Institut für Dynamik der Flugsysteme	Bl.:
	IB.Nr.: 515-83/3

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

Freigabe: Die Bearbeiter: Unterschriften:

Dr. M.K. Horn

i.A. Klaus Hch Well

Dr. K.H. Well
Der Abteilungsleiter

Klaus Hch Well

Der stellv. Institutsdirektor:

Dr. Ing. J. Ackermann
Der Institutsdirektor:

J. Ackermann

Dieser Bericht enthält

134 Blatt davon
11 Bilder
3 Diagramme Tabellen

Ort: Oberpfaffenhofen	Datum: 28.1.1983	Bearbeiter: Dr. Horn	Zeichen Dr.Ho/rm

ABSTRACT

numerical analysis, Runge-Kutta algorithms, iteration schemes

The RKF45T software package, a fifth order Runge-Kutta integration package with step size control, is adapted to include automatic stops whenever a zero of any user-supplied, auxiliary function, PHI, vanishes. The program is capable of switching between the PHI components to locate all zeros if several components have vanishing points within a given step. The user is allowed to redefine the PHI function or the ODE expression as each zero point is located. The RKF45T program is described thoroughly with numerous examples illustrating the use of the program. The program listing is included in the report.

(This paper is printed by GML from the dataset df65.trap.text(apr).)

PREFACE

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

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

in which extensive use of linearly interpolated tabular data is made, treating the system truly as a piecewise continuous problem by halting the integration for equation updates as each table grid point is isolated. (See reference [3].)

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 [4].)

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. (Current report)

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 [5].)

CONTENTS

1. INTRODUCTION

During the numerical solution of the initial value problem

$$(1) \qquad \frac{dy}{dt} = f(t,y) , \qquad y(t_o) = y_o ,$$

one may frequently request information about the solution whenever a component of a constraint function vanishes, i.e., whenever PHI(J)=0. Examples of such constraint functions may range from isolating specific values of a particular dependent variable to analyzing complicated functions involving the independent variable, the dependent variables, and their derivatives. Typical examples include locating important points along the integration path, such as perigee or apogee of an orbit, as well as isolating singularities or stopping the solution at specified values of a dependent variable (or a function of the dependent variables). A priori knowledge of the conditions for satisfying such constraints, however, is generally unavailable. Thus, some iterative procedure must be used to adjust the integration step size in order to locate the values of the independent variable at which the components of the constraint vector vanish. The cost of such iterations, however, may be prohibitive if the problem requires frequent analysis of such constraints. An algorithm, then, is needed which can stop the numerical integration in an efficient manner whenever a user-supplied, constraint equation is satisfied.

While Runge-Kutta (RK) algorithms are effective in solving certain classes of ordinary differential equations (ODEs), their efficiency depends upon their ability to use large step sizes while generating the solution. Such iterative schemes, requiring repeated reductions in the step size, however, could demand too much additional computing time. Scaled Runge-Kutta algorithms exist which enable one to determine the solution anywhere within a given integration step at only a slight increase in computing time [2]. These scaled methods, which are used in conjunction with existing RK algorithms, are ideally suited for iterative schemes and are used to evaluate the solution at intermediate points throughout any integration step.

The RKF45 software package, developed by H.A. Watts and L.F. Shampine [6] is modified to provide an efficient integration stop whenever a component of a user-supplied, constraint vector, PHI(J), vanishes. The majority of the analysis of this constraint function, PHI is treated in additional subroutines (TRAPPD and associated routines), thus requiring only minor modifications to the original RKF45 system. To distinguish between the original and modified versions, the package which handles the constraint functions is referred to as RKF45T and is, in fact, the RKF45 package when the trapping option is not activated.

Zeros of the PHI components are identified by detecting sign changes as the PHI vector is monitored after each integration step. Once a sign change is observed (or once a zero has been "stepped on"), the integration is temporarily halted, and the analysis is shifted into separate subroutines for isolating the vanishing point (or points). If derivatives (or derivative estimates) of PHI are provided, a Newton-Rhapson (or secant) method will be used to isolate T*, the value of the independent variable at which a PHI component vanishes. If the derivative values provided produce unacceptable T* estimates, a false-position (or perhaps half interval) estimate is used to determine T*. If several components vanish within a given step (not necessar-

ily at the same point) each zero will be isolated in order in the direction of the integration. (The trapping iteration is capable of switching between the PHI components.) The user is informed of the vanishing of a PHI component through an update call to SUBPHI (the user-supplied subroutine for evaluating the PHI and PHIP components.) If more than one component vanishes at a specific point, a separate update call is made to SUBPHI for each vanishing component. During the update call, the user may print information, change the PHI expressions, or even change the differential equation system itself. Several modes of the trapping option are available so that the analysis may remain within the integration routine or may return to the user after an update, depending upon the requested mode.

Certain types of PHI functions may pose particular difficulties, for example, multiple zeros within a given integration step, relatively flat PHI components (having a large convergence region), or "bouncing" PHI functions (ones which do not change sign as they pass through zero). Several precautionary measures and several emergency features have been included to handle these situations, but the nature of the problem lies with the PHI component itself (or with its relation to the numerical solution of the differential equations) rather than with the iteration procedure. The user must understand the vectors being analyzed in order to interpret the trapping results (or difficulties).

This report presents a description of the basic (RKF45) program, as well as the documentation of the modifications to the original software package. The program listing is given in Appendix A. (Further documentation of the original RKF45 package is contained in comment cards in the program listing). This report is structured so that the user may refer to specific sections without reading the entire report (with related sections being referenced.) Examples illustrating the use of the RKF45T package are presented.

This report is actually one of a series of four reports designed to be used in the solution of the optimal control problem when discontinuities occur in the ODE or in higher derivatives of the ODE. Volume 1 of this set [3] illustrates the use of the RKF45T package for problems in which extensive use is made of linearly interpolated tabular data, with RKF45T halting the integration at each grid point in the tables. Volume 2 [4] demonstrates the use of RKF45T when a small number of discontinuities occur in the ODE, with an inequality constraint, depending upon t, y, and y', analyzed using the RKF45T package. Volume 4 of the set [5] describes the tabular structure to be used in Volume 1. The current volume illustrates further uses of the RKF45T package.

2. THE PHI FUNCTION

The inclusion of the PHI vector in the RKF45T package gives the user a great deal of programming flexibility during the solution of a system of ODEs. Reasons for temporarily halting the integration may be as simple as requesting dense output at specified increments of the independent variable or as complicated as isolating particular values of expressions in the dependent variable and its derivatives. Even the simplest of stopping conditions, however, can often be analyzed more efficiently as a component of the PHI vector than as a separate stopping condition in the driving program.

The problem of locating a specified value of a given expression, $\Phi(t,y,y') = \Phi_o$, may be formulated as one of locating the vanishing points of the function

$$PHI = \Phi(t,y,y') - \Phi_o$$

By treating a vector PHI, the RKF45T program is able to locate the vanishing points for any number of stopping conditions during the integration, switching between the components of PHI to isolate the zeros in the direction of the integration. Added flexibility is provided in an updating feature which gives the user the opportunity to print information and even to change the PHI function or the differential equations whenever a zero component of PHI is isolated. These updates may be made without returning to the driving program.

2.1 The Structure of the PHI Vector

The user-supplied PHI vector may be expressed as a function of t, y, and y'. While the analysis of this vector is fairly straightforward, the user should keep in mind that certain difficulties can arise while isolating the zeros of a PHI component even though the integration is proceeding smoothly. Difficulties in the PHI analysis might occur, for example, due to effects from errors in the integration. These difficulties occur regardless of the iteration scheme used. Thus, the user must have some understanding of the PHI vector being provided. (Emergency analysis can be activated, but this, in general, is inefficient. See §§9.2.2, 9.3, and 10.5.)

2.2 Sample PHI Components

Examples of simple but useful PHI components might included:

- Independent variable increments, PHI(1)=T-TPRINT, where TPRINT is increased each time PHI(1)=0 is isolated,
- Dependent variable stops, PHI(2)=Y(3)-YPRINT,
- A function of T, Y, and Y', say perigee or apogee of an elliptic orbit, i.e., V•R=0, PHI(3) = YP(1)*Y(1) + YP(2)*Y(2) + YP(3)*Y(3),

  or

- Grid values for interpolating tabular data, PHI(4)=(MACHUP-M)*(M-MACHLO) where MACHUP and MACHLO are entries in a table and where M may depend upon T, Y, and Y'. MACHUP and MACHLO may be incremented when PHI(4) vanishes, so that the bracketing values of the new region are used.

Such functions change sign as they pass through a zero point identifying conditions for activating an option to isolate a zero. The identification process may encounter difficulties if the PHI components change sign more than once within a given step or if they "bounce" on the zero value, i.e., if PHI has the same sign on both sides of a zero. A good understanding of the PHI components, however, will avoid the need to monitor the PHI vector carefully.

2.3 Convergence Regions

The user is seeking the value T* for which a component of PHI, say PHI(J), vanishes. In general, however, an iterative process can only approximate the zero of PHI(J). Thus, a neighborhood of T* exists in which any value of T will satisfy the requirements that |PHI(J)| be less than a prescribed tolerance. (While this tolerance may be quite small, certain restrictions, e.g., the integration accuracy, dictate a limiting lower bound. Therefore, a convergence region exists whose size depends upon the tolerance requested and the nature of the PHI component itself.(See Figure 1.)

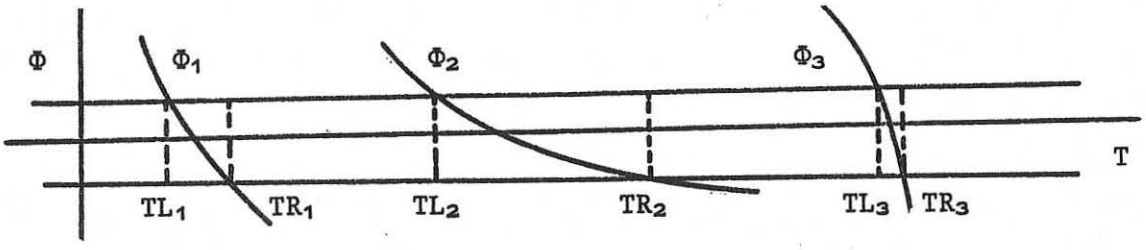


Figure 1. Convergence regions for Φ functions. $|\Phi_i| <$ *tolerance* for $T^* \in (TL_i, TR_i)$, i=1,2,3.

2.4 Analysis of the PHI Function

Communication between the RKF45T package and the user occurs, in part, through the calling sequence of the user-supplied subroutine, SUBPHI, which computes the PHI component values (See §4.1.2). During the solution of the ODE system, SUBPHI is referenced after each successful integration step to evaluate the PHI components. The integration continues with the PHI vector monitored in SETRAP until a sign change is detected in any component or until a component has vanished at the end of a given step. At this point, the integration is halted and the analysis is switched to the TRAPPD subroutine to isolate the components of PHI vanishing within that step.

In subroutine TRAPPD an iterative procedure is used to isolate the first component of PHI to change signs. (If the zero is located at the end of the integration step, no iterations are required.) The user is informed of the vanishing point through an update call to SUBPHI, which identifies the component and gives the user certain information about that function. (The procedure used in subroutine TRAPPD is described in § 5.) If multiple components of PHI vanish at the same point, a separate update is made for each vanishing component. The trapping routine may then continue to locate further vanishing components within that step or may return to the basic integration package (and possibly to the user) depending upon the mode of operation selected.

3. THE RKF45T STRUCTURE

The RKF45T integration package, with an option to stop the integration should a constraint vector be satisfied, is a modification of the RKF45 system developed by H.A. Watts and L.F. Shampine [6]. The core of the integration system is a fifth order Runge-Kutta method due to E. Fehlberg [1], having an embedded fourth order solution for step size control. Modifications to the RKF45 subroutines have been inserted in blocks as far as possible, to avoid reformulating the existing modular structure of the RKF45 package. The majority of the work done in analyzing a constraint function is managed through an additional subroutine, TRAPPD, which is referenced once the integrator (through subroutine SETRAP) detects a sign change (or a vanishing value) of a component of the constraint vector, PHI. TRAPPD references a number of subroutines during its normal operation. Additional subroutines, SCALED, VANISH, and PANIC, are major subroutines which treat particular situations arising during the analysis in TRAPPD or SETRAP. The subroutines FLAGCK and OUTFLG are attached to the integrator itself for analysis of the operation parameters. In addition, the user may switch modes of operation easily, even activating the trapping option after the integration has begun.

3.1 Subroutines in the RKF45T Package

The basic tasks of each of the subroutines in the RKF45T package are described below:

RKF45T is an interfacing routine between the driving program and RKFST. RKF45T partitions the WORK array and references FLAGCK for possible adjustments to some internal parameters before referencing RKFST. Before returning to the user RKF45T again references FLAGCK (for further adjustments to internal flags) and OUTFLG to print warning messages if the integration or trapping procedure has not gone as planned.

RKFST is the decision making routine for the solution of the ODE. RKFST analyzes the step size, deciding the initial step length, estimating the accuracy of each step, and making further adjustments to the step size. Flags are set and adjusted in RKFST to determine the success or failure of the integration. RKFST references SETRAP after each step to monitor the PHI vector.

FEHL evaluates the Runge-Kutta solution at T + H.

SETRAP serves as the interfacing routine between RKFST and TRAPPD, monitoring the PHI vector after each integration step and deciding when TRAPPD should be referenced.

TRAPPD organizes the search for the zeros of PHI(INDEX), referencing BOUNCD, TSTAR, SHIFTI, and MULTOP to perform simple tasks.

VANISH studies PHI functions which may have vanished throughout the trapping interval (or which have vanished at the initial conditions).

PANIC is an emergency routine which studies the PHI components throughout the interval if problems have arisen during the analysis.

SCALED computes the solution at any given T value within a given integration step.

FLAGCK makes internal IFLAG adjustments.

OUTFLG prints warning messages designated by IFLAG.

3.2 An Overview of the RKF45T analysis

The RKF45 package is a well tested, production software package. The modifications forming the RKF45T system would still be considered to be a research code. Thus, problems encountered during the use of the program are probably due to the modifications in the code or to user error.

The software package evaluates the derivative, selects the step size automatically (including the initial step size), and advances the solution in the manner requested by the user. Any problems encountered during the solution, whether due to difficulties in the ODE system or due to user input or response, are indicated by the parameter IFLAG. Certain values of IFLAG indicate that problems have arisen, which are sufficiently severe, that a continued attempt at integrating will terminate the program. (See Table 1.) During the solution, the dependent variables, Y, and their derivatives, YP, (stored in WORK array locations 1 through NEQN) always correspond to the value, T, of the independent variable, although T may not have reached TOUT or may not even have been advanced a single step since being referenced by the driving program (as indicated by the parameter IFLAG.)

If a constraint vector PHI is being analyzed, each component is monitored as the integration proceeds. Whenever any component changes sign over an integration step or vanishes at the end of the step, the integration is temporarily interrupted and the analysis is shifted to the TRAPPD subroutine to isolate the zeros of PHI or to permit updating of any vanishing component of PHI. Once the PHI vector has been analyzed, the user is informed of the location of vanishing values of PHI isolated by TRAPPD, and the integration is continued or the solution is returned to the driving program, according to the user's specifications.

4. CALLING SEQUENCES FOR THE RKF45T PACKAGE AND USER-SUPPLIED SUBROUTINES

A basic understanding of the parameters used in referencing the RKF45T package and the user-supplied subroutines, F and SUBPHI, is needed before the actual communication between the user and the RKF45T package can be discussed. In this section these parameters, including the various modes of operation will be defined. The actual interaction between the user's program and the RKF45T package is discussed in §7.

4.1 User-Supplied Subroutines

The user is required to supply two subroutines which are used in conjunction with the RKF45T package, both being listed in an external reference statement in the referencing program.

4.1.1 Subroutine F

Subroutine F, having calling sequence F(T,Y,YP), evaluates YP, the derivative of vector, Y. The input parameters are:

T the independent variable, and

Y the dependent variable, dimensioned NEQN in the referencing subroutines

with the returned values

YP the user-supplied derivative of the vector Y (eqn. 1), dimensioned NEQN in the referencing subroutines.

Parameters T, Y, YP are *double precision* variables.

4.1.2 Subroutine SUBPHI

Subroutine SUBPHI, having calling sequence SUBPHI(NPHI, INDEX, T, Y, YP, PHI, PHIP, KOUNTR, UPDATE, IVAN, BOUNCE, ABSER) evaluates the user supplied constraint vector, PHI, for analysis in the RKF45T program. The rather lengthy calling sequence provides considerable flexibility for the user, so that SUBPHI may perform far more tasks than simply evaluating the PHI function. The input parameters are listed below.

NPHI gives the dimension of the constraint vector PHI (and of its derivative PHIP),

INDEX equals zero if the analysis of PHI is still handled in RKFST/SETRAP or designates the component of PHI currently being analyzed in TRAPPD if a sign change in a PHI component has been observed. (If INDEX is not zero, the analysis has shifted into the TRAPPD routine.) INDEX may change values during the trapping process as the iteration converges to the first component (and eventually to each component) that experiences a sign change over the integration step being analyzed,

T is the independent variable,

Y is a vector of the dependent variables, dimensioned NEQN in the referencing subroutines,

YP is the derivative of Y, a vector dimensioned NEQN in the referencing subroutines,

KOUNTR counts the number of zeros isolated by TRAPPD (KOUNTR=0 indicates the initialization phase in SUBPHI),

UPDATE a logical parameter, designates the status of the trapping procedure,

IVAN a logical parameter, indicates whether or not a PHI component has vanished throughout the interval from T to T*,

BOUNCE a logical parameter, indicates whether or not a PHI component has "bounced" on a zero, and

ABSER is a user-supplied absolute error tolerance used in defining the convergence of the PHI vector (the default value being DMAX1{RELERR,ABSERR}, integration tolerances for RKF45T).

Returned values (user-supplied) for UPDATE=.FALSE. are:

PHI the constraint vector, dimensioned NPHI in referencing routines, and

PHIP the derivative (or derivative estimate) of the PHI vector, dimensioned NPHI in the referencing routines. (If the derivative or derivative estimate of any PHI is unavailable, that PHIP value should be set equal to zero.)

Again, T, Y, YP, as well as, ABSER, PHI, and PHIP, are *double precision* variables, while UPDATE, IVAN, and BOUNCE are *logical* variables.

The parameters, INDEX, KOUNTR, UPDATE, BOUNCE, and IVAN, are provided for the user's benefit and may be changed after the initial call to SUBPHI without affecting the integration or trapping routines. Because the primary function of these parameters is to communicate information to the user, they will be discussed in greater detail in §7.

4.2 Calling Sequence and Operating Options for RKF45T

The initial and final interaction between the user and the RKF45T package occurs through the calling sequence of RKF45T. If a PHI vector is being analyzed, further communication occurs when the vanishing point of a PHI component is isolated, but the information about the ultimate success or failure of the integration is generally given by the RKF45T parameters. Since the returned solution and flags depend upon the mode of operation selected, the user should fully understand the information returned by the RKF45T program as well as the various modes of operation.

4.2.1 The RKF45T Calling Sequence

The user supplies the following information to the integration package:

F, SUBPHI — user-supplied subroutines (given in external reference statements in the driving program) with calling sequences described in §4.1,

NPHI — the number of components of vector, PHI.

NEQN — the number of components of the dependent variable, Y,

Y — the initial value of the dependent variable, Y, dimensioned NEQN

T — the initial value of the independent variable,

TOUT — the desired value of the independent variable upon return to the user (in a continuous mode of operation) or the indicator for the integration direction (and the limiting value of T in a step-by-step mode),

RELERR — an accuracy estimate for the relative error in the solution,

ABSERR — an accuracy estimate for the absolute error in the solution,

IFLAG — a flag designating the mode of operation selected (See § 4.2.2.),

WORK — a work array that masks information which is needed by the program but which is generally not needed by the user (See Table 2.a.), and

IWORK — an integer work array that masks information which is needed by the program but which is generally not needed by the user (See Table 2.b.).

Returned parameters are:

T — the current value of the independent variable (not necessarily TOUT)

Y — the value of the dependent variable at T

IFLAG — a flag designating the status of the integration,

WORK — the work array with new values which are generally not needed by the user (See Table 2.a.), and

IWORK — the integer work array with new values which are generally not needed by the user (See Table 2.b.).

Parameters Y, T, TOUT, RELERR, ABSERR, and WORK are *double precision* variables.

4.2.2 RKF45T Modes of Operation

The RKF45T integration package may be operated in several modes. More specifically, the integrator itself may be operated in two different ways: (1) in a step-by-step mode, with the solution being returned to the user after each step, or (2) in a continuous mode, with the solution being returned to the

user when the integration has reached TOUT. In addition, there are several trapping options, corresponding, in part, to the mode used in the integration. The trapping option may be activated during the initial call to the integrator or may be started during the solution of the ODE system.

Each mode of operation is identified by the parameter IFLAG. In the basic integrator, IFLAG=-1 activates the step-by-step mode, while IFLAG=1 activates the continuous mode. Upon return to the user IFLAG is set equal to -2 or +2, respectively, unless TOUT has been reached in the step-by-step mode, in which case IFLAG=2. Similar modes of operation are available with the trapping option, but with two options for both the step-by-step mode and the continuous mode. Also, similar flags are used: (1) IFLAG=-10 or -15 for the step-by-step mode or (2) IFLAG=10 or 15 for the continuous mode, with return values of -20, -25, 20, and 25, respectively. (For the "late start" option the starting flag is shifted -1 or +1, depending upon the mode used. The shifted values are then reset to standard IFLAG values by the RKF45T package. (See § 4.2.2.6.)) The two, step-by-step modes may return to the user before the integration has reached TOUT, whereas the continuous modes return the solution at TOUT (unless difficulties are encountered during the integration.) The user should study the two continuous modes carefully. Although each is designed to handle a different type of problem, one of the two modes is generally sufficient to treat any problem. Both of these methods isolate all vanishing points of PHI throughout the interval of integration, providing communication between the software package and the user whenever a component of PHI vanishes. (Thus, printing data, changing the PHI function, and even changing the ODE system are possible without ever returning to the driving program.)

4.2.2.1 Step-by-Step Mode (with Trapping Option)

The simplest mode of operation is the step-by-step mode (IFLAG=-10). This option is also the least efficient mode and should be used only when the solution must be carefully monitored. The solution will be advanced a single step and returned to the user. If a component of the PHI function vanishes during the step, the solution is returned at the "trapped" point. If several components of PHI vanish during the step, the solution is returned at the value of T corresponding to the first PHI component to vanish. After a component of PHI has been "trapped", an update is made in the SUBPHI routine, giving the user an immediate opportunity to identify the component. (Also, if several components vanish at that value of T, an update call will be made for each component.) Although the solution is returned at the "trapped" point, the conditions at the end of the step, TF, YF, and YPF, are stored in the WORK array if the user should need them. (See §7.2.) If no component of PHI has "vanished" during the step, the solution is returned at the end of the step. If TOUT is reached, IFLAG will be reset to +2. Otherwise, IFLAG=-20, which is the value needed for continuing the integration in this mode until TOUT is reached.

4.2.2.2 Trapping Step-by-Trapping Step Mode

A similar mode of operation is the trapping step-by-trapping step mode. IFLAG=-15 activates a mode of operation, which is a combination of the step-by-step and the continuous integration options. The integration itself is advanced in the continuous mode, with a return to the user being activated whenever a vanishing component of PHI has been isolated within a particular step. Conditions and update procedures are similar to those for the

step-by-step mode. If TOUT is reached, IFLAG will be reset to +2. Otherwise, IFLAG=-25, which is the value needed for continuing the integration until TOUT is reached. Values at the end of the integration step, TF, YF, YPF, are returned in the WORK array locations. (See § 7.2.) The efficiency of the trapping step-by-trapping step mode decreases with the increasing number of returns to the user, due, in part, to the overhead involved in referencing the integration subroutines. The user should investigate the possibilities of using a fully continuous mode before selecting the trapping step-by-trapping step mode.

4.2.2.3 Single Trapping Mode (Continuous)

The single trap mode of operation isolates each vanishing value of PHI, advancing the solution from the most recently trapped point. If several components vanish at a specific value of T, all components are identified, each in a separate update call to SUBPHI. Should several components of the PHI vector vanish within a given step, but at different values of T, this mode of operation will isolate the first vanishing point and continue the integration from that point. The further vanishing values will be isolated on subsequent steps. This mode of operation permits update in both the PHI function and the differential equation system. Because the ODE may be updated, however, the PHI vector (and its zero points) may also change after the trapped point. Thus, the integration and PHI function analysis must be started again at the trapped point.

4.2.2.4 Multiple Trapping Mode (Continuous)

The second form of continuous integration, the multiple trap option, searches for all vanishing values of PHI throughout a given step and then advances the solution from the end of the step taken. The multiple trap option does not allow updates in the differential equation system, but does permit updates in the PHI function itself. If an update in the PHI function occurs, however, further vanishing values of PHI are sought only between the last trapped point and the end of the step. (If the newly introduced PHI function vanishes over a previously studied region, these values are not considered to be applicable to the problem. Otherwise, the new PHI function should have been analyzed as that region was being studied.) This mode of operation is ideally suited to treat dense output at specified increments of the independent variable, because the requested output point may be shifted forward at each update of that PHI component, allowing all points to be isolated throughout the step with no further derivative evaluation expense. (See example 1, §10.)

4.2.2.5 Combination Trapping Mode (Continuous)

The multiple trap mode may be used in conjunction with the single trap mode. This combination option is actually a multiple trap mode which stops locating additional zeros within a given step once a PHI component, designated as a single trap component, is isolated. The PHI vector is partitioned, with the first M components being analyzed in the multiple trap manner and the remaining NPHI-M components, in the single trap manner. Properties of each option apply to their corresponding PHI components.

The combination option is activated using the same IFLAG values as in the multiple trapping option. The information for designating the combination option (as opposed to the normal multiple trapping option) occurs in subrou-

tine SUBPHI. (This rather indirect means of activating the option helps avoid further deviations from the standard RKF45 calling sequence, and permits users of the standard continuous options to ignore the mode completely.) The user must partition the PHI vector so that the first M components are those which will be analyzed in the multiple trapping sense. The limit M is conveyed by the user through the parameter INDEX on the first call to SUBPHI (at which time KOUNTR=0). Failure to supply an initial INDEX (when IFLAG=15) simply applies the multiple trap mode to the entire vector, i.e., the default value in RKF45T is NPHI. (Once information from INDEX is obtained on the initial call to SUBPHI, INDEX is reset (to zero) in RKFST, and user changes to the parameter no longer affect the RKF45T package.) Activating the combination mode is most easily described by example. (See example 2, §10.)

4.2.2.6 The Late Start Option

The user may choose to advance the ODE solution for some time before activating the trapping option. Rather than reinitializing the entire integration, the user may select IFLAG so that only the section of the program needed to activate the trapping option is set. The standard trapping options are still available, but the IFLAG designation is IFLAG=-16,-11,11,16, corresponding to the options IFLAG=-15,-10,10,15, respectively. During the initialization, IFLAG is reset to its appropriate value. Since the options are the same, the "late start" is not considered to be an actual mode of operation and so is not generally mentioned with the other options throughout this report.

5. GENERAL PROCEDURE IN SUBROUTINE TRAPPD

Subroutine TRAPPD handles the major part of the analysis of the vector function PHI once any component of PHI experiences a sign change over a given step or vanishes at the end of a step. To simplify the analysis in TRAPPD, a number of subroutines are referenced to perform basic tasks such as selecting the new zero estimate point or shifting indices. In addition, several fundamental routines are referenced, namely, PANIC, SCALED, and VANISH, to treat particular important conditions which may arise. TRAPPD, then, may be considered as the final judge concerning the success of the search.

TRAPPD attempts to locate the zeros of the PHI components which have changed sign over the integration step or which have vanished at the end of the step (within the specified tolerance). Since several components of PHI may have changed sign over a given step, INDEX indicates that particular component (having experienced a sign change) which has the largest magnitude at the most recently analyzed point across the boundary. INDEX may be shifted between components as the trapping procedure continues, assuring that the component which most strongly violates the sign change restriction is being trapped. Thus, the zeros of the PHI components are isolated in order in the direction of integration.

5.1 PHI Vanishes at TF

Several situations may exist upon entry into TRAPPD. The simplest conditions to analyze occurs when PHI(INDEX) has "vanished" at TF, i.e., when |PHIF(INDEX)| is less than the prescribed tolerance. The integrator has managed to step on a zero of at least one component of PHI, and no trapping iteration is necessary. An update is made for PHI(INDEX) and for any other PHI component which may also have vanished, and the analysis is returned to the RKFST routine. (If another component of PHI had changed signs during the integration step, but had failed to vanish at TF, INDEX would have designated that component. Thus, if |PHIF(INDEX)| < tolerance, several components may have vanished at TF, but none should have vanished between T and TF.)

5.2 The Trapping Iteration

The detailed analysis in TRAPPD occurs when at least one component of PHI has changed sign and has not yet vanished. If several zeros occur within the step, the INDEX parameter will be shifted automatically until the first vanishing component is isolated. The trapping portion of the subroutine estimates the vanishing point, T*, calls SCALED to evaluate the solution at T*, shifts INDEX if another component, having experienced a sign change, is of greater magnitude, and continues "trapping" until convergence is achieved. The analysis is then shifted to the update section of the routine, where the user is given the opportunity to update the PHI function, the solution, or the differential equations, depending upon the mode of operation designated by the user.

5.3 The Update of PHI Components

Once the first vanishing component of PHI has been isolated, the analysis shifts to the update section during which SUBPHI is referenced, giving the user a chance to update conditions without returning to the driving program. Before the user has access to the situation, however, TRAPPD sets certain flags and analyzes the PHI function to inform the user of possible difficulties. (See § 6.5.) A separate update call is made to SUBPHI for *each* component of the PHI vector which has vanished at the current value of T*, with the parameter INDEX indicating the component of PHI being updated.

5.4 Trapping Additional Values within a Given Step

If the multiple trapping mode of operation is used, all vanishing values of PHI within a given step are isolated. Once the first vanishing component of PHI is trapped and updated, the PHI vector is reevaluated at TF (to assure correct values, since PHI may have been updated). Then the PHI vector is analyzed to see if any components have changed sign between the trapped point and TF. If additional components need analyzing, the trapping iteration is repeated.

If a combination of the multiple and single trapping modes is used, further vanishing points are sought until a zero of a PHI component corresponding to the single trap mode is isolated. The integration will then be continued from this zero point.

6. DETAILS OF SUBROUTINE TRAPPD

The analysis in subroutine TRAPPD is handled in several blocks, with additional subroutines being referenced to treat special problems which may arise. TRAPPD is referenced whenever a component of PHI has either changed sign over an integration step or vanished within the specified tolerance at the end of the step, i.e., at TF. If several components of PHI have vanished at TF or changed sign over a given step, the parameter INDEX designates the component from this set, having the largest magnitude at TF. All decisions for estimating the first vanishing point for the PHI components will be made using PHI(INDEX). As the trapping iteration continues, however, INDEX may be switched between components as the values of PHI across the boundary change, assuring that the component which most strongly violates the sign change criterion is being trapped. Trapped and related subroutines are listed in Appendix A.

6.1 The Initialization Block for TRAPPD

The initialization part of TRAPPD contains two safety features: (1) the "bouncing" analysis, and (2) the PANIC option. In addition, a standard printing option is available, although the option should generally be activated only if the PHI analysis encounters difficulties. The initialization block labels the PHI components and sets other parameters used internally by the trapping analysis.

The user may need to understand the labeling of the points used in the trapping analysis. Bracketing values of T for locating T* (the predicted value at which PHI(INDEX) will vanish) are established in RKFST and points are labeled before entry into TRAPPD. TRAPPD then reduces the bracketing interval until the zero point is isolated. PHI and PHIP vectors are associated with each labeled point. The solution, however, is stored only at T, TF, and T2 (where T2 is the current estimate of T*). The initial conditions are labeled "L", the final conditions, "R". (If the emergency analysis has been used in RKF45T, "L" and "R" are end points of a substep of the integration step length, h. Otherwise, "L" conditions are those at "0", and "R" conditions are those at "F".) T* estimates are labeled "2" with the initial "2" values being those at "L". (See Figure 2.)

Subroutine BOUNCE is referenced upon entry into TRAPPD in order to see if any component of PHI has been labeled improperly. The idea of a "bouncing" PHI component, i.e., a component which does not change sign as it passes through zero, is detrimental to the analysis in TRAPPD (and in general, the only means of detecting such a zero is to step on it). If such a zero has been isolated, however, an incorrect sign may have been imposed by TRAPPD, and a correction is in order. (The detection of a bouncing PHI function is not as

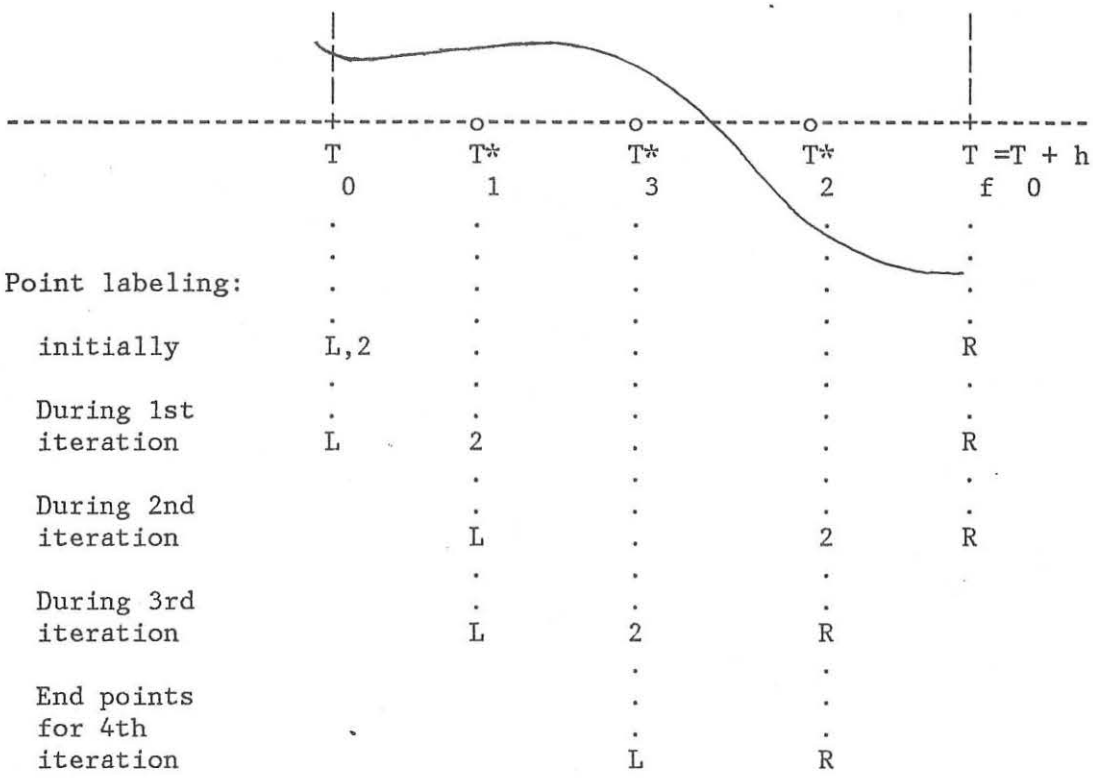


Figure 2. Point labelling during trapping iteration. "L" denotes left end point, "R", right end point, and "2", the current estimate of T*, the vanishing point.

far-fetched as one might think, particularly in analyzing constraint equations in which the solution stays within a narrow band about the constraint for a while and then diverges.) The bouncing analysis is protection which requires little computing time and which is necessary for some of the proposed applications of RKF45T. The problem of "bouncing" PHI functions and the BOUNCE analysis is described thoroughly in §9.9.

An emergency feature, PANIC, is also available in the initialization block of TRAPPD (if no PANIC feature is active in RKFST). This feature is not recommended for general use. The user may reference PANIC to print information about PHI throughout the integration step. This emergency option is difficult to activate (intentionally) because of the amount of additional computing time required. If major difficulties occur during the trapping procedure, e.g., if the maximum number of derivative evaluations is exceeded, TRAPPD references PANIC itself in order to print out PHI values throughout the step before terminating the analysis. Means of activating the PANIC option are given in §§ 8.5, 9.2.2, and 9.3.)

6.2 Determining the Trapping Status of PHI

Upon entry into TRAPPD, the current status of the PHI components must be checked. Since several components may have changed sign over the interval or vanished at TF, the parameter INDEX indicates the component of PHI which most strongly violates the convergence criteria for the vector. Thus, if PHI(INDEX) satisfies the convergence criterion at TF, the components of PHI have failed to change sign over the integration step or have vanished (within the prescribed tolerance) at TF. In this case, the trapping iteration is not activated. Instead an immediate update in PHI is possible. An additional safety check concerning the PHI vector is made. If PHI(INDEX) has changed signs (without vanishing at TF), the trapping option is activated. If PHI(INDEX) has neither changed sign nor vanished at TF, an error has occurred in the analysis in SETRAP, and TRAPPD should not have been referenced. In this case a warning message is printed, and the program is terminated.

6.3 Choosing the Iteration Step Size

The trapping routine must estimate the value of T* for which PHI(INDEX) will vanish. The ideal situation is to have a sufficiently smooth PHI function over the current bounds (T, TF), so that a Newton-Rhapson iteration may be used. In practice, however, one must take certain precautionary measures. Possible choices of the T* estimate are:

1. Newton-Rhapson (or secant method if derivative approximations are used)

2. False-Position, or

3. Half-Interval

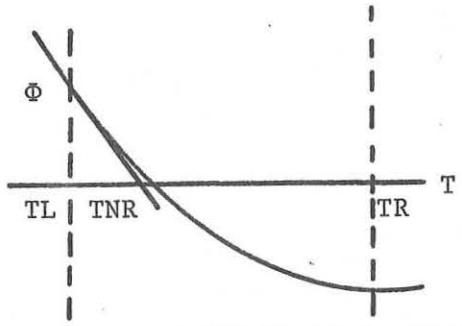
The Newton-Rhapson estimate will be chosen as long as the estimated value of T* remains within the trapping bounds. If this value of T* does violate the trapping bounds, the false-position estimate will be used to give the iteration a "kick" and hopefully to move the trapping bounds out of the problem area. (See Figure 3.)

The Newton-Rhapson estimate needs further discussion. The user is asked to provide derivative values of PHI (or derivative estimates) in the calling sequence of SUBPHI. To "deactivate" the Newton-Rhapson estimate, the PHIP values may be set to zero. TRAPPD will then reset them to the unit round-off value which will give an invalid Newton estimate. If the PHIP values are difficult to generate, secant values may be determined easily in SUBPHI by merely using the PHI values before and after the current PHI evaluation, giving a reasonable derivative estimate and requiring the storage of only the T value from the previous step. (See example 2, §10.)

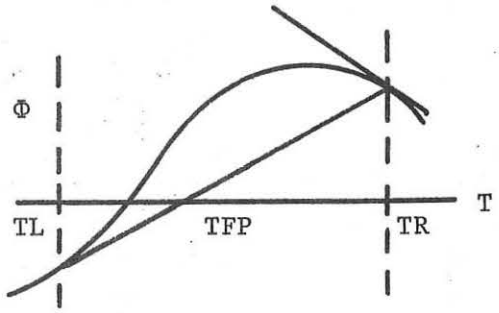
The Newton-Rhapson estimate is initially generated from the end point at which |PHI(INDEX)| is smaller, unless PHI(INDEX) vanishes at T, in which case the "R" bound, i.e., TF, is used. During the iteration, the Newton estimate is made from the previous T* point (unless T* has not yet moved outside of the "vanished region" about TL.)

An initially vanishing PHI(INDEX) can occur if the component has vanished on the previous step and changes sign over the current step. Because PHI(INDEX)

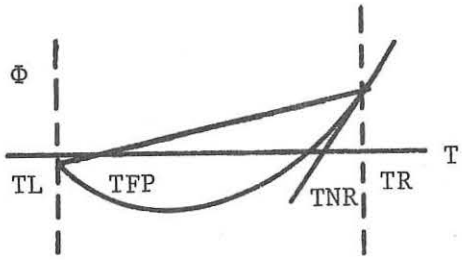
is near zero at "TL" (within the prescribed tolerance), the false position estimate will remain close to TL, thus necessitating the half interval estimate if the Newton-Rhapson value is unacceptable or repeats a previous T* value. (See Figure 3.)

