIOUS STEP.  IF
C                                        THE STEP IS TO BE REPEATED
C                                        SET KOUNTR=-2.  (THIS IS A
C                                        DUMMY VALUE, AND THE PREVIOUS
C                                        VALUE OF KOUNTR WILL BE
C                                        RESTORED.)
C                    BOUNCE=.FALSE. ===> PHI(INDEX) DID NOT BOUNCE ON
C                                        A ZERO
C
C     OUTPUT PARAMETERS:
C
C       PHI          USER-SUPPLIED VECTOR OF STOPPING CONDITIONS
C       PHIP         DERIVATIVE OF PHI WRT T
C       ABSER        USER-SUPPLIED, ABSOLUTE ERROR TOLERANCE
C                    DEFAULT VALUE=ABSERR, THE INTEGRATION TOLERANCE
C                    (MAX(ABSERR,RELERR))
C
C-----------------------------------------------------------------------
C     USER CHANGES:
C
C        (1) THE USER MUST SUPPLY PROPER DIMENSIONING
C            IN COMMON BLOCKS--TLIMIT AND GRIDBD
C            AND DIMENSIONING OF TRAPPING PARAMETERS
C
C        (2) THE USER MUST SUPPLY THE APPROPRIATE NUMBER
C            OF PHI AND PHI COMPONENTS. (PATTERN IS CLEAR.)
C
C        (3) THE USER MAY SUPPLY ADDITIONAL PHI VECTORS
```

```
C            NOT RELATED TO THE TABLES, AND MUST SUPPLY ANY
C            ADDITIONAL ANALYSIS NEEDED FOR THESE COMPONENTS.
C
C-----------------------------------------------------------------------
C     PROGRAMMER:  M.K. HORN, DFVLR-OBERPFAFFENHOFEN, JUNE, 1982.
C-----------------------------------------------------------------------
      IMPLICIT REAL*8 (A-H,O-Z)
C
      DIMENSION Y(NEQN),YP(NEQN)
      DIMENSION PHI(NPHI),PHIP(NPHI)
      COMMON/IDENT/ITER8,KNT,KNTFI,KNTBI
      COMMON/CRKF45/IOPT,IDUM45(4)
      COMMON/FSTEP/ITOPH
      COMMON/PRINTR/IPR
      DATA MODE1/1/,MODE2/2/,MODE3/3/,MODE4/4/
      DATA IPRINT/0/
      DATA ZAPP/1.D-10/
C
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
C     USER SUPPLIED DIMENSIONS:
C     DIMENSION OF TRPR,TRPRP, TRMAG,        =     NUMBER OF TRAPPING
C                 BOUNDL, AND BOUNDU               PARAMETERS
C
C     SEE TABLIM FOR DIMENSIONING VALUES OF COMMONS TLIMIT AND GRIDBD
C
C     USER SUPPLIES "NTRPPR", THE NUMBER OF TRAPPING PARAMETERS
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
      DIMENSION TRPR(2),TRPRP(2),SCALE(2)
      DIMENSION TRMAG(2),BOUNDL(2),BOUNDU(2)
      DATA NTRPR/2/
      COMMON/TLIMIT/INDIC(4,2),NGE(4,2),NCOMP(4),NTABLE
      COMMON/GRIDBD/GRIDL( 4, 2),GRIDU( 4, 2),IBL( 4, 2),IBU( 4, 2)
C-----------------------------------------------------------------------
      LOGICAL UPDATE,IVAN,BOUNCE,UPPER,PHIUPD
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
C     FUNCTION STATEMENTS
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
      FCT(X,XU,XL,XSCALE) = (XU - X)*(X-XL)* XSCALE
      FCTP(X,XU,XL,XSCALE)=(-2.D0*X + XU+XL)* XSCALE
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
C
C     TRAPPING PARAMETERS MUST BE EVALUATED--PERFORMED IN PHIPAR--
C     BEFORE PHI ANALYSIS CAN BEGIN
C
      CALL PHIPAR(MODE1,NEQN,T,Y,YP,NTRPR,TRPR,TRPRP)
C-----------------------------------------------------------------------
C
      IF (UPDATE)  GO TO 50
C
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
C     NORMAL MODE OF OPERATION--UPDATE=.FALSE.
C-----------------------------------------------------------------------
```