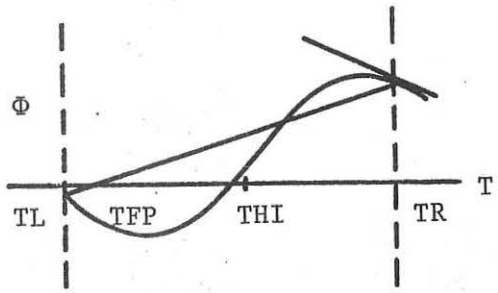


(a) Newton-Rhapson estimate, TNR, starting from TL



(b) Newton-Rhapson estimate, TNR, starting from TR (unacceptable). Switch to False-Position





(c) Φ vanishes at TL. If the false-position estimate, TFP, is unacceptable, use Newton-Rhapson estimate, TNR, starting from TR or half interval estimate, THI.

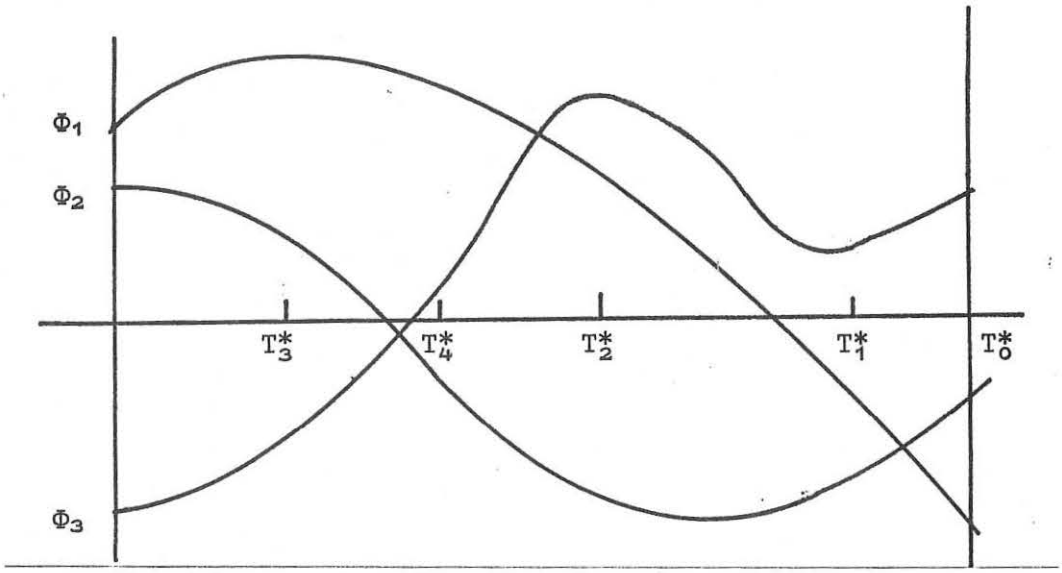
Figure 3. T* estimates for the vanishing point of Φ = PHI(INDEX) TNR = Newton-Rhapson estimate, TF = False-postion estimate, and THI = Half-interval estimate.

The selected T* value determines the iteration interval, (T, T*) which must be larger than TBOUND*TAVG, where TBOUND is related to the machine precision, and where TAVG is an average magnitude of T. If the interval is too small, IFLAG is set equal to 97, the iteration is halted, and the subroutine PANIC will be referenced to print out values of PHI and T throughout the interval before the program is returned to the user. The values of T, Y, and YP will be set to conditions at TL, the last point analyzed for which PHI(INDEX) did not change sign over the interval. If the integration interval is of acceptable length, the iteration proceeds by calling subroutine SCALED to generate the scaled solution at T*.

6.4 INDEX Shift

Once subroutine SCALED has determined the solution at T*, the most recent iteration point, the PHI vector must be analyzed to assure that INDEX indicates that component which most strongly violates the convergence criteria. The only possiblilty of a shift in INDEX occurs when multiple components of PHI experience a sign change over the interval (TL, T*). (See Figure 4.)

If several components do change sign over the (TL,T*) interval, INDEX designates the component of that set having the largest magnitude at T*. If INDEX has been shifted, iteration parameters are reset and the convergence test is made.



Iteration :	0	1	2	3	4
INDEX:	1	2	3	3	2

Figure 4. Changes in parameter INDEX. Decisions for chosing T* are based on the characteristics of PHI(INDEX). (In this example, T* values are selected for illustrative purposes and not by a Newton-Rhapson or false-position procedure.)

6.5 The Update of PHI Components

Once the first vanishing component of PHI has been isolated, the analysis shifts to the update section during which SUBPHI is referenced, giving the user a chance to update conditions without returning to the driving program. Before the user has access to the situation, however, TRAPPD sets certain flags and analyzes the PHI function to inform the user of possible difficulties. TRAPPD references VANISH to determine whether or not the solution has vanished throughout the entire integration step. (See §8.3.) Then, TRAPPD references SUBPHI for *each* component of PHI which has vanished, passing important information to the user through the parameters UPDATE, INDEX, IVAN, and KOUNTR. During the updating portion (UPDATE=.TRUE.), UPDATE indicates

that TRAPPD is in its updating mode, while INDEX=J, indicates that PHI(J) has vanished within the prescribed tolerance (with separate calls to SUBPHI for each PHI(J) which has vanished). The logical parameter, IVAN, is included to avoid possible confusion in iterpreting the updating procedure (particularly if PHI or the differential equations system is being altered). During a given integration step, a particular PHI component may have vanished throughout the entire step. IVAN=.TRUE. informs the user that PHI(INDEX) has vanished throughout the entire step. IVAN=.FALSE. informs the user that PHI(INDEX) has passed out of any "vanished" region during the integration step. The additional term, KOUNTR, is a counting parameter. For the first call to SUBPHI (during the intialization process), KOUNTR=0, giving the user a flag for setting any initialization parameters needed in evaluating terms in the user-supplied subroutine SUBPHI. Each time a zero of a PHI component has been isolated, KOUNTR is incremented by unity before the update call to SUBPHI. If several components of PHI vanish at one value of T, KOUNTR will be incremented as each component is updated. These four parameters are included to aid the user, who may change them without affecting the RKF45T system.

The PHI component being updated will be treated as if it were passing through zero (with an artificially imposed sign if necessary.) If the component actually bounces on a zero (and the user does not correct the sign error, the analysis will continue with the incorrect sign. The sign correction will occur on a subsequent step due to the analysis in subroutine BOUNCD, and the user will be given the opportunity to repeat the step if the incorrect sign has caused computational difficulties. The "bouncing" analysis is described thoroughly in §9.9.

7. THE RKF45T SYSTEM COMMUNICATION WITH THE USER

While a general overview of the RKF45T package is needed to understand the basic workings of the program and to analyze any difficulties arising during the solution, another view of the software package is equally important, namely, the "black box" view. The communication between the user and the RKF45T system occurs through the calling sequence of both the RKF45T and the SUBPHI subroutines. Parameters are included which inform the user of the general progress of the integration or trapping, as well as of the occurrence of possible difficulties. Thus, both calling sequences must be carefully studied, so that the user can take full advantage of the information at hand.

7.1 User Input into RKF45T

The parameters for the RKF45T calling sequence are described in §4.2.1, and, along with the description of the modes of operation (§4.2.2), define the input required to solve the ODE. Briefly, the first two parameters, F and SUBPHI, define the functions to be analyzed during the solution of the ODE system, with the next two parameters, NPHI and NEQN, defining the dimensions of the system. The remaining parameters necessary for defining the initial value problem, Y and T (the initial values of the dependent variables and the independent variable, respectively), are listed next, with TOUT designating both the direction of the integration and the final value of T requested. The

relative and absolute precision estimate of the integration is requested through RELERR and ABSERR, respectively. IFLAG, which designates the mode of operation of the integrator, is described in §4.2.2. The WORK and IWORK arrays store parameters needed by the RKF45T system, the partitioning of WORK being given in Table 2.a and that of IWORK, in Table 2.b.

7.2 Information Returned through the RKF45T Calling Sequence

The RKF45T calling sequence returns the solution, Y, at the current value of the independent variable, T. In addition, YP, the derivative of Y, is also available at T, but is stored in WORK array locations, 1 through NEQN. Thus, as the solution is advanced, the initial values of T, Y, and YP are updated, so that T, Y, and YP always correspond to the current point, regardless of the condition of the integration. The parameter IFLAG informs the user of the success, temporary suspension, or failure of the integration and should be carefully monitored. Values of IFLAG, returned to the user, are listed in Table 1, along with a description of the difficulties encountered. (These values are also found in the program listing.)

The goal of the RKF45T integrator is to reach TOUT. Unless IFLAG indicates an error, this goal is achieved in the continuous integration mode, as well as in the continuous trapping modes. The step-by-step mode (both with and without the trapping option) and the trapping step-by-trapping step mode, however, may return the solution at any intermediate point even though the integration is proceeding normally. Thus, for the step-by-step mode IFLAG is set equal to 2 when TOUT is achieved. If the user wishes to continue the integration in the original mode, IFLAG must be set equal to -2, -20, or -25, respectively.

If the user sets RELERR too small (though still positive), RKFST resets the parameter to a minimum acceptable value and returns this value to the driving program with a warning IFLAG=3. The integration will proceed as long as the user does not change RELERR to a smaller number, and IFLAG will be reset automatically in RKFST. If the user fails to supply a convergence tolerance for PHI or if he supplies an unacceptable value, the integration tolerance will be used (i.e., tolerance=DMAX1{RELERR, ABSERR}).

Changes in IFLAG should be monitored to determine the current status of the integration. The user should *not* blindly accept the values of T, Y, and YP as being the conditions at TOUT (or as being an acceptable solution at T). Values of IFLAG returned by RKF45T are listed in Table 1, along with the value of T, (to which Y and YP correspond).

The parameters stored in the WORK and IWORK arrays are also changed during the solution of the problem. The location of these parameters is listed in Tables 2.a and 2.b. Of greatest importance to the user are the storage locations of

Parameter:		Location:
YP,	the derivative of Y at T	1, ..., NEQN
H,	the predicted step length for the subsequent step	NEQN + 1
TF,	the value of T at the end of the previous step (the trapping option may have halted the integration between T and TF)	10 NEQN + 2
YF,	the value of Y at TF	10 NEQN + 3 through 11 NEQN + 2
YPF,	the value of YP at TF	11 NEQN + 3 through 12 NEQN + 2

7.3 Communication through the SUBPHI Calling Sequence

The calling sequence for subroutine SUBPHI provides a great deal of information for the user's benefit, particularly during the update portion of the program. The user is required to provide the PHI and PHIP values. (If PHIP expressions are not known, estimates or zero values are acceptable, where a zero value will activate a false position estimate in the trapping routine.) The remaining parameters in the calling sequence have been included for the user's benefit and are not needed by the RKF45T system. The user is not required to respond to any parameter in the calling sequence, other than by supplying the PHI and PHIP values.

7.3.1 Initialization of the Trapping Option

If the trapping feature of the RKF45T package is being used, an initialization call is made to SUBPHI before the first integration step is made. This initialization call is identified by the parameter KOUNTR, KOUNTR=0. The user may wish to convey certain information to the RKF45T package during this initial call to SUBPHI. (Default values of parameters needed by RKF45T have already been set so that the system will attempt to solve the problem, but the nature of the PHI function may require special treatment for an effective solution.) Upon return to the RKFST system, KOUNTR is incremented by unity. Thus, the user may include an initialization block within SUBPHI, using KOUNTR=0 as the designating flag. (KOUNTR will remain greater than zero unless the integration is reinitialized, unless the bouncing analysis gives a warning through KOUNTR, or unless the user tampers with the value.)

7.3.1.1 User Requested Tolerances

The convergence of the trapping iteration may not need to be as precise as the integration accuracy. (In fact, for certain classes of PHI functions, integration accuracy may not be attainable.) Thus, the user may want to spe-

cify a relative error and/or absolute error tolerance for the PHI components. This separate error tolerance, ABSER, may be set during the initial call to SUBPHI with the default value (or with an unacceptable value) being set equal to the requested integration tolerance, DMAX1(RELERR,ABSERR). The parameters are then protected on further calls to SUBPHI so that changes by the user will not affect their values in the RKF45T system. If a relative error convergence is appropriate, the user should scale the PHI values by suitable factors.

7.3.1.2 Initialization of the Combination (Multiple and Single Trap) Option

The user may wish to activate the combination trapping mode (§4.2.2.5) which uses the multiple trapping mode for PHI(J), J ≤ M, and the single trapping mode for PHI(M+1), ..., PHI(NPHI). The combination trapping mode (which is a special form of the multiple trapping mode), is activated with the same IFLAG as the multiple trapping mode (IFLAG=15). During the initial call to SUBPHI (KOUNTR=0), the user simply resets INDEX, INDEX=M. Failure to identify the partitioning value M or failure to supply an acceptable value of M applies the multiple trapping option to the entire vector, i.e., the default value is NPHI. (Changes to INDEX affect RKF45T only at the initial call to SUBPHI. Otherwise, the parameter is protected in the program and is used merely to inform the user of the current value of PHI being analyzed in TRAPPD or being updated after a successful trap.)

7.3.2 Update Calls to SUBPHI

Once the trapping option has isolated a vanishing value of PHI, an update call is made to SUBPHI so that the user may make any adjustments desired. The logical parameter UPDATE=.TRUE., identifies the update mode while the parameter INDEX designates the component of PHI currently being updated. If several components of PHI have vanished at a given value of T, an update call will be made for each vanishing component. KOUNTR is incremented by unity after each update call and designates the number of vanishing points of PHI isolated during the integration (unless the user has tampered with the values). KOUNTR is provided solely for the user's information and may be adjusted freely without affecting the RKF45T package. If users wish to ignore the updating feature, they should simply activate an immediate return to TRAPPD whenever UPDATE=.TRUE.

TRAPPD assumes that PHI will change sign as it passes through zero. Therefore, if PHI(INDEX) has vanished within the specified tolerance but has not "stepped over" the zero point into the adjacent area, a sign change in PHI will be imposed artificially. If the integration step size is so small that the subsequent step has also failed to cross over the zero point, subroutine VANISH will detect the condition and the sign of PHI will reflect conditions at the begining of the step. (See Figure 5 and §9.5.) If updates are being made in the PHI function or in the differential equations, the user may wish to monitor IVAN so that the system is not updated twice in the same vanishing region. If the user changes the value of PHI at update, the user-supplied sign will be kept.

If a "bouncing" PHI component has disturbed the trapping analysis, an update call is made to SUBPHI indicating a corrected sign for PHI(INDEX). In this update case (and in no other case), BOUNCE=.TRUE.. The use may request that

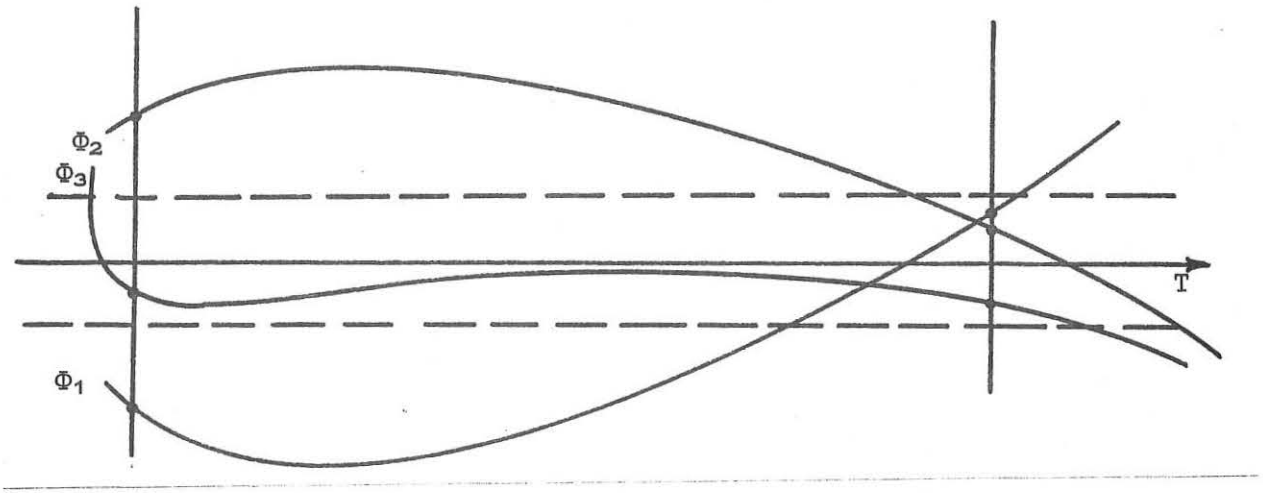


Figure 5. The sign of PHI(INDEX) at update. PHI(1) has crossed the boundary into the "new" region. PHI(2) has not yet crossed the boundary. Therefore an "artificial" sign change must be imposed. PHI(3) has vanished throughout the entire step.

the step be repeated by adjusting KOUNTR. Details of the "bouncing" analysis and update are given in §9.9.

8. RKFST AND AUXILIARY SUBROUTINES FOR THE TRAPPD SYSTEM

While subroutine TRAPPD performs the majority of the analysis concerning the PHI vector, several additional subroutines are referenced to treat particular situations. In addition, RKFST handles some of the PHI vector analysis before switching into the TRAPPD system. Subroutine SCALED evaluates the scaled solution at each point requested by TRAPPD, while VANISH studies the solution throughout the interval during the updating portion of TRAPPD. PANIC is an emergency routine referenced by the RKFST system if TRAPPD is unable to isolate a vanishing component of PHI.

8.1 Subroutine RKFST

The structure of RKFST is basically that of RKFS in the RKF45 package. Certain changes have been made to analyze the PHI vector before switching the analysis to TRAPPD, but these changes have been inserted in two major blocks so as to affect the RKFST structure as little as possible. Any additional changes involve recognizing flags for the trapping option and are minor in nature. If the trapping option is not activated, an internal flag, LFLAG=0) skirts the PHI vector analysis with only minor flag indicators being checked during the integration.

In the initialization block in RKFST, a section has been inserted to evaluate the PHI vector and to set the parameters needed by the trapping options. This section can be activated separately from the initialization process for the integration, making the "late start" possible without restarting the entire integration. (This basically avoids reevaluating the estimated starting value for the step size, which, in theory, is less accurate than the step size predicted after a successful integration step.)

The second large "block" inserted into the integration analysis is actually a subroutine call to SETRAP. All additional analysis of the PHI vector, occurring before TRAPPD is referenced, occurs in SETRAP. After the PHI function is evaluated at the end of the step (in SETRAP), the analysis will be shifted into the TRAPPD system if any component has changed sign over the step or has vanished at the end of the step. Otherwise, the analysis returns to RKFST to continue the integration. (The condition in which a PHI component vanishes at the beginning of the first integration step is handled in this block instead of in the initialization block since the analysis requires information within the first step rather than just at the beginning of the step. The listing for RKF45T and SETRAP is found in Appendix A.

8.2 Subroutines TRAPPD, SHIFTI, TSTAR, MULTOP, and BOUNCD

Subroutine TRAPPD organizes the search for the zeros of the PHI components. To streamline the analysis in TRAPPD, several auxiliary subroutines are referenced to perform such tasks as selecting the T* values, i.e., the vanishing point estimates, switching INDEX values, etc. TRAPPD itself has been described thoroughly in 6. The additional subroutines used in the analysis in TRAPPD include: SHIFTI, TSTAR, MULTOP, and BOUNCD. SHIFTI checks to see if TRAPPD should switch to a new PHI component in the zero search, i.e., SHIFTI checks to see if a new PHI component more strongly violates the boundary conditions than PHI(INDEX). (Index shifting is described in 6.4.) TSTAR uses the approach given in §6.3 to select the new T* value, the estimate of the vanishing point of PHI(INDEX). MULTOP is referenced if the multiple trapping (or combination trapping) option is in use to see if additional zeros lie between the just isolated zero and TF. BOUNCD is referenced at the beginning of each TRAPPD call to see if PHI actually "bounced" on the previously isolated zero point rather than passing through zero. (The problem of a bouncing PHI component is discussed in §9.9.)

8.3 Subroutine SCALED

Subroutine SCALED is referenced to perform two tasks, with ISCALE being the designating parameter. For ISCALE=1, SCALED generates the additional derivative evaluations needed to form the scaled solution. For ISCALE=2, SCALED evaluates the actual scaled solution. Since scaled solutions of orders four and five are available, the user must set the logical parameter, FIFTH=.FALSE. to form the fourth order solution or FIFTH=.TRUE. to form the fifth order solution. (See Table 3.) The procedure followed by SCALED is essentially the same for both the fourth and fifth order solutions, although the F2, ..., F10 vectors, which are, in part, storage parameters, may be labeled differently. SCALED is listed in Appendix A.

8.4 Subroutine VANISH

The convergence criteria imposed upon the vector PHI actually defines a region in which that component is said to vanish, i.e., there exists an interval over which the PHI component satisfies the convergence criteria. Thus, if a vanishing component of PHI is isolated on one step and that same component vanishes on the subsequent step, the user does not know whether or not the solution remained within the vanished region or whether the solution has passed out of the vanished region and "stepped on another zero". Such information may be important to the user, particularly if the PHI vector or ODE system is being updated. (See Figure 6.)

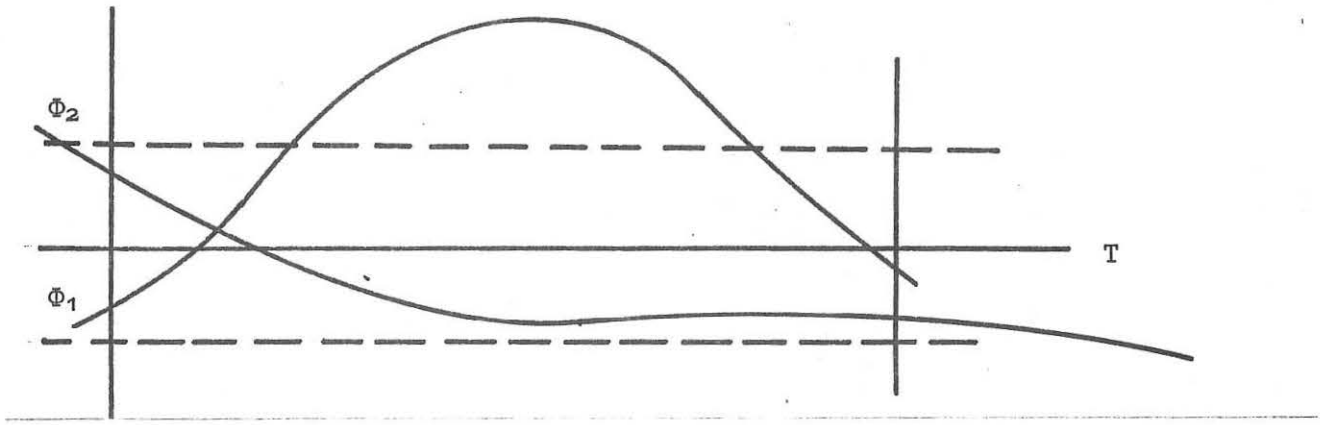


Figure 6. Two types of PHI components vanishing at the beginning and end of a given step. PHI(1) vanishes at both end points of the step but not throughout the entire step, while PHI(2) "vanishes" everywhere within the step. Two distinct zeros of PHI(1) have been located, while a "convergence neighborhood" of PHI(2) has been found. At update of PHI(2), IVAN=.TRUE. to indicate that the subroutine has "vanished" throughout the step.

The updating procedure in TRAPPD begins by referencing VANISH. If no component has vanished both initially and finally, the analysis is returned to TRAPPD to continue the update. Otherwise, each component which has vanished initially and finally is studied to see if it has also vanished throughout the entire trapping interval. If the multiple trapping option is being used, the initial value of T for the trapping bound is the most recent value isolated by TRAPPD. Otherwise, the initial value for T corresponds to the beginning of the integration step. The other trapping bound is the current "vanishing point" isolated by the TRAPPD routine, i.e., T2 if the trapping iteration has been used, or TF if the integrator "stepped on a zero" and no trapping was necessary. Three equally spaced subdivisions are used as the mesh to test for "total" vanishing, although this number may be increased by the user if desired. (See Table 3.)

Subroutine VANISH is also referenced in a special mode if a PHI component has vanished during the initialization of the integration problem. The sign of the PHI component needs to be set properly for trapping identification. The user, however, may not be able to set the initial PHI values correctly. Round-off from a zero value or an incorrect sign on zero (+0 or -0) could result in an improper interpretation of trapping conditions. Thus, VANISH is referenced by SETRAP, and the sign of the vanishing component at the initial condition is set equal to the sign of the component at the first nonvanishing substep generated in VANISH or at TF if the solution vanished throughout the entire step. An update call to SUBPHI is then made from RKFST with the corrected sign for component PHI(INDEX). The user may change PHI(INDEX) during this update call, and RKF45T will make no further changes to the updated component.

8.5 Subroutine PANIC

An emergency routine is provided to study the PHI vector throughout the integration step, including output of T, Y, YP, PHI, and PHIP. PANIC is referenced by the RKF45T system if subroutine TRAPPD fails to isolate a zero of the PHI function once conditions have indicated that a PHI component has changed signs over an integration step. (Exceeding the maximum limit on iterations and reducing the trapping bounds below an acceptable limit are the two errors which cause PANIC to be referenced by RKF45T.) The user may also activate the PANIC option, but, in general, the use of the subroutine is not recommended and is deliberately awkward to start. Subroutine PANIC is listed in Appendix A.

The user may activate PANIC in two ways. If a user wishes to monitor PHI whenever the analysis shifts into subroutine TRAPPD, the logical parameter, NOTFAL, which is normally set equal to .FALSE. in a data statement in TRAPPD must be reset to equal .TRUE.. With this change, PANIC will be referenced each time the analysis switches to subroutine TRAPPD, i.e., each time RKFST detects that a PHI component has changed sign over the interval or that a PHI component has vanished at the end of the integration step. The PANIC subroutine will print the values of T, Y, YP, PHI, and PHIP at ten equally spaced points throughout the integration step including at TF. (The density of output spacing may be adjusted by the user by changing the parameters POINTS and NPOINT (double precision and integer values). (See comment cards in the program listing of PANIC). The conditions at TF generated by the defining RK algorithm are then printed, since the scaled solution (either of fourth or fifth order accuracy) is not identical to the solution given by the defining formula and so may give slightly different PHI values.

The second means of activating PANIC analyzes the PHI function over each integration step, although the printing option may be suppressed. Such a mode of operation may be of importance if the user fears that the PHI function is oscillating frequently during the integration. "Frequently" is a relative term, which is actually defined in terms of the integration step size. (See example 5, §10.) This PANIC mode is activated in SETRAP through the use of the logical parameter, NOTFAL. (NOTFAL in SETRAP and TRAPPD are different parameters and are neither held in common nor passed between subroutines in the calling sequence. If PANIC is referenced through SETRAP, it will not be referenced in TRAPPD, regardless of the value of NOTFAL. (See §§9.2.2 and 9.3, and the PANIC listing in Appendix A.) If NOTFAL is reset to equal .TRUE., SETRAP will reference PANIC after each successful integration step. A

mesh refinement will be made in PANIC (normally set to ten subintervals), and the PHI vector will be studied at each point, in sequence, from T to TF. Any sign change over a subinterval will activate an immediate return to RKFST and TRAPPD with T and the bracketing subinterval value as limits for the trapping iteration.

A warning is in order. If the multiple trap option (§4.2.2.4) is used, further vanishing points will be sought between any isolated zero and TF. Thus, the thorough search of the PANIC option may be "washed over" by the TRAPPD routine searching for additional roots. Thus, the multiple trapping option will be reset to a single trap option if the emergency feature is used in RKFST. *To reiterate,* the panic options are *emergency* options and should be used only as such. A warning message will be printed on the initial call to PANIC to inform the user that an emergency procedure is being activated.

8.6 Subroutine FLAGCK

The parameter IFLAG designates the mode of operation of the RKF45T package. The number of available options in RKF45T has been increased from that of the RKF45 with IFLAG designating both the integration and the trapping mode. In RKFST, however, IFLAG needs to direct only the integration decisions with the trapping option decisions made in subroutine TRAPPD. FLAGCK is again referenced to reset IFLAG to a standard value for integration with a second flag, LFLAG, being set to identify the trapping option. (LFLAG=0 implies no trapping option, while LFLAG=-1, -2, 1, 2, identifies the step-by-step trapping mode, the trapping step-by-trapping step mode, the single trapping mode, and the multiple (or combination) trapping mode, respectively. FLAGCK is referenced before a return to the user. If a non-trapping mode has been used, no changes are made to IFLAG. If a trapping mode has been used, IFLAG will be reset to indicate that the trapping mode is in effect. If difficulties have been encountered, IFLAG will not be reset and will serve as a warning to the user.

8.7 Subroutine OUTFLG

The IFLAG parameter should be carefully monitored since any change from a standard value indicates that difficulties have been encountered during the integration or during the trapping iteration. Some values of IFLAG will cause the program to terminate if the user fails to respond to the designated problem. Other values serve as minor warnings to the user. Subroutine OUTFLG will print a warning message, identifying the possible cause of the difficulties. These messages are available in the program listing. OUTFLG is provided simply to help the user recognize possible difficulties quickly (and without having to monitor IFLAG in the driving routine). If IFLAG is an appropriate value for continuing, no message is printed. OUTFLG is activated by setting the logical parameter, FLGOUT=.TRUE. in subroutine RKF45T. The RKF45T package has been modified so that any terminal error gives a printed warning (with IFLAG value) before the program is stopped.

9. SPECIAL FEATURES, SPECIAL PROBLEMS

Although the analysis of the user-supplied PHI vector is, in theory, straightforward, the actual iteration process may encounter difficulties due to the nature of the PHI function. To demand that the user supply a "sufficiently smooth" PHI vector is a restriction that will often be ignored. To protect the iteration process from all "insufficiently smooth" PHI components, however, is impossible. Thus, a compromise must be reached.

An attempt has been made to provide the user with sufficient options to cover a wide range of applications of user-supplied stopping conditions, handling the additional analysis in an efficient manner while still including a good number of safety features. If the user suspects that certain difficulties may arise, he may include special emergency options in the analysis. Because these emergency features may reduce the efficiency of the RKF45T package, they should be used only when the added expense will be offset by the information gained. Because these features are to be used only in emergencies, they are deliberately difficult to activate.

Prospective users of the RKF45T package should note that even well behaved PHI components may be difficult to analyze. Examples of such problems are given in §10. The difficulties encountered in the analysis of such a function are generally due to the nature of the PHI vector (or due to its relation to the solution or the manner in which the solution is generated) and will be encountered regardless of the iterative procedure.

9.1 Controlling Constants

Many of the special features of the RKF45T package involve constants which must be set by the user. Some of the constants are machine dependent number or safety limits and need to be set only once for a given computing system. Others have "standard" supplied values, which may be reset by the informed and/or desperate user. Additional constants activate certain emergency options and must be reset from their standard values. All are given in data statements in their respective subroutines. Table 3 gives standard values of these constants and describes their purpose, with further clarification of some parameters available in the program listing.

9.2 Special Features in RKFST

Most refinements to the RKF45T package occur in subroutines which have been attached to the basic integration package. Two special features, however, are activated in SETRAP.

9.2.1 Vanishing PHI Components at the Initial Conditions

Special analysis of the initial conditions must be treated since the user may not know that a PHI component vanishes initially and since the sign of the vanishing component (+0 or -0) may not be properly set for continuing into the next region. These "initial conditions" refer to the PHI vector at the beginning of the integration (or at the first step using the trapping mode if the "late start" option is being used) and not to the conditions at the beginning of each step. If any component of PHI has vanished initially, VANISH is referenced in a special mode, and the sign of the zero is given that of the value of the PHI component at the first substep. An update call is then made to SUBPHI for each vanishing component at which time the user may alter the sign. Any user-altered sign will be used to identify the next sign change.

9.2.2 The PANIC Option Activated through SETRAP

The PANIC option, described in Section 8.4, should be used only as an emergency measure. This option refines the step size mesh, giving dense output of the PHI vector within the step. The first subinterval, (T_{i-1}, T_i), indicating a sign change in (or the vanishing of) a component of PHI, will activate a return to SETRAP and will shift the analysis to TRAPPD with T_{i-1} and T_i as the bracketing values for the trapping interval. This PANIC scheme should be activated only if the user fears the PHI function experiences multiple zeros within single integration steps. (Such zeros might go undetected altogether or at least part of the zeros might be overlooked.) (See example 5, §10.) To activate the option, the user must reset NOTFAL in the data statement SETRAP. A print-no print option is also available for this mode being activated in subroutine PANIC. JPRINT=0 suppresses the printing altogether, while JPRINT=1 prints T and the PHI values throughout the step. Regardless of the printing option selected, a warning message will be printed at the first call to PANIC from SETRAP. The mesh size for the subintervals is regulated by the parameters POINTS and NPOINT given in a data statement in PANIC. (Standard values are 10.0D0 and 10, respectively.)

9.3 The PANIC Option in TRAPPD

Users may activate the PANIC option upon entry into TRAPPD, by setting NOTFAL=.TRUE.. (The standard value, NOTFAL=.FALSE., is set in a data statement in TRAPPD. See Table 3.) This emergency feature will print values of the T, Y, YP, PHI, and PHIP at a specified number of points throughout the integration step. The analysis will then be continued in the normal fashion in TRAPPD. Printing options in PANIC should also be set. The standard printing option, IPRINT=1, prints T and PHI throughout the interval. Setting IPRINT=2 gives additional information, Y, YP, and PHIP throughout the step. The number of output points may be changed from the standard value (10) by changing POINTS and NPOINT in the data statement in PANIC.

9.4 Order of the Scaled Solution

Two scaled solutions are available in the SCALED subroutine, one of 4th and one of 5th order accuracy. In general, the 4th order solution is sufficiently accurate and requires only one additional derivative evaluation for a given step. The fifth order solution, requiring 5 additional derivative evaluations, is also included. The logical parameter, FIFTH designates the scaled solution to be generated. FIFTH=.FALSE. causes the 4th order scaled solution to be evaluated with FIFTH=.TRUE. giving the 5th order scaled solution.

9.5 PHI Vanishes throughout a Step

Certain difficulties may arise in the analysis of the PHI function if a component vanishes at the beginning of a step as well as at the end of the step (or at a trapped point). The component may have vanished on the previous step (giving the vanishing value at the beginning of the step), passed out of the "vanishing region" during the step, and again vanished at TF (or T*). This component may also have remained within the "vanished" region throughout the step. If the integration is being advanced using large steps, the former situation is quite possible, while if the integration is encountering difficulties, the naturally restricted step size could be so small that the PHI function has still remained within the vanishing region. A PHI component which vanishes throughout the entire step may also represent a function which has a very large radius of curvature, and consequently a very large convergence region. (See example 6, §10.)

Subroutine VANISH is referenced to study components which have vanished at T and T*. (See §8.3.) The component is studied at NPOINT evenly spaced points through out the substep, (T, T*), and the user is informed if the component vanishes at each of the inserted values (IVAN=.TRUE.) during the update call to SUBPHI.

9.6 Print Options

Various print options are available throughout the RKF45T system. Most of these options involve the emergency features of the program or warnings when difficulties have arisen. Subroutine OUTFLG has been included to print out values of IFLAG and clarifications when IFLAG indicates difficulties have arisen during the solution of the ODE or during the trapping iteration. No messages are printed if the IFLAG indicates that the solution is proceeding normally. OUTFLG is referenced by subroutine RKF45T only if FLGOUT is set equal to .TRUE..

Printing options are also available in TRAPPD with IOPT=0 suppressing the print statements. IOPT=1 will give conditions upon entry into TRAPPD and upon update of each vanishing component.

Subroutine PANIC also includes print statements which are described in §§8.4, 9.2.2, and 9.3.

9.7 User Update of the PHI Components and the Differential Equation System

The user is given the opportunity to update the PHI vector or the differential equation system whenever a vanishing value of a PHI component is isolated. An update call is made for *each* component of PHI to have vanished (i.e., to have satisfied the tolerance criteria) at the point T*. The parameter INDEX designates the value of PHI currently being updated. Because this updating feature can provide such computational efficiency and flexibility in programming, the user should understand the updating portion thoroughly.

9.7.1 Sign of the Vanishing Component

Since a zero is seldom trapped exactly, some confusion about the sign of the component may arise even though the magnitude is properly bounded, i.e., the value of PHI(INDEX) may or may not have passed through the zero yet. *TRAPPD assumes that the PHI component will change sign as it passes through zero and makes an artificial sign change if PHI has not yet changed signs.* Thus, when PHI enters the update portion of SUBPHI, the trapped component reflects the sign of PHI in the region (T*, TF). (If the component vanished throughout the entire region (T, T*), the sign is held constant, since the component has remained in a "vanished" region.) If the user chooses to alter the PHI component during the update, TRAPPD assumes that the user has given the altered components the proper sign and makes *no* further adjustments to the PHI vector. The user may actually change any component of PHI during the update, but this might affect the analysis of other PHI components. (See §9.7.2.) Thus, the user may overwrite the artificial sign change, if the particular PHI component is not correctly described and may make any additional changes desired.

9.7.2 Changes in the PHI Vector

During an update call, only one PHI component is identifed as having vanished, namely PHI(INDEX). If additional components have vanished, these will be updated in separate calls to SUBPHI. In some applications, the user may only wish to print information which would require no changes in the PHI components. Other applications may actually require changes to the PHI vector. During an update call for PHI(INDEX), the user may change any component of PHI, but in doing so, may destroy information gained during the trapping process. More specifically, the vector PHI represents the PHI vector at T*, the value of T for which at least one component of PHI vanishes. The updating process analyzes |PHI(J)| as J increases, J=1,...,NPHI. Each vanishing component, INDEX, activates an update call to SUBPHI at which time the user may change any component of PHI. If changes are made to PHI(J), J > INDEX, however, the information gained during the trapping iteration is altered before the user is informed of the status of that component at T*. Thus, such changes should be made only if the user fully understands the PHI function at T*. In general, the user should only update the component PHI(INDEX). Any changes to a PHI component *must* carry the correct sign (including +0 or -0). Otherwise, further trapping regions will not be properly identified.

9.7.3 Updates in the Differential Equation System

During an update call, the user may actually wish to change the ODE system itself. For example, in a transfer orbit problem, one would want to increment the velocity at perigee of a particular revolution. (See example 2, §10.) Such a change involves altering the state vector and, consequently, the differential equations. Such changes are permissible in the single trap option (or in the single trap partition of PHI in the combination option) since the solution is advanced from T*, the trapping point. Of course, such changes are also possible in the step-by-step or trapping step-by-trapping step modes since any trapped solution activates a return to the user with T=T*. The multiple trap option, however, is *not* an acceptable mode for an ODE update since the solution is advanced from the end of the step, where conditions in TRAPPD are still described by the original state vectors and corresponding ODE values.

If the user changes the state vector or the differential equations, he must reference F so that the proper derivative values are assigned at T*, and he must reevaluate the PHI vector to reflect the altered state variables.

9.8 Difficulties with the PHI Function

The analysis in TRAPPD and associated subroutines assumes that the PHI function is "well behaved" and reasonably well understood. Basically, TRAPPD assumes that a PHI component does not "bounce" on a zero (i.e., TRAPPD assumes that a PHI component changes sign as it passes through zero) and that any component vanishes only once during a given integration step. In addition, rather than isolating the actual zero of a PHI component, TRAPPD isolates a point, T*, within a convergence region (TL, TR) (such that |PHI(INDEX)| < TOLERANCE at T*.) If the radius of curvature of the PHI component is large, T* may be some distance from the actual vanishing point of PHI(INDEX), and, in fact, an accurate estimate of the vanishing point may be impossible to achieve. Further difficulties may be encountered in the trapping iteration itself if discontinuities of PHI or large values of |PHIP| occur. Such difficulties will plague the user regardless of whether he uses his own iterative scheme or a supplied routine.

Additional options are included in the TRAPPD system to help the user analyze the PHI function if difficulties occur. The PANIC mode permits the user to study the PHI function over a refined mesh at each integration step, helping to isolate multiple zeros over a given step. (This refined mesh gives a much denser output of PHI that may help trap "bouncing" components since these are isolated only if the zero occurs near a mesh point. A far safer way to locate such a vanishing point, however, is for the user to include the derivative of that PHI component as an additional PHI component, i.e., to seek the vanishing point of PHIP. (See example 7, §10.) Discontinuities or steep derivatives which predict iteration steps sizes less than the permitted value will activate the PANIC routine to print out the PHI vector throughout the step if the iteration fails to converge. Thus, the user has some help in locating difficulties. The large convergence region, however, may pose a more difficult problem. The user may gain some advantage by scaling the PHI function using some sufficiently large value. If the zero location is strongly affected by integration errors, there may be nothing that the user can do to locate the zero precisely. During the update portion of TRAPPD, the user is informed if

a component has vanished over an entire integration step. In addition, the PANIC option can also provide a good deal of information about the magnitude of PHI throughout the step, so that the user can monitor the convergence region to some extent. An additional PANIC call could be inserted during the update portion of TRAPPD to study the convergence region, but this difficulty is not anticipated for the current applications and so this option is not included in RKF45T, although the user could make such a call from SUBPHI.

9.9 The Bouncing PHI Function

A PHI component which bounces, i.e., a component which fails to change sign as it passes through zero, poses particular difficulties for the RKF45T package, since the zero detection technique requires a sign change in the PHI component or an integration step coinciding with the zero point. Thus, in general, such a zero will go undetected. The only case for which a bouncing PHI component will be detected and updated is that in which the zero is stepped upon. The simplest remedy for detecting the zeros of a bouncing PHI component is to include the derivative of PHI as an additional PHI component. Then the zero of the derivative should be detected and so the zero of the desired bouncing PHI function.

If a bouncing zero is detected further difficulites in the RKF45T analysis arise. The RKF45T pacakge assumes that the PHI component will change sign as it passes through zero and will "adjust" the sign artificially to reflect the conditions it assumes will exist across the boundary. (See §7.3.2.) In the case of a bouncing PHI function this sign adjustment is incorrect. The user may override this sign change (or change the value of PHI altogether) and the user-changed value will be used. In case the user does not recognize a zero as a "bouncing zero", however, further safety checks are included in the RKF45T package. (While the RKF45T program is not designed to "trap" bouncing zeros, the isolation of such a zero could confuse things greatly if the special analysis were not included.)

The idea of a bouncing PHI function is not as far-fetched as one might think. An example is presented in [4] in which an inequality constraint is handled by analyzing an additional ODE which imposes a strong penalty when the constraint is violated and which has no contribution when the constraint is not violated. The difficulty in analyzing such a function comes from the fact that the optimal solution estimates generally stay near such boundaries during a portion of the integration path. The user does not know in which direction the ODE solution will diverge from the constraint (since in searching for the optimal solution, the constraint equations may be violated). Thus, an important application exists in which such a boundary analysis needed.

The PHI components are checked at each entry into TRAPPD to see if a component "bounced" (and consequently was given the incorrect sign) on the previous step. The requirements for the detection of a bouncing function are that: $|PHI(I)| <$ tolerance at T and that $|PHI(I)| >$ tolerance at TF. If a function bounces and yet remains in the "vanished" region throughout the integration step, the detection will be delayed until the solution again passes out of the "vanished" region at TF. Any PHI component which has been identified as a "bouncing" component over a given step will not be analyzed by the TRAPPD subroutine over that step. Instead, the user will be informed of the bouncing nature of the component and will be given the opportunity to have the step

repeated with whatever updated conditions he wishes to insert in the ODE or PHI systems.

The bouncing component update is identified in SUBPHI by BOUNCE=.TRUE., UPDATE=.TRUE.. The parameter, KOUNTR, will be given the dummy value, KOUNTR=-1. If changes in the ODE or PHI expressions are needed, the user should make these changes and reference F if the derivatives need to be reevaluated. *If the step is to be repeated, KOUNTR must be reset to KOUNTR=-2.* (KOUNTR will be reset to its previous value upon return to TRAPPD.) If the user has demanded that the step be repeated, the analysis will be returned to RKFST, and the step repeated with the user-supplied, corrected version of the ODE or PHI system.

10. APPLICATIONS

The RKF45T package is applied to a variety of problems to illustrate the use of its various options. Several of these examples represent simple integration problems which can be handled by the Runge-Kutta method exactly (neglecting round-off error) so that the user may see the affects of the trapping option without having the results contaminated by integration errors. Other examples are more representative of practical applications. The basic equations are given for each example as well as the essential components of the corresponding SUBPHI routine and a summary of the results. Appendix B lists the actual SUBPHI program used for each example along with the output from the problem. The types of problems studied are:

1. the two body problem (elliptic orbit) requesting dense output at specified values of the independent variable,

2. the two body problem (elliptic orbit) transfer to a higher orbit after a specified number of revolutions,

3. the restricted problem of three bodies with multiple stopping conditions,

4. an ODE expression involving tabular data, with stops at the table grid values,

5. a highly oscillatory PHI function,

6. a function having a large convergence region, and

7. a "Bouncing" PHI function.

10.1 Dense Output at Specified Values of the Independent Variable

Users often monitor the solution of an ODE system by demanding dense output at specified values of the independent variable. Such demands may severely restrict the natural selection of the step size, i.e., the RK method is able to take much larger steps than the user is permitting. By handling the dense

output restriction in a SUBPHI subroutine, however, the integrator is able to chose its own step size, and the user receives the information requested. Consider the two-body problem, elliptic orbit. The equations of motion are $y'' + \mu\, y/r^3 = 0$, where y is a vector and where $r = (y_1^2 + y_2^2 + y_3^2)^{1/2}$ with μ, a constant. A normalized version of the problem (in two dimensions) with $\mu=1$, having a semi-major axis of unit length and period of 2π, has the following initial conditions, expressed in terms of the eccentricity, e, $0 \le e < 1$: $y_1(0)=1-e$, $y_2'(0)=0$, $y_1'(0)=0$, and $y_2'(0)= \{(1+e)/(1-e)\}^{1/2}$, the conditions being given at perigee.

Consider the normalized two body problem, eccentricity e=0.6, in which the user wishes dense output at specified values of the independent variable. If the time increment, TINCR, is constant, the problem can be handled using the following SUBPHI program:

```
Subroutine Name:   DENSE1
Option:            Multiple trap (IFLAG=15,25)
NPHI:              1
COMMON block:      TIME1/TINCR/
Initialization:    Set TPR = TINCR

PHI components     PHI(1)  = T - TPR
                   PHIP(1) = 1.0D0

Update:            Print information;
                   increment TPR, TPR = TPR + TINCR;
                   reevaluate PHI(1)
```

A similar problem involving uneven spacing, requires a vector of output values (say, TPRINT) to be held in common with the driving routine or to be stored in SUBPHI. Otherwise, the procedure is the same.

```
Subroutine Name:   DENSE2
Option:            Multiple trap (IFLAG=15,25)
NPHI:              1
COMMON block:      TIME1/TPRINT(50)
Initialization:    Set TPR = TPRINT(1)

PHI components     PHI(1)  = T - TPR
                   PHIP(1) = 1.0D0

Update:            Print information;
                   increment TPR, TPR = TPRINT(KOUNTR+1);
                   update PHI, PHI(1)=T-TPR
```

In both problems the initialization block is identified by the parameter KOUNTR=0. ABSER is not supplied (default value being integration precision) since the Newton-Rhapson iteration converges to TPR in one iteration. In DENSE2 the counting parameter, KOUNTR, is used to increment the TPR value, although a separate counter could have been set in the initialization block. At each update call the time print-out value is shifted forward. The user must also evaluate the updated PHI components. Listings of DENSE1 and DENSE2 are given in Appendix B along with a representative portion of the output.

Treating the problem of dense output at specified values of T by reducing the step size to correspond to the output point, i.e., by setting IFLAG=1, TOUT=DT, and then incrementing TOUT upon return, TOUT=TOUT+DT, until the final time is reached, proves to be a far less efficient means of solving the

problem, with the efficiency dropping markedly as the number of points increases. (See below.)

The TRAPPD analysis is written assuming that PHI depends upon Y'. Thus, once the scaled solution is evaluated, a derivative call is also made before SUBPHI is referenced to determine the PHI components. In DENSE1 and DENSE2, however, the PHI vector is independent of Y' so that additional savings in computing time could be made (one function call per output point). However, because most practical PHI and PHIP functions include Y and Y', and because some options included in TRAPPD require Y', the "wasted" evaluations for the given example are considered an acceptable price. If the user is trapping PHI functions *all* of which are solely dependent upon time, and if these functions involve a large number of stops, he should consider duplicating (and renaming) subroutine TRAPPD and deactivating the F call after the SCALED call in the modified routine. All other subroutines would remain the same. The TRAPPD package itself should be kept as is, since this is the more general program.

	Number of Output Points					
	20	50	100	200	500	1000
NFE without trapping	247	307	601	1195	2977	5947
NFE with trapping	207	252	302	403	703	1203
NEXTRA	38	77	127	228	528	1028

(NEXTRA counts the evaluations required by the trapping iteration and *is included* in the NFE with trapping count.)

10.2 Update in the ODE System Using the Combination Mode

The user may wish to change the differential equation system at certain points during the solution. These points, themselves may not be clearly defined, i.e., one may need an iterative procedure to locate them. Consider the two body problem (example 1) in which the user wants to increment the velocity at perigee during the second revolution to send the satellite into a higher orbit.

During the solution, other information may also be requested which requires no changes to the ODE during the update procedure, such as in example 1. By properly structuring the PHI vector, the user may apply the multiple trap mode to part of the PHI vector and the single trap mode to the remainder of the vector. This combination mode is a special form of the multiple trap mode which iterates using the multiple trap option until a T* value corresponding to a single trap component of PHI, is isolated. The integration is then continued from T*.

In the combination mode, the user must structure the PHI vector so that the first MPART components are those to be trapped using the multiple trap mode. The remaining NPHI-MPART components will be trapped in the single trap fash-

ion. Since the combination mode is a special form of the multiple trap mode, the designating IFLAG is also 15 with the normal returned value being 25. The value of MPART is conveyed to RKF45T during the initialization block in SUBPHI (designating flag, KOUNTR=0) by setting INDEX=MPART. (Failure to set INDEX=MPART will apply the multiple trap option to all components of PHI, i.e., the default value is NPHI.)

The combination option is applied to the two body problem, an earth-satellite problem, with initial conditions given at an arbitrary point along the orbit path. Upon the second pass through perigee, the satellite will be boosted into a higher orbit by incrementing the velocity. The RKF45T package will continue to integrate the new system of equations until the final specified time. For this problem, μ=398601.3D0 km^3/s^2, with initial conditions: Y(1)=-.7196D+4 km, Y(2)=-.1546D+4km, Y(3)=-.9840D+3 km, Y(4)=-.4201D+1 km/s, Y(5)=-.8359D+1 km/s, and Y(6)=-.2074D+1 km/s.

The velocity increments to boost the satellite into a higher orbit require a change in the ODE system. Therefore, the single trap mode is appropriate for the corresponding PHI component. Other stopping conditions may involve no updates to the state vector or differential equations. These could be better handled in the multiple trap mode. Thus, the combination mode is appropriate. In addition to perigee and apogee conditions, two additional stopping conditions are requested for print out: specified values of T (even spacing) and vanishing values of R". The PHI representations for the T and R" stops are clear. Both the perigee and apogee stops, however, are defined by V•R=0. The R" value may be used to distinguish between the positions, with R" < 0 at apogee, R" > 0, at perigee.

For the combination mode, the PHI vector is structured so that the T and R" stops are listed first with the apogee-perigee stops being the final component in the array. IFLAG=15 with INDEX set equal to 2 during the initialization block in SUBPHI. These flags activate the combination mode, with PHI(1) and PHI(2) trapped in the multiple trap manner and PHI(3), in the single trap manner. During the update of PHI(3), R" is used to identify perigee. A counter, IPER, is set equal to zero initially and incremented each time perigee is reached. When IPER=2, the transfer orbit conditions are activated.

Subroutine Name:	TRANSF
Option:	Combination mode (IFLAG=15,25, MPART=INDEX=2)
NPHI:	3
COMMON block:	TIME/TINCR
Initialization:	To activate the combination mode set INDEX=2 (MPART is set equal to INDEX in RKF45T); Set counter, IPER=0 for counting passes through perigee; Set TPR=TINCR
PHI components	PHI(1) = T - TPR PHI(2) = R" PHI(3) = V • R PHIP(1) = 1.0D0 PHIP(2) = D(R")/DT (secant approximation) PHIP(3) = V'• R + V • V
Update:	Print information; if INDEX=1, time print out update: increment TPR, TPR = TPR + TINCR; if INDEX=2, no additional changes;

if INDEX=3, and position is apogee, RETURN;
if INDEX=3, and position is perigee, increment perigee counter, IPER=IPER+1. If IPER=2, increment velocity, Y', reevalutate F(T,Y,Y') with new conditions, reevaluate PHI(3), RETURN.

Details of the independent variable stop are given in example 1. The vanishing R" points involve the second derivative of a scalar function,

$$R'' = \{Y_1Y_1'' + Y_2Y_2'' + Y_3Y_3'' + Y_1'^2 + Y_2'^2\ Y_3'^2\}/R - R'^2/R$$

where $R = \{Y_1^2 + Y_2^2 + Y_3^2\}^{1/2}$ and where $R' = \{Y_1Y_1' + Y_2\ Y_2' + Y_3Y_3'\}/R$. The R" values are of great importance as the perigee-apogee indicator.

During the initialization block in SUBPHI, the perigee counter and the first T print out value are set. ABSERR, the error tolerance, is ignored which imposes the integration tolerances. During the perigee update for the transfer orbit, the state vector is changed (increased by 5%) necessitating the evaluation of both the PHI(3) and PHIP(3) values. The SUBPHI program, TRANSF, and results are given in Appendix B. Figure 7 illustrates the stopping conditions (schematically) and the transfer orbit.

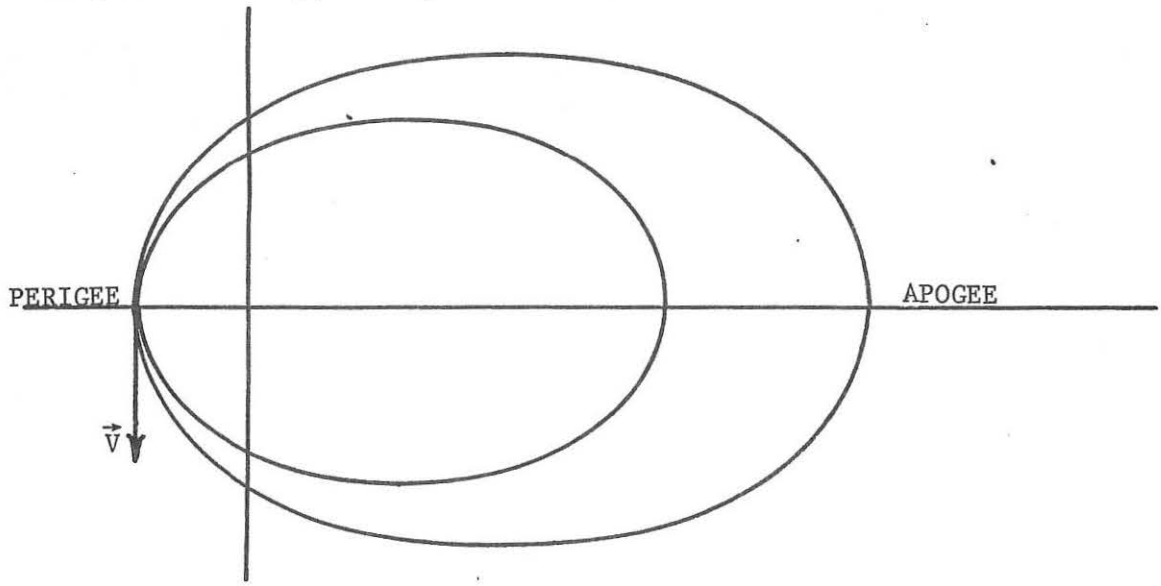


Figure 7. Vanishing PHI(3) values for transfer orbit problem. Perigee and apogee are located.

10.3 Large system of Stopping Conditions

The user may wish to employ a large system of stopping conditions involving printing, PHI updates, and/or ODE updates. Thus, single, multiple, or combination options could be applicable depending upon the problem. The restricted problem of three bodies (a body of negligible mass, orbiting around two "heavy" bodies) is a good test problem, since at certain points several functions of practical interest vanish, e.g., as the radius reaches a minimum, a

velocity component vanishes, and V•R = 0. Such a problem provides a good test of RKF45T's ability to switch between components during a given step.

The equations of motion for the restricted three body problem are: $Y_1'=Y_3$, $Y_2'=Y_4$, along with

$$Y_3'=2Y_2'+Y_1-\mu^*(Y_1+\mu)/R1^3 \ -\mu(Y_1-\mu^*)/R2^3, \text{ and}$$

$$Y_4'=-2Y_1'+Y_2-\mu^*Y_2/R1^3 \ - \mu Y_2/R2^3,$$

with initial conditions:

$$Y_1(0)=1.2,\ Y_2(0)=0,\ Y_3(0)=0, \text{ and } Y_4(0)=-1.04935750983031990726.$$

The constant $\mu=1/82.45$, with $\mu^*=1-\mu$, $R1^2=\{(Y_1+\mu)^2 + Y_2^2\}$ and $R2^2=\{(Y_1-\mu^*)^2 + Y_2^2\}$.

The satellite will recover initial conditions at T=6.19216933131963970674. A necessary (but not sufficient) condition for the success of the ODE solution is that the Jacobi integral, $J = \{Y_1'^2+Y_2'^2-Y_1^2-Y_2^2\}/2 - \mu^*/R1 - \mu/R2$, remain constant during the integration.

The restricted three body problem is solved with eight stopping conditions imposed. The first five components have physical significance, being vanishing values of the state vector or points where the velocity and position vector are perpendicular. The expressions for the corresponding PHIP components are clear. The remaining stopping conditions are more artificial in nature, being non-zero specified values of the dependent variables: $Y_1=0.5$, $Y_2=-0.6$, and $V_1=1.0$). The Jacobi integral could be included as a PHI function, but the user really only wants to know *if* the value explodes and does not need to know exactly *where*. Thus, monitoring the value at each integration step is really a sufficient analysis. The structure of SUBPHI is simple since only print out is requested. An additional section is added, however, to monitor the Jacobi integral during the integration (flags: INDEX=0, UPDATE=.FALSE.)

Subroutine Name:	RP3B1, RP3B2
Option:	Multiple trap (IFLAG=15,25)
NPHI:	8
COMMON block:	None
Initialization:	Set parameters for monitoring the Jacobi integral

PHI components

```
PHI(1) = V • R
PHI(2) = Y(1)
PHI(3) = Y(2)
PHI(4) = Y(3)
PHI(5) = Y(4)
PHI(6) = Y(1) -0.5D0
PHI(7) = Y(2) +0.6D0
PHI(8) = Y(3) - 1.0D0

PHIP(1) = V'• R + V • V
PHIP(I+1) = YP(I), I = 1,2,3,4
PHIP(I+5) = YP(I), I = 1,2,3
```

Update: Print information;
no changes are made to the ODE or state vector

The Jacobi integral is monitored in both RP3B1 and RP3B2 but is not listed as a PHI component.

Two SUBPHI programs, RP3B1 and RP3B2, are presented having identical stops but different monitoring sections for the Jacobi integral. Output, along with listings is presented in Appendix B. Figure 8 illustrates the stopping conditions for the given PHI components. Several comments about the presented results are needed. If the exact solution could be generated, several components of PHI would vanish simultaneously at specific values of T along the particle path. The initial condition stops (1,2,3) for example occur at T=0. Stops (7,8,9), (14,15,16), and (20,21,22) should also occur simultaneously. In addition, there should be three stops at TF. These stopping conditions are: (1) the velocity in the Y_1 or Y_2 direction vanishes, (2) the velocity and the radius are perpendicular, and (3) either Y_1 or Y_2 vanishes (depending upon the particular T value). (In the print out in Appendix B, Y_1 is designated X and Y_2,Y, with corresponding velocities being V and V .) Stops (1,2,3), occurring at the initial conditions, are not contaminated by integration errors, and so the updates are simultaneous. As the integration proceeds, however, errors in position and velocity perturb the zero points of the PHI components from their true values. Thus, with strict tolerances imposed, the three stopping conditions are satisfied at three different values of T. Stops (7,8,9) and (20,21,22), at Y_1=0 (X=0), occur during the most "difficult" part of the solution, i.e., the magnitude of the position vector reaches a minimum at this point causing a "near singularity" in the differential equations and, consequently, severely reduced integration step size to maintain the requested accuracy. In this region, the "simultaneous" vanishing points occur within a time space of 0.0026, whereas the stop (14,15,16) (at maximum position vector) occurs within a time space of 0.00016. (The requested integration and trapping tolerances are 1.D-06 with the global error achieved at TF being less than 1.8D-04.) The remaining multiple stopping point illustrates an important difficulty. A PHI function which vanishes at the final conditions may not be detected because errors in the solution have shifted the zero slightly beyond the final time. (These zeros, which would be isolated on subsequent steps, have not yet been detected at TF.) Thus, only the stop, Y =0, which precedes its true zero, is actually detected. The user should always be aware of vanishing difficulties at the final time.

10.4 Tabular Data Expressions in ODE Systems

In engineering applications, the differential equations often include parameters which are determined from tabular data. The drag coefficient, for example, may be expressed as a function of Mach number, altitude and lift coefficient. If the tables are treated using constant values or bilinear interpolation, discontinuities in the function or its derivatives occur at the grid mesh defining the table. Even the slight discontinuities in slopes in the tables may be amplified further by the expressions involved in the ODE, seriously affecting the integration accuracy.

In theory, the user should stop at each discontinuity, since piecewise continuity is needed for a valid RK algorithm. In practice, however, a good software package may be able to isolate the discontinuities to some degree, giving a reasonable estimate of the solution. The "natural trapping" ability of the given integrator involves rejecting steps near discontinuities until the step size is small enough that the abrupt changes no longer violate the

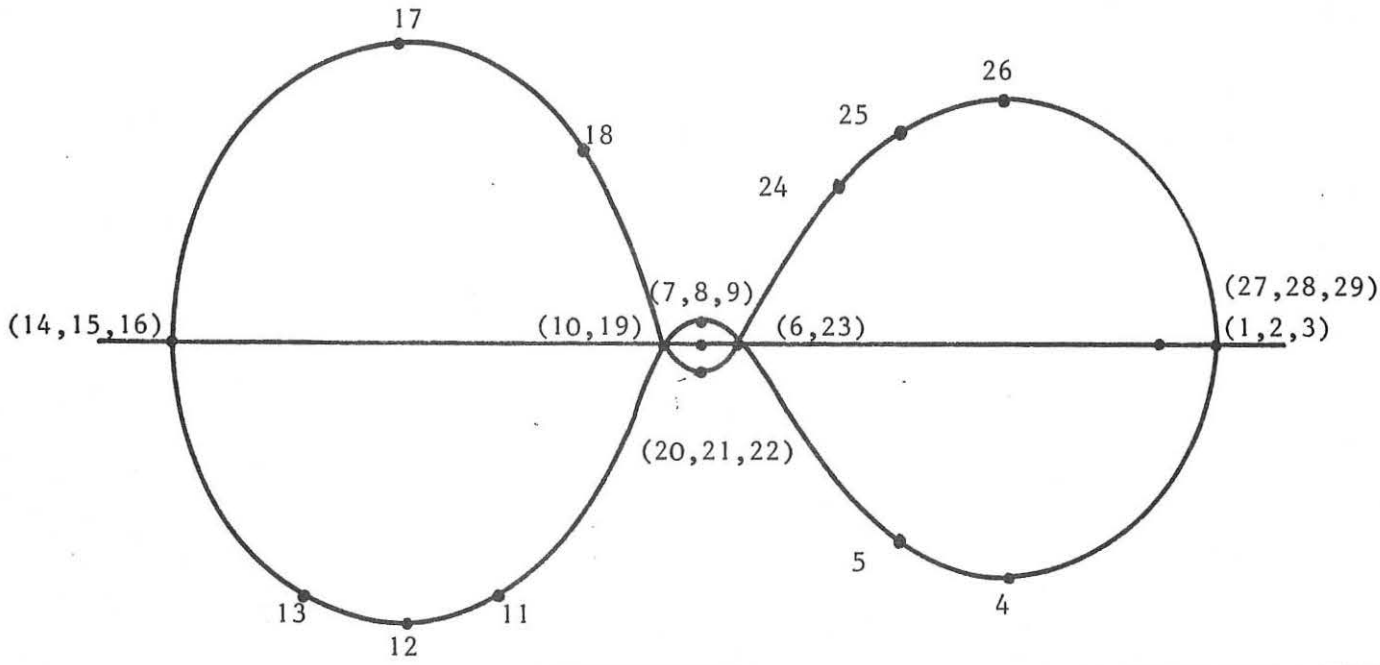


Figure 8. Orbit for the restricted problem of three bodies with stopping points marked.

specified tolerance bounds. Although this "natural trapping" feature of a program requires no analysis by the user, the convergence regions are vaguely defined and execution is inefficient. Thus, the user should have some understanding of the effect of the interpolation scheme on the integration.

Consider an example involving the integration of a two dimensional step function, similar to the integration of a set of building blocks. Setting YP(1)=AA1*T and YP(2)=AA1 with $Y_1(0) = Y_2(0) = 0.50D0$, (where AA1 is the value of the step function at the particular (Y_1, Y_2) point (See Figure 9.)), gives a solution that is a set of connected parabolic or straight line segments. The simplicity of the problem allows one to analyze the effects of the discontinuities on the "trapped" and the "untrapped" problem, i.e., the integration is exact (except for round-off) and the grid crossings can be determined analytically to check the accuracy of the trapping procedure.

The application of the trapping option requires that the particular plane segment covering the solution point at T0 represent the F function over the entire domain until a grid value is isolated. Then the F function is updated to reflect the new region of the table, i.e., as the PHI components stop on the grid entries, the plane segments are changed. The PHI components are written as parabolas with with zeros at the current bracketing bounds. Thus, PHI(1)=(X1-Y(1))*(Y(1)-X0) and PHI2=(Y1-Y(2))*(Y(2)-Y0), where X0, X1, Y0, Y1 are the grid values which bracket Y(1) and Y(2), respectively. Any point within the current bounding values gives PHI(I) > 0, while all points outside of the current bounds give PHI(I) < 0. The iteration converges to the X1 or Y1 bound if the corresponding PHIP < 0 and to the X0 or Y0 bound if the PHIP > 0. The F value is held constant until a boundary is trapped. The bounds (X0, X1) or (Y0, Y1) are shifted and the plane representing F is reset and held at the

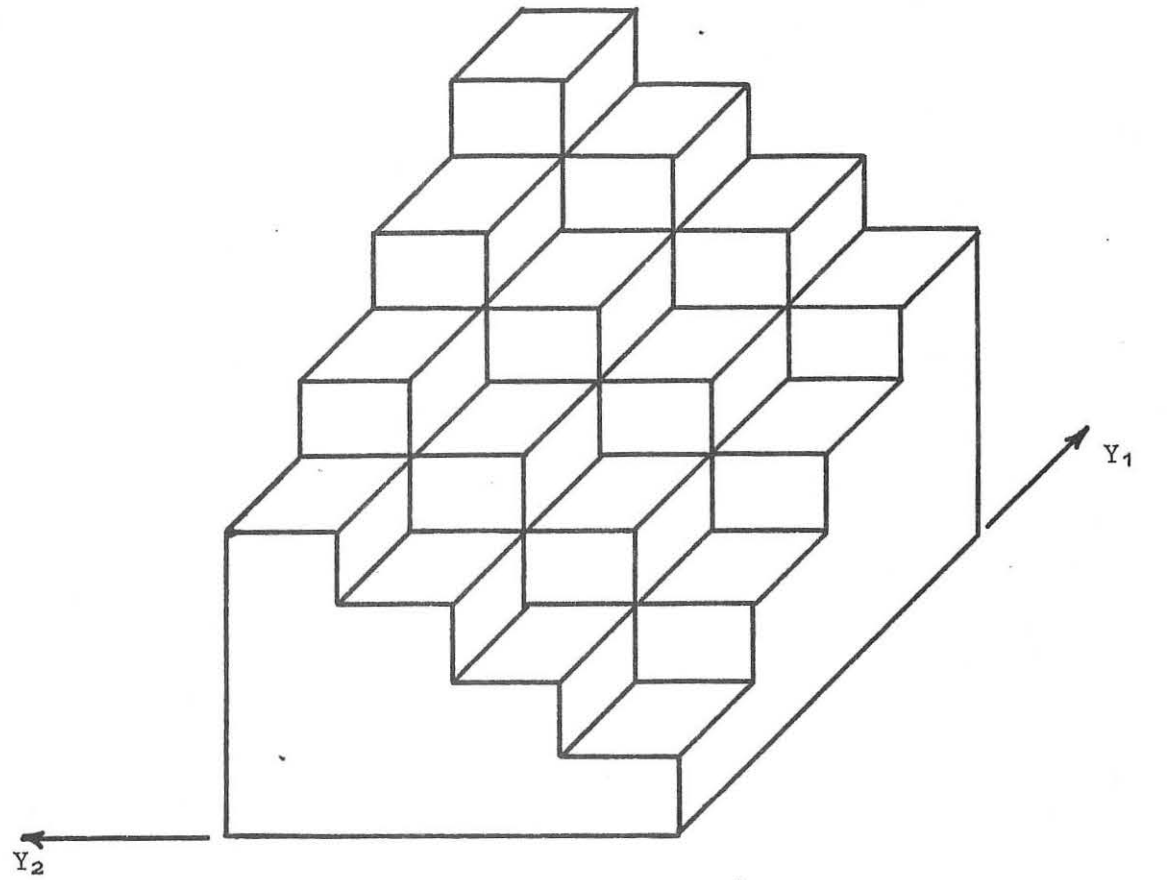


Figure 9. Discontinuous ODE expression, a step function.

new value until the next boundary is crossed. (See Figure 10.) (A challenging problem involving extensive use of tabular data is presented in [3])

Subroutine Name:	TABLE
Option:	Single trap (IFLAG=10,20)
NPHI:	2
COMMON block:	GRID/AA,XX,YY,IX,IY/
	FVAL/AA1/
Initialization:	Set X0, X1, Y0, Y1; set ABSER
PHI components	PHI(1) = (X1-Y(1))*(Y(1)-X0)
	PHI(2) = (Y1-Y(2))*(Y(2)-Y0)
	PHIP(1) = ((X1+X0)-2*Y(1))*YP(1)
	PHIP(2) = ((Y1+Y0)-2*Y(2))*YP(2)
Update:	Print T, Y(1), Y(2);
	shift bracketing indices;
	shift X0, X1, Y0, Y1, AA1;
	evaluate F, set PHI(INDEX)=0, and evaluate PHIP(INDEX)

The grid partition is defined in the main program as XX(I), YY(J), with the F values being AA(I,J), I=1,...,9, J=1,...,9, and being held in common. IX, IY

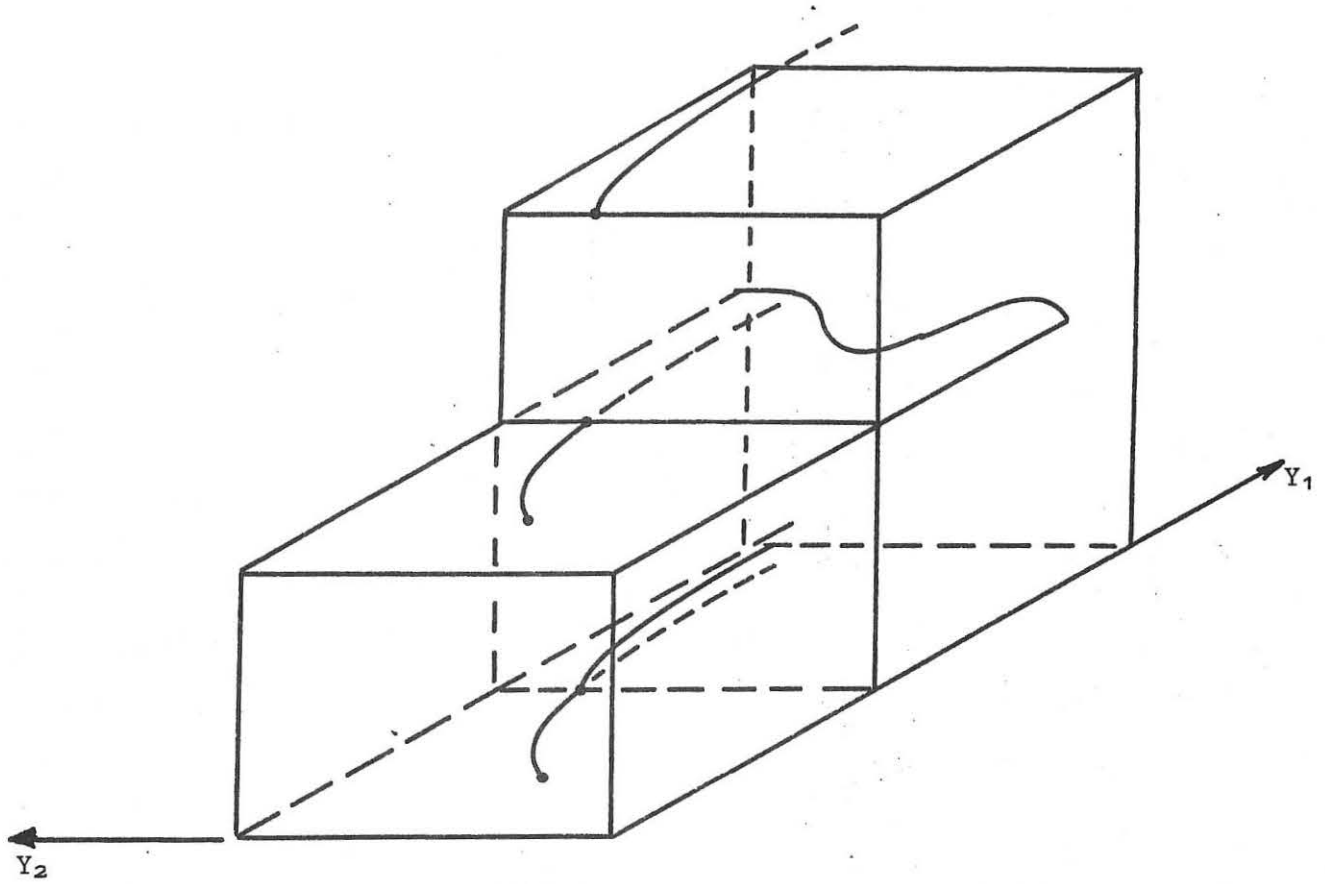


Figure 10. The plane surfaces defining the F function. The plane surface remains constant until the grid value is isolated. At update, the plane surface is redefined and this value will be used until the next grid value is isolated.

define the indices for the bracketing values, XX(IX) ≤ Y(1)<XX(IX+1), YY(IY) ≤ Y(2)<YY(IY+1), and AA1 = AA(IX,IY). X0 and X1 are the current bounds for Y(1), with Y0 and Y1 being those for Y(2). During update, IX or IY is shifted forward or backward depending upon which border is crossed. The X0, X1, Y0, Y1 values as well as the AA1 value are shifted to the new XX, YY values. (If the trapping iteration converged to a point not yet across the boundary, the "trapped value" of Y(INDEX) is used instead of the corresponding XX or YY value to give the proper sign to the new PHI component.) ABSER is set automatically to integration precision.

The grid values have been set in unit increments for easy identification in the output. In addition, the simplicity of the defining problem permits the trapping precision to be checked analytically. Thus, the error due to the trapping precision can be seen. Appendix B contains the SUBPHI listing along with the computed results showing the trapping stops and corresponding coordinates. The problem has also been run with exact stopping conditions, the time corresponding to the grid crossings being listed along with the integration results at T=1.0. The results achieved by "natural" trapping using RKF45 are also presented showing the number of additional derivative evalu-

ations required when the integrator attempts to locate the points of discontinuity itself.

10.5 A Highly Oscillatory Problem

During the solution of an ODE system, a PHI function may vanish several times within a given step. Since the trapping procedure is activated only if a sign change is observed, zeros of the PHI component could easily go undetected. The user may activate the PANIC mode during the integration so the solution is monitored throughout each integration step. Any detected sign change or vanishing component will activate TRAPPD. This PANIC analysis is expensive since the derivative is calculated at each substep for the PHI evaluation. If PANIC is referenced in SETRAP, the user should apply the ***single trap*** continuous mode (or the step-by-step modes). Since the multiple trap mode is likely to step over additional zeros, the option will be shifted to the single-trap option if the panic option is active in SETRAP.

The term "high oscillatory" is a relative one, depending in part, upon the integration step size. Thus, the analysis of PHI(1)=SIN(OSC*PI*T) poses no difficulties as long as the integration step size is less than 1/OSC. If the natural step size of the integration, however, steps over two or more zeros, these vanishing points (or part of these points) may go undetected. As an example, consider an ODE system with a polynomial solution of order less than five. Because the truncation error terms are zero in the RKF45T package, large step sizes will be attempted (restricted only by an imposed limit on the allowable step size increase in RKFST). The PHI function is sinusoidal with a period of 2/OSC. Two examples, OSC=5.0D0 and OSC=10.0D0, are presented with PANIC activated in SETRAP by setting NOTFAL=.TRUE.. The example OSC=5.0D0 is repeated without the PANIC option.

Subroutine Name:	OSCIL8
Option:	Single trap (IFLAG=10,20)
PANIC Option:	in SETRAP, NOTFAL=.TRUE.
NPHI:	1
COMMON block:	NOSC/OSC
ODE:	$Y' = 4\ T^3 + 3\ T^2 + 2\ T + 1$, Y=0 at T=0
Initialization:	none
PHI components	PHI(1) = DSIN(OSC*PI*T)
	PHIP(2) = DCOS(OSC*PI*T)*OSC*PI
Update:	Print T, PHI and Y

Without the use of the PANIC option in SETRAP, only the initial and final zeros were isolated. All intermediate points were missed. With the PANIC option in RKFST, however, all zeros were isolated for both OSC=5.0D0 and OSC=10.0D0. The OSC=10.0D0 is included since the PANIC mesh points coincide with the zeros of the PHI function. SUBPHI and output for both values of OSC are given in Appendix B.

As long as the mesh refinement in PANIC isolates only one zero per substep, no difficulties should be encountered. If multiple zeros occur within a mesh grid, the user should consider using a step-by-step mode and monitor PHI at

each step. If PHI is actually oscillating so quickly over the integration steps, another integration method may be preferable.

10.6 A Large Convergence Region

One of the most difficult problems to analyze concerns determining the vanishing point when the convergence region is large. If the derivative of a PHI component is nearly zero in the vacinity of a vanishing point, the convergence region can be quite large, and locating the zero point accurately may be impossible (regardless of the iteration procedure.) Detection of the problem, however, is quite simple since the user need only monitor the PHIP values. In some applications the vanishing region of PHI is the important feature rather than the exact zero location (e.g., interpolation from tabular data or bounding a control parameter) in which case the large convergence region poses no difficulty.

To illustrate a PHI component having a flat convergence region a simple polynomial is considered, PHI(1)=(T-2)**IPOW. The zero occurs at T=2 with the flatness controlled by the parameter IPOW. The ODE is not of importance for the example. Consider the two body problem (example 1) with eccentricity=0, i.e., a circular orbit, and apply the PANIC option in SETRAP to better identify the convergence region.

```
Subroutine Name:   FLAT
Option:            Single trap (IFLAG=10,20)
PANIC Option:      in SETRAP, NOTFAL=.TRUE.
NPHI:              1
COMMON block:      EXPO/IPOW,POWER,IPOWM1
Initialization:    none

PHI components     PHI(1)  =  (T-2.0D0)**IPOW

                   PHIP(2) = POWER*(T-2.0D0)**IPOWM1

Update:            Print T, PHI and PHIP
```

The values of IPOW, IPOWM1(=IPOW-1), and POWER (double precision value of IPOW) are held in common with the driving program. The example is run with IPOW=3, 5, and 9 to show the degree of flatness of the PHI component. PHIP is also monitored to inform the user of the flatness. The example is also run without the PANIC option in RKFST to show the zero estimate. In such cases, PHIP provides an important indication of the flatness difficulties. Greater accuracy is achieved using PANIC because the limits are closer to the zero points. The accuracy using the standard option, however, is quite acceptable. No warning message has been printed concerning "vanishing throughout the step" since the problem was contrived to show the convergence region of a single zero. In general, the user would include such a warning which would give another indication of the extreme flatness of the PHI function.

10.7 A Bouncing PHI Function

The TRAPPD analysis is not designed to handle a "bouncing" PHI component, i.e., a vanishing component which does not change signs as it passes through zero. (The example in §10.6, for IPOW=2,4,..., is an example of such a bouncing function.) The user may finesse the location of the zeros, however, by including the corresponding PHIP as an additional PHI component. The location of the PHIP zero will automatically locate the zero of the PHI component by "stepping" on it. The major difficulty in the TRAPPD strategy comes with the sign adjustment. If the user reevaluates the PHI(INDEX) component at update, then he must give PHI(INDEX) the sign associated with the region "across the boundary" or the same zero will be trapped again. If no update of PHI (or reevaluation of PHI) is made, TRAPPD automatically adjusts the sign (even for a "bouncing" function).

Polynomials provide a simple but effective test for "bouncing" PHI components. Three such polynomials (in even powers) are studied. Their derivatives are also given as PHI components. The zeros to be located are T=3, 6, 10, and 12 (See PHI statements). An extraneous zero will also be isolated since the PHIP component (treated as a PHI component) has an additional zero not corresponding to an original PHI zero). Since the zeros of the PHIP components being analyzed as PHI components are not of interest no print out has been made of these zeros. Figure 11 illustrates the PHI(I) components, I=1,3,5 being studied..

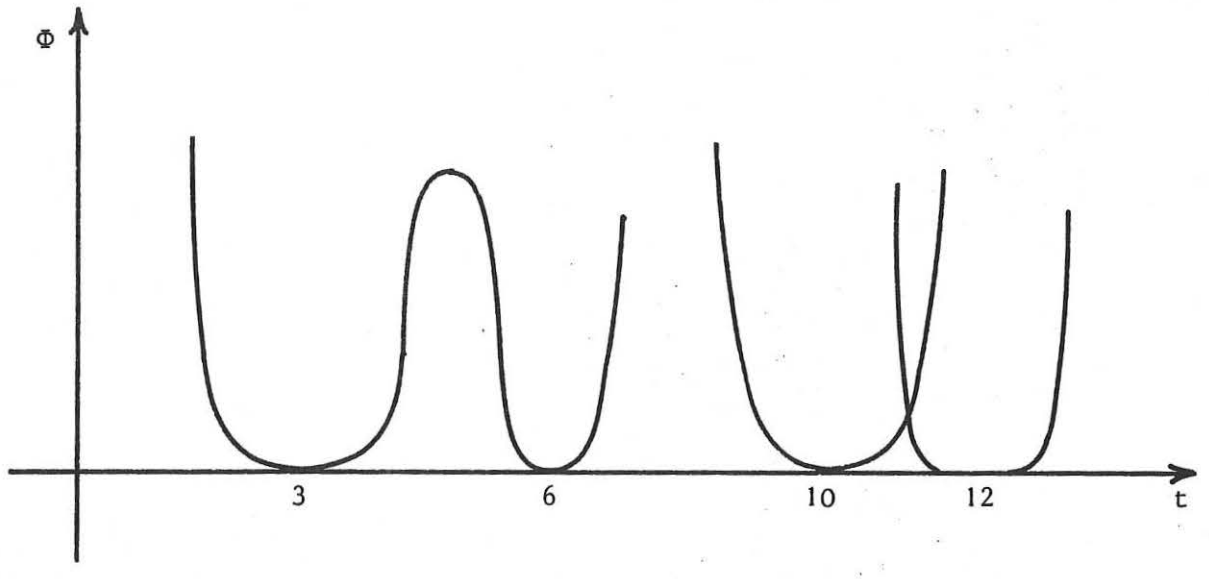


Figure 11. PHI components which "bounce" on zero".

Subroutine Name:	BOUNCE
Option:	Single trap (IFLAG=10,20)
PANIC Option:	in SETRAP, NOTFAL=.TRUE.
NPHI:	6

```
COMMON block:       none
Initialization:     none

PHI components      PHI(1) = (T-3.0D0)**4 * (T-6.0D0)**2
                    PHI(3) = (T-10.D0)**6
                    PHI(5) = (T-12.D0)**8

                    with derivatives also a PHI components
                    PHI(2) = PHIP(1)
                    PHI(4) = PHIP(2)
                    PHI(6) = PHIP(3)
Update:             Print T, PHI and PHIP
```

PHI(I), I=2,4,6 are essential for this approach to the problem. (The derivatives, PHIP(I), I=2,4,6, may be approximated by secants or may be set equal to zero activating a false position estimate of the zero points in TRAPPD). All zeros of the PHI components are isolated. The lack of accuracy in some zeros comes from the flatness of the PHI curves. (See example 6.) rather than from the "bouncing" effects. The SUBPHI listing and output for this example are given in Appendix B.

11. CONCLUSIONS

The RKF45T software package extends the RKF45 package to include intermediate integration stops whenever components of a user-supplied function vanish. These PHI components may be a function of the independent variable, the dependent variables, or their derivatives. Should multiple zeros occur at a given point or within a given step, each zero will be isolated in order, i.e., in the direction of the integration. After each zero is isolated, an update call is made to SUBPHI (the user-supplied subroutine for evaluating the PHI components) at which time the user may print information or change the PHI functions or even the differential equations themselves. Several modes of operation are available so that the user may execute the integration in a step-by-step fashion or in a continuous fashion (corresponding to the RKF45 options) with intermediate communication to the user occurring at the update for each vanishing PHI component. Safety features are included which may be activated if the PHI function analysis indicates difficulties are occurring. The trapping feature performs well for the examples presented. The success of this feature, however, may depend upon the user's understanding of the PHI components before applying the trapping options. The examples presented show a wide range of applications, but are, in part, academic in nature. Practical examples involving the solution of of optimal control problems with discontinuities in the differential equations or higher derivatives are given in [3] and [4].

Table 1. IFLAG values returned from the RKF45T package.

-2	T + 1 step	Step-by-step integration without trapping option. Integration is proceeding normally. TOUT has not been reached.
+2	TOUT	Continuous integration without trapping option, or step-by-step or trapping step-by-trapping step option with TOUT having been reached. (Normal mode for advancing the solution in the continuous mode after TOUT has been reached. For continuing in the step-by-step or trapping step-by-trapping step mode, IFLAG must be reset. In all cases, TOUT must be reset.)
-25	T*	Trapping step-by-trapping step mode stopping at a zero value of a PHI component. (Before the return to the driving program, an update call was made to SUBPHI identifying the component of PHI which had vanished.)
-20	T + 1 step	Step-by-step mode with trapping option. If a zero has vanished at the current T value, an update call has already been made to SUBPHI identifying the component of PHI which has vanished.)
+20	TOUT	Continuous integration with trapping option. Update calls to SUBPHI have informed the user of each vanishing point. PHI, Y, and the differential equations may be altered at each update call. The integration is continued from the trapped point each time a zero is isolated.
+25	TOUT	Continuous integration with trapping option. Update calls to SUBPHI have informed the user of each vanishing point. PHI and Y may be altered at each update call but no updates in the differential equation expressions are permitted. If several zeros lie within the given step all should be detected by the RKF45T package. The integration is continued from the end of the integration step rather than from the isolated zero point.

IFLAG values if difficulties have been encountered during the integration or trapping.

+94	TL	The trapping option is being used. Too many iterations were required in TRAPPD. PANIC has printed out information concerning PHI, and the solution is returned at TL, the last point analyzed which has not passed through the zero point. Calling RKF45T without changing IFLAG will terminate the program. Resetting IFLAG will continue the integration and zero search but may waste computing time or lead to an "infinite loop" because of difficulties in the PHI component.

+97	TL	The trapping option is being used. The trapping bounds have become too close. The same exit procedure is used as with IFLAG = 94.
+3	Initial T	RELERR is too small. RKFST has reset the value. The integration will continue if RKF45T is referenced again. No ste has been taken. (IFLAG will be reset by RKF45T.)
+4	T < TOUT	More than 3000 derivative evaluations (more than 500) steps have been used in generating the solution. If RKF54T is referenced again, the function counter will be reset to 0 and the integratio will be continued. (IFLAG will be reset by RKF45T.)
+5	T < TOUT	A pure relative error is being requrested, and the solution is identically zero. An absolute error toleracne must be used to continue the integration. The step-by-step mode may be a good way to proceed. *Terminal error* if ABSERR remanis zero.
+6	T < TOUT	The requested accuracy could not be acheived using the minimum allowable step size. *Terminal error*.
+7	T < TOUT	Too much output is restricting the natural step size choice. RKF45T may be inefficient for solving this problem. Perhaps the user should consider the trapping option 20, 25 with PHI(J)=T-TOUT, in which TOUT is updated at each zero point of PHI. (See example 1, §10.) A step-by-step mode may also be applicable. *Terminal error* if the user does not reset IFLAG.
+8	Initial T	Invalid input parameters, occurring if: NEQN ≤ 0, T=TOUT and |IFLAG| .ne. 1 (T may equal TOUT when the problem is initialized. This sets up internal parameters for the integration including the initial derivative evaluation, resets IFLAG = -2 or +2, and returns to the user for printing out initial YP values.). RELERR or ABSERR < 0, IFLAG error for the non-trapping option: IFLAG = 0, IFLAG < -2, or IFLAG < 8 for the trapping option: IFLAG ≠ -25, -20, -15, -10, 10, 15, 20, 25.

Table 2.a. The partitioning of the WORK array for the RKF45T package

1	1	NEQN	YP values at T
K1M	NEQN+1	--	Step size estimate, H
K1	NEQN+2	2 NEQN+1	F1, ODE evaluation
K2	2 NEQN+2	3 NEQN+1	F2, ODE evaluation
K3	3 NEQN+2	4 NEQN+1	F3, ODE evaluation
K4	4 NEQN+2	5 NEQN+1	F4, ODE evaluation
K5	5 NEQN+2	6 NEQN+1	F5, ODE evaluation
K6	6 NEQN+2	7 NEQN+1	F6, ODE evaluation
K7	7 NEQN+2	8 NEQN+1	F7, ODE evaluation
K8	8 NEQN+2	9 NEQN+1	F8, ODE evaluation
K9	9 NEQN+2	10 NEQN+1	F9, ODE evaluation
KTF	10 NEQN+2	--	TF, T value at end of integration step (for trapping option)
K10	10 NEQN+3	11 NEQN+2	YF, solution at TF
K11	11 NEQN+3	12 NEQN+2	YPF, derivative of Y at TF
K12	12 NEQN+3	13 NEQN+2	Y2, solution at T2
K13	13 NEQN+3	14 NEQN+2	YP2, derivative of Y at T2
K14	14 NEQN+3	15 NEQN+2	PHI0, PHI vector at T
K15	14 NEQN+3 + NPHI	15 NEQN+2 + 2 NPHI	PHIP0, derivative of PHI at T
K16	14 NEQN+3 + 2 NPHI	14 NEQN+2 + 3 NPHI	PHI2, PHI vector at T2
K17	14 NEQN+3 + 3 NPHI	14 NEQN+2 + 4 NPHI	PHIP0, derivative of PHI at T2
K18	14 NEQN+3 + 4 NPHI	14 NEQN+2 + 5 NPHI	PHIF, PHI vector at TF
K19	14 NEQN+3 + 5 NPHI	14 NEQN+2 + 6 NPHI	PHIPF, derivative of PHI at TF
K20	14 NEQN+3 + 6 NPHI	14 NEQN+2 + 7 NPHI	PHIR, PHI vector at TR

K21	14 NEQN+3 + 7 NPHI	14 NEQN+2 + 8 NPHI	PHIPR, derivative of PHI at TR
K22	14 NEQN+3 + 7 NPHI	14 NEQN+2 + 8 NPHI	PHIL, PHI vector at TL
K23	14 NEQN+3 + 8 NPHI	14 NEQN+2 + 9 NPHI	PHIPL, derivative of PHI at TL
K24	14 NEQN+3 + 9 NPHI	14 NEQN+2 +10 NPHI	PHIB, information about bouncing PHI components
K25	14 NEQN+3 +10 NPHI	14 NEQN+2 +11 NPHI	PHIV, information about vanishing PHI components
K26	14 NEQN+3 +11 NPHI	14 NEQN+2 +12 NPHI	PHIPV, derivative of PHIV
KSAVRE	14 NEQN+3 +13 NPHI	--	RELERR value saved
KSAVAE	14 NEQN+3 +13 NPHI	--	ABSERR value saved

Table 2.b. The partitioning of the IWORK array for the RKF45T package

1	NFE	The number of derivative evaluations used by RKF45T (NFE includes NEXTRA)
2	NREJ	The number of rejected steps encountered in RKF45T during the solution of the ODE
3	NEXTRA	The number of derivative evaluations required by the trapping option (NEXTRA is included in NFE)
4	KOP	Counting index in RKFST to guard agains too many output points
5	INIT	Index in RKFST which shifts analysis into or away from the initialization block in RKFST
6	IFLAG	A flag which defines the mode of operation for RKF45T
7	KFLAG	Safety flag which protects IFLAG between calls to RKFST
8	LFLAG	Designating flag in RKFST for the trapping option (If the trapping option is used, IFLAG is reset to a standard RKF45 value, and LFLAG defines the trapping mode of operation.)
9	ISTART	Initializing flag for the trapping option if the "late start" option is being used
10	KOUNTR	Counting parameter for updates in the PHI components (and for identifying the initialization stage in the user-supplied subroutine, SUBPHI

Table 3. Pre-set and user-supplied constants for the RKF45T package.

Notation: S = subroutine name,
SV = standard value
AV = additional value(s)

FIFTH — designates the order of the scaled solution generated in SCALED. FIFTH=.FALSE. gives a 4th order solution; FIFTH=.TRUE. gives a 5th order solution.
(S: SCALED, SV: .FALSE., AV: .TRUE.)

FLGOUT — controls referncing of OUTFLG to print out integration error messages. FLGOUT=.TRUE., references OUTFLG.
(S: RKF45T, SV: .TRUE., AV: .FALSE.)

IOPT — controls printing option in TRAPPD. IOPT=0 suppresses printing option. IOPT=1 prints entry and exit conditions.
(S: TRAPPD, SV: 0 , AV: 1)

IPRINT — controls printing in PANIC when referenced by TRAPPD. IPRINT=1, prints T and PHI; IPRINT=2, prints T, PHI, PHIP, Y, and YP.
(S: PANIC, SV: 1 , AV: 2)

JPRINT — controls printing in PANIC when referenced by SETRAP. JPRINT=0, suppresses all printing; JPRINT=1 prints T and PHI; IPRINT=2, prints T, PHI, PHIP, Y, and YP.
(S: PANIC, SV: 0 , AV: 1, 2)

MAXIT — gives the maximum number of iterations permitted on each trapping attempt.
(S: TRAPPD, SV: 10, AV: user-supplied in data statement)

NOTFAL — controls the referencing of PANIC (from SETRAP and TRAPPD). (NOTFAL may have different values in SETRAP and TRAPPD.) If NOTFAL=.TRUE. PANIC will be referenced; if NOTFAL=.FALSE., PANIC will not be referenced. (If difficulties are encountered in the trapping analysis in TRAPPD (e.g., too many iteration are being required), then PANIC will be referenced regardless of the value of NOTFAL.)
(S: SETRAP, TRAPPD, SV: .FALSE., AV: .TRUE.)

NPOINT — controls the number of increments for the partition of the integration step (integer value of POINTS)
(S: PANIC, SV: 10, AV: user-supplied in a data statement)
(S: VANISH, SV: 4, AV: user-supplied in a data statement)

POINTS — controls the number of increments for the partition of the integration step (souble precision value of NPOINT)
(S: PANIC, SV: 10.0D0, AV: user-supplied in a data statement)
(S: VANISH, SV: 4.0D0, AV: user-supplied in a data statement)

REMIN — is the tolerance threshold for the relative error tests. Bounding values are vaguely deinded, say, REMIN > MAX{U, 1.0D-12}
(S: RKFST, SV: given above, AV: user-supplied in a data statement)

TBOUND is the limiting iteration step size in TRAPPD (being related to unit round-off (see parameter U) or set to a somewhat larger value by the user.
(S: TRAPPD, SV: 26*U or user-supplied)

U is the unit round-off for the particular computer system. (See computer listing.)
(S: RKFST, SV: machine dependent (user-supplied))

12. REFERENCES

[1] Fehlberg, E. Low-Order Classical Runge-Kutta Formulae with Step Size Control and Their Application to Some Heat Transfer Problems.
NASA TR R-315 (July 1969).

[2] Horn, M.K. Scaled Runge-Kutta Algorithms for Handling Dense Output.
DFVLR-FB 81-13 (April, 1981).

[3] Horn, M.K. A Fortran Program for Solving State/Control-Constraint Optimal Control Problems with System Equations Having Expressions Involving Tabular Data.
DFVLR-IB 515/1 (1983).

[4] Horn, M.K. A Numerical Solution of State/Control-Constraint Optimal Control Problems with Piecewise Continuous Derivatives Using RKF45T
DFVLR-IB 515/2 (1983).

[5] Horn, M.K. Subroutines for Handling Tabular Data Used in System Equations.
DFVLR-IB 515/4 (1983).

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

APPENDIX A. PROGRAM LISTING FOR RKF45T AND RELATED SUBROUTINES

```
      SUBROUTINE RKF45T(F,SUBPHI,NPHI,NEQN,Y,T,TOUT,RELERR,ABSERR,
     1            IFLAG,WORK,IWORK)
C
C     IS A FEHLBERG FOURTH-FIFTH ORDER RUNGE-KUTTA METHOD WITH AN
C     OPTION TO HALT THE INTEGRATION IF ANY COMPONENT OF A USER
C     SUPPLIED VECTOR FUNCTION VANISHES, PHI(J) = FUNCTION(T,Y,YP),
C
C     SUBROUTINES RKF45 AND RKFS,
C     WRITTEN BY H.A.WATTS AND L.F.SHAMPINE
C                SANDIA LABORATORIES
C                ALBUQUERQUE, NEW MEXICO,
C     HAVE BEEN MODIFIED SLIGHTLY (INTO RKF45T AND RKFST) WITH AN
C     ADDITIONAL SUBROUTINE, TRAPPD, WHICH STOPS THE INTEGRATION
C     IF ANY COMPONENT OF PHI VANISHES.  THE MODIFICATIONS TO
C     RKF45 AND RKFS AND THE TRAPPD ROUTINE HAVE BEEN ADDED BY
C                M.K. HORN
C                DFVLR-OBERPFAFFENHOFEN
C
C     RKF45T IS PRIMARILY DESIGNED TO SOLVE NON-STIFF AND MILDLY STIFF
C     DIFFERENTIAL EQUATIONS WHEN DERIVATIVE EVALUATIONS ARE INEXPENSIVE
C     RKF45T SHOULD GENERALLY NOT BE USED WHEN THE USER IS DEMANDING
C     HIGH ACCURACY.
C-----------------------------------------------------------------------
C     ABSTRACT
C-----------------------------------------------------------------------
C     SUBROUTINE RKF45T INTEGRATES A SYSTEM OF NEQN FIRST ORDER
C     ORDINARY DIFFERENTIAL EQUATIONS OF THE FORM
C              DY(I)/DT = F(T,Y(1),Y(2),...,Y(NEQN))
C              WHERE THE Y(I) ARE GIVEN AT T
C     WITH AN OPTION TO TO STOP THE INTEGRATION WHEN ANY COMPONENT OF
C     A CONSTRAINT VECTOR, PHI0(J), A FUNCTION OF T, Y, AND YP IS
C     SATISFIED.
C
C     TYPICALLY THE SUBROUTINE IS USED TO INTEGRATE FROM T TO TOUT BUT
C     CAN BE USED AS A ONE-STEP INTEGRATOR TO ADVANCE THE SOLUTION A
C     SINGLE STEP IN THE DIRECTION OF TOUT OR TO THE VALUE OF T FOR
C     WHICH A COMPONENT OF THE PHI VECTOR VANISHES.  ON RETURN THE
C     PARAMETERS IN THE CALL LIST ARE SET FOR CONTINUING THE INTEGRA-
C     TION.  THE USER HAS ONLY TO CALL RKF45T AGAIN (AND PERHAPS DEFINE
C     A NEW VALUE FOR TOUT).  ACTUALLY, RKF45T IS AN INTERFACING ROUTINE
C     WHICH CALLS SUBROUTINE RKFST FOR THE SOLUTION.  RKFST IN TURN
C     CALLS SUBROUTINE FEHL WHICH COMPUTES AN APPROXIMATE SOLUTION
C     OVER ONE STEP.
C
C     RKF45T USES THE RUNGE-KUTTA-FEHLBERG (4,5) METHOD DESCRIBED IN
C     THE REFERENCES
C     E. FEHLBERG, LOW-ORDER CLASSICAL RUNGE-KUTTA FORMULAS WITH STEPSIZ
C                  CONTROL, NASA TR R-315.
C                  ALSO IN COMPUTING, 6(1970),PP.61-71.
C
C     L.F. SHAMPINE AND H.A. WATTS, PRACTICAL SOLUTION OF ORDINARY
C                  DIFFERENTIAL EQUATIONS BY RUNGE-KUTTA METHODS.
C                  SANDIA LABORATORIES REPORT SAND76-0585.
```

```
C
C     THE PERFORMANCE OF RKF45 IS ILLUSTRATED IN THE REFERENCE
C     L.F. SHAMPINE, H.A. WATTS, S. DAVENPORT, SOLVING NON-STIFF ORDINAR
C                  DIFFERENTIAL EQUATIONS-THE STATE OF THE ART.
C                  SANDIA LABORATORIES REPORT SAND78-0182 .  ALSO IN
C                  SIAM REVIEW,18(1976),PP.376-411.
C
C
C
C     THE PARAMETERS REPRESENT
C     F -- SUBROUTINE F(T,Y,YP) TO EVALUATE DERIVATIVES YP(I)=DY(I)/DT
C     SUBPHI -- SUBROUTINE SUBPHI
C           A SUBROUTINE FOR DETERMINING A CONSTRAINT VECTOR USED FOR
C           TEMPORARILY STOPPING THE INTEGRATION (AT PHI(J)=0)
C     NEQN -- NUMBER OF EQUATIONS TO BE INTEGRATED
C     NPHI -- NUMBER OF COMPONENTS OF THE PHI VECTOR
C     Y( ) -- SOLUTION VECTOR AT T
C     T -- INDEPENDENT VARIABLE
C     TOUT -- OUTPUT POINT AT WHICH SOLUTION IS DESIRED
C     RELERR,ABSERR -- RELATIVE AND ABSOULTE ERROR TOLERANCES FOR LOCAL
C           ERROR TEST. AT EACH STEP THE CODE REQUIRES THAT
C                ABS(LOCAL ERROR) .LE. RELERR*ABS(Y) + ABSERR
C           FOR EACH COMPONENT OF THE LOCAL ERROR AND SOLUTION VECTORS
C     IFLAG -- INDICATOR FOR STATUS OF INTEGRATION
C     WORK( ) -- ARRAY TO HOLD INFORMATION INTERNAL TO RKF45 WHICH IS
C           NECESSARY FOR SUBSEQUENT CALLS.  MUST BE DIMENSIONED
C           AT LEAST 3+14*NEQN+5*NPHI
C     IWORK( ) -- INTEGER ARRAY USED TO HOLD INFORMATION INTERNAL TO
C           RKF45 WHICH IS NECESSARY FOR SUBSEQUENT CALLS.  MUST BE
C           DIMENSIONED AT LEAST 10
C
C
C-----------------------------------------------------------------------
C     FIRST CALL TO RKF45T
C-----------------------------------------------------------------------
C
C     THE USER MUST PROVIDE STORAGE IN HIS CALLING PROGRAM FOR THE ARRAY
C     IN THE CALL LIST  -  Y(NEQN), WORK(3+14*NEQN+5*NPHI) , IWORK(10),
C     DECLARE F AND SUBPHI IN AN EXTERNAL STATEMENT, SUPPLY SUBROUTINE
C     F(T,Y,YP) AND SUBPHI
C
C     AND INITIALIZE THE FOLLOWING PARAMETERS
C     NEQN -- NUMBER OF EQUATIONS TO BE INTRGRATED.  (NEQN .GE. 1)
C     Y( ) -- VECTOR OF INITIAL CONDITIONS
C     T -- STARTING POINT OF INTEGRATION, MUST BE A VARIABLE
C        T=TOUT IS ALLOWED ON THE FIRST CALL ONLY, IN WHICH CASE
C        RKF45T RETURNS WITH IFLAG=2 IF CONTINUATION IS POSSIBLE.
C     RELERR,ABSERR -- RELATIVE AND ABSOLUTE LOCAL ERROR TOLERANCES
C        WHICH MUST BE NON-NEGATIVE.  RELERR MUST BE A VARIABLE WHILE
C        ABSERR MAY BE A CONSTANT.  THE CODE SHOULD NORMALLY NOT BE
C        USED WITH RELATIVE ERROR CONTROL SMALLER THAN ABOUT 1.E-8 .
C        TO AVOID LIMITING PRECISION DIFFICULTIES THE CODE REQUIRES
C        RELERR TO BE LARGER THAN AN INTERNALLY COMPUTED RELATIVE
C        ERROR PARAMETER WHICH IS MACHINE DEPENDENT.  IN PARTICULAR,
C        PURE ABSOLUTE ERROR IS NOT PERMITTED.  IF A SMALLER THAN
C        ALLOWABLE VALUE OF RELERR IS ATTEMPTED, RKF45 INCREASES
C        RELERR APPROPRIATELY AND RETURNS CONTROL TO THE USER BEFORE
C        CONTINUING THE INTEGRATION.
```

```
C
C     IFLAG -- INDICATES THE MODE OF OPERATION OF THE PROGRAM.  IF
C        THE INTEGRATOR IS TO BE USED WITHOUT THE TRAPPING OPTION,
C        IFLAG = +1 OR -1 INITIALLY. IF THE TRAPPING OPTION IS TO BE
C        USED, IFLAG = -15, -10, 10, OR 15 INITIALLY (SEE BELOW).
C
C     IFLAG -- +1,-1 INDICATOR TO INITIALIZE THE CODE FOR EACH NEW
C        PROBLEM.  NORMAL INPUT IS +1.  THE USER SHOULD SET IFLAG=-1
C        ONLY WHEN ONE-STEP INTEGRATOR CONTROL IS ESSENTIAL.  IN THIS
C        CASE, RKF45T ATTEMPTS TO ADVANCE THE SOLUTION A SINGLE STEP
C        IN THE DIRECTION OF TOUT EACH TIME IT IS CALLED.  SINCE THIS
C        MODE OF OPERATION RESULTS IN EXTRA COMPUTING OVERHEAD, IT
C        SHOULD BE AVOIDED UNLESS NEEDED.
C
C     IFLAG OPTIONS--IF THE CONSTRAINT VECTOR, PHI, IS TO BE ANALYZED.
C        IF IFLAG = -15, -10, +10, +15, RKF45T RESETS IFLAG EQUAL TO
C        -1, -1, +1, OR +1, RESPECTIVELY, AND INITIALIZES OTHER
C        PARAMETERS TO ACTIVATE THE TRAPPING OPTION.  UPON RETURN TO
C        TO USER, IFLAG IS RESET TO -25, -20, 20, OR 25, RESPECTIVLEY.
C
C     IFLAG = 15--INTEGRATES FROM T TO TOUT, STOPPING INTERNALLY AT
C                 T = T* IF PHI(J) = 0 AT T*, AND THEN CONTINUING THE
C                 INTEGRATION FROM CONDITIONS AT THE END OF THE STEP
C                 IN WHICH PHI(J) VANISHED.  IF FURTHER COMPONENTS OF
C                 PHI VANISH WITHIN A GIVEN STEP, THESE WILL ALSO
C                 BE TRAPPED WITHOUT FURTHER INTEGRATION.  UPDATE
C                 CALLS ARE MADE TO SUBPHI SO THAT THE USER MAY PRINT
C                 INFORMATION OR UPDATE PHI, BUT THE DIFFERENTIAL
C                 EQUATIONS SHOULD NOT BE ALTERED SINCE THE SOLUTION
C                 IS ADVANCED FROM T+H AND F(T+H) WOULD NOT REFLECT
C                 THE CHANGES.  UPDATES IN THE PHI FUNCTION ARE POS-
C                 SIBLE, SINCE PHI AT T* AND T+H ARE REEVALUATED AT
C                 OR AFTER UPDATE, MAKING FURTHER TRAPPINGS POSSIBLE,
C                 BUT ANALYSIS OF THE PHI FUNCTION WILL OCCUR ONLY
C                 FOR VALUES OF T > T* .  UPON RETURNING, IFLAG EQUALS
C                 25, WHICH IS THE NORMAL MODE FOR CONTINUING.
C
C           = 10--INTEGRATES FROM T TO TOUT, STOPPING INTERNALLY AT
C                 T = T* IF PHI(J) = 0 AT T*, AND THEN CONTINUING THE
C                 INTEGRATION FROM CONDITIONS AT T*. UPDATE CALLS TO
C                 SUBPHI ARE MADE DURING WHICH THE USER MAY OUTPUT IN-
C                 FORMATION OR ALTER THE PHI FUNCTION.  THE SOLUTION
C                 VECTOR AND/OR THE DIFFERENTIAL EQUATIONS MAY ALSO
C                 BE CHANGED SINCE F(T,Y,YP) IS REEVALUATED AND THE
C                 INITITAL VALUE PROBLEM IS ESSENTIALLY RESTARTED.
C                 EXTREME CARE, HOWEVER,  SHOULD BE EXERCISED IF THE
C                 DIFFERENTIAL SYSTEM IS ALTERED.  IF PHI VANISHES
C                 AT POINTS BETWEEN T* AND T+H, THESE VALUES ARE
C                 TRAPPED BY CONTINUED INTEGRATION.  UPON RETURNING,
C                 IFLAG EQUALS 20, WHICH IS THE NORMAL MODE FOR CON-
C                 TINUING.
C
C           =-10--INTEGRATES FROM T TOWARDS TOUT, STEP BY STEP, RETURN-
C                 ING TO THE MAIN PROGRAM AT T* INSTEAD OF T+H IF
C                 PHI(J) = 0 AT T*, WHERE T .LT. T* .LT. T+H.  VECTORS
C                 Y AND YP AT T+H ARE RETURNED IN WORK ARRAY LOCATIONS
C                 1*NEQN+2 TO 2*NEQN+1 AND 2*NEQN+2 TO 3*NEQN+1,
C                 RESPECTIVELY, WITH T+H RETURNED IN WORK(3*NEQN+2).
```

```
C                UPDATES IN BOTH THE PHI VECTOR AND THE DIFFERENTIAL
C                EQUATIONS SYSTEM ARE POSSIBLE AS IN THE IFLAG=10
C                OPTION.  UPON RETURNING, IFLAG EQUALS -20, WHICH IS
C                THE NORMAL MODE FOR CONTINUING.
C
C           =-15--INTEGRATES FROM T TOWARDS TOUT RETURNING TO THE
C                MAIN PROGRAM AT T = T*, IF PHI(J) = 0 AT T* .  THE
C                VECTORS Y AND YP AT T+H ARE RETURNED IN WORK ARRAY
C                LOCATIONS  1*NEQN+2 TO 2*NEQN+1 AND 2*NEQN+2 TO
C                3*NEQN+1, RESPECTIVELY, WITH T+H RETURNED IN
C                WORK(3*NEQN+2).  UPDATES IN BOTH THE PHI VECTOR AND
C                THE DIFFERENTIAL EQUATIONS SYSTEM ARE POSSIBLE AS IN
C                THE IFLAG=10 OPTION.  UPON RETURNING, IFLAG EQUALS
C                -25, WHICH IS THE NORMAL MODE FOR CONTINUING.
C
C-------------------------------------------------------------------------
C     OUTPUT FROM RKF45T
C-------------------------------------------------------------------------
C
C     Y( ) -- SOLUTION AT T
C     T -- LAST POINT REACHED IN INTEGRATION.
C     IFLAG = 2 -- INTEGRATION REACHED TOUT. INDICATES SUCCESSFUL RETURN
C                  AND IS THE NORMAL MODE FOR CONTINUING INTEGRATION.
C           =-2 -- A SINGLE SUCCESSFUL STEP IN THE DIRECTION OF TOUT
C                  HAS BEEN TAKEN.  NORMAL MODE FOR CONTINUING
C                  INTEGRATION ONE STEP AT A TIME.
C           = 3 -- INTEGRATION WAS NOT COMPLETED BECAUSE RELATIVE ERROR
C                  TOLERANCE WAS TOO SMALL. RELERR HAS BEEN INCREASED
C                  APPROPRIATELY FOR CONTINUING.
C           = 4 -- INTEGRATION WAS NOT COMPLETED BECAUSE MORE THAN
C                  3000 DERIVATIVE EVALUATIONS WERE NEEDED.  THIS
C                  IS APPROXIMATELY 500 STEPS
C           = 5 -- INTEGRATION WAS NOT COMPLETED BECAUSE SOLUTION
C                  VANISHED MAKING A PURE RELATIVE ERROR TEST
C                  IMPOSSIBLE.  MUST USE NON-ZERO ABSERR TO CONTINUE.
C                  USING THE ONE-STEP INTEGRATION MODE FOR STEP
C                  IS A GOOD WAY TO PROCEED.
C           = 6 -- INTEGRATION WAS NOT COMPLETED BECAUSE REQUESTED
C                  ACCURACY COULD NOT BE ACHIEVED USING SMALLEST
C                  ALLOWABLE STEPSIZE. USER MUST INCREASE THE ERROR
C                  TOLERANCE BEFORE CONTINUED INTEGRATION CAN BE
C                  ATTEMPTED.
C           = 7 -- IT IS LIKELY THAT RKF45 IS INEFFICIENT FOR SOLVING
C                  THIS PROBLEM.  TOO MUCH OUTPUT IS RESTRICTING THE
C                  NATURAL STEPSIZE CHOICE.  USE THE ONE-STEP INTEGRATOR
C                  MODE.
C           = 8 -- INVALID INPUT PARAMETERS
C                  THIS INDICATOR OCCURS IF ANY OF THE FOLLOWING IS
C                  SATISFIED - NEQN .LE. 0
C                              T=TOUT AND IFLAG .NE. +1 OR -1
C                              RELERR OR ABSERR .LT. 0.
C                              IFLAG .EQ. 0 OR .LT. -2 OR .GT.8
C           = 20-- INTEGRATION HAS RETURNED USING TRAPPD OPTION WITH
C                  TOUT HAVING BEEN REACHED. THIS IS THE NORMAL MODE
C                  FOR CONTINUING INTEGRATION USING THE TRAPPING OP-
C                  TION. DURING THE INTEGRATION, THE SOLUTION WAS AD-
C                  VANCED FROM T* IN ANY STEP IN WHICH THE A COMPO-
C                  NENT OF THE VECTOR PHI VANISHED.
```

```
C           = 25-- INTEGRATION HAS RETURNED USING TRAPPD OPTION WITH
C                  TOUT HAVING BEEN REACHED.  THIS IS THE NORMAL MODE
C                  FOR CONTINUING INTEGRATION USING THE TRAPPING OP-
C                  TION.  DURING THE INTEGRATION THE SOLUTION WAS AD-
C                  VANCED FROM THE END OF ANY STEP IN WHICH THE TRAP-
C                  PING OPTION WAS APPLIED TO THE VECTOR PHI.
C           =-20-- INTEGRATION HAS RETURNED USING TRAPPD OPTION AFTER
C                  TAKING A SINGLE STEP IN THE DIRECTION OF TOUT. IF THE
C                  TRAPPING OPTION WERE NOT ACTIVATED, CONDITIONS ARE
C                  RETURNED AT T+H.  IF A COMPONENT OF PHI VANISHED
C                  WITHIN THE STEP CONDITIONS ARE RETURNED AT T=T*,
C                  WHERE PHI(J) = 0 AT T* FOR AT LEAST ONE VALUE OF J .
C                  IN THIS CASE, Y AND YP AT THE END OF THE STEP HAVE
C                  BEEN STORED IN WORK ARRAY LOCATIONS 1*NEQN+2 TO
C                  2*NEQN+1 AND 2*NEQN+2 TO 3*NEQN+1 WITH T STORED IN
C                  WORK(3*NEQN+2).  NORMAL MODE FOR CONTINUING
C                  INTEGRATION ONE STEP AT A TIME  USING THE TRAPPD
C                  OPTION.
C           =-25-- INTEGRATION HAS RETURNED USING TRAPPD OPTION AFTER
C                  REACHING TOUT OR T*, THE FIRST VALUE OF T FOR WHICH
C                  A COMPONENT OF PHI VANISHED.  IF TOUT WERE NOT
C                  REACHED, CONDITIONS AT THE END OF THE STEP IN
C                  WHICH THE COMPONENT OF PHI VANISHED WILL HAVE
C                  BEEN STORED IN WORK ARRAY LOCATIONS 1*NEQN+2 TO
C                  2*NEQN+1 AND 2*NEQN+2 TO 3*NEQN+1 WITH T STORED IN
C                  WORK(3*NEQN+2).  NORMAL MODE FOR CONTINUING
C                  INTEGRATION FROM ONE T* TO THE NEXT USING THE
C                  TRAPPD OPTION.
C           = 94-- ERROR HAS ARISEN IN TRAPPD ROUTINE.  TOO MANY
C                  ITERATIONS WERE USED IN ATTEMPTING TO ISOLATE
C                  A VANISHING COMPONENT OF PHI.  CONDITIONS ARE
C                  RETURNED AT THE LAST ESTABLISHED POINT FOR WHICH
C                  THE PHI COMPONENT HAD NOT YET CHANGED SIGN.
C           = 97-- ERROR HAS ARISEN IN TRAPPD ROUTINE.  TRAPPING
C                  LIMITS ON PHI ARE TOO CLOSE TO PERMIT FURTHER
C                  TRAPPING.  CHECK PHI FUNCTION OR ITS DERIVATIVE
C                  FOR POSSIBLE DISCONTINUITIES.
C     WORK( ),IWORK( ) -- INFORMATION WHICH IS USUALLY OF NO INTEREST
C                  TO THE USER BUT NECESSARY FOR SUBSEQUENT CALLS
C                  WORK(1),...,WORK(NEQN) CONTAIN THE FIRST DERIVATIVES
C                  OF THE SOLUTION VECTOR Y AT T.  WORK(NEQN+1) CONTAINS
C                  THE STEP SIZE H TO BE ATTEMPTED ON THE NEXT STEP.
C                  IWORK(1) CONTAINS THE DERIVATIVE EVALUATION COUNTER.
C
C
C-----------------------------------------------------------------------
C     SUBSEQUENT CALLS TO RKF45T
C-----------------------------------------------------------------------
C
C     SUBROUTINE RKF45T RETURNS WITH ALL INFORMATION NEEDED TO CONTINUE
C     THE INTEGRATION.  IF THE INTEGRATION REACHED TOUT, THE USER NEED
C     ONLY DEFINE A NEW TOUT AND CALL RKF45T AGAIN. IN THE ONE-STEP
C     INTEGRATOR MODE (IFLAG=-2) THE USER MUST KEEP IN MIND THAT EACH
C     STEP TAKEN IS IN THE DIRECTION OF THE CURRENT TOUT. UPON REACHING
C     TOUT INDICATED BY CHANGING IFLAG TO 2), THE USER MUST THEN DEFINE
C     A NEW TOUT AND RESET IFLAG TO -2 TO CONTINUE IN THE ONE STEP INTE-
C     GRATOR MODE.
C
```

```
C
C     IF THE TRAPPING OPTION HAS BEEN USED, SIMILAR FLAGS ARE SET
C     (IFLAG = +20, OR +25 FOR REACHING TOUT OR IFLAG = -20 OR -25
C     IN THE STEP-BY-STEP MODE OR TRAPPING STEP-BY-TRAPPING STEP
C     MODE.)  TO DEACTIVATE THE TRAPPING OPTION, THE USER NEED ONLY
C     SET IFLAG = -2 OR +2 IF THE INTEGRATION IS BEING CONTINUED OR
C     SET IFLAG = -1 OR +1 IF A NEW INTEGRATION IS TO BE STARTED.
C
C     IF THE INTEGRATION WERE NOT COMPLETED BUT THE USER STILL WANTS
C     TO CONTINUE (IFLAG=3,4 CASES), HE JUST CALLS RKF45T AGAIN. WITH
C     IFLAG=3, THE RELERR PARAMETER HAS BEEN ADJUSTED APPROPRIATELY FOR
C     CONTINUING THE INTEGRATION.  IN THE CASE OF IFLAG=4 THE FUNCTION
C     COUNTER WILL BE RESET TO 0 AND ANOTHER 3000 FUNCTION EVALUATIONS
C     WILL BE ALLOWED. (IF TRAPPING OPTIONS WERE USED ON THE PREVIOUS
C     STEP, IFLAG WILL BE RESET IN RKFST TO ACTIVATE THESE OPTIONS.)
C
C     HOWEVER, IN THE CASE IFLAG=5, THE USER MUST FIRST ALTER THE
C     ABSOLUTE ERROR CRITERION TO USE A POSITIVE VALUE OF ABSERR.
C     IFLAG=5 MAY OCCUR IN RKF45T WHEN A PURE RELATIVE ERROR TEST
C     IS USED TO CHECK THE ACCURACY OF THE SOLUTION. IF ALL COMPO-
C     NENTS OF THE SOLUTION VANISH AND NO ABSOLUTE ERROR TEST IS
C     USED, IFLAG IS SET EQUAL TO 5 .  THE USER MUST SPECIFY
C     A POSITIVE VALUE OF ABSERR BEFORE THE INTEGRATION
C     CAN PROCEED.  IF HE DOES NOT, EXECUTION IS TERMINATED.
C
C     ALSO, IN THE CASE IFLAG=6, IT IS NECESSARY FOR THE USER TO RESET
C     IFLAG TO 2 (OR -2 WHEN THE ONE-STEP INTEGRATION MODE IS BEING USED
C     AS WELL AS INCREASING EITHER ABSERR, RELERR OR BOTH BEFORE THE
C     INTEGRATION CAN BE CONTINUED.  IF THIS IS NOT DONE, EXECUTION WILL
C     BE TERMINATED.  THE OCCURRENCE OF IFLAG=6 INDICATES A TROUBLE SPOT
C     (SOLUTION IS CHANGING RAPIDLY, SINGULARITY MAY BE PRESENT) AND IT
C     IS OFTEN INADVISABLE TO CONTINUE.
C
C     IF IFLAG=7 IS ENCOUNTERED, THE USER SHOULD USE THE ONE-STEP
C     INTEGRATION MODE WITH THE STEPSIZE DETERMINED BY THE CODE OR
C     CONSIDER SWITCHING TO THE ADAMS CODES DE/STEP,INTRP.  IF THE USER
C     INSISTS UPON CONTINUING THE INTEGRATION WITH RKF45, HE MUST RESET
C     IFLAG TO 2 BEFORE CALLING RKF45 AGAIN.  OTHERWISE, EXECUTION WILL
C     BE TERMINATED.
C
C     IF IFLAG=8 IS OBTAINED, INTEGRATION CAN NOT BE CONTINUED UNLESS
C     THE INVALID INPUT PARAMETERS ARE CORRECTED.
C
C     IT SHOULD BE NOTED THAT THE ARRAYS WORK,IWORK CONTAIN INFORMATION
C     REQUIRED FOR SUBSEQUENT INTEGRATION.  ACCORDINGLY, WORK AND IWORK
C     SHOULD NOT BE ALTERED.
C
C-----------------------------------------------------------------------
C
      IMPLICIT REAL*8 (A-H,O-Z)
C
      DIMENSION Y(NEQN),WORK(1),IWORK(10)
C
      EXTERNAL F,SUBPHI
      LOGICAL FLGOUT
      DATA FLGOUT/.TRUE./
C
C
```

```
      INDIC = 0
      CALL FLAGCK(IFLAG,INDIC,IWORK(8),IWORK(9))
C
      IF (IWORK(8) .EQ. 0)  NPHI = 1
C
C
C     COMPUTE INDICES FOR THE SPLITTING OF THE WORK ARRAY
C
      K1M = NEQN + 1
      K1 = K1M + 1
      K2 = K1 + NEQN
      K3 = K2 + NEQN
      K4 = K3 + NEQN
      K5 = K4 + NEQN
C
C     PARTITION FOR DIMENSIONING ADDITIONAL F VECTORS
C
      K6 = K5 + NEQN
      K7 = K6 + NEQN
      K8 = K7 + NEQN
      K9 = K8 + NEQN
C
C     PARTITION FOR RETURNING TF (FOR TRAPPING OPTIONS (IFLAG < -2))
C
      KTF = K9 + NEQN
C
C     PARTITION FOR DIMENSIONING ADDITIONAL Y VECTORS
C
      K10 = KTF + 1
      K11 = K10 + NEQN
      K12 = K11 + NEQN
      K13 = K12 + NEQN
C
C     PARTION FOR DIMENSIONING PHI0(K)-S
C
      K14 = K13 + NEQN
      K15 = K14 + NPHI
      K16 = K15 + NPHI
      K17 = K16 + NPHI
      K18 = K17 + NPHI
      K19 = K18 + NPHI
      K20 = K19 + NPHI
      K21 = K20 + NPHI
      K22 = K21 + NPHI
      K23 = K22 + NPHI
      K24 = K23 + NPHI
      K25 = K24 + NPHI
      K26 = K25 + NPHI
C
C     PARTION FOR SAVRE,SAVAE, TZERO
C
      KSAVRE  = K26 + NPHI
      KSAVAE  = KSAVRE + 1
      KTZERO  = KSAVAE + 1
C
C-----------------------------------------------------------------------
C     THIS INTERFACING ROUTINE MERELY RELIEVES THE USER OF A LONG
C     CALLING LIST VIA THE SPLITTING APART OF TWO WORKING STORAGE
```

```
C     ARRAYS.  IF THIS IS NOT COMPATIBLE WITH THE USERS COMPILER,
C     SHE MUST USE RKF45T DIRECTLY.
C-----------------------------------------------------------------------
C
C
      CALL RKFST(F,SUBPHI,NPHI,NEQN,Y,T,TOUT,RELERR,ABSERR,IFLAG,
     1           WORK(1),WORK(K1M),WORK(K1),WORK(K2),WORK(K3),
     2           WORK(K4),WORK(K5),WORK(K6),WORK(K7),WORK(K8),
     3           WORK(K9),WORK(KTF),WORK(K10),WORK(K11),WORK(K12),
     4           WORK(K13),WORK(K14),WORK(K15),WORK(K16),WORK(K17),
     5           WORK(K18),WORK(K19),WORK(K20),WORK(K21),WORK(K22),
     6           WORK(K23),WORK(K24),WORK(K25),WORK(K26),
     7           WORK(KSAVRE),WORK(KSAVAE),WORK(KTZERO),
     8           IWORK(1),IWORK(2),IWORK(3),IWORK(4),IWORK(5),
     9           IWORK(6),IWORK(7),IWORK(8),IWORK(9),IWORK(10))
C
      IF (FLGOUT)  CALL OUTFLG(IFLAG)
C
C     ADJUST IFLAG IF TRAPPED OPTION HAS BEEN USED
C
      INDIC = 1
      CALL FLAGCK(IFLAG,INDIC,IWORK(8),IWORK(9))
C
      RETURN
      END
C
C
      SUBROUTINE RKFST(F,SUBPHI,NPHI,NEQN,Y,T,TOUT,RELERR,ABSERR,
     1           IFLAG,YP,H,F1,F2,F3,F4,F5,F6,F7,F8,F9,TF,YF,YPF,
     2           Y2,YP2,PHI0,PHIP0,PHI2,PHIP2,PHIF,PHIPF,
     3           PHIR,PHIPR,PHIL,PHIPL,PHIB,PHIV,PHIPV,SAVRE,SAVAE,
     4           TZERO,      NFE,NREJ,NEXTRA,KOP,INIT,JFLAG,KFLAG,
     5           LFLAG,LSTART,KOUNTR)
      IMPLICIT REAL*8  (A-H,O-Z)
C     FEHLBERG FOURTH-FIFTH ORDER RUNGE-KUTTA METHOD
C
C     RKFST INTEGRATES A SYSTEM OF FIRST ORDER ORDINARY DIFFERENTIAL
C     EQUATIONS AS DESCRIBED IN THE COMMENTS FOR RKF45T.
C     THE ARRAYS YP,F1,F2,F3,F4, AND F5 (OF DIMENSION AT LEAST NEQN) AND
C     THE VARIABLES H,SAVRE,SAVAE,NFE,KOP,INIT,JFLAG,AND KFLAG ARE USED
C     INTERNALLY BY THE CODE AND APPEAR IN THE CALL LIST TO ELIMINATE
C     LOCAL RETENTION OF VARIABLES BETWEEN CALLS.  ACCORDINGLY, THEY
C     SHOULD NOT BE ALTERED.  ITEMS OF POSSIBLE INTEREST ARE
C           YP - DERIVATIVE OF SOLUTION VECTOR AT T
C           H  - AN APPROPRIATE STEP SIZE TO BE USED FOR THE NEXT STEP
C           NFE- COUNTER ON THE NUMBER OF DERIVAITVE FUNCTION EVALUATION
C
C
C-----------------------------------------------------------------------
C
      LOGICAL HFAILD,OUTPUT,UPDATE,IVAN,FIND,EVALF,BOUNCE
C
      DIMENSION Y(NEQN),YP(NEQN),F1(NEQN),F2(NEQN),F3(NEQN),F4(NEQN),
     1          F5(NEQN),Y2(NEQN),YP2(NEQN),YF(NEQN),YPF(NEQN),
     2          F6(NEQN),F7(NEQN),F8(NEQN),F9(NEQN)
      DIMENSION PHI0(NPHI),PHIP0(NPHI),PHIF(NPHI),PHIPF(NPHI)
      DIMENSION PHI2(NPHI),PHIP2(NPHI),PHIB(NPHI),PHIV(NPHI),PHIPV(NPHI)
      DIMENSION PHIR(NPHI),PHIPR(NPHI),PHIL(NPHI),PHIPL(NPHI)
```

```
C
C-----------------------------------------------------------------------
C*********************************-----------------**********************----------
      COMMON/FSTEP/ITOPH
C*********************************-----------------**********************----------
C-----------------------------------------------------------------------
C
      EXTERNAL F,SUBPHI
C
C-----------------------------------------------------------------------
C
C     THE COMPUTER UNIT ROUND OFF ERROR U IS THE SMALLEST POSITIVE VALUE
C     REPRESENTABLE IN THE MACHINE SUCH THAT 1.+U .GT. 1.
C                    VALUES TO BE USED ARE
C                U = 9.5E-7          FOR IBM 360/370
C                U = 1.5E-8          FOR UNIVAC 1108
C                U = 7.5E-9          FOR POP-10
C                U = 7.1E-15         FOR CDC 6000 SERIES
C                U = 2.2E-16         FOR IBM 360/370 DOUBLE PRECISION
      DATA U/2.2D-16/
C-----------------------------------------------------------------------
C
C     REMIN IS A TOLERANCE THRESHOLD WHICH IS ALSO DETERMINED BY THE
C     INTEGRATION METHOD.  IN PARTICULAR, A FIFTH ORDER METHOD WIL
C     GENERALLY NOT BE CAPABLE OF DELIVERING ACCURACIES NEAR
C     PRECISION ON COMPUTERS WITH LONG WORDLENGTHS.
C
      DATA REMIN/1.D-12/
C-----------------------------------------------------------------------
      DATA MAXNFE/3000/
C
C     PROTECTION FOR IFIRST PARAMETER
      IFIRST = 2
      LFLAGS = LFLAG
C*********************************-----------------**********************----------
      ITOPH = 0
C*********************************-----------------**********************----------
C
C
C     CHECK INPUT PARAMETERS
C
      IF (NEQN.LT.1)  GO TO 10
      IF ((RELERR.LT.0.0D0)  .  OR.  (ABSERR.LT. 0.0D0))  GO TO 10
      MFLAG = IABS(IFLAG)
      IF ((MFLAG . GE. 1)  .AND.  (MFLAG .LE. 8)) GO TO 20
C
C     INVALID INPUT
C
   10 IFLAG = 8
      RETURN
C
C     IF THIS THE FIRST CALL
   20 IF (MFLAG .EQ. 1)  GO TO 50
C
C     CHECK CONTINUATION POSSIBILITIES
      IF ((T .EQ. TOUT)  .AND.  (KFLAG .NE. 3)) GO TO 10
      IF (MFLAG .NE. 2)  GO TO 25
C
```

```
C     IFLAG = +2 OR -2
      IF (KFLAG .EQ. 3)  GO TO 45
      IF (INIT.EQ.0)  GO TO 45
      IF (KFLAG .EQ.4)  GO TO 40
      IF ((KFLAG .EQ. 5)  .AND.  (ABSERR.EQ. 0.0D0)) GO TO 30
      IF ((KFLAG .EQ. 6)  .AND.  (RELERR .LE. SAVRE)  . AND.
     1     (ABSERR .LE. SAVAE))  GO TO 30
      GO TO 50
C
C     IFLAG = 3,4,5,6,7, OR 8
   25 IF (IFLAG .EQ. 3)  GO TO 45
      IF (IFLAG .EQ. 4)  GO TO 40
      IF ((IFLAG .EQ. 5) .AND. (ABSERR .GT. 0.0D0))  GO TO 45
C
C     INTEGRATION CANNOT BE CONTINUED SINCE USER DID NOT RESPOND TO
C     THE INSTRUCTIONS PERTAINING TO IFLAG=5,6,7, OR 8
C
   30 CONTINUE
C
      STOP
C
C-----------------------------------------------------------------------
C
C     RESET FUNCTION EVALUATION COUNTER
   40 NFE=0
      IF (MFLAG .EQ. 2)  GO TO 50
C
C     RESET FLAG VALUE FORM PREVIOUS CALL
   45 IFLAG = JFLAG
      IF (KFLAG .EQ. 3)  MFLAG=IABS(IFLAG)
C
C     SAVE INPUT IFLAG AND SET CONTINUATION FLAG FOR SUBSEQUENT
C     INPUT CHECKING
   50 JFLAG = IFLAG
      KFLAG = 0
C
C     SAVE RELERR AND ABSERR FOR CHECKING INPUT ON SUBSEQUENT CALLS
      SAVRE = RELERR
      SAVAE = ABSERR
C
C     RESTRICT RELATIVE ERROR TOLERANCE TO BE AT LEAST AS LARGE AS
C     2U+REMIN TO AVOID LIMITING PRECISION DIFFICULTIES ARRISING FROM
C     IMPOSSIBLE ACCURACY REQUESTS
C
      RER = 2.0D0*U + REMIN
      IF (RELERR.GE.RER)  GO TO 55
C
C     RELATIVE ERROR TOLERANCE TOO SMALL
      RELERR = RER
      IFLAG = 3
      KFLAG = 3
      RETURN
C
   55 U26 = 26.D0*U
      DT = TOUT - T
      IF (MFLAG .EQ. 1)  GO TO 60
      IF (INIT .EQ. 0)  GO TO 65
      IF (LSTART .EQ. 1)  GO TO 62
```

```
      GO TO 80
C
C
C-----------------------------------------------------------------------
C
C                     SET INITIALIZATION COMPLETION INDICATOR,INIT
C                     SET INDICATOR FOR TOO MANY OUTPUT POINTS,KOP
C                     EVALUATE INITIAL DERIVATIVES
C                     SET COUNTER FOR FUNCTION EVALUATIONS,NFE
C                     ESTIMATE STARTING STEP SIZE
C                     SET PARAMETERS FOR ACTIVATING CONSTRAINT
C                       FUNCTION IF THIS OPTION IS BEING USED
C
   60 INIT = 0
      KOP = 0
C
      A = T
      CALL F(A,Y,YP)
      NFE = 1
   62 CONTINUE
C
      NEXTRA = 0
      NREJ = 0
      IVAN = .FALSE.
      JUSTR = 0
      IF (LFLAG .EQ. 0)  GO TO 64
C
C-----------------------------------------------------------------------
C     INITIALIZATION FOR TRAPPING OPTION
C-----------------------------------------------------------------------
C
      ABSER = 0.5D0 * (ABSERR + RELERR)
      KOUNTR = 0
C
C
      UPDATE = .FALSE.
      IVAN   = .FALSE.
      BOUNCE = .FALSE.
      INDEX = 0
C
      CALL SUBPHI(NPHI,INDEX,NEQN,A,Y,YP,PHI0,PHIP0,KOUNTR,UPDATE,
     1          IVAN,BOUNCE,ABSER)
C
      MPART = NPHI
      IF (LFLAG .EQ. 2 .AND. INDEX .GT. 0 .AND. INDEX .LT. NPHI)
     1                       MPART = INDEX
      INDEX = 0
      IF (ABSER .LT. RER)  ABSER = RER
C
      DO 63 J = 1,NPHI
      IF (DABS(PHI0(J)) .LT. U26)  PHI0(J) = DSIGN(U26,PHI0(J))
   63 CONTINUE
      KOUNTR = 1
      IFIRST = 1
      IF (LSTART .EQ. 1)  GO TO 80
C-----------------------------------------------------------------------
   64 CONTINUE
      IF (T .NE. TOUT)  GO TO 65
```

```
      IFLAG = 2
      RETURN
C
C
   65 INIT = 1
      H = DABS(DT)
      TOLN = 0.0D0
      DO 70 K = 1,NEQN
      TOL = RELERR*DABS(Y(K)) + ABSERR
      IF (TOL .LE. 0.D0)  GO TO 70
      TOLN = TOL
      YPK = DABS(YP(K))
      IF (YPK*H**5 .GT. TOL )  H = (TOL/YPK)**0.2D0
   70 CONTINUE
      IF (TOLN .LE. 0.D0)  H = 0.D0
      H = DMAX1(H,U26*DMAX1(DABS(T),DABS(DT)))
C
C-----------------------------------------------------------------------
C
C     INSERT NEW BLOCK FOR INITIAL STEP SIZE ESTIMATE--HSTART
C
C     PRINT 1588,H
      ETOL = 0.5D0*(ABSERR+RELERR)
      BIG = DSQRT(1.0D+10)
      CALL HSTART(F,NEQN,T,TOUT,Y,YP,ETOL,5,
     1            U,BIG,F1,F2,F3,F4,DUM1,NFE   ,H)
C
C-----------------------------------------------------------------------
C
      JFLAG = ISIGN(2,IFLAG)
C
C
C-----------------------------------------------------------------------
C
C     SET STEP SIZE FOR INTEGRATION IN THE DIRECTION FROM T TO TOUT
C
   80 H = DSIGN(H,DT)
C
C     TEST TO SEE IF RKF45 IS BEING SEVERELY IMPACTED BY TOO MANY
C     OUTPUT POINTS
C
      IF (DABS(H) .GE. 2.D0*DABS(DT)) KOP = KOP + 1
      IF (KOP .NE. 100)  GO TO 85
C
C     UNNECESSARY FREQUENCY OF OUTPUT
C
      KOP = 0
      IFLAG = 7
      RETURN
C
C
   85 IF (DABS(DT) .GT. U26*DABS(T)) GO TO 95
C
C     IF TOO CLOSE TO OUTPUT POINT, EXTRAPOLATE AND RETURN
C
      DO 90 K = 1,NEQN
   90 Y(K) = Y(K) + DT*YP(K)
      A = TOUT
```

```
      CALL F(A,Y,YP)
      NFE = NFE + 1
C
C*****************************************************************************
      PRINT 1600,T
 1600 FORMAT(///,' ********* USING EXTRAPOLATED SOLUTION AT T = ',D15.7,
     1           ' *********',//)
C*****************************************************************************
      GO TO 300
C
C
C     INITIALIZE OUTPUT INDICATOR
C
   95 OUTPUT = .FALSE.
C
C     TO AVOID PREMATURE UNDERFLOW IN THE TOLERANCE FUNCTION,
C     SCALE THE ERROR TOLERANCES
C
      SCALE = 2.D0/RELERR
      AE = SCALE*ABSERR
C
C
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
C
C     STEP BY STEP INTEGRATION
C
  100 HFAILD = .FALSE.
C
C     SET SMALLEST ALLOWABLE STEPSIZE
C
      HMIN = U26*DABS(T)
C
C     ADJUST STEPSIZE IF NECESSARY TO HIT THE OUTPUT POINT.
C     LOOK AHEAD TWO STEPS TO AVOID DRASTIC CHANGES IN THE STEPSIZE AND
C     THUS LESSEN THE IMPACT OF OUTPUT POINTS ON THE CODE.
C
      DT = TOUT - T
      IF (DABS(DT) .GE. 2.D0*DABS(H)) GO TO 200
      IF (DABS(DT) .GT. DABS(H)/0.9D0) GO TO 150
C
C     THE NEXT SUCCESSFUL STEP WILL COMPLETE THE INTEGRATION TO THE
C     OUTPUT POINT
C
      OUTPUT = .TRUE.
      H = DT
      GO TO 200
C
  150 H = 0.5D0*DT
C
C
C
C-----------------------------------------------------------------------
C     CORE INTEGRATOR FOR TAKING A SINGLE STEP
C-----------------------------------------------------------------------
C     THE TOLERANCES HAVE BEEN SCALED TO AVOID PREMATURE UNDERFLOW IN
C     COMPUTING THE ERROR TOLERANCE FUNCTION ET.
C     TO AVOID PROBLEMS WITH ZERO CROSSINGS, RELATIVE ERROR IS MEASURED
```

```
C     USING THE AVERAGE OF THE MAGNITUDES OF THE SOLUTION AT THE
C     BEGINNING AND END OF A STEP.
C     THE ERROR ESTIMATE FORMULA HAS BEEN GROUPED TO CONTROL LOSS OF
C     SIGNIFICANCE
C     TO DISTINGUISH THE VARIOUS ARGUMENTS, H IS NOT PERMITTED
C     TO BECOME SMALLER THAN 26 UNITS OF ROUND OFF IN T.
C     PRACTICAL LIMITS ON THE CHANGE IN THE STEP SIZE ARE ENFORCED TO
C     SMOOTH THE STEP SIZE SELECTION PROCESS AND TO AVOID EXCESSIVE
C     CHATTERING ON PROBLEMS HAVING DISCONTINUITIES.
C     TO PREVENT UNNECESSARY FAILURES, THE CODE USES 9/10 THE STEP SIZE
C     IT ESTIMATES WILL SUCCEED.
C     SINCE LOCAL EXTRAPOLATION IS BEING USED AND EXTRA CAUTION SEEMS
C     WARRANTED.
C-----------------------------------------------------------------------
C
C
C     TEST NUMBER OF DERIVATIVE FUNCTION EVALUATIONS
C     IF OKAY, TRY TO ADVANCE THE INTEGRATION FROM T TO T + H
C
  200 IF (NFE .LE. MAXNFE)  GO TO 220
C
C     TOO MUCH WORK
      IFLAG = 4
      KFLAG = 4
      RETURN
C
C     ADVANCE AN APPROXIMATE SOLUTION OVER ONE STEP OF LENGTH H
C
  220 CALL FEHL(F,NEQN,Y,T,H,YP,F1,F2,F3,F4,F5,F1)
      NFE = NFE + 5
C
C
C     COMPUTE AND TEST ALLOWABLE TOLERANCES VERSUS LOCAL ERROR ESTIMATES
C     AND REMOVE SCALING OF TOLERANCES.  NOTE THAT RELATIVE ERROR IS
C     MEASURED WITH RESPECT TO THE AVERAGE OF THE MAGNITUDES OF THE
C     SOLUTION AT THE BEGINNING AND END OF THE STEP.
C
      EEOET = 0.0D0
      DO 225 K = 1,NEQN
      ET = DABS(Y(K)) + DABS(F1(K)) + AE
      IF (ET.GT.0.D0)  GO TO 224
C
C     INAPPROPRIATE ERROR TOLERANCE
      IFLAG = 5
      KFLAG = 5
      RETURN
C
  224 EE = DABS((-2090.D0*YP(K)+(21970.D0*F3(K)-15048.D0*F4(K)))+
     1                       (22528.D0*F2(K)-27360.D0*F5(K)))
  225 EEOET = DMAX1(EEOET,EE/ET)
C
      ESTTOL = DABS(H)*EEOET*SCALE/752400.D0
C
      IF (ESTTOL .LE. 1.D0)  GO TO 230
C
C
C     UNSUCESSFUL STEP
C                       REDUCE THE STEP SIZE, TRY AGAIN
```

```
C                       THE DECREASE IS LIMITED TO A FACTOR OF 1/10
C
      HFAILD = .TRUE.
      OUTPUT = .FALSE.
      S = 0.1D0
      IF (ESTTOL .LT. 59049.D0)  S = 0.9D0/ESTTOL**0.2D0
C
C
      NREJ = NREJ + 1
      H = S*H
      IF (DABS(H) .GT. HMIN) GO TO 200
C
C     REQUESTED ERROR UNATTAINABLE AT SMALLEST ALLOWABLE STEP SIZE
C
      IFLAG = 6
      KFLAG = 6
      RETURN
C
C     SUCCESSFUL STEP
C                       STORE SOLUTION AT T + H
C                       AND EVALUATE DERIVATIVES THERE
C
  230 CONTINUE
      IF (LFLAG .EQ. 0)  GO TO 269
C-----------------------------------------------------------------------
C     CALL SETRAP TO SET UP AND CHECK CONDITIONS FOR REFERENCING TRAPPD
C-----------------------------------------------------------------------
      CALL       SETRAP(F,SUBPHI,NPHI,NEQN,TOUT,H,ABSER,TZERO,
     1                  T,Y,YP,PHI0,PHIP0,   TF,YF,YPF, PHIF,PHIPF,
     2                  T2,Y2,YP2,PHI2,PHIP2,
     2                  PHIL,PHIPL,PHIR,PHIPR,PHIB,PHIV,PHIPV,
     3                  F1,F2,F3,F4,F5,F6,F7,F8,F9,
     4                  REMIN,U26,
     5                  IFLAG,LFLAG,JUSTR,KOUNTR,IFIRST,NFE,NEXTRA,
     6                  MPART,
     7                  OUTPUT,UPDATE,BOUNCE,IVAN,FIND,HFAILD)
C
      IF (IFLAG .GT. 2)   RETURN
      GO TO 275
C-----------------------------------------------------------------------
  269 CONTINUE
      T = T + H
      DO 270 K = 1,NEQN
  270 Y(K) = F1(K)
      A = T
C*************************----------------*************************----------
      ITOPH = 1
C*************************----------------*************************----------
      CALL F(A,Y,YP)
C*************************----------------*************************----------
      ITOPH = 0
C*************************----------------*************************----------
      NFE = NFE + 1
C
C
C                       CHOOSE NEXT STEP SIZE
C                       THE INCREASE IS LIMITED TO A FACTOR OF 5
C                       IF STEP FAILURE HAS JUST OCCURRED, NEXT
```

```
C                          STEP SIZE IS NOT ALLOWED TO INCREASE
C
  275 CONTINUE
      S = 5.0D0
C*********************************************************************************************
      HSAVE =H
C*********************************************************************************************
      IF (ESTTOL .GT. 1.889568D-4)  S = 0.9D0/ESTTOL**0.2D0
      IF (HFAILD) S = DMIN1(S,1.D0)
C
      H = DSIGN(DMAX1(S*DABS(H),HMIN),H)
C
C-----------------------------------------------------------------------
C     END OF CORE INTEGRATOR
C
C
C
C     SHOULD WE TAKE ANOTHER STEP
C
      IF (OUTPUT) GO TO 300
      IFLAG = ISIGN(2,IFLAG)
C
C     IF LFLAG=2 AND JUSTR=1, TRAPPD HAS JUST ISOLATED A ZERO OF PHI
C        AND THE USER SELECTED THE OPTION TO RETURN (IFLAG=-15,-25)
C
      IF (LFLAG .EQ. -1)  GO TO 280
      IF (LFLAG .EQ. -2 .AND. JUSTR .EQ. 0)  GO TO 100
      IF (IFLAG .GT. 0)   GO TO 100
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
C
  280 CONTINUE
C
C     INTEGRATION SUCCESSFULLY COMPLETED
C
C     ONE-STEP MODE
      IFLAG = -2
      RETURN
C
C     INTERVAL MODE
  300 T = TOUT
      IFLAG = 2
      RETURN
C
      END
      SUBROUTINE FEHL(F,NEQN,Y,T,H,YP,F1,F2,F3,F4,F5,S)
C
C     FEHLBERG FOURTH-FIFTH ORDER RUNGE-KUTTA METHOD
C
C-----------------------------------------------------------------------
C     FEHL INTEGRATES A SYSTEM OF NEQN FIRST ORDER
C     ORDINARY DIFFERENTIAL EQUATIONS OF THE FORM
C              DY(I)/DT = F(T,Y(1),---,Y(NEQN))
C     WHERE THE INITIAL VALUES Y(I) AND THE INITIAL DERIVATIVES
C     YP(I) ARE SPECIFIED AT THE STARTING POINT T.  FEHL ADVANCES
C     THE SOLUTION OVER THE FIXED STEP H AND RETURNS
C     THE FIFTH ORDER (SIXTH ORDER ACCURATE LOCALLY) SOLUTION
C     APPROXIMATION AT T+H IN ARRAY S(I)
```

```
C
C     F1,---F5 ARE ARRAYS OF DIMENSION NEQN WHICH ARE NEEDED
C     FOR INTERNAL STORAGE
C     THE FORMULAS HAVE BEEN GROUPED TO CONTROL LOSS OF SIGNIFICANCE.
C     FEHL SHOULD BE CALLED WITH AN H NOT SMALLER THAN 13 UNITS OF
C     ROUNDOFF IN T SO THAT THE VARIOUS INDEPENDENT ARGUMENTS CAN BE
C     DISTINGUISHED.
C-----------------------------------------------------------------------
C
      IMPLICIT REAL*8 (A-H,O-Z)
C
      COMMON/SAFETY/IPR
      DIMENSION Y(NEQN),YP(NEQN),F1(NEQN),F2(NEQN),F3(NEQN),F4(NEQN),
     1          F5(NEQN),S(NEQN)
C
      IPR = 0
      CH = H/4.D0
      DO 221 K = 1,NEQN
  221 F5(K) = Y(K) + CH*YP(K)
      CALL F(T+CH,F5,F1)
C
      CH = 3.D0*H/32.D0
      DO 222 K = 1,NEQN
  222 F5(K) = Y(K) + CH*(YP(K) + 3.D0*F1(K))
      CALL F(T+3.D0*H/8.D0,F5,F2)
C
      CH = H/2197.D0
      DO 223 K = 1,NEQN
  223 F5(K) = Y(K) + CH*(1932.D0*YP(K) + (7296.D0*F2(K)
     1         - 7200.0D0*F1(K)))
      CALL F(T+12.D0*H/13.D0,F5,F3)
C
      CH = H/4104.D0
      DO 224 K = 1,NEQN
  224 F5(K) = Y(K) + CH*((8341.D0*YP(K)-845.D0*F3(K))+
     1                          (29440.D0*F2(K)-32832.D0*F1(K)))
      CALL F(T+H,F5,F4)
C
      CH = H/20520.D0
      DO 225 K = 1,NEQN
  225 F1(K) = Y(K)+CH*((-6080.D0*YP(K) + (9295.D0*F3(K)-5643.D0*F4(K)))
     1                       +  (41040.D0*F1(K)-28352.D0*F2(K)))
      CALL F(T+H/2.D0,F1,F5)
C
C     COMPUTE APPROXIMATE SOLUTION AT T + H
C
      CH = H/7618050.D0
      DO 230 K = 1,NEQN
  230 S(K) = Y(K) +CH*((902880.D0*YP(K) + (3855735.D0*F3(K)
     1     -1371249.D0*F4(K))) + (3953664.D0*F2(K) + 277020.D0*F5(K)))
      IPR = 1
C
      RETURN
      END
C
C-----------------------------------------------------------------------
      SUBROUTINE SETRAP(F,SUBPHI,NPHI,NEQN,TOUT,H,ABSER,TZERO,
     1                  T,Y,YP,PHI0,PHIP0,   TF,YF,YPF, PHIF,PHIPF,
```

```
     2                  T2,Y2,YP2,PHI2,PHIP2,
     2                  PHIL,PHIPL,PHIR,PHIPR,PHIB,PHIV,PHIPV,
     3                  F1,F2,F3,F4,F5,F6,F7,F8,F9,
     4                  REMIN,U26,
     5                  IFLAG,LFLAG,JUSTR,KOUNTR,IFIRST,NFE,NEXTRA,
     6                  MPART,
     7                  OUTPUT,UPDATE,BOUNCE,IVAN,FIND,HFAILD)
C-----------------------------------------------------------------------
      IMPLICIT REAL*8 (A-H,O-Z)
C
      DIMENSION Y(NEQN), YP(NEQN),YF(NEQN),YPF(NEQN),Y2(NEQN),YP2(NEQN)
      DIMENSION F1(NEQN),F2(NEQN),F3(NEQN),F4(NEQN), F5(NEQN), F6(NEQN)
      DIMENSION F7(NEQN),F8(NEQN),F9(NEQN)
      DIMENSION  PHI0(NPHI), PHIF(NPHI), PHI2(NPHI)
      DIMENSION  PHIL(NPHI), PHIR(NPHI), PHIB(NPHI), PHIV(NPHI)
      DIMENSION PHIP0(NPHI),PHIPF(NPHI),PHIP2(NPHI)
      DIMENSION PHIPL(NPHI),PHIPR(NPHI),            PHIPV(NPHI)
      LOGICAL   OUTPUT,UPDATE,BOUNCE,IVAN,NOTFAL,FIND,HFAILD,EVALF
      DATA MODE0/0/,MODE2/2/
C-----------------------------------------------------------------------
      DATA NOTFAL/.FALSE./
C-----------------------------------------------------------------------
      COMMON/FSTEP/ITOPH
C-----------------------------------------------------------------------
      EXTERNAL F,SUBPHI
C-----------------------------------------------------------------------
C     ANALYZE CONDITIONS FOR SUBROUTINE TRAPPD
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
C
      JUSTR = 0
      TF = T + H
      IF (OUTPUT)   TF = TOUT
C
      DO 235 J = 1,NEQN
  235 YF(J) = F1(J)
C*****************************----------*************************----------
      ITOPH = 1
C*****************************----------*************************----------
      CALL F(TF,YF,YPF)
C*****************************----------*************************----------
      ITOPH = 0
C*****************************----------*************************----------
      NFE = NFE + 1
C
      UPDATE = .FALSE.
      BOUNCE = .FALSE.
      IVAN   = .FALSE.
      INDEX = 0
      TOLER = ABSER
      CALL SUBPHI(NPHI,INDEX,NEQN,TF,YF,YPF,PHIF,PHIPF,KOUNTR,UPDATE,
     1          IVAN,BOUNCE,TOLER)
C
      DO 200 J = 1,NPHI
      IF (DABS(PHIF(J))  .LT.   U26)  PHIF(J)  = DSIGN(U26,PHIF(J))
      IF (DABS(PHIPF(J)) .LT.   U26)  PHIPF(J) = DSIGN(U26,PHIPF(J))
  200 CONTINUE
      INDEX = 0
```

```
      EVALF = .FALSE.
C
      IF (IFIRST .EQ. 2)  GO TO 258
C-----------------------------------------------------------------------
C     FIRST STEP (IFIRST = 1).
C     SEE IF A PHI COMPONENT VANISHED AT THE INITIAL  T  VALUE.
C-----------------------------------------------------------------------
C
C     IF THE EMERGENCY FEATURE (NOTFAL = .TRUE.) IS USED, THE INTEGRA-
C     WILL BE RESTARTED FROM EACH TRAPPED POINT EVEN IF THE MULTPLE
C     TRAP OPTION IS DESIGNATED BY THE USER
C
      IF (NOTFAL .AND. LFLAG .EQ. 2)  LFLAG = 1
C
      ITEST = 0
      DO 245 I = 1,NPHI
      PHIV(I) = -1.0D0
      IF (DABS(PHI0(I)) .LT. ABSER)  ITEST = 1
  245 CONTINUE
C
C     TZERO, THE MOST RECENT VANISHING POINT, NEEDS A DUMMY VALUE AT
C     THE BEGINNING OF THE INTEGRATION.  SET TZERO EQUAL TO A POINT
C     IN THE OPPOSITE DIRECTION OF THE INTEGRATION.
C
      TZERO = T - H
C
      IF (ITEST .EQ. 0)  GO TO 258
C
C-----------------------------------------------------------------------
C     A PHI COMPONENT VANISHED AT THE INITIAL T VALUE.
C     STUDY THE COMPONENT FOR POSSIBLE SIGN ERROR.
C-----------------------------------------------------------------------
C
      T2 = TF
C
C     CALL PANIC TO ESTABLISH SUB-MESH STEP SIZE IF NOTFAL
C     (EMERGENCY OPTION IS BEING USED)
C
      IF (NOTFAL)  CALL PANIC(F,SUBPHI,NPHI,NEQN,NFE,INDEX,T,Y,YP,
     1                        PHI0,PHIP0,TF,YF,YPF,PHIF,PHIPF,
     2                        T2,Y2,YP2,PHI2,PHIP2,PHIL,PHIPL,
     3                        PHIR,PHIPR,TL,TR,PHIB,
     4                        F1,F2,F3,F4,F5,F6,F7,F8,F9,
     5                        ABSER,KOUNTR,NEXTRA,EVALF,FIND,
     6                        MODE0,IFIRST,U26)
C
      IND = 0
      CALL VANISH(F,SUBPHI,NPHI,NEQN,NFE,IND,T,Y,YP,
     1            T,T2,TF,PHI0,PHIP0,PHI2,PHIF,PHIPF,PHIR,PHIPR,
     2            PHIV,PHIPV,F1,F2,F3,F4,F5,F6,F7,F8,F9,ABSER,KOUNTR,
     3            EVALF,IVAN,NEXTRA,U26,MODE2)
C
      DO 256 I = 1,NPHI
      IF (DABS(PHI0(I)) .GT. ABSER)   GO TO 255
C
C
      IF (DABS(PHI0(I)) .LT. U26)   PHI0(I) = U26
      PHI0(I) = DSIGN(PHI0(I),PHIV(I))
```

```
      PHIV(I) = +1.0D0
C     UPDATE COMPONENT WHICH VANISHED AT INITIAL CONDITIONS
      KOUNTR = KOUNTR + 1
      IND = I
      UPDATE = .TRUE.
      BOUNCE = .FALSE.
      IVAN = .FALSE.
      TOLER = ABSER
C
      CALL SUBPHI(NPHI,IND,NEQN,T,Y,YP,PHI0,PHIP0,KOUNTR,UPDATE,
     1            IVAN,BOUNCE,TOLER)
      GO TO 256
  255 CONTINUE
      PHIV(I) = -1.0D0
  256 CONTINUE
C
      DO 257 J = 1,NPHI
      IF (DABS(PHI0(J))  .LT.   U26)  PHI0(J)  = DSIGN(U26,PHI0(J))
      IF (DABS(PHIP0(J)) .LT.   U26)  PHIP0(J) = DSIGN(U26,PHIP0(J))
  257 CONTINUE
      TZERO = T
      I=2
C-----------------------------------------------------------------------
C
  258 CONTINUE
C
      IFIRST = 2
C
C     SET CONDITIONS AT "L" EQUAL TO THOSE AT "0"
C     SET CONDITIONS AT "R" EQUAL TO THOSE AT "F"
C
      DO 259 J = 1,NEQN
      Y2(J) = Y(J)
      YP2(J) = YP(J)
  259 CONTINUE
C
      DO 260 J = 1,NPHI
      PHIL(J)  = PHI0(J)
      PHI2(J)  = PHI0(J)
      PHIR(J)  = PHIF(J)
      PHIPL(J) = PHIP0(J)
      PHIP2(J) = PHIP0(J)
      PHIPR(J) = PHIPF(J)
  260 CONTINUE
      T2 = T
      TL = T
      TR = TF
C
      FIND = .FALSE.
      IF (NOTFAL)  CALL PANIC(F,SUBPHI,NPHI,NEQN,NFE,INDEX,T,Y,YP,
     1                        PHI0,PHIP0,TF,YF,YPF,PHIF,PHIPF,
     2                        T2,Y2,YP2,PHI2,PHIP2,PHIL,PHIPL,
     3                        PHIR,PHIPR,TL,TR,PHIB,
     4                        F1,F2,F3,F4,F5,F6,F7,F8,F9,
     5                        ABSER,KOUNTR,NEXTRA,EVALF,FIND,
     6                        MODE2,IFIRST,U26)
C
C
```

```
      IF (NOTFAL .AND. .NOT.FIND)  GO TO 266
      IF (FIND)  HFAILD = .TRUE.
C
C     IF THE EMERGENCY FEATURE DETECTED A ZERO OVER A SUBSTEP,
C     CONDITIONS AT "L" AND "R" ARE RESET IN PANIC.  POINT "2"
C     IS SET EQUAL TO "L" (INCLUDING Y2,YP2)
C
      PHIMAX = 0.0D0
      DO 262 J = 1,NPHI
C
      IF (DABS(PHIR(J)) .LT.  ABSER)  GO TO 261
      IF (PHIL(J)*PHIR(J) .GT. 0.0D0)  GO TO 262
  261 CONTINUE
C
C     THE JTH COMPONENT OF PHI EXPERIENCES A SIGN CHANGE OR PHIF(J)
C     VANISHES (PHI0(J) WILL NEVER BE IDENTICALLY ZERO AT THIS POINT
C     BECAUSE TRAPPD SETS ZERO VALUES OF PHI EQUAL TO UNIT ROUNDOFF
C     WITH APPROPRIATE SIGN)
C
      IF (DABS(PHIR(J)) .LT. PHIMAX) GO TO 262
C
      INDEX = J
      PHIMAX = DABS(PHIR(J))
C
  262 CONTINUE
C
      IF (INDEX .EQ. 0)  GO TO 266
C
C-----------------------------------------------------------------------
C
C     A COMPONENT OF PHI VANISHED AT T  OR BETWEEN T  AND T
C                                    F             0      F
C-----------------------------------------------------------------------
C
      CALL TRAPPD(F,SUBPHI,NPHI,NEQN,NFE,INDEX,IFLAG,T,Y,YP,
     1     PHI0,PHIP0,TF,YF,YPF,PHIF,PHIPF,T2,Y2,YP2,
     2     PHI2,PHIP2,PHIR,PHIPR,PHIL,PHIPL,TL,TR,PHIB,PHIV,PHIPV,
     3     F1,F2,F3,F4,F5,F6,F7,F8,F9,ABSER,TZERO,
     4     KOUNTR,MPART,UPDATE,OUTPUT,FIND,EVALF,
     5     REMIN,U26,LFLAG,NEXTRA)
C
      JUSTR = 1
      IF (IABS(IFLAG) .GT. 2)  RETURN
C
C     CONDITIONS AT THE END OF THE STEP HAVE BEEN SET IN TRAPPD
C
      RETURN
C-----------------------------------------------------------------------
  266 CONTINUE
C
C-----------------------------------------------------------------------
C     NO COMPONENT OF PHI HAS CHANGED SIGN
C-----------------------------------------------------------------------
C
      IVAN = .FALSE.
      DO 267 J = 1,NEQN
      Y(J) = YF(J)
  267 YP(J) = YPF(J)
```

```
      T = TF
      DO 268 J = 1,NPHI
      PHIV(J) = -1.0D0
      PHI0(J) = PHIF(J)
  268 PHIP0(J) = PHIPF(J)
C
      RETURN
      END
C-----------------------------------------------------------------------
      SUBROUTINE FLAGCK(IFLAG,INDIC,LFLAG,LSTART)
C-----------------------------------------------------------------------
C
      IF (INDIC .EQ. 1)  GO TO 50
C
C     ADJUSTMENTS TO IFLAG IF TRAPPD ROUTINE IS TO BE USED
C
      MFLAG = IABS(IFLAG)
      LSTART = 0
C
C
      IF (MFLAG .LE. 25)  GO TO 10
C
C     ERROR IN TRAPPING PHI ON PREVIOUS STEP
      PRINT 1985,IFLAG,LFLAG
 1985 FORMAT(//,' TRAPPING ERROR ON PREVIOUS STEP OR USER INPUT ERROR',
     1       /,' IFLAG  = ',I3,4X,/,' LFLAG = ',I3,/)
      RETURN
C
   10 CONTINUE
C
      IF (MFLAG .GE. 3 .AND. MFLAG .LE. 8)  RETURN
C
      LFLAG = 0
      IF (MFLAG .LE. 2)  RETURN
C
C     TRAPPING OPTION TO BE USED.  NO PROBLEMS WITH PREVIOUS IFLAGS.
C
      ISGN = ISIGN(1,IFLAG)
      IF (MFLAG .EQ. 11 .OR. MFLAG .EQ. 16)  GO TO 12
C
      GO TO 14
   12 CONTINUE
C
      LSTART = 1
      MFLAG = MFLAG - 1
      IFLAG = IFLAG - ISGN
   14 CONTINUE
C
      II = MFLAG - MFLAG/10*10
C
      IF (II .EQ. 0)  LFLAG = ISGN
      IF (II .EQ. 5)  LFLAG = ISGN*2
      IF (LFLAG .EQ. 0)   GO TO 30
C
C
C     MUST CHANGE IFLAG TO +1,2 OR -1,-2  TO CALL RKFST, IFLAG WILL
C     BE RESET BEFORE RETURN TO USER UNLESS ERROR OCCURS
C
```

```
      IFLAG =  IFLAG / 10
C
      RETURN
C
   30 CONTINUE
C
C     INACCEPTABLE VALUE OF IFLAG
      STOP
C
   50 CONTINUE
C
C     ADJUST IFLAG IF TRAPPED OPTION HAS BEEN USED
C     IF AN ERROR HAS OCCURRED, THE TRAPPING OPTION WILL BE
C     SWITCHED OFF TEMPORARILY AND IFLAG WILL INDICATE THE PROBLEM
C
C     LFLAG WILL STORE INFORMATION ON TRAPPING OPTION PREVIOUSLY
C     USED.
C
      IF (IFLAG .GT. 2)  RETURN
C
      IF (LFLAG .EQ. 0)  RETURN
      IF (IFLAG .EQ. 2 .AND. LFLAG .LT. 0)  RETURN
C
C     TRAPPING OPTION WAS USED SUCCESSFULLY--RESET IFLAG
C
      ISGN = ISIGN(1,IFLAG)
      IFLAG = ISGN*15 + 5*LFLAG
C
      RETURN
      END
C
C     OUTFLG PRINTS WARNING MESSAGES IF IFLAG INDICATES THAT
C            RKF45T HAS ENCOUNTERED DIFFICULTIES
C
      SUBROUTINE OUTFLG(IFLAG)
C
      LOGICAL ISKIRT
      DATA ISKIRT/.TRUE./
C
C-----------------------------------------------------------------------
C
C     IFLAG = 3 OR =4 IS A MINOR WARNING TO THE USER AND CONDITIONS
C                     HAVE BEEN (OR WILL BE) RESET IN RKFST FOR
C                     CONTINUING. IF A WARNING MESSAGE IS NOT WANTED
C                     FOR THESE VALUES OF IFLAG, SET ISKIRT = .TRUE.
C
C-----------------------------------------------------------------------
C
      IF (IABS(IFLAG) .EQ. 2)  RETURN
C
      IF (ISKIRT)  GO TO 30
C
      IF (IFLAG .EQ. 3)  PRINT 1500
 1500 FORMAT(/,54H THE USER SUPPLIED, RELATIVE ERROR TOLERANCE WAS TOO S
     1        , 6HMALL.
     2       ,/,54H RELERR HAS BEEN INCREASED TO A SUITABLE VALUE FOR CON
     3        , 8HTINUING.,/,
     4         53H SIMPLY RECALL RKF45T (NO CHANGE TO IFLAG IS NEEDED).
```

```
     5         ,/)
C
      IF (IFLAG .EQ. 4)  PRINT 1501
 1501 FORMAT(/,54H LIMITING NUMBER OF DERIVATIVE EVALUATIONS HAS BEEN EX
     1         ,6HCEEDED,
     2        /,54H MAXNFE = 3000 PERMITS APPROXIMATELY 500 STEPS TO BE A
     3         ,9HTTEMPTED.,/)
C
   30 CONTINUE
C
      IF (IFLAG .EQ. 5)  PRINT 1502
 1502 FORMAT(/,54H A COMPONENT OF THE SOLUTION HAS VANISHED (AT BOTH END
     1         ,12HS OF A STEP),
     2        /,46H MAKING A PURE RELATIVE ERROR TEST IMPOSSIBLE.
     3        /,41H A NON-ZERO VALUE OF ABSERR MUST BE USED.
     4        /,48H THE STEP-BY-STEP MODE IS A GOOD WAY TO PROCEED.,/)
C
      IF (IFLAG .EQ. 6)  PRINT 1503
 1503 FORMAT(/,54H THE REQUESTED INTEGRATION ACCURACY COULD NOT BE ACHIE
     1         , 3HVED
     2       ,/,40H USING THE SMALLEST ALLOWABLE STEP SIZE.
     3       ,/,54H INTEGRATION TOLERANCES MUST BE INCREASED BEFORE THE S
     4         ,25HOLUTION CAN BE ATTEMPTED.,/)
C
      IF (IFLAG .EQ. 7)  PRINT 1504
 1504 FORMAT(/,54H TOO MUCH OUTPUT IS RESTRICTING THE NATURAL STEP SIZE
     1         , 7HCHOICE.
     2      ,//,54H THE MULTIPLE TRAP OPTION MAY BE USEFUL (IFLAG=15,25),
     3       ,/,47H WITH TOUT AS THE TEMPORARY STOPPING CONDITION.
     4       ,/,54H INTERACTION WITH THE USER IS POSSIBLE DURING UPDATE
     5       ,/,40H IN THE USER SUPPLIED SUBROUTINE SUBPHI.
     6      ,//,34H OTHERWISE, USE THE ONE-STEP MODE.,/)
C
      IF (IFLAG .EQ. 8)  PRINT 1505,IFLAG
 1505 FORMAT(/,26H INVALID INPUT PARAMETERS:
     1      ,//,12H NEQN .LE. 0
     2       ,/,33H T = TOUT AND IFLAG .NE. +1 OR -1
     3       ,/,28H RELERR OR ABSERR .LT. 0.0D0
     4       ,/, 3H OR
     5       ,/,46H IFLAG  HAS BEEN SET TO AN INACCEPTABLE VALUE.
     6       ,/, 9H IFLAG = ,I3,/)
C
      IF (IFLAG .EQ. 94)  PRINT 1506
 1506 FORMAT(/,54H DIFFICULTIES WERE ENCOUNTERED USING THE TRAPPING OPTI
     1         ,3HON.,/,31H TOO MANY ITERATIONS WERE USED.
     2       ,/,53H CONDITIONS HAVE BEEN RETURNED AT THE LAST VALUE OF T
     3       ,/,31H BEFORE THE BORDER WAS CROSSED. ,/)
C
      IF (IFLAG .EQ. 97)  PRINT 1507
 1507 FORMAT(/,54H DIFFICULTIES WERE ENCOUNTERED USING THE TRAPPING OPTI
     1         ,3HON.,/,46H THE TRAPPING ITERATION BOUNDS WERE TOO CLOSE.
     2       ,/,56H CONDITIONS HAVE BEEN RETURNED AT THE LAST VALUE OF T
     3       ,/,31H BEFORE THE BORDER WAS CROSSED. ,/)
C
C                123456789 123456789 123456789 123456789 123456789 1234
C                0         1         2         3         4         5
C
      RETURN
```

```
      END
C
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
      SUBROUTINE TRAPPD(F,SUBPHI,NPHI,NEQN,NFE,INDEX,IFLAG,
     1   T,Y,YP,PHI0,PHIP0,TF,YF,YPF,PHIF,PHIPF,T2,Y2,YP2,
     2   PHI2,PHIP2,PHIR,PHIPR,PHIL,PHIPL,TL,TR,PHIB,PHIV,PHIPV,
     3   F1,F2,F3,F4,F5,F6,F7,F8,F9,ABSER,TZERO,
     4   KOUNTR,MPART,UPDATE,OUTPUT,FIND,EVALF,
     5   REMIN,ZAPP,LFLAG,NEXTRA)
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
      IMPLICIT REAL*8 (A-H,O-Z)
C
      DIMENSION Y(NEQN),YP(NEQN),Y2(NEQN),YP2(NEQN)
      DIMENSION YF(NEQN),YPF(NEQN),F1(NEQN),F2(NEQN)
      DIMENSION F3(NEQN),F4(NEQN),F5(NEQN)
      DIMENSION F6(NEQN),F7(NEQN),F8(NEQN),F9(NEQN)
      DIMENSION PHI0(NPHI),PHIP0(NPHI),PHIF(NPHI),PHIPF(NPHI)
      DIMENSION PHI2(NPHI),PHIP2(NPHI),PHIB(NPHI)
      DIMENSION PHIR(NPHI),PHIPR(NPHI),PHIL(NPHI),PHIPL(NPHI)
      DIMENSION PHIV(NPHI),PHIPV(NPHI)
      COMMON/CRKF45/IOPT,JOPT,IDUM(3)
      DATA TBOUND/1.0D-10/
      DATA MAXIT/25/
      DATA MODE0/0/,MODE1/1/,MODE2/2/
C
      LOGICAL UPDATE,IVAN,EVALF,ENDPT,OUTPUT,PUTOUT,
     1                NOTFAL,FIND,BOUNCE,SEARCH
      DATA NOTFAL/.FALSE./
C
      EXTERNAL F,SUBPHI
C
C-----------------------------------------------------------------------
C     STANDARD PRINTING OPTION--IOPT = 1
C-----------------------------------------------------------------------
      IF (IOPT .EQ. 0) GO TO 777
      PRINT 901,NFE,NEXTRA
      PRINT 760,T,TF,INDEX
      IF (FIND)  PRINT 763,TL,TR,PHIL(INDEX),PHIR(INDEX)
      DO 762 J = 1,NPHI
  762 PRINT 761,J,PHI0(J),J,PHIF(J)
  777 CONTINUE
C
      IF (.NOT. FIND)  EVALF = .FALSE.
C-----------------------------------------------------------------------
C     CALL BOUNCD TO SEE IF ANY COMPONENT OF PHI HAS BOUNCED ON A
C     ZERO ON THE PREVIOUS STEP.
C-----------------------------------------------------------------------
      CALL       BOUNCD(F,SUBPHI,NPHI,NEQN,T,Y,YP,PHI0,PHIP0,
     1                  TF,YF,YPF,PHIF,PHIPF,T2,Y2,YP2,PHI2,PHIP2,
     2                  PHIL,PHIPL,PHIR,PHIPR,TL,TR,PHIB,
     3                  F1,F2,F3,F4,F5,F6,F7,F8,F9,ABSER,ZAPP,
     4                  ITEST,INDEX,NFE,NEXTRA,KOUNTR,
     5                  EVALF,FIND)
C
      IF (ITEST .EQ. 1)  RETURN
      IF (INDEX .EQ. 0)  RETURN
```

```
C-----------------------------------------------------------------------
C     PANIC OPTION FOR STUDYING PHI THROUGHOUT THE INTEGRATION STEP
C     (EMERGENCY FEATURE--PANIC NOT REFERENCED IF EMERGENCY FEATURE
C     IS USED IN RKFST)
C-----------------------------------------------------------------------
      IF (.NOT. FIND  .AND. NOTFAL)
     1             CALL PANIC(F,SUBPHI,NPHI,NEQN,NFE,INDEX,T,Y,YP,
     2             PHI0,PHIP0,TF,YF,YPF,PHIF,PHIPF,T2,Y2,YP2,PHI2,PHIP2,
     3             PHIL,PHIPL,PHIR,PHIPR,TL,TR,PHIB,
     4             F1,F2,F3,F4,F5,F6,F7,F8,F9,ABSER,KOUNTR,
     5             NEXTRA,EVALF,FIND,MODE1,MODE0,ZAPP)
C
C-----------------------------------------------------------------------
C     INITIALIZATION BLOCK--BRACKETING VALUES HAVE BEEN SET IN RKFST
C-----------------------------------------------------------------------
C
  620 CONTINUE
C
      STEP = TF - T
      TLSTAR = TL
      TMAG = 0.5D0 * (DABS(TR) + DABS(TL))
      ITERAT = 0
      TOLER = ABSER
      ENDPT = .FALSE.
      PUTOUT = .FALSE.
C
      PHILL  = PHIL(INDEX)
      PHIPLL = PHIPL(INDEX)
      PHIRR  = PHIR(INDEX)
      PHIPRR = PHIPR(INDEX)
C-----------------------------------------------------------------------
C     ROUTE ANALYSIS THROUGH PROPER SECTION
C-----------------------------------------------------------------------
C     IF |PHIR(INDEX)| .LT. TOLERANCE--GO TO 300 FOR UPDATE PREPARATION
C     IF PHI(INDEX) CHANGED SIGNS FROM TL TO TR--GO TO 55  TO ITERATE
C           UNTIL A ZERO IS TRAPPD.
C     (OTHERWISE, AN ERROR IN THE ANALYSIS WAS MADE IN RKFST--STOP)
C
      IF (DABS(PHIR(INDEX)) .LT. TOLER)          GO TO 300
      IF (PHIL(INDEX)*PHIR(INDEX) .LT. 0.0D0)    GO TO 55
      PRINT 902,INDEX,PHIL(INDEX),PHIR(INDEX)
      STOP
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
C     AT LEAST ONE COMPONENT OF PHI HAS BEEN BRACKETED.
C     BEGIN ANALYSIS FOR ISOLATING THE ZERO OF PHI(INDEX).
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
   55 CONTINUE
C-----------------------------------------------------------------------
C     GENERATE THE ADDITIONAL F VALUES IF THEY HAVE NOT BEEN EVALUATED
C-----------------------------------------------------------------------
      IF (.NOT. EVALF)  CALL SCALED(F,NEQN,Y,YP,T,SIGMA,STEP,F2,F3,F4,
     1                    F5,F6,F7,F8,F9,F1,MODE1,NFE,NEXTRA,Y2)
      EVALF = .TRUE.
C-----------------------------------------------------------------------
C     LSTART DETERMINES THE SIDE FROM WHICH THE NEWTON APPROXIMATION
C           IS MADE.
```

```
C            IF PHI(INDEX) VANISHED ON THE PREVIOUS STEP, THE
C            THE FIRST NEWTON ESTIMATE IS MADE FROM THE RIGHT SIDE.
C            OTHERWISE, THE ITERATION STARTS FROM THE SIDE FOR
C            |PHI(INDEX)| IS SMALLER.
C            FOR SUBSEQUENT ITERATIONS, START FROM THE PREVIOUS T*
C            LSTART = 0       START FROM LEFT  HAND SIDE  (TL SIDE)
C            LSTART = 1       START FROM RIGHT HAND SIDE  (TR SIDE)
C-----------------------------------------------------------------------
      LSTART = 0
      IF (DABS(PHIL(INDEX)) .GT. DABS(PHIR(INDEX)))  LSTART = 1
      IF (PHIV(INDEX) .GT. 0.0D0)                    LSTART = 1
C-----------------------------------------------------------------------
C     STORE Y AND YP AT T0 IN F2 AND F3 (WHICH ARE NO LONGER NEEDED FOR
C        GENERATING THE SCALED SOLUTION
C-----------------------------------------------------------------------
      DO 116 J = 1,NEQN
      F2(J) = Y2(J)
      F3(J) = YP2(J)
  116 CONTINUE
C
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
C     MAJOR ITERATION LOOP
C         EXIT WHEN |PHI(INDEX)| <  TOLERANCE
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
  120 CONTINUE
C
      ITERAT = ITERAT + 1
      IF (ITERAT .GT. MAXIT)  GO TO 650
C
C-----------------------------------------------------------------------
C     CHOOSE NEW INTEGRATION STEP SIZE TO LOCATE THE ZERO OF
C     PHI(INDEX) USING A FALSE-POSITION OR NEWTON-RHAPSON METHOD
C-----------------------------------------------------------------------
      CALL       TSTAR(Z,T,TL,TR,TF,TLSTAR,TZERO,TMAG,PHLAST,
     1                 INDEX,NPHI,PHI2,PHIV,PHILL,PHIPLL,PHIRR,PHIPRR,
     2                 ABSER,TBOUND,ZAPP,LSTART,ITERAT,ISHIFT)
C-----------------------------------------------------------------------
C     Z = T* ESTIMATE HAS BEEN SELECTED, SET    H = Z - T
C-----------------------------------------------------------------------
      H = Z - T
C-----------------------------------------------------------------------
C     IF (TL, TR) BOUNDS ARE TOO CLOSE, EXIT THROUGH 648
C-----------------------------------------------------------------------
      IF (DABS(TL-TR) .LT. 0.5D0*DABS((TR+TL))*TBOUND)  GO TO 648
C-----------------------------------------------------------------------
C     EVALUATE THE SOLUTION AT T2 = T + H USING THE SCALED ALGORITHM
C-----------------------------------------------------------------------
      SIGMA = H/STEP
      CALL SCALED(F,NEQN,Y,YP,T,SIGMA,STEP,F2,F3,F4,F5,F6,F7,F8,F9,F1,
     1       MODE2,NFE,NEXTRA,Y2)
      T2 = Z
      CALL F(T2,Y2,YP2)
      NFE = NFE + 1
      NEXTRA = NEXTRA + 1
C-----------------------------------------------------------------------
C     EVALUATE PHI AND PHIP AT T2
```

```
C-----------------------------------------------------------------------
      UPDATE = .FALSE.
      BOUNCE = .FALSE.
      IVAN = .FALSE.
      INDX = INDEX
      PHLAST = PHI2(INDEX)
      CALL SUBPHI(NPHI, INDX,NEQN,T2,Y2,YP2,PHI2,PHIP2,KOUNTR,UPDATE,
     1     IVAN,BOUNCE,TOLER)
      IF (JOPT .EQ. 1)  PRINT 1523,T,INDEX,PHILL,INDEX,PHI2(INDEX),PHIRR
 1523 FORMAT(' T=',D15.7,2X,'INDEX=',I3,/,' PHILL=',D15.7,2X,'PHI2(',I3,
     1       ') = ',D15.7,2X,'PHIRR=',D15.7)
      TOLER = ABSER
C-----------------------------------------------------------------------
C     ADJUSTMENTS TO PHI2, PHIP2:
C     IF |PHI2|, |PHIP2| < UNIT ROUND-OFF, SET EQUAL TO UNIT ROUND-OFF
C     IF PHI(J) BOUNCED, GIVE PHI2(J) THE SIGN OF PHIR(J)
C-----------------------------------------------------------------------
      DO 154 L = 1,NPHI
      IF (DABS(PHI2(L))  .LT. ZAPP)     PHI2(L)  = DSIGN(ZAPP,PHI2(L))
      IF (DABS(PHIP2(L)) .LT. ZAPP)     PHIP2(L) = DSIGN(ZAPP,PHIP2(L))
  154 CONTINUE
      DO  155 J = 1,NPHI
      IF (PHIB(J) .GT. 0.0D0)   PHI2(J) = DSIGN(PHI2(J),PHIR(J))
  155 CONTINUE
C
C-----------------------------------------------------------------------
C     CALL SHIFTI TO SEE IF INDEX SHOULD BE SHIFTED
C-----------------------------------------------------------------------
      CALL       SHIFTI(NPHI,T2,PHI2,PHIP2,PHIL,PHIPL,PHIR,PHIPR,PHIF,
     1                  PHIB,ABSER,LSTART,
     2                  PHLAST,PHILL,PHIPLL,PHIRR,PHIPRR,INDEX,ISHIFT)
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
C     CHECK FOR CONVERGENCE
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
      IF (DABS(PHI2(INDEX)) .LT. TOLER) GO TO 188
C
C-----------------------------------------------------------------------
C     CONTINUE ITERATING--CONVERGENCE WAS NOT ACHIEVED
C     ADJUST TL AND TR
C-----------------------------------------------------------------------
  180 CONTINUE
      IF (PHI2(INDEX)*PHIR(INDEX) .GT. 0.0D0)  GO TO 185
C-----------------------------------------------------------------------
C     PHI(T2) IS NOT ACROSS THE BOUNDARY--SHIFT "L" TO "2"
C-----------------------------------------------------------------------
      TL = T2
      PHILL = PHI2(INDEX)
      PHIPLL = PHIP2(INDEX)
      LSTART = 0
C
C     STORE Y,YP AT TL IN F2,F3 WHICH ARE NO LONGER NEEDED IN SCALED
C
      DO 182 J = 1,NEQN
      F2(J) = Y2(J)
      F3(J) = YP2(J)
  182 CONTINUE
```

```
      GO TO 120
C
  185 CONTINUE
C-----------------------------------------------------------------------
C     PHI(T2) IS ACROSS THE BOUNDARY--SHIFT "R" TO "2"
C-----------------------------------------------------------------------
      TR = T2
      PHIRR = PHI2(INDEX)
      PHIPRR = PHIP2(INDEX)
      LSTART = 1
      GO TO 120
C
  188 CONTINUE
C-----------------------------------------------------------------------
C     CONVERGENCE HAS BEEN ACHIEVED.
C     SAFETY CHECK--IS SOME COMPONENT ACROSS THE BOUNDARY AND OUTSIDE
C     THE CONVERGENCE RANGE ?
C-----------------------------------------------------------------------
C
      ISTOP = 0
      DO 178 J = 1,NPHI
      IF (PHIB(J) .GT. 0.0D0)             GO TO 178
      IF (PHIL(J)*PHI2(J) .GT. 0.0D0)     GO TO 178
      IF (DABS(PHI2(J)) .LT. TOLER)   GO TO 178
C
C     COMPONENT J CHANGED SIGNS FROM TL TO T2 BUT HAS NOT YET VANISHED
C
      PRINT 927,J,INDEX
      PHIDIF = DABS(DABS(PHIL(J)) - TOLER)
      PRINT 925,J,PHIL(J),PHI2(J),PHIR(J),PHIDIF,TOLER
      ISTOP = 1
  178 CONTINUE
      IF (ISTOP .EQ. 0)  GO TO 400
      PRINT 1411
      STOP
C
  300 CONTINUE
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
C     PREPARE FOR EXIT--SOLUTION VANISHED AT T = TR SO NO ITERATIONS
C                       WERE REQUIRED
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
      IF (FIND .AND. TR .NE. TF)  GO TO 308
C-----------------------------------------------------------------------
C     TR AND TF ARE THE SAME POINT
C-----------------------------------------------------------------------
C
      ENDPT = .TRUE.
      IF (OUTPUT)  PUTOUT = .TRUE.
      DO 303 J = 1,NEQN
      Y2(J) = YF(J)
      YP2(J) = YPF(J)
  303 CONTINUE
      T2 = TF
      DO 306 J = 1,NPHI
      PHI2(J) = PHIF(J)
      PHIP2(J) = PHIPF(J)
```

```
  306 CONTINUE
      GO TO 400
C
C-----------------------------------------------------------------------
C     TR AND TF ARE NOT THE SAME POINT (TR SET BY EMERGENCY FEATURE
C     IN RKFST--A SUBSTEP OF THE INTEGRATION STEP WAS ANALYZED)
C     (THE SOLUTION HAS BEEN DELETED AT TR AND MUST BE RE-EVALUATED.)
C-----------------------------------------------------------------------
  308 CONTINUE
C
      SIGMA = (TR-T)/STEP
      T2 = TR
      CALL SCALED(F,NEQN,Y,YP,T,SIGMA,STEP,F2,F3,F4,F5,F6,F7,F8,F9,F1,
     1       MODE2,NFE,NEXTRA,Y2)
      CALL F(T2,Y2,YP2)
      NFE = NFE + 1
      NEXTRA = NEXTRA + 1
      DO 310 I = 1,NPHI
      PHI2(I) = PHIR(I)
      PHIP2(I) = PHIPR(I)
  310 CONTINUE
C
  400 CONTINUE
C
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
C     TRAPPD WAS SUCCESSFUL AND THE FUNCTION WILL BE UPDATED
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
C     EACH COMPONENT OF PHI WILL BE CHECKED TO SEE IF IT HAS
C     "VANISHED."    IF PHI(J) HAS "VANISHED," SUBPHI WILL BE
C     REFERENCED TO UPDATE THIS PARTICULAR COMPONENT, I.E.,
C     INDEX WILL INDICATE THAT COMPONTENT WHICH IS BEING UPDATED.
C
C     SUBROUTINE TRAPPD ASSUMES THAT PHI WILL CHANGE SIGN
C     AS IT CROSSES A BOUNDARY UNLESS THE COMPONENT HAS
C     REMAINED IN THE "VANISHED" REGION FROM T TO T2.  THEREFORE,
C     IF PHI(J) HAS BEEN TRAPPED (AND IS ESSENTIALLY ZERO), THE
C     SIGN IS CHANGED TO REFLECT CONDITIONS ACROSS THE BORDER.
C     THIS SIGN ADJUSTMENT IS MADE BEFORE ENTRY INTO SUBPHI FOR
C     UPDATING.  IF PHI(J) VANISHED THROUGHOUT (T,T2), THE SIGN
C     REMAINS THE SAME.  (THE SIGN OF A BOUNCING COMPONENT HAS
C     BEEN GIVEN THE SIGN OF PHIR DURING THE TRAPPING ANALYSIS.)
C
C     IF THE USER CHANGES THE SIGN OR MAGNITUDE OF PHI DURING UPDATE,
C     NEITHER WILL BE ALTERED UPON RE-ENTRY INTO TRAPPD AS LONG AS THE
C     MAGNITUDE IS GREATER THAN UNIT ROUND-OFF. IF THE USER-SUPPLIED
C     VALUE IS LESS THAN UNIT ROUND-OFF, THE VALUE WILL BE SET EQUAL
C     TO UNIT-ROUNDOFF WITH THE USER-SUPPLIED SIGN.
C-----------------------------------------------------------------------
      CALL VANISH(F,SUBPHI,NPHI,NEQN,NFE,INDEX,T,Y,YP,TLSTAR,T2,TF,
     2            PHI0,PHIP0,PHI2,PHIL,PHIPL,PHIR,PHIPR,PHIV,PHIPV,
     3            F1,F2,F3,F4,F5,F6,F7,F8,F9,
     4            ABSER,KOUNTR,EVALF,IVAN,NEXTRA,ZAPP,MODE1)
C
C     NOTE: PHIPL IS CHANGED IN VANISH
C- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
      UPDATE = .TRUE.
```

```
C- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C-----------------------------------------------------------------------
      DO 410 J = 1,NPHI
C-----------------------------------------------------------------------
      PHIV(J) = -1.0D0
      IF (DABS(PHI2(J)) .GT. TOLER)  GO TO 410
      PHIV(J) = +1.0D0
C-----------------------------------------------------------------------
C     THE JTH COMPONENT OF PHI HAS VANISHED--PHIPL(J) > 0 INDICATES
C        THAT PHI(J) VANISHED THROUGHOUT (T,T2)
C-----------------------------------------------------------------------
      IND = J
      IVAN = .FALSE.
      IF (PHIPL(J) .GT. 0.0D0)  IVAN = .TRUE.
C-----------------------------------------------------------------------
C     CHECK SIGN OF PHI(J) FOR POSSIBLE ADJUSTMENT
C-----------------------------------------------------------------------
      SGN = -1.D0
      IF (IVAN) SGN = +1.0D0
C-----------------------------------------------------------------------
C     PHI2(J) AND PHIL(J) WILL HAVE OPPOSITE SIGNS IF PHI(J) DID NOT
C     VANISH THROUGHOUT (TLSTAR, T2).  IF PHI(J) DID VANISH THROUGHOUT
C     THIS INTERVAL, PHI2(J) WILL BE GIVEN THE SIGN OF PHIL(J).
C-----------------------------------------------------------------------
      PHI2(J) = SGN*DSIGN(PHI2(J),PHIL(J))
C-----------------------------------------------------------------------
C     SAVE PHI2(J) VALUE BEFORE ENTERING PHI.  IF THE USER CHANGES PHI2
C     AND THE PRINTING OPTION IS ACTIVE, OUTPUT WILL ALSO BE GIVEN
C     AFTER THE SUBPHI CALL.
C-----------------------------------------------------------------------
      SAVEPH = PHI2(J)
C-----------------------------------------------------------------------
C     STANDARD PRINTING OPTION--IOPT = 1
C-----------------------------------------------------------------------
      IF (IOPT .EQ. 0) GO TO 783
      PRINT 780,T2,INDEX
      DO 778 JJ = 1,NPHI
  778 PRINT 779,JJ,PHI2(JJ)
  783 CONTINUE
C-----------------------------------------------------------------------
C     CALL SUBPHI TO UPDATE PHI2
C-----------------------------------------------------------------------
      UPDATE = .TRUE.
      BOUNCE = .FALSE.
      INDD = IND
      CALL SUBPHI(NPHI, INDD,NEQN,T2,Y2,YP2,PHI2,PHIP2,KOUNTR,UPDATE,
     1          IVAN,BOUNCE,TOLER)
      TOLER = ABSER
      KOUNTR = KOUNTR + 1
      DIFF = DABS(PHI2(J) - SAVEPH)
C-----------------------------------------------------------------------
C     STANDARD PRINTING OPTION:
C    IF CONDITIONS ARE UPDATED, THEY WILL BE REPRINTED (FOR IOPT=1)
C-----------------------------------------------------------------------
      IF (IOPT .EQ. 0 .OR. DIFF .LT. ZAPP) GO TO 785
      PRINT 780,T2,INDEX
      DO 784 JJ = 1,NPHI
  784 PRINT 779,JJ,PHI2(JJ)
```

```
  785 CONTINUE
C-----------------------------------------------------------------------
  410 CONTINUE
C-----------------------------------------------------------------------
      TZERO = T2
C-----------------------------------------------------------------------
C     MAKE MAGNITUDE ADJUSTMENT IN CASE A USER SUPPLIED, UPDATE VALUE
C     OF PHI IS LESS THAN UNIT-ROUNDOFF.  (USER SUPPLIED SIGN REMAINS.)
C-----------------------------------------------------------------------
      DO 411 L = 1,NPHI
      IF (DABS(PHI2(L))  .LT. ZAPP)     PHI2(L)  = DSIGN(ZAPP,PHI2(L))
  411 IF (DABS(PHIP2(L)) .LT. ZAPP)     PHIP2(L) = DSIGN(ZAPP,PHIP2(L))
C-----------------------------------------------------------------------
      UPDATE = .FALSE.
      INDEX = 0
      IF (LFLAG .LT. 2)  GO TO 450
C-----------------------------------------------------------------------
C     IF MULTIPLE TRAPPING OPTION IS ACTIVE (LFLAG=2), CALL MULTOP
C     TO IDENTIFY FURTHER ZEROS.
C-----------------------------------------------------------------------
      JLIM = MPART + 1
      IF (JLIM .GT. NPHI)  GO TO 448
      DO 447 J = JLIM,NPHI
      IF (PHIV(J) .GT. 0)  GO TO 450
  447 CONTINUE
  448 CONTINUE
C
C     ANY UPDATES OCCURRED WITH COMPONENTS IN THE MULTITRAPPING
C     OPTION
C
      CALL       MULTOP(F,SUBPHI,NPHI,NEQN,T,Y,YP,PHI0,PHIP0,
     1                  T2,Y2,YP2,PHI2,PHIP2,TF,YF,YPF,PHIF,PHIPF,PHIB,
     2                  TL,TR,PHIL,PHIPL,PHIR,PHIPR,
     3                  PHILL,PHIPLL,PHIRR,PHIPRR,TLSTAR,TZERO,
     4                  ABSER,ZAPP,F1,F2,F3,F4,F5,F6,F7,F8,F9,
     5                  INDEX,MPART,KOUNTR,NFE,NEXTRA,ITERAT,
     6                  OUTPUT,PUTOUT,ENDPT,SEARCH,FIND)
C-----------------------------------------------------------------------
      IF (SEARCH)  GO TO 620
      RETURN
C
  450 CONTINUE
C
C-----------------------------------------------------------------------
C     PREPARE FOR EXIT--SOLUTION IS RETURNED AT THE TRAPPED POINT
C                       (LFLAG = -2,-1, OR 1)
C-----------------------------------------------------------------------
C
      T = T2
      DO 415 J = 1,NEQN
      Y(J) = Y2(J)
  415 YP(J) = YP2(J)
C
      DO 416 J = 1,NPHI
      PHI0(J) = PHI2(J)
  416 PHIP0(J) = PHIP2(J)
C
  455 CONTINUE
```

```
C
C     IF AN OUTPUT POINT OCCURRED IN RKFST AND THE TRAPPED POINT
C     OCCURRS AT TF, OUTPUT WILL STILL BE .TRUE., OTHERWISE
C     OUTPUT = .FALSE.
C
      OUTPUT = .FALSE.
      IF (PUTOUT)  OUTPUT = .TRUE.
C
      RETURN
C
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
C     TERMINAL ERRORS ENCOUNTERED--PREPARE FOR FINAL EXIT.
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
C     THE DIFFERENCE BETWEEN THE TRAPPING BOUNDS IS BELOW
C     AN ACCEPTABLE TOLERANCE--TERMINAL ERROR--IFLAG = 97.
C     OR
C     ITERATION REQUIRES TOO MANY STEPS--TERMINAL ERROR--IFLAG = 94
C-----------------------------------------------------------------------
  648 CONTINUE
      IFLAG = 97
      GO TO 652
  650 CONTINUE
      IFLAG = 94
  652 CONTINUE
C
C-----------------------------------------------------------------------
C     RETURN SOLUTION AT TL
C-----------------------------------------------------------------------
C
      CALL PANIC(F,SUBPHI,NPHI,NEQN,NFE,INDEX,T,Y,YP,
     1             PHI0,PHIP0,TF,YF,YPF,PHIF,PHIPF,T2,Y2,YP2,PHI2,PHIP2,
     2             PHIL,PHIPL,PHIR,PHIPR,TL,TR,PHIB,
     3             F1,F2,F3,F4,F5,F6,F7,F8,F9,ABSER,KOUNTR,
     4             NEXTRA,EVALF,FIND,MODE1,MODE0,ZAPP)
C
      DO 356 J = 1,NEQN
      Y(J) = F2(J)
      YP(J) = F3(J)
  356 CONTINUE
      T = TL
      DO 367 J = 1,NPHI
      PHI0(J) = PHIL(J)
      PHIP0(J) = PHIPL(J)
  367 CONTINUE
C
  760 FORMAT(' ENTERED TRAPPD WITH T = ',D15.7,' AND  TF = ',D15.7,
     1    2X,' INDEX = ',I3)
  761 FORMAT(' PHI0(',I2,') = ',D15.7,2X,' PHIF(',I2,') = ',D15.7)
  763 FORMAT(' FROM PANIC ROUTINE--TL = ',D15.7,3X,' TR = ',D15.7,/,
     1         ' PHIL(INDEX = ',D15.7,3X,'PHIR(INDEX) = ',D15.7)
  779 FORMAT(' PHI0(',I2,') = ',D15.7)
  780 FORMAT(' CONDITIONS-----AFTER SUCESSFUL TRAP--T = ',D15.7,2X,
     1        ' INDEX = ',I3)
  901 FORMAT(' NFE = ',I5,2X,' NEXTRA = ',I5)
  902 FORMAT(//,' TRAPPED WAS REFERENCED WHEN PHI NEITHER VANISHED NOR',
     1       ' CHANGED SIGN',/,' INDEX = ',I3,2X,'PHIL(INDEX) = ',
```

```
     2       D15.7,2X,'PHIR(INDEX) = ',D15.7,//,' TERMINAL ERROR',//)
  925 FORMAT(' J,PHIL,PHI2,PHIR,|PHI2-TOLER|,TOLER',/,2X,I3,
     1      3(2X,D15.7),/,2(2X,D15.7))
  927 FORMAT('  TERMINAL ERROR  WITH COMPONENT J = ',I3,/,
     1    ' INDEX = ',I3)
 1411 FORMAT(//,' TERMINAL ERROR IN TRAPP--ATTEMPT TO EXIT WITHOUT '
     1       ,' COMPLETE TRAPPING',/)
C
      RETURN
      END
C-----------------------------------------------------------------------
      SUBROUTINE BOUNCD(F,SUBPHI,NPHI,NEQN,T,Y,YP,PHI0,PHIP0,
     1                  TF,YF,YPF,PHIF,PHIPF,T2,Y2,YP2,PHI2,PHIP2,
     2                  PHIL,PHIPL,PHIR,PHIPR,TL,TR,PHIB,
     3                  F1,F2,F3,F4,F5,F6,F7,F8,F9,ABSER,ZAPP,
     4                  ITEST,INDEX,NFE,NEXTRA,KOUNTR,
     5                  EVALF,FIND)
C-----------------------------------------------------------------------
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION Y(NEQN),YP(NEQN),YF(NEQN),YPF(NEQN),Y2(NEQN),YP2(NEQN)
      DIMENSION  F1(NEQN),F2(NEQN),F3(NEQN),F4(NEQN)
      DIMENSION  F5(NEQN),F6(NEQN),F7(NEQN),F8(NEQN),F9(NEQN)
      DIMENSION  PHI0(NPHI), PHIF(NPHI), PHI2(NPHI), PHIB(NPHI)
      DIMENSION  PHIL(NPHI), PHIR(NPHI)
      DIMENSION PHIP0(NPHI),PHIPF(NPHI),PHIP2(NPHI)
      DIMENSION PHIPL(NPHI),PHIPR(NPHI)
C
      DATA MODE0/0/,MODE3/3/
      LOGICAL EVALF,FIND,UPDATE,IVAN,BOUNCE
C
      EXTERNAL F,SUBPHI
C
C-----------------------------------------------------------------------
C     BOUNCING FUNCTION ANALYSIS
C-----------------------------------------------------------------------
C     CALL PANIC (IN SPECIAL MODE) TO SEE IF ANY COMPONENT OF PHI
C     MAY HAVE "BOUNCED" ON A ZERO.
C
      CALL PANIC(F,SUBPHI,NPHI,NEQN,NFE,INDEX,T ,Y ,YP ,PHI0,PHIP0,
     1           TF,YF,YPF,PHIF,PHIPF,      T2,Y2,YP2,PHI2,PHIP2,
     2           PHIL,PHIPL,PHIR,PHIPR,TL,TR,PHIB,
     3           F1,F2,F3,F4,F5,F6,F7,F8,F9,ABSER,KOUNTR,
     4           NEXTRA,EVALF,FIND,MODE3,MODE0,ZAPP)
C
      ITEST = 0
      DO 602 J = 1,NPHI
      IF (PHIB(J) .LT. 0.0D0)  GO TO 602
C
C     COMPONENT J "BOUNCED" ON A ZERO--CALL SUBPHI IN AN UPDATE MODE
C     WITH CONDITIONS AT T IN CASE ANY CHANGES NEED TO BE MADE.
C
C     LOGICAL INDICATOR--BOUNCE=.TRUE.
C
C     DUMMY COUNTER, KNT, IS SENT OVER INSTEAD OF KOUNTR
C     KNT = -1.  IF KNT IS RETURNED AS -2 FROM ANY UPDATE CALL,
C          THE ANALYSIS WILL RETURN TO RKFST AND THE INTEGRATION
C          STEP WILL BE REPEATED.  (THIS ALLOWS THE USER TO CHANGE
C          THE ODE SYSTEM AT TO IF THE BOUNCING FUNCTION HAS CAUSED
```

```
C          DIFFICULTIES)
      KNTR = -1
      UPDATE = .TRUE.
      BOUNCE = .TRUE.
      IVAN = .FALSE.
      TOLER = ABSER
      INDX = J
      CALL SUBPHI(NPHI, INDX,NEQN,T2,Y2,YP2,PHI0,PHIP0,KNTR  ,UPDATE,
     1     IVAN,BOUNCE,TOLER)
      DO 601 L = 1,NPHI
      IF (DABS(PHI0(L))  .LT. ZAPP)     PHI0(L)  = DSIGN(ZAPP,PHI0(L))
      IF (DABS(PHIP0(L)) .LT. ZAPP)     PHIP0(L) = DSIGN(ZAPP,PHIP0(L))
      IF (PHIB(L) .GT. 0)               PHI0(L) = DSIGN(PHI0(L),PHIR(L))
  601 CONTINUE
      IF (KNTR .EQ. -2)  ITEST = 1
  602 CONTINUE
C
      IF (ITEST .EQ. 0)  RETURN
C
C     INTEGRATION IS TO BE CONTINUED FROM T--RETURN TO RKFST AND REPEAT
C        THE INTEGRATION STEP
C
      T = T2
      DO 603 J = 1,NEQN
      Y(J) = Y2(J)
      YP(J) = YP2(J)
  603 CONTINUE
C
      DO 604 J = 1,NPHI
      PHI0(J) = PHI2(J)
      PHIP0(J) = PHIP2(J)
  604 CONTINUE
C
      RETURN
      END
C-----------------------------------------------------------------------
      SUBROUTINE SHIFTI(NPHI,T2,PHI2,PHIP2,PHIL,PHIPL,PHIR,PHIPR,PHIF,
     1                  PHIB,ABSER,LSTART,
     2                  PHLAST,PHILL,PHIPLL,PHIRR,PHIPRR,INDEX,ISHIFT)
C-----------------------------------------------------------------------
      IMPLICIT REAL*8 (A-H,O-Z)
C
      COMMON/CRKF45/IDUM1(2),IOPT,IDUM2(2)
C
      DIMENSION  PHI2(NPHI), PHIL(NPHI), PHIR(NPHI), PHIF(NPHI)
      DIMENSION PHIP2(NPHI),PHIPL(NPHI),PHIPR(NPHI), PHIB(NPHI)
      LOGICAL RSHIFT
C
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
C     SEE IF INDEX SHOULD BE SHIFTED
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
C
      ISHIFT = 0
C
C     INDEX DESIGNATES THE COMPONENT OF PHI(J) CURRENTLY BEING
C     TRAPPED.
```

```
C-----------------------------------------------------------------------
C     SAFETY CHECK:
C     HAS A NON-DETECTED ZERO BEEN FOUND ON THIS STEP?
C     (BOUNCING COMPONENTS WILL NOT BE ANALYZED)
C-----------------------------------------------------------------------
      RSHIFT = .FALSE.
      DO 156 J = 1,NPHI
      IF (PHIB(J) .GT. 0.0D0)  GO TO 156
      IF (DABS(PHI2(J)) .LT. ABSER)   GO TO 156
      IF (PHIL(J)*PHI2(J) .LT. 0.0D0 .AND. PHIL(J)*PHIR(J) .GT. 0.0D0)
     1            RSHIFT = .TRUE.
  156 CONTINUE
      IF (.NOT. RSHIFT)  GO TO 159
C
C-----------------------------------------------------------------------
C     A NON-DETECTED ZERO HAS BEEN FOUND.  THE SIGN OF PHIR WILL BE
C     CHANGED TO MATCH THAT OF PHI2 FOR THESE COMPONENTS.
C-----------------------------------------------------------------------
C
      DO 158 J = 1,NPHI
      IF (PHIL(J)*PHIR(J) .GT. 0.0D0 .AND. PHI2(J)*PHIL(J) .LT. 0.0D0)
     1GO TO 157
      GO TO 158
  157 CONTINUE
      IF (IOPT .EQ. 1)     PRINT 1531,J,PHIL(J),PHI2(J),PHIR(J),PHIF(J)
      PHIR(J) =-PHIR(J)
  158 CONTINUE
  159 CONTINUE
C
      IF (NPHI .EQ. 1)  RETURN
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
C     CHECK TO SEE IF INDEX SHOULD BE RESET
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
      DO 160 J = 1,NPHI
      IF (PHIB(J) .GT. 0.0D0)  GO TO 160
      IF (PHI2(J)*PHIL(J) .GT. 0.0D0)  GO TO 160
      GO TO 162
  160 CONTINUE
C
C-----------------------------------------------------------------------
C     NO COMPONENT OF PHI IS ACROSS THE BORDER AT "2"
C-----------------------------------------------------------------------
C
      RETURN
C
  162 CONTINUE
C
C-----------------------------------------------------------------------
C     THE BORDER WAS CROSSED.  SHOULD INDEX BE SHIFTED ?
C-----------------------------------------------------------------------
C
      IND = INDEX
      INDEX = J
      IF (J .EQ. NPHI)  GO TO 168
      JP1 = J + 1
      DO 165 J = JP1,NPHI
```

```
      IF (PHIB(J)   .GT. 0.0D0)          GO TO 165
      IF (PHIL(J)*PHIR(J) .GT. 0.0D0)    GO TO 165
      IF (PHI2(J)*PHIR(J) .LT. 0.0D0)    GO TO 165
C
C-----------------------------------------------------------------------
C     COMPONENT J WAS TRAPPD AND T2 IS ACROSS THE BOUNDARY.
C-----------------------------------------------------------------------
C
      IF ( DABS(PHI2(J)) .LT. DABS(PHI2(INDEX)) )   GO TO 165
      IF (J .NE. IND .AND. IOPT .EQ. 1)
     1                            PRINT 972,IND,J,IND,PHI2(IND),J,PHI2(J)
      INDEX = J
  165 CONTINUE
  168 CONTINUE
C
      IF (IND .EQ. INDEX)  RETURN
C
C-----------------------------------------------------------------------
C     INDEX HAS BEEN SHIFTED
C-----------------------------------------------------------------------
C
      ISHIFT = 1
      PHILL = PHIL(INDEX)
      PHIPLL = PHIPL(INDEX)
      PHIRR = PHIR(INDEX)
      PHIPRR = PHIPR(INDEX)
      IF (LSTART .EQ. 0)      PHLAST = PHILL
      IF (LSTART .EQ. 1)      PHLAST = PHIRR
C
  972 FORMAT(/,' INDEX IS BEING CHANGED IN TRAPPD:   INDEX = ',I3,2X,
     1       ' J = ',I3,/,
     2       ' PHI(',I3,') = ',D15.7,2X,' PHI(',I3,') = ',D15.7)
 1531 FORMAT(' UNDETECTED ZERO--I,PHI(I)--L,2,R,F',/,I3,4(1X,D13.6))
C
      RETURN
      END
C-----------------------------------------------------------------------
      SUBROUTINE TSTAR(Z,T,TL,TR,TF,TLSTAR,TZERO,TMAG,PHLAST,
     1                 INDEX,NPHI,PHI2,PHIV,PHILL,PHIPLL,PHIRR,PHIPRR,
     2                 ABSER,TBOUND,ZAPP,LSTART,ITERAT,ISHIFT)
C-----------------------------------------------------------------------
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION PHI2(NPHI),PHIV(NPHI)
      COMMON/CRKF45/IDUM1(3),IOPT,IDUM2
C
      DATA FRACT/0.50D0/
      LOGICAL RATDIF
C
C-----------------------------------------------------------------------
C     Z1      GIVES NEWTON-ESTIMATE.
C     ZCHORD  GIVES FALSE-POSITION ESTIMATE.
C     LSTART  DESIGNATES THE SIDE FROM WHICH THE NEWTON ESTIMATE IS
C             EVALUATED.
C-----------------------------------------------------------------------
      IF (LSTART .EQ. 0)        Z1 = TL - PHILL/PHIPLL
      IF (LSTART .EQ. 1)        Z1 = TR - PHIRR/PHIPRR
      ZCHORD = TL - PHILL * (TR-TL)/(PHIRR-PHILL)
      Z = Z1
```

```
      IF ((TR-Z)*(Z-TL) .LT. ZAPP)        Z = ZCHORD
      IF (IOPT .EQ. 1)                    PRINT 1504,LSTART,Z1,ZCHORD,Z
C
C-----------------------------------------------------------------------
C     SAFETY CHECKS:  IS T* NEAR THE LAST LOCATED ZERO POINT (TZERO),
C                     OR IS CONVERGENCE SLOW ?
C-----------------------------------------------------------------------
C
C     IF THE DISTANCE BETWEEN Z AND THE LAST ZERO POINT IS 5% OF THE
C       ORIGINAL INTERATION INTERVAL, GIVE THE PROCDEURE A FRACTIONAL-
C       INTERVAL KICK TO GET IT OUT OF THIS REGION.
C
      RATDIF = .FALSE.
      IF (PHIV(INDEX)    .GT.  0.0D0  .AND.
     1    DABS(Z-TZERO) .LT. 0.05*DABS(TF-TLSTAR))     GO TO 140
C
  120 CONTINUE
      IF (ITERAT .LE. 3)        GO TO 150
      IF (ISHIFT .EQ. 1)        GO TO 150
      IF (LSTART .EQ. 0 )       RATIO = DABS(PHILL/PHLAST)
      IF (LSTART .EQ. 1 )       RATIO = DABS(PHIRR/PHLAST)
      IF (RATIO .LE. 0.20D0)    GO TO 150
      RATDIF = .TRUE.
C
  140 CONTINUE
      IF (IOPT .EQ. 1)         PRINT 1500,Z,TLSTAR,TL,TR,PHILL,PHIRR,
     1                                    PHLAST,RATIO
C
C-----------------------------------------------------------------------
C     TROUBLE SHOOTING BLOCK:
C        IF PHIO(INDEX) IS NEAR ZERO (I.E., IF IT VANISHED ON THE PRE-
C        VIOUS STEP)  CONVERGENCE DIFFICULTIES ARISE IF THE LEFT END
C        END POINT IS USED.  GIVE THE ITERATION PROCESS A "FRACTIONAL-
C        INTERVAL" KICK TO GET PHI OUT OF THE "VANISHED" REGION.
C
C        IF CONVERGENCE IS SLOW AFTER 2 ITERATIONS (IF LESS THAN A
C        DIGIT OF ACCURACY HAS BEEN ACHIEVED DURING THE LATEST ITERA-
C        TION) GIVE THE ITERATION PROCESS A "FRACTIONAL-INTERVAL"
C        KICK.
C-----------------------------------------------------------------------
C
C     ZCHORD  GIVES A SPECIFIED FRACTION OF |TR-TL| AS ESTIMATE
C     Z1      GIVES NEWTON-RHAPSON ESTIMATE STARTING FROM TR
C
      IF (LSTART .EQ. 0)    ZCHORD = TL + FRACT * (TR - TL)
      IF (LSTART .EQ. 1)    ZCHORD = TR - FRACT * (TR - TL)
      Z1 = TR - PHIRR/PHIPRR
      IF (RATDIF)                                          Z1 = ZCHORD
      IF (DABS(Z1-TZERO) .LT. 0.05*DABS(TF-TLSTAR))        Z1 = ZCHORD
      IF ( (TR-Z1)*(Z1-TL) .LT. 0.0D0)                     Z1 = ZCHORD
      Z = Z1
      IF (IOPT .EQ. 1)  PRINT 1501,TL,Z,TR,TLSTAR
      RETURN
  150 CONTINUE
      IF (IOPT .EQ. 1)   PRINT 1502,TL,Z,TR
C
 1500 FORMAT(/,' DIFFICULTIES IN ESTIMATING TSTAR:',/,
     1       ' CURRENT EST. = ',D23.16,2X,'PREVIOUS EST. = ',
```

```
     2       D23.16,/,' TL = ',D23.16,2X,'TR = ',D23.16,/,' PHILL=',
     3       D15.7,2X,'PHIRR=',D15.7,2X,'PHLAST=',D15.7,2X,
     4       /,' RATIO= ',D15.7)
 1501 FORMAT(/,' DIFFICULTIES WITH THE LEFT BOUND SEEM TO OCCUR:',/,
     1       ' (START FROM R.H.S. AND DO NOT USE FALSE-POSITION)',/,
     2 ' TL=',D15.7,2X,'Z=',D23.16,/,' TR=',D15.7,2X,'TLSTAR=',D23.17)
 1502 FORMAT(/,' T* ESTIMATE:',/,' TL=',D15.7,2X,'Z=',D15.7,2X,'TR=',
     1       D15.7)
 1504 FORMAT(/,' TSTAR ESTIMATES:',/,' LSTART=',I3,2X,' Z1=',D23.16,2X,
     1       ' ZCHORD=',D23.16,/,' SELECTED Z=',D23.16)
C
      RETURN
      END
C-----------------------------------------------------------------------
      SUBROUTINE MULTOP(F,SUBPHI,NPHI,NEQN,T,Y,YP,PHI0,PHIP0,
     1                  T2,Y2,YP2,PHI2,PHIP2,TF,YF,YPF,PHIF,PHIPF,PHIB,
     2                  TL,TR,PHIL,PHIPL,PHIR,PHIPR,
     3                  PHILL,PHIPLL,PHIRR,PHIPRR,TLSTAR,TZERO,
     4                  ABSER,ZAPP,F1,F2,F3,F4,F5,F6,F7,F8,F9,
     5                  INDEX,MPART,KOUNTR,NFE,NEXTRA,ITERAT,
     6                  OUTPUT,PUTOUT,ENDPT,SEARCH,FIND)
C-----------------------------------------------------------------------
      IMPLICIT REAL*8 (A-H,O-Z)
C
      DIMENSION Y(NEQN), Y2(NEQN),YF(NEQN),YP(NEQN),YP2(NEQN),YPF(NEQN)
      DIMENSION  PHI0(NPHI), PHI2(NPHI), PHIF(NPHI), PHIB(NPHI),
     1           PHIL(NPHI), PHIR(NPHI)
      DIMENSION PHIP0(NPHI),PHIP2(NPHI),PHIPF(NPHI),
     1          PHIPL(NPHI),PHIPR(NPHI)
      DIMENSION F1(NEQN),F2(NEQN),F3(NEQN),F4(NEQN),F5(NEQN),
     1          F6(NEQN),F7(NEQN),F8(NEQN),F9(NEQN)
C
      COMMON/CRKF45/IDUM1(4),IOPT
C
      LOGICAL OUTPUT,PUTOUT,ENDPT,SEARCH,BOUNCE,UPDATE,IVAN,FIND,EVALF
      DATA EVALF/.TRUE./
C
      EXTERNAL F,SUBPHI
C
C-----------------------------------------------------------------------
C     MULTIPLE TRAP OPTION IS BEING USED.  IF T2 = TR,  GO TO 435.
C-----------------------------------------------------------------------
C
C-----------------------------------------------------------------------
C     CHECK TO SEE IF ANY COMPONENTS  STILL NEED TO BE TRAPPD
C     (OPTION IFLAG =  15, 25, (LFLAG=2), IS BEING USED)
C-----------------------------------------------------------------------
C     REEVAULATE PHIF AND PHIPF SINCE THE USER MAY HAVE
C     CHANGED THEM.  CONDITIONS AT T2 WILL BECOME LEFT CONDITIONS
C     (USER SHOULD HAVE ALREADY MADE ANY DESIRED CHANGES TO PHI2
C     DURING TRAPPING UPDATES)
C-----------------------------------------------------------------------
      UPDATE = .FALSE.
      BOUNCE = .FALSE.
      IVAN   = .FALSE.
      SEARCH = .FALSE.
      INDX = 0
      TOLER = ABSER
```

```
      CALL SUBPHI(NPHI, INDX,NEQN,TF,YF,YPF,PHIF,PHIPF,KOUNTR,UPDATE,
     1          IVAN,BOUNCE,TOLER)
      TOLER = ABSER
C-----------------------------------------------------------------------
C     IF ANY COMPONENT OF PHIF IS ZERO--CHANGE TO UNIT ROUND-OFF
C-----------------------------------------------------------------------
      DO 410 J = 1,NPHI
      IF (DABS(PHIF(J))  .LT. ZAPP)     PHIF(J)  = DSIGN(ZAPP,PHIF(J))
      IF (DABS(PHIPF(J)) .LT. ZAPP)     PHIPF(J) = DSIGN(ZAPP,PHIPF(J))
  410 CONTINUE
      IF (.NOT. ENDPT)  GO TO 420
C-----------------------------------------------------------------------
C     IF T IS AT THE END POINT, RETURN.
C-----------------------------------------------------------------------
      T = TF
      DO 412 J = 1,NEQN
      Y(J) = YF(J)
  412 YP(J) = YPF(J)
      DO 414 J=1,NPHI
      PHI0(J) = PHIF(J)
  414 PHIPF(J) = PHIPF(J)
      RETURN
C-----------------------------------------------------------------------
C     SET NEW INITIAL CONDITIONS:  TZERO BECOMES TLSTAR
C-----------------------------------------------------------------------
  420 CONTINUE
C     T  = T2
      TL = T2
      TR = TF
      TLSTAR = T2
      TZERO  = T2
      DO 424 J = 1,NPHI
      PHIL(J) = PHI2(J)
      PHIPL(J) = PHIP2(J)
C     PHI0(J) = PHI0(J)
C     PHIP0(J) = PHIP0(J)
      PHIR(J) = PHIF(J)
      PHIPR(J) = PHIPF(J)
  424 CONTINUE
C     DO 425 J = 1,NEQN
C     Y(J) = Y2(J)
C     YP(J) = YP2(J)
C 425 CONTINUE
C-----------------------------------------------------------------------
C     HAS ANY COMPONENT OF PHI CHANGED SIGN FROM T2 TO TF ?
C-----------------------------------------------------------------------
      INDEX = 0
      PHIMAX = 0.0D0
      DO 428 J = 1,MPART
      IF (DABS(PHIF(J)) .LT. TOLER)    GO TO 427
      IF (PHIL(J)*PHIF(J) .GT. 0.0D0)  GO TO 428
  427 CONTINUE
      IF (DABS(PHIF(J)) .LT. PHIMAX)   GO TO 428
      INDEX = J
      PHIMAX = PHIF(J)
  428 CONTINUE
C
      IF (INDEX .EQ. 0)  GO TO 435
```

```
 1515 FORMAT(//,' TRAPPING CONTINUES WITH NEW COMPONENT--INDEX = ',I4,
     1       /,' PHIL(INDEX) = ',D15.7,2X,' PHIR(INDEX) = ',D15.7)
C
C-----------------------------------------------------------------------
C     CALL BOUNCD TO SEE IF ANY COMPONENT OF PHI HAS BOUNCED ON A
C     ZERO ON THE PREVIOUS STEP.
C-----------------------------------------------------------------------
      CALL       BOUNCD(F,SUBPHI,NPHI,NEQN,T,Y,YP,PHI0,PHIP0,
     1                  TF,YF,YPF,PHIF,PHIPF,T2,Y2,YP2,PHI2,PHIP2,
     2                  PHIL,PHIPL,PHIR,PHIPR,TL,TR,PHIB,
     3                  F1,F2,F3,F4,F5,F6,F7,F8,F9,ABSER,ZAPP,
     4                  ITEST,INDEX,NFE,NEXTRA,KOUNTR,
     5                  EVALF,FIND)
C
      IF (ITEST .EQ. 1)   GO TO 455
      IF (INDEX .EQ. 0)   GO TO 435
C-----------------------------------------------------------------------
C     AT LEAST ONE COMPONENT OF PHI HAS STILL CHANGED SIGNS
C-----------------------------------------------------------------------
      ITERAT = 0
      PHILL = PHI2(INDEX)
      PHIPLL = PHIP2(INDEX)
      PHIRR = PHIF(INDEX)
      PHIPRR = PHIPF(INDEX)
      DO 434 J = 1,NEQN
      F2(J) = Y2(J)
  434 F3(J) = YP2(J)
      LSTART = 1
      IF (IOPT .EQ. 1)   PRINT 1515,INDEX,PHILL,PHIRR
C
      SEARCH = .TRUE.
      RETURN
C
  435 CONTINUE
C-----------------------------------------------------------------------
C     PREPARE FOR EXIT--SOLUTION RETURNED AT TF
C-----------------------------------------------------------------------
      INDEX = 0
      T = TF
      DO 438 J = 1,NEQN
      Y(J) = YF(J)
  438 YP(J) = YPF(J)
C
      DO 439 J = 1,NPHI
      PHI0(J) = PHIF(J)
  439 PHIP0(J) = PHIPF(J)
C
  455 CONTINUE
C
C     IF AN OUTPUT POINT OCCURRED IN RKFST AND THE TRAPPED POINT
C     OCCURRS AT TF, OUTPUT WILL STILL BE .TRUE., OTHERWISE
C     OUTPUT = .FALSE.
C
      OUTPUT = .FALSE.
      IF (PUTOUT)  OUTPUT = .TRUE.
C
      RETURN
      END
```

```
C
C-----------------------------------------------------------------------
      BLOCK DATA
C-----------------------------------------------------------------------
      IMPLICIT REAL*8 (A-H,O-Z)
C
      COMMON/CRKF45/ITRAP,JTRAP,ISHFTI,ITSTAR,IMULTI
C
C     ITRAP  ==> PRINTING IN TRAPPD, INPUT AND OUTPUT
C     JTRAP  ==> PRINTING IN TRAPPD, CONDITIONS AFTER EACH PHI
C                                    EVALUATION
C     ISHFTI ==> PRINTING IN SHIFTI
C     ITSTAR ==> PRINTING IN TSTAR
C     IMULIT ==> PRINTING IN MULTIPLE TRAP FEATURE (AFTER EACH UDPATE)
C
      DATA ITRAP/0/,JTRAP/0/,ISHFTI/0/,ITSTAR/0/,IMULTI/0/
C
      END
C-----------------------------------------------------------------------
C
      SUBROUTINE VANISH(F,SUBPHI,NPHI,NEQN,NFE,INDEX,T,Y,YP,
     1   TLSTAR,T2,TF,PHI0,PHIP0,PHI2,PHIL,PHIPL,PHIR,PHIPR,PHIV,PHIPV,
     3   F1,F2,F3,F4,F5,F6,F7,F8,F9,
     4   ABSER,KOUNTR,EVALF,IVAN,NEXTRA,ZAPP,IROUTE)
C
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
C     SUBROUTINE FOR CHECKING TO SEE IF A COMPONENT OF PHI HAS
C          VANISHED THROUGHOUT THE INTERVAL
C
C     USED IN CONJUNCTION WITH TRAPPD--IROUTE = 1
C       OR IN CONJUNCTION WITH RKFST --IROUTE = 2  IF PHI VANISHES AT
C                                                  INITIAL CONDITIONS
C-----------------------------------------------------------------------
      IMPLICIT REAL*8 (A-H,O-Z)
C
      DIMENSION Y(NEQN),YP(NEQN)
      DIMENSION F1(NEQN),F2(NEQN),F3(NEQN),F4(NEQN),F5(NEQN)
      DIMENSION F6(NEQN),F7(NEQN),F8(NEQN),F9(NEQN)
      DIMENSION PHI0(NPHI),PHIP0(NPHI),PHI2(NPHI)
      DIMENSION PHIL(NPHI),PHIPL(NPHI),PHIR(NPHI),PHIPR(NPHI)
      DIMENSION PHIV(NPHI),PHIPV(NPHI)
      DATA POINTS,NPOINT/4.0D0,4/
C-----------------------------------------------------------------------
C     IF THE USER WISHES TO INCREASE THE NUMBER OF TEST POINTS
C     THROUGHOUT THE INTERVAL, POINTS AND NPOINT NEED TO BE CHANGED.
C     SUGGESTED VALUES ARE POINTS = 4.0D0, NPOINT = 4
C     (NPOINT IS THE INTEGER VALUE OF POINTS.)
C-----------------------------------------------------------------------
      DATA IPRINT/0/
      DATA ISCAL1,ISCAL2/1,2/
      EXTERNAL F,SUBPHI
      LOGICAL UPDATE,IVAN,EVALF,BOUNCE
C
C
      TOLER = ABSER
      STEP = TF - T
C
```

```
      IF (IROUTE .EQ. 2)  GO TO 23
C
C-----------------------------------------------------------------------
C
C     CHECK TO SEE IF ANY COMPONENT OF PHI HAS VANISHED INITIALLY
C           AND FINALLY
C
C-----------------------------------------------------------------------
C
C     CHECK EACH COMPONENT OF PHI TO SEE IF ANY HAVE VANISHED
C           INITIALLY AND FINALLY
C
C     ITEST = 0  IF NO PHI(J) VANISHED INITIALLY AND FINALLY
C           = 1  IF AT LEAST ONE PHI(J) VANISHED INITIALLY AND FINALLY
C
C
      ITEST = 0
      DO 22 J = 1,NPHI
C
      IF (DABS(PHI2(J)) .GT.  TOLER)  GO TO 21
C
C     PHI(J) VANISHED AT T2. DID PHI0(J) ALSO VANISH ?
C
      IF (DABS(PHI0(J)) .GT. TOLER)  GO TO 21
C
C     PHI(J) VANISHED INITIALLY AND "FINALLY" FOR COMPONENT J
C
      PHIPL(J) = 1.0D0
      ITEST = 1
      GO TO 22
C
   21 CONTINUE
      PHIPL(J) = -1.0D0
   22 CONTINUE
C
      IF (ITEST .EQ. 0)  RETURN
C
   23 CONTINUE
C
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
C
C     PHI NEEDS TO BE STUDIED THROUGHOUT THE INTERVAL
C
C     A COMPONENT OF PHI HAS VANISHED INITIALLY AND FINALLY
C
C     OR A COMPPONENT OF PHI HAS VANISHED AT THE INITIAL CONDITIONS
C
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
C
C
      DIST  =  T2 - TLSTAR
      IPART = 1
      IF (.NOT. EVALF)    CALL SCALED(F,NEQN,Y,YP,T,1.0D0,STEP,
     1           F2,F3,F4,F5,F6,F7,F8,F9,F1,ISCAL1,NFE,NEXTRA,F2)
      EVALF = .TRUE.
C
```

```
      SIGINC = 1.0D0/POINTS
      FRACT = SIGINC
C
   24 CONTINUE
C
      TSTAR = TLSTAR + FRACT * DIST
      SIGMA = (TSTAR - T) / STEP
C
      CALL SCALED(F,NEQN,Y,YP,T,SIGMA,STEP,F2,F3,F4,F5,F6,F7,F8,F9,F1,
     1            ISCAL2,NFE,NEXTRA,F2)
C
      CALL F(TSTAR,F2,F3)
C
      NFE = NFE + 1
      NEXTRA = NEXTRA + 1
C
      UPDATE = .FALSE.
      BOUNCE = .FALSE.
      IVAN   = .FALSE.
      INDX = INDEX
      CALL SUBPHI(NPHI, INDX,NEQN,TSTAR,F2,F3,PHIV,PHIPV,KOUNTR,
     1            UPDATE,IVAN,BOUNCE,TOLER)
      TOLER = ABSER
      DO 25 J = 1,NPHI
      IF (DABS(PHIV(J)) .LT. ZAPP)  PHIV(J) = DSIGN(ZAPP,PHIV(J))
   25 CONTINUE
C
      IF (IROUTE .EQ. 2)  RETURN
C
C          A PHI FUNCTION HAS VANISHED AT TLSTAR AND T2
C          SEE IF THE SUBINTERVAL HAS ALTERED
C          THE END POINT "VANISHING" STATUS
C
      ITEST = 0
      DO 30 J = 1,NPHI
      IF (PHIPL(J) .LT. 0.0D0)  GO TO 30
C
C     PHI(J) HAS VANISHED THROUGHOUT THE INTERVAL UP TO T*
C
      IF (DABS(PHIV(J)) .GT. TOLER)  GO TO 28
      IF (IPRINT .EQ. 1)  PRINT 1503,J,PHIV(J)
 1503 FORMAT(' COMPONENT J VANISHED--J,PHI = ',I3,2X,D15.7)
C
      ITEST = 1
      GO TO 30
C
   28 CONTINUE
      PHIPL(J) = -1.0D0
C
   30 CONTINUE
C
      IF (ITEST .EQ. 0)  RETURN
C
C     A COMPONENT OF PHI STILL APPEARS TO VANISH WITHIN THE INTERVAL
C                 CONTINUING ITERATING
C
   48 CONTINUE
C
```

```
C
C     INCREMENT SIGMA AND CONTINUE INTERVAL STUDY IF SIGMA < 1
C
      IPART = IPART + 1
      FRACT = FRACT + SIGINC
C
      IF (IPART .LT. NPOINT)   GO TO 24
C
   50 CONTINUE
C
      RETURN
      END
C-----------------------------------------------------------------------
C
C     SUBROUTINE FOR DETERMINING RK45 SOLUTION AT INTERMEDIATE
C          POINTS WITHIN A GIVEN INTEGRATION STEP
C
C-----------------------------------------------------------------------
C
      SUBROUTINE SCALED(F,NEQN,Y,YP,T,SIGMA,STEP,F2,F3,F4,F5,F6,F7,F8,
     1                  F9,F10,ISCALE,NFE,NEXTRA,Y2)
C
      IMPLICIT REAL*8 (A-H,O-Z)
C
      DIMENSION Y(NEQN),YP(NEQN),Y2(NEQN)
      DIMENSION F2(NEQN),F3(NEQN),F4(NEQN),F5(NEQN),F6(NEQN),F7(NEQN)
      DIMENSION F8(NEQN),F9(NEQN),F10(NEQN)
C
      DIMENSION CC6(4),CC7(4),CC8(4),CC9(4),CC10(4)
C
      LOGICAL FIFTH
      DATA FIFTH/.FALSE./
C
      DATA A6,A7,A8,A9,A10/0.24D0,0.75D0,0.58D0,0.975D0,0.58D0/
C
      DATA B60,B63,B64,B65,B70,B73,B74,B75,B76/
     1           .1447948800000000D+00,  -.2768699345454545D+00,
     2           .1966694400000000D+00,   .1754056145454545D+00,
     3           .2223303125000000D-01,   .4967529346590909D+00,
     4          -.3255963750000000D+00,   .5661040909090909D-01,
     5           .5000000000000000D+00/
C
      DATA B80,B83,B84,B85,B86,B87/
     1           .3882639666666667D-01,  -.4846961481818182D+00,
     2           .2153539066666667D+00,  -.1894841551515152D+00,
     3           .5000000000000000D+00,   .5000000000000000D+00/
C
      DATA B90,B93,B94,B95,B96,B97,B98/
     1           .4462967382812500D-01,  -.3201639884588068D+00,
     2           .2211626640625000D+00,  -.4706283494318182D+00,
     3           .5000000000000000D+00,   .5000000000000000D+00,
     4           .5000000000000000D+00/
C
      DATA B100,B103,B104,B105,B106,B107,B108,B109/
     1           .5064642791666667D-01,  -.1001758849886363D+01,
     2           .1096885316666667D+00,  -.5785761096969697D+00,
     3           .5000000000000000D+00,   .5000000000000000D+00,
     4           .5000000000000000D+00,   .5000000000000000D+00/
```

```
C
      DATA CC6/ .6932920226914295D+01,  -.1887192785610976D+02,
     1          .1883923504042140D+02,  -.6538563137673370D+01/
      DATA CC7/ .4638216070742022D+01,  -.2138664616173266D 02,
     1          .3067196377461660D+02,  -.1366995599982912D 02/
      DATA CC8/-.2293180126824485D+01,   .6848856763028249D+01,
     1         -.4867906461976630D+01,   .4523758829678437D+00/
      DATA CC9/-.8195943096485590D+00,   .3947241560261452D+01,
     1         -.6162658362778917D+01,   .3140208082944671D+01/
      DATA CC10/-.4333472382845521D+01,   .2197991355681816D+02,
     1          -.3223001998104773D+02,   .1465110562055352D+02/
C
      EXTERNAL F
C
      IF (ISCALE .EQ. 2)  GO TO 50
C
      IF (FIFTH)  GO TO 22
C
C     FOR THE FOURTH ORDER, SCALED SOLUTION COMPUTE F6 AND THEN
C     STORE F2 AND F3 IN F7 AND F8 LOCATIONS
C
      TIME = T +  STEP
C
      DO 16 J = 1,NEQN
      F10(J) = Y(J) + STEP * ( YP(J) + F4(J) + 4.0D0*F5(J))/6.0D0
   16 CONTINUE
C
      CALL F(TIME,F10,F6)
C
C
      NFE = NFE + 1
      NEXTRA = NEXTRA + 1
C
      DO 20 J = 1,NEQN
      F7(J) = F2(J)
      F8(J) = F3(J)
   20 CONTINUE
C
      RETURN
C
   22 CONTINUE
C
C
C     FIFTH ORDER, SCALED SOLUTION
C     COMPUTE F6,F7,F8,F9,F10 AND STORE F10 IN F1 LOCATION
C
      TIME = T + A6 * STEP
C
      DO 26 J = 1,NEQN
      F10(J) = Y(J) + STEP * ( B60 * YP(J) + B63 * F3(J) + B64 * F4(J)
     1      + B65 * F5(J))
   26 CONTINUE
C
      CALL F(TIME,F10,F6)
C
      TIME = T + A7 * STEP
C
      DO 27 J = 1,NEQN
```

```
      F10(J) = Y(J) + STEP * ( B70 * YP(J) + B73 * F3(J) + B74 * F4(J)
     1      + B75 * F5(J) + B76 * F6(J))
   27 CONTINUE
C
      CALL F(TIME,F10,F7)
C
      TIME = T + A8 * STEP
C
      DO 28 J = 1,NEQN
      F10(J) = Y(J) + STEP * ( B80 * YP(J) + B83 * F3(J) + B84 * F4(J)
     1      + B85 * F5(J) + B86 * F6(J) + B87 * F7(J))
   28 CONTINUE
C
      CALL F(TIME,F10,F8)
C
      TIME = T + A9 * STEP
C
      DO 29 J = 1,NEQN
      F10(J) = Y(J) + STEP * ( B90 * YP(J) + B93 * F3(J) + B94 * F4(J)
     1      + B95 * F5(J) + B96 * F6(J) + B97 * F7(J) + B98 * F8(J))
   29 CONTINUE
C
      CALL F(TIME,F10,F9)
C
      TIME = T + A10 * STEP
C
      DO 30 J = 1,NEQN
      F3(J) = Y(J) + STEP * (B100 * YP(J) + B103 * F3(J) + B104 * F4(J)
     1      + B105 * F5(J) + B106 * F6(J) + B107 * F7(J) + B108 * F8(J)
     2      + B109 * F9(J))
   30 CONTINUE
C
      CALL F(TIME,F3,F10)
C
C
      NFE = NFE + 5
      NEXTRA = NEXTRA + 5
      RETURN
C
   50 CONTINUE
C
      SST = SIGMA*STEP
C
      IF (FIFTH)  GO TO 55
C
C     FORM C-COEFFICIENTS FOR GIVEN SIGMA, FOURTH ORDER SOLUTION
C
C
      C2 = SIGMA*((7168.0D0/1425.0D0) + SIGMA*((-4096.0D0/513.0D0)
     1                                + SIGMA*(14848.0D0/4275.0D0) ))
C
      C3 = SIGMA*((-28561.0D0/8360.0D0)+ SIGMA*((199927.0D0/22572.0D0)
     1                                  + SIGMA*(-371293.0D0/75240.0D0)))
C
      C4 = SIGMA*((57.0D0/50.0D0) + SIGMA*(-3.0D0
     1                            + SIGMA*(42.0D0/25.0D0) ))
C
      C5 = SIGMA*((-96.0D0/55.0D0) + SIGMA*((40.0D0/11.0D0)
```

```
     1                              + SIGMA*(-102.0D0/55.0D0) ))
C
      C6 = SIGMA*( 1.5D0 + SIGMA*(-4.0D0 + SIGMA*2.5D0 ))
C
C
      C0 = 1.0D0 - (SIGMA*((301.0D0/120.0D0) + SIGMA*((-269.0D0/108.0D0)
     1           +  SIGMA*(311.0D0/360.0D0) )))
C
C
C     EVALUATE THE FOURTH ORDER, SCALED SOLUTION
C
      DO 51 J = 1,NEQN
      Y2(J) = Y(J) + SST * (C0*YP(J) + C2*F7(J) + C3*F8(J) + C4*F4(J)
     1                      + C5*F5(J) + C6*F6(J))
   51 CONTINUE
C
C
C
      RETURN
C
   55 CONTINUE
C
C     FORM C-COEFFICIENTS FOR GIVEN SIGMA, FIFTH ORDER SOLUTION
C
C
      C6 = SIGMA*(CC6(1) + SIGMA*(CC6(2) + SIGMA*(CC6(3)
     1                      + SIGMA*CC6(4))))
C
      C7 = SIGMA*(CC7(1) + SIGMA*(CC7(2) + SIGMA*(CC7(3)
     1                      + SIGMA*CC7(4))))
C
      C8 = SIGMA*(CC8(1) + SIGMA*(CC8(2) + SIGMA*(CC8(3)
     1                      + SIGMA*CC8(4))))
C
      C9 = SIGMA*(CC9(1) + SIGMA*(CC9(2) + SIGMA*(CC9(3)
     1                      + SIGMA*CC9(4))))
C
      C10 = SIGMA*(CC10(1) + SIGMA*(CC10(2) + SIGMA*(CC10(3)
     1                      + SIGMA*CC10(4))))
C
      C0 = 1.0D0 - (C6 + C7 + C8 + C9 + C10)
C
C     EVALUATE THE SOLUTION
C
      DO 73 J = 1,NEQN
      Y2(J) = Y(J) + SST * (C0*YP(J) + C6*F6(J) + C7*F7(J) + C8*F8(J)
     1                      + C9*F9(J) + C10*F10(J))
   73 CONTINUE
C
C
      RETURN
      END
C
C    CHANGES TO BOUNCING FUNCTION ANALYSIS--5.5.82
C
C-----------------------------------------------------------------------
      SUBROUTINE PANIC(F,SUBPHI,NPHI,NEQN,NFE,INDEX,
     1   T,Y,YP,PHI0,PHIP0,TF,YF,YPF,PHIF,PHIPF,T2,Y2,YP2,
```

```
     2   PHI2,PHIP2,PHIL,PHIPL,PHIR,PHIPR,TL,TR,PHIB,
     3   F1,F2,F3,F4,F5,F6,F7,F8,F9,
     4   ABSER,KOUNTR,NEXTRA,EVALF,FIND,IROUTE,IWARN,U26)
C-----------------------------------------------------------------------
C
C     PANIC SHOULD BE USED ONLY IN CRISIS (OR NEAR CRISIS) SITUATIONS
C
C     PANIC IS REFERENCED BY RKF45T (TRAPPD) WHEN THE TRAPPING PROCEDURE
C     FAILS TO CONVERGE.
C
C     THE USER MAY ACTIVATE THE USE OF PANIC IN TWO MODES BY CHANGING
C     THE PARAMETER NOTFAL IN EITHER SETRAP OR TRAPPD. IN BOTH ROUTINES
C     NOTFAL IS .FALSE. FOR NORMAL (NON-CRISIS) SITUATIONS.
C
C
C     AN ADDITIONAL MODE OF PANIC IS AVAILABLE TO CHECK FOR ERRORS
C     CAUSED BY BOUNCING FUNCTIONS.  THE USER IS NOT CONCERNED WITH
C     THIS MODE OF PANIC.
C
C-----------------------------------------------------------------------
C     PANIC--REFERENCED BY TRAPPD
C-----------------------------------------------------------------------
C     PANIC  PRINTS INFORMATION ABOUT PHI THROUGHOUT ANY STEP ON WHICH
C            TRAPPD HAS BEEN ACTIVATED.  INFORMATION ABOUT THE FIRST
C            VALUE OF T FOR WHICH PHI HAS CHANGED SIGNS IS RETURNED TO
C            TRAPPD FOR USE IN THE ITERATION PROCESS.
C
C            PRINT OPTIONS:  IPRINT = 1   T AND PHI ARE PRINTED AT EACH
C                                         SUBSTEP
C                            IPRINT = 2   T, Y, YP, PHI, PHIP ARE
C                                         PRINTED AT EACH SUBSTEP
C
C                            IPRINT = --  ANY OTHER VALUE WILL BE RESET
C                                         TO IPRINT=1
C-----------------------------------------------------------------------
C     PANIC--REFERENCED BY RKF45T
C-----------------------------------------------------------------------
C     PANIC  CHECKS THE PHI FUNCTION THROUGHOUT THE INTERVAL FOR
C            POSSIBLE MULTIPLE VANISHING POINTS WITHIN A GIVEN STEP.
C            THE CHECK IS MADE AFTER EACH INTEGRATION STEP WHETHER
C            OR NOT THE ANALYSIS INDICATED THAT TRAPPD SHOULD BE
C            REFERENCED.  PRINTING IS OPTIONAL, ALTHOUGH A WARNING
C            MESSAGE IS PRINTED ON THE FIRST CALL TO PANIC.
C
C
C            PRINT OPTIONS:  JPRINT = 0   PRINTING IS SUPPRESSED
C                            JPRINT = 1   T AND PHI ARE PRINTED AT
C                                         EACH SUBSTEP
C                            JPRINT = 2   T, Y, YP, PHI, PHIP ARE
C                                         PRINTED AT EACH SUBSTEP
C-----------------------------------------------------------------------
C     PANIC--REFERENCED IN "BOUNCING MODE"
C-----------------------------------------------------------------------
C            A ZERO HAS BEEN DETECTED, AND TRAPPED WILL BE REFERENCED.
C            IF THE PREVIOUS ZERO WAS A "BOUNCING" ZERO, THE INCORRECT
C            SIGN IMPOSED INDICATES A ZERO WHICH DOES NOT EXIST.  PANIC
C            WILL DETECT ANY BOUNCING COMPONENT SO THAT THE SIGN CAN
C            BE CORRECTED.
```

```
C
C            NO PRINTING OPTIONS ARE ASSOCIATED WITH THE BOUNCING
C            ANALYSIS.
C-----------------------------------------------------------------------
C     POINT SPACING IN PANIC:
C-----------------------------------------------------------------------
C
C     THE STANDARD NUMBER OF OUTPUT POINTS IS 10.  THE USER MAY
C     INCREASE OR DECREASE THIS NUMBER  BY CHANGING THE PARAM-
C     ETERS POINTS AND NPOINT IN THE GIVEN DATA STATEMENT.
C
C-----------------------------------------------------------------------
C
      IMPLICIT REAL*8 (A-H,O-Z)
C
      DIMENSION Y(NEQN),YP(NEQN),Y2(NEQN),YP2(NEQN)
      DIMENSION YF(NEQN),YPF(NEQN),F1(NEQN),F2(NEQN)
      DIMENSION F3(NEQN),F4(NEQN),F5(NEQN)
      DIMENSION F6(NEQN),F7(NEQN),F8(NEQN),F9(NEQN)
      DIMENSION PHI0(NPHI),PHIP0(NPHI),PHIF(NPHI),PHIPF(NPHI)
      DIMENSION PHI2(NPHI),PHIP2(NPHI),PHIL(NPHI),PHIPL(NPHI)
      DIMENSION PHIR(NPHI),PHIPR(NPHI),PHIB(NPHI)
C
      LOGICAL UPDATE,IVAN,EVALF,FIND,BOUNCE
C-----------------------------------------------------------------------
      DATA POINTS,NPOINT/10.0D0,10/
      DATA IPRINT,JPRINT/0,0/
      DATA JOPT/1/
C-----------------------------------------------------------------------
      DATA ISCALE,JSCALE/1,2/
C
      EXTERNAL F,SUBPHI
C
C-----------------------------------------------------------------------
C     SET SUB-STEP LENGTH AND T2, THE FIRST POINT TO BE ANALYZED.
C         (IF IROUTE=1, RETURN TO SETRAP--ONLY THE SUB-STEP LENGTH IS
C          SOUGHT.)
C-----------------------------------------------------------------------
C
      DSIG = 1.0D0/POINTS
      STEP = TF - T
      T2 = T + DSIG*STEP
      IF (IROUTE .EQ. 0)  RETURN
C-----------------------------------------------------------------------
C     PRINTING PARAMETER SAFETY CHECKS:
C-----------------------------------------------------------------------
      IPR = IPRINT
      IF (IPR .LT. 1 .OR. IPR .GT. 2)  IPR = 1
      JPR = JPRINT
      IF (JPR .LT. 0 .OR. JPR .GT. 2)  JPR = 1
C
      IF (IROUTE .EQ. 3)  GO TO 600
C-----------------------------------------------------------------------
C     A STANDARD PANIC OPTION IS BEING USED
C        (FROM EITHER SETRAP OR TRAPPD)
C-----------------------------------------------------------------------
C     INITIALIZATION AND SAFETY CHECKS:
C-----------------------------------------------------------------------
```

```
      IF (IWARN .EQ. 1)  PRINT 900
  900 FORMAT(/,' THE PANIC OPTION IS BEING USED AT EACH STEP IN RKF45T'
     1       ,/,' TO CHECK FOR MULTIPLE ZEROS OF A PHI COMPONENT WITHIN'
     2         ,' A GIVEN STEP',//,' THIS IS VERY INEFFICIENT.',//)
C-----------------------------------------------------------------------
C     PRINTING BLOCK:
C-----------------------------------------------------------------------
      IF (IROUTE .EQ. 2 .AND. JPR .EQ. 0)  GO TO 10
      PRINT 901
      PRINT 902,INDEX
      PRINT 903,T
      PRINT 906,PHI0
      IF (IROUTE .EQ. 2 .AND. JPR .EQ. 1)  GO TO 10
      IF (IROUTE .EQ. 1 .AND. IPR .EQ. 1)  GO TO 10
      PRINT 904,Y
      PRINT 905,YP
C-----------------------------------------------------------------------
   10 CONTINUE
      INDEX = 0
      NEXT = 0
C-----------------------------------------------------------------------
C     GENERATE THE ADDITIONAL F EVALUATIONS IF NOT ALREADY DETERMINED:
C-----------------------------------------------------------------------
      IF (.NOT. EVALF) CALL SCALED(F,NEQN,Y,YP,T,SIGMA,STEP,F2,
     1                F3,F4,F5,F6,F7,F8,F9,F1,ISCALE,NFE,NEXT,Y2)
      EVALF = .TRUE.
C
C-----------------------------------------------------------------------
C     SUB-STEP POINTS WILL BE LABELED "L" UNTIL A ZERO IS TRAPPED
C       I.E., POINT "L" WILL BE MOVED TOWARDS "F" AS A ZERO IS SOUGHT.
C
C-----------------------------------------------------------------------
C     PHIPL IS USED AS A TEMPORARY INDICATOR AND WILL BE RESET BEFORE
C       RETURNING TO SETRAP OR TRAPPD
C
C     STORE VANISHING INFORMATION IN PHIPL:
C           PHIPL > 0 ,  COMPONENT DID NOT VANISH OR CHANGE SIGN
C           PHIPL < 0 ,  COMPONENT EITHER VANISHED OR CHANGED SIGN
C-----------------------------------------------------------------------
      DO 12 J = 1,NPHI
      PHIL(J) = PHI0(J)
      PHIPL(J) = +1.0D0
   12 CONTINUE
      TL = T
C
      DO  36 JJ= 1,NPOINT
C-----------------------------------------------------------------------
C     EVALUATE THE SOLUTION AND PHI FUNCTION AT THE JJTH SUBSTEP:
C-----------------------------------------------------------------------
      SIGMA = DSIG*DFLOAT(JJ)
      CALL SCALED(F,NEQN,Y,YP,T,SIGMA,STEP,F2,F3,F4,F5,F6,F7,F8,F9,F1,
     1            JSCALE,NFE,NEXT,F2)
      T2 = T + SIGMA*STEP
      IF (JJ .EQ. NPOINT)  T2 = TF
      CALL F(T2,F2,F3)
      NFE = NFE + 1
C
      UPDATE = .FALSE.
```

```
      BOUNCE = .FALSE.
      IVAN   = .FALSE.
      TOLER = ABSER
      INDX = INDEX
      CALL SUBPHI(NPHI,INDX,NEQN,T2,F2,F3 ,PHI2,PHIP2,KOUNTR,UPDATE,
     1         IVAN,BOUNCE,TOLER)
      TOLER = ABSER
      DO 15 KK = 1,NPHI
      IF (DABS(PHI2(KK))  .LT.  U26)  PHI2(KK) =DSIGN(U26,PHI2(KK))
      IF (DABS(PHIP2(KK)) .LT.  U26)  PHIP2(KK)=DSIGN(U26,PHIP2(KK))
   15 CONTINUE
      UPDATE = .FALSE.
C-----------------------------------------------------------------------
      IF (FIND)  GO TO 29
C-----------------------------------------------------------------------
C     NO ZERO HAS BEEN ISOLATED YET.  CHECK THE CURRENT POINT:
C-----------------------------------------------------------------------
      DO 18 I = 1,NPHI
      IF (.NOT. (DABS(PHI2(I))   .LT. TOLER .OR.
     1           PHI2(I)*PHIL(I) .LT. 0.0D0) )     GO TO 18
C
C     PHI2(I) HAS VANISHED OR CHANGED SIGN OVER THE SUBSTEP,
C     SET FIND=.TRUE. AND SET PHIPL(I) INDICATOR
C
      FIND = .TRUE.
      PHIPL(I) = -1.0D0
   18 CONTINUE
C
      IF (FIND)  GO TO 25
C-----------------------------------------------------------------------
C     NO ZERO HAS BEEN ISOLATED ON THIS STEP EITHER.
C     CONTINUE ITERATING--THIS STEP WILL NOT SET TRAPPING BOUNDS.
C-----------------------------------------------------------------------
      TL = T2
      DO 23 J = 1,NEQN
      Y2(J) = F2(J)
   23 YP2(J) = F3(J)
      DO 24 J = 1,NPHI
      PHIL(J) = PHI2(J)
   24 CONTINUE
      GO TO 29
C
   25 CONTINUE
C
C-----------------------------------------------------------------------
C     A ZERO WAS TRAPPED ON THIS SUBSTEP.  SET TRAPPING BOUNDS (CONDI-
C     TIONS AT "R".  (IF IROUTE=2, THE ANALYSIS WILL BE RETURNED TO
C     SETRAP WITH THE ZERO BRACKETED.)
C-----------------------------------------------------------------------
      TR = T2
      IF (JJ .EQ. NPOINT)  TR = TF
      DO 26  J = 1,NPHI
      PHIR(J) = PHI2(J)
      PHIPR(J) = PHIP2(J)
   26 CONTINUE
C
   29 CONTINUE
C-----------------------------------------------------------------------
```

```
C     STANDARD PRINTING BLOCK:
C-----------------------------------------------------------------------
      IF (IROUTE .EQ. 2 .AND. JPR .EQ. 0)  GO TO 33
      PRINT 903,T2
      PRINT 906,PHI2
      PRINT 907,PHIP2
      PRINT 1588,FIND
 1588 FORMAT(' FIND = ',L2)
      IF (IROUTE .EQ. 2 .AND. JPR .EQ. 1)  GO TO 33
      IF (IROUTE .EQ. 1 .AND. IPR .EQ. 1)  GO TO 33
      PRINT 904,Y
      PRINT 905,YP
C
   33 CONTINUE
C-----------------------------------------------------------------------
C     IF IROUTE=2 AND A ZERO WAS BRACKETED, EXIT DO LOOP (DO 36 ...)
C-----------------------------------------------------------------------
      IF (IROUTE .EQ. 2 .AND. FIND)  GO TO 42
C
   36 CONTINUE
C-----------------------------------------------------------------------
C     END OF DO LOOP FOR CHECKING SUBSCRIPTS.  (EARLY EXIT OCCURS
C     IF IROUTE=2 AND A ZERO IS BRACKETED DURING A SUB-STEP.)
C-----------------------------------------------------------------------
C     STANDARD PRINTING BLOCK:
C-----------------------------------------------------------------------
      IF (IROUTE .EQ. 2 .AND. JPR .EQ. 0)  GO TO 40
      PRINT 908
      PRINT 903,TF
      PRINT 906,PHIF
      PRINT 907,PHIPF
      IF (IROUTE .EQ. 2 .AND. JPR .EQ. 1)  GO TO 40
      IF (IROUTE .EQ. 1 .AND. IPR .EQ. 1)  GO TO 40
      PRINT 904,YF
      PRINT 905,YPF
   40 CONTINUE
C-----------------------------------------------------------------------
C     PREPARE FOR EXIT:
C-----------------------------------------------------------------------
      IF (FIND)  GO TO 42
C-----------------------------------------------------------------------
C     NO ZERO WAS DETECTED.  RESET CONDITIONS AT "0" WHICH HAVE BEEN
C        ALTERED DURING THE PANIC ANALYSIS.
C-----------------------------------------------------------------------
      DO 41 J = 1,NPHI
      PHIL(J) = PHI0(J)
      PHIPL(J) = PHIP0(J)
      PHI2(J) = PHI0(J)
      PHIP2(J) = PHIP0(J)
      PHIR(J) = PHIF(J)
      PHIPR(J) = PHIPF(J)
   41 CONTINUE
      TL = T
      T2 = T
      TR = TF
      RETURN
C
   42 CONTINUE
```

```
C-----------------------------------------------------------------------
C     A ZERO WAS DETECTED.  SET BRACKETING CONDITIONS "L", "2", AND
C        "R" BEFORE RETURNING.
C-----------------------------------------------------------------------
C     DETERMINE INDEX--CONSIDER ONLY COMPONENTS WHO HAVE VANISHED
C       AT CURRENT JJ VALUE OR HAVE CHANGED OVER THE LAST SUBSTEP
C       (DESIGNATED BY PHIPL < 0)
C-----------------------------------------------------------------------
      INDEX = 0
      PHIMAX = 0.0D0
      DO 45 I = 1,NPHI
      IF (PHIPL(I) .GT. 0.0D0)  GO TO 45
      IF (DABS(PHIR(I)) .LT. TOLER)  GO TO 43
      IF (PHIR(I)*PHIL(I) .GT. 0.0D0) GO TO 45
   43 CONTINUE
C
C     COMPONENT "I" HAS VANISHED OR CHANGED SIGNS AT T2--CHECK TO SEE
C        IF INDEX SHOULD BE SHIFTED TO "I"
C
      IF (DABS(PHIR(I)) .LT. PHIMAX)  GO TO 45
      INDEX = I
      PHIMAX = DABS(PHIR(I))
   45 CONTINUE
C
      IF (TL .EQ. T)  GO TO 66
C-----------------------------------------------------------------------
C     BRACKETING DID NOT OCCUR ON THE FIRST SUB-STEP.  THE SOLUTION
C     VECTOR AT TL HAS BEEN "LOST".  GENERATE CONDITIONS AT "TL".
C-----------------------------------------------------------------------
      T2 = TL
      SIGMA = (TL - T)/(TF - T)
      CALL SCALED(F,NEQN,Y,YP,TL,SIGMA,STEP,F2,F3,F4,F5,F6,F7,F8,F9,F1,
     1            JSCALE,NFE,NEXT,Y2)
      CALL F(TL,Y2,YP2)
      NFE = NFE + 1
C
      UPDATE = .FALSE.
      BOUNCE = .FALSE.
      IVAN   = .FALSE.
      CALL SUBPHI(NPHI,INDX,NEQN,TL,Y2,YP2,PHI2,PHIP2,KOUNTR,UPDATE,
     1         IVAN,BOUNCE,TOLER)
      TOLER = ABSER
C
      DO 65 I = 1,NPHI
      PHIL(I) = PHI2(I)
      IF (DABS(PHIL(I)) .LT. U26)  PHIL(I) = DSIGN(U26,PHIL(I))
   65 PHIPL(I) = PHIP2(I)
      RETURN
C
C-----------------------------------------------------------------------
C     BRACKETING OCCURRED ON THE FIRST SUB-STEP.  THE SOLUTION AT TL
C     IS STILL KNOWN.  ONLY PHIL, PHIPL, PHI2,PHIP2 NEED TO BE SET.
C-----------------------------------------------------------------------
   66 CONTINUE
      DO 67 I = 1,NPHI
      PHIL(I) = PHI0(I)
      PHIPL(I) = PHIP0(I)
      PHI2(I) = PHI0(I)
```

```
   67 PHIP2(I) = PHIP0(I)
      T2 = T
C
      RETURN
C
  600 CONTINUE
C
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
C     SPECIAL LOOP FOR INVESTIGATING POSSIBLE "BOUNCING FUNCTIONS"
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
C     USE PHIP0(I) TO STORE "BOUNCING" INFORMATION--THEN
C         RECOMPUTE IT BEFORE RETURNING TO TRAPPD
C         PHIP0(I) < 0,   NO BOUNCING CANDIDATES EXIST
C         PHIP0(I) > 0,   AT LEAST ONE BAOUNCING CANDIDATE EXISTS
C
C         (CRITERION FOR "BOUNCING", |PHIL(I)|<TOL, |PHIPR(I)| > TOL
C          AND PHIL(I)*PHIR(I) < 0 (SIGN CHANGE)
C
C     PHIB(I) CONTAINS INFORMATION ABOUT THE "BOUNCING" STATUS OF
C         COMPONENT "I"
C         PHIB(I) < 0,    NO BOUNCING INDICATED YET
C         PHIB(I) > 0,    BOUNCING INDICATED
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
      TOLER = ABSER
C-----------------------------------------------------------------------
      DO 602 I = 1,NPHI
      PHIB(I) = -1.0D0
      PHIP0(I) = -1.0D0
      IF (PHIL(I)*PHIR(I) .GT. 0.0D0)        GO TO 602
      IF (DABS(PHIL(I)) .LT. TOLER .AND.
     1    DABS(PHIR(I)) .GT. TOLER       )   PHIP0(I) = 1.0D0
  602 CONTINUE
C-----------------------------------------------------------------------
C     DOES A BOUNCING CANDIDATE EXIST ?  (IS PHIP)(I) > 0, FOR ANY "I"?)
C-----------------------------------------------------------------------
      ITEST = 0
      DO 603 I = 1,NPHI
      IF (PHIP0(I) .GT. 0.0D0)  ITEST = 1
  603 CONTINUE
C
      IF (ITEST .EQ. 0)  RETURN
C
C-----------------------------------------------------------------------
C     AT LEAST ONE COMPONENT MUST BE STUDIED FOR "BOUNCING" DIFFICULTIES
C     GENERATE ADDITIONAL F EVALUATIONS IF NOT YET DETERMINED
C-----------------------------------------------------------------------
      IF (.NOT. EVALF)    CALL SCALED(F,NEQN,Y,YP,T,SIGMA,STEP,F2,
     1                    F3,F4,F5,F6,F7,F8,F9,F1,ISCALE,NFE,NEXTRA,Y2)
      EVALF = .TRUE.
C-----------------------------------------------------------------------
      DO  620 JJ= 1,NPOINT
C-----------------------------------------------------------------------
C     SIGMA IS A FRACTION OF THE INTEGRATION STEP.  IF THE BRACKETING
C     INDICES WERE DETERMINED BY "PANIC" ANALYSIS, SIGMA IS A FRACTION
C     OF THE INTEGRATION "SUB-STEP" FROM STANDARD PANIC ANALYSIS.
```

```
C-----------------------------------------------------------------------
      SIGMA = DSIG * DFLOAT(JJ)
      IF (FIND)   SIGMA = (TL + SIGMA*(TR-TL)  - T) / STEP
      IF (FIND .OR. JJ .LT. NPOINT)  GO TO 606
C-----------------------------------------------------------------------
C     "R" = "F" AND THE SOLUTION IS ALREADY KNOWN
C-----------------------------------------------------------------------
      DO 604 I = 1,NEQN
      F2(I) = YF(I)
  604 F3(I) = YPF(I)
      DO 605 I = 1,NPHI
      PHI2(I) = PHIR(I)
      PHIP2(I) = PHIPR(I)
  605 CONTINUE
      T2 = TR
      GO TO 607
C-----------------------------------------------------------------------
  606 CONTINUE
C-----------------------------------------------------------------------
C     "R" WAS DETERMINED BY THE PANIC ANALYSIS AND THE SOLUTION HAS
C     BEEN LOST:
C     CALL SCALED TO COMPUTE THE SOLUTION AT SUBSTEP JJ
C-----------------------------------------------------------------------
      CALL SCALED(F,NEQN,Y,YP,T,SIGMA,STEP,F2,F3,F4,F5,F6,F7,F8,F9,F1,
     1            JSCALE,NFE,NEXTRA,F2)
      T2 = T + SIGMA*STEP
      CALL F(T2,F2,F3)
      NFE = NFE + 1
C
      UPDATE = .FALSE.
      BOUNCE = .FALSE.
      IVAN   = .FALSE.
      INDX = INDEX
      CALL SUBPHI(NPHI,INDX,NEQN,T2,F2,F3,PHI2,PHIP2,KOUNTR,UPDATE,
     1         IVAN,BOUNCE,TOLER)
      TOLER  = ABSER
C
C-----------------------------------------------------------------------
  607 CONTINUE
C
C-----------------------------------------------------------------------
C     CHECK PHI COMPONENTS FOR "BOUNCING" CORRECTIONS AT SUBSTEP JJ
C-----------------------------------------------------------------------
C
      DO 608 I = 1,NPHI
      IF (PHIP0(I) .LT. 0.0D0)  GO TO 608
C
      IF (DABS(PHI2(I)) .LT. TOLER)                    GO TO 608
C
C     PHI(I) IS OUT OF THE VANISHED REGION--DOES IT HAVE THE SAME
C         SIGN AS PHIR(I) --I.E., DOES IT "BOUNCE" ?
C     IF PHI(I) BOUNCES, SET PHIB(I) = +1.0D0
C
      PHIP0(I) = -1.0D0
      PHIL(I) = DSIGN(PHIL(I),PHI2(I))
      IF (PHI2(I)*PHIR(I) .GT. 0.0D0)  PHIB(I) = +1.0D0
      IF (FIND .AND. T .NE. TL)  GO TO 608
C
```

```
C     NO EMERGENCY FEATURE WAS USED IN SETRAP    OR
C     THE EMERGENCY FEATURE WAS USED IN SETRAP AND A ZERO WAS DETECTED
C     ON THE FIRST SUB-STEP
C
C     ADJUST THE SIGN OF PHI0(J)
C
      PHI0(I) = DSIGN(PHI0(I),PHI2(I))
C
  608 CONTINUE
C-----------------------------------------------------------------------
C
      ITEST = 0
      DO 610 I = 1,NPHI
      IF (PHIP0(I) .GT. 0.0D0)  ITEST = 1
  610 CONTINUE
C
C     IF NO MORE "BOUNCING" CANDIDATES EXIST, EXIT DO LOOP--GO TO 622
C
      IF (ITEST .EQ. 0)  GO TO 622
  620 CONTINUE
C
C     NORMAL EXIT OF DO LOOP--BOUNCING CANDIDATE STILL EXITED ON THE
C                             FINAL SUBSTEP
C
  622 CONTINUE
C
C
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
C     INDEX ANALYSIS
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
C
      INDOLD = INDEX
      INDEX = 0
      PHIMAX = 0.0D0
      DO 704 I = 1,NPHI
      IF (PHIB(I)  .GT. 0.0D0)          GO TO 704
      IF (PHIR(I)*PHIL(I) .GT. 0.0D0)   GO TO 704
      IF (DABS(PHIR(I)) .LT. PHIMAX)    GO TO 704
      PHIMAX = DABS(PHIR(I))
      INDEX = I
  704 CONTINUE
C
      IF (INDEX .NE. INDOLD .AND. JOPT .EQ. 1) PRINT 705,INDOLD,INDEX
  705 FORMAT(' INDEX = ',I3,' WAS INCORRECTLY IMPOSED--NEW INDEX = ',I4)
C
C     RESET CONDITIONS AT "2" EQUAL TO THOSE AT "L"  (Y2 AND YP2
C        HAVE NOT BEEN CHANGED)
C
      DO 710 I = 1,NPHI
      PHI2(I) = PHIL(I)
      PHIP2(I) = PHIPL(I)
  710 CONTINUE
      T2 = TL
C
C-----------------------------------------------------------------------
C
```

```
  901 FORMAT(' ENTERING PANIC')
  902 FORMAT(//,' IN PANIC--COMPONENT OF PHI BEING ANALYZED = ',I3,/)
  903 FORMAT(' T = ',D15.7)
  904 FORMAT(' Y    = ',4(2X,D15.7),/,(8X,4(2X,D15.7)))
  905 FORMAT(' YP   = ',4(2X,D15.7),/,(8X,4(2X,D15.7)))
  906 FORMAT(' PHI  = ',4(2X,D15.7),/,(8X,4(2X,D15.7)))
  907 FORMAT(' PHIP = ',4(2X,D15.7),/,(8X,4(2X,D15.7)))
  908 FORMAT(' LEAVING PANIC')
C
      RETURN
      END
```

APPENDIX B. SUBPHI SUBROUTINES AND RESULTING OUTPUT FOR EXAMPLES IN §10.

Example 1, dense output equal spacing in T

```
      SUBROUTINE DENSE1(NPHI,INDEX,NEQN,T,Y,YP,PHI,PHIP,KOUNTR,UPDATE,
     1                  IVAN,RELER,ABSER)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION Y(4),YP(4),PHI( 1),PHIP( 1)
      LOGICAL UPDATE,IVAN
C
      COMMON/TIME1/TINCR
C
      IF (UPDATE)  GO TO 100
      IF (KOUNTR .GT. 0)  GO TO 12
C
C     INITIALIZATION BLOCK:
      TPR  = TINCR
   12 CONTINUE
C-----------------------------------------------------------------------
      PHI(1) = T - TPR
      PHIP(1) =   1.0D0
C-----------------------------------------------------------------------
      RETURN
  100 CONTINUE
C
C     UPDATE
      PRINT 516,T,Y(1),Y(3),Y(2),Y(4)
      TPR = TPR + TINCR
C
C     IN MULITPLE TRAP MODE, PHI WILL BE UPDATED BY TRAPPD.
C     IN OTHER MODES, THE USER SHOULD REEVALUATE PHI SINCE ITS VALUE
C     CHANGES WITH THE CHANGE IN TPR.
C
  516 FORMAT(/,' INTERMEDIATE OUTPUT--T = ',D15.7,/,22X,
     1       'Y1=',D15.7,2X,'Y2=',D15.7,/,22X,
     2       'Y3=',D15.7,2X,'Y4=',D15.7)
      RETURN
```

```
      END
C
```

```
TWO BODY PROBLEM--ELLIPTIC ORBIT, ECC=0.1
DENSE OUTPUT, TIME INCREMENT =   0.3141593D 00

(some output deleted)

OPERATION PARAMETERS AND BOUNDARY CONDITIONS

RELERR =   0.1000000D-05   ABSERR =   0.1000000D-05   IFLAG =   15
T      =   0.0000000D 00   TF     =   0.6283185D 01
Y(1)   =   0.9000000D 00   Y(2)   =   0.0000000D 00
Y(3)   =   0.0000000D 00   Y(4)   =   0.1105542D 01


INTERMEDIATE OUTPUT--T =   0.3141593D 00
                     Y1=  0.8399578D 00  Y2= -0.3767007D 00
                     Y3=  0.3395789D 00  Y4=  0.1032275D 01

INTERMEDIATE OUTPUT--T =   0.6283185D 00
                     Y1=  0.6698828D 00  Y2= -0.6914165D 00
                     Y3=  0.6349843D 00  Y4=  0.8299182D 00

INTERMEDIATE OUTPUT--T =   0.9424778D 00
                     Y1=  0.4164354D 00  Y2= -0.9029608D 00
                     Y3=  0.8520312D 00  Y4=  0.5418264D 00

INTERMEDIATE OUTPUT--T =   0.1256637D 01
                     Y1=  0.1148030D 00  Y2= -0.9980995D 00
                     Y3=  0.9717582D 00  Y4=  0.2184158D 00

INTERMEDIATE OUTPUT--T =   0.1570796D 01
                     Y1= -0.1993454D 00  Y2= -0.9852670D 00
                     Y3=  0.9900608D 00  Y4= -0.9787924D-01

INTERMEDIATE OUTPUT--T =   0.1884956D 01
                     Y1= -0.4949687D 00  Y2= -0.8837882D 00
                     Y3=  0.9140837D 00  Y4= -0.3780634D 00

INTERMEDIATE OUTPUT--T =   0.2199115D 01
                     Y1= -0.7476645D 00  Y2= -0.7155781D 00
                     Y3=  0.7580980D 00  Y4= -0.6052274D 00

INTERMEDIATE OUTPUT--T =   0.2513274D 01
                     Y1= -0.9397271D 00  Y2= -0.5009346D 00
                     Y3=  0.5402737D 00  Y4= -0.7708017D 00

INTERMEDIATE OUTPUT--T =   0.2827433D 01
                     Y1= -0.1059391D 01  Y2= -0.2573625D 00
                     Y3=  0.2806395D 00  Y4= -0.8710278D 00

INTERMEDIATE OUTPUT--T =   0.3141593D 01
                     Y1= -0.1099992D 01  Y2=  0.2075454D-04
                     Y3= -0.1968451D-04  Y4= -0.9045380D 00
```

```
(Output for 3.14159 < T < 6.2831 has been deleted.)

 INTERMEDIATE OUTPUT--T =   0.6283185D 01
                      Y1=  0.8999798D 00  Y2= -0.1021791D-03
                      Y3=  0.8616911D-04  Y4=  0.1105556D 01
```

Example 1, dense output unequal spacing in T

```
 C
       SUBROUTINE DENSE2(NPHI,INDEX,NEQN,T,Y,YP,PHI,PHIP,KOUNTR,UPDATE,
      1                  IVAN,RELER,ABSER)
       IMPLICIT REAL*8 (A-H,O-Z)
       DIMENSION Y(4),YP(4),PHI( 1),PHIP( 1)
       LOGICAL UPDATE,IVAN
       COMMON/TIME2/TPRINT(50)
 C
       IF (UPDATE)  GO TO 100
 C
       IF (KOUNTR .GT. 0)  GO TO 12
       TPR   = TPRINT(1)
    12 CONTINUE
 C
 C-----------------------------------------------------------------------
       PHI(1) = T - TPR
       PHIP(1) =   1.0D0
 C-----------------------------------------------------------------------
       RETURN
   100 CONTINUE
 C
 C     UPDATE
       PRINT 516,T,Y(1),Y(3),Y(2),Y(4)
       TPR = TPRINT(KOUNTR+1)
 C
 C     IN MULITPLE TRAP MODE, PHI WILL BE UPDATED BY TRAPPD.
 C     IN OTHER MODES, THE USER SHOULD REEVALUATE PHI SINCE ITS VALUE
 C     CHANGES WITH THE CHANGE IN TPR.
 C
   516 FORMAT(/,' INTERMEDIATE OUTPUT--T = ',D15.7,/,22X,
      1       'Y1=',D15.7,2X,'Y2=',D15.7,/,22X,
      2       'Y3=',D15.7,2X,'Y4=',D15.7)
       RETURN
       END
```

```
 TWO BODY PROBLEM--ELLIPTIC ORBIT, ECC=0.1
 DENSE OUTPUT UNEVEN SPACING

 (some output deleted)
```

```
OPERATION PARAMETERS AND BOUNDARY CONDITIONS

RELERR =   0.1000000D-05   ABSERR =   0.1000000D-05   IFLAG =   15
T      =   0.0000000D 00   TF     =   0.6283185D 01
Y(1)   =   0.9000000D 00   Y(2)   =   0.0000000D 00
Y(3)   =   0.0000000D 00   Y(4)   =   0.1105542D 01


INTERMEDIATE OUTPUT--T =   0.1000000D 00
                     Y1=  0.8938362D 00  Y2= -0.1230922D 00
                     Y3=  0.1103019D 00  Y4=  0.1097975D 01

INTERMEDIATE OUTPUT--T =   0.2200000D 00
                     Y1=  0.8703366D 00  Y2= -0.2677368D 00
                     Y3=  0.2405447D 00  Y4=  0.1069224D 01

INTERMEDIATE OUTPUT--T =   0.4900000D 00
                     Y1=  0.7569118D 00  Y2= -0.5637749D 00
                     Y3=  0.5128778D 00  Y4=  0.9325251D 00

INTERMEDIATE OUTPUT--T =   0.6000000D 00
                     Y1=  0.6891170D 00  Y2= -0.6668679D 00
                     Y3=  0.6111621D 00  Y4=  0.8524277D 00

INTERMEDIATE OUTPUT--T =   0.6600000D 00
                     Y1=  0.6475560D 00  Y2= -0.7178635D 00
                     Y3=  0.6608675D 00  Y4=  0.8039059D 00

INTERMEDIATE OUTPUT--T =   0.7500000D 00
                     Y1=  0.5797717D 00  Y2= -0.7869198D 00
                     Y3=  0.7297455D 00  Y4=  0.7256920D 00

INTERMEDIATE OUTPUT--T =   0.8000000D 00
                     Y1=  0.5395550D 00  Y2= -0.8212732D 00
                     Y3=  0.7648897D 00  Y4=  0.6798258D 00

INTERMEDIATE OUTPUT--T =   0.9000000D 00
                     Y1=  0.4543357D 00  Y2= -0.8811394D 00
                     Y3=  0.8281186D 00  Y4=  0.5839259D 00

INTERMEDIATE OUTPUT--T =   0.9800000D 00
                     Y1=  0.3822211D 00  Y2= -0.9204380D 00
                     Y3=  0.8716557D 00  Y4=  0.5041103D 00

INTERMEDIATE OUTPUT--T =   0.1000000D 01
                     Y1=  0.3637254D 00  Y2= -0.9290651D 00
                     Y3=  0.8815353D 00  Y4=  0.4838348D 00

INTERMEDIATE OUTPUT--T =   0.3100000D 01
                     Y1= -0.1099278D 01  Y2= -0.3434921D-01
                     Y3=  0.3759444D-01  Y4= -0.9039508D 00

INTERMEDIATE OUTPUT--T =   0.3220000D 01
                     Y1= -0.1097450D 01  Y2=  0.6478456D-01
                     Y3= -0.7088700D-01  Y4= -0.9024476D 00
```

```
 INTERMEDIATE OUTPUT--T =   0.3490000D 01
                      Y1= -0.1050092D 01  Y2=  0.2848796D 00
                      Y3= -0.3103776D 00  Y4= -0.8633178D 00


(Output for 3.4900 < T < 3.9800 has been deleted.)


 INTERMEDIATE OUTPUT--T =   0.3980000D 01
                      Y1= -0.8186657D 00  Y2=  0.6487301D 00
                      Y3= -0.6918563D 00  Y4= -0.6671274D 00
```

Example 2, two body problem, transfer orbit

```
      SUBROUTINE TRANSF(NPHI,INDEX,NEQN,T,Y,YP,PHI,PHIP,KOUNTR,UPDATE,
     1                  IVAN,RELER,ABSER)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION Y(6),YP(6),PHI(NPHI),PHIP(NPHI),HEAD(5)
      COMMON/TIME/TINCR
      DATA HEAD/8HT-TPR = ,8HR" =     ,8HV . R = ,8HAPOGEE  ,8HPERIGEE /
      LOGICAL UPDATE,IVAN
C
      IF (UPDATE)  GO TO 100
C
      IF (KOUNTR .GT. 0)  GO TO 12
C-----------------------------------------------------------------------
C     INITIALIZATION BLOCK
C-----------------------------------------------------------------------
C
C     COMBINATION TRAPPING MODE IS BEING USED
C           INDEX=1 AND 2 ARE TRAPPED IN MULTIPLE TRAPPING MODE
C           INDEX=3 IS TRAPPED IN SINGLE TRAPPING MODE
C
C   >>>>>>>>>>>> SET INDEX=2 IN INITIAL. BLOCK <<<<<<<<<<<<
C-----------------------------------------------------------------------
      IPER = 0
      IPRINT = 0
      INDEX = 2
      TPR = TINCR
C     PARAMETERS FOR SECANT APPROXIMATION FOR R'''
      TSAVE = T
      RPP  =0.0D0
      RPPP =0.0D0
   12 CONTINUE
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
C     DETERMINE R, R**2, AND R' TO FORM R" (THEN FORM R''')
C-----------------------------------------------------------------------
      R2 = (Y(1)**2+Y(2)**2+Y(3)**2)
      R = DSQRT(R2)
      RP = (Y(1)*YP(1)+Y(2)*YP(2)+Y(3)*YP(3))/R
```

```
      RPPSV = RPP
      RPP = (YP(1)**2+YP(2)**2+YP(3)**2 + Y(1)*YP(4)+Y(2)*YP(5)
     1         + YP(3)*YP(6))/R - RP*RP/R
C
C     SECANT APPROXIMATION FOR R'''
      DELTT = T - TSAVE
      RPPP = 0.0D0
      IF (KOUNTR .GT. 0 .AND. DELTT .GT.  1.D-12)
     1                                      RPPP = (RPP - RPPSV)/DELTT
C
C-----------------------------------------------------------------------
   20 CONTINUE
C-----------------------------------------------------------------------
      PHI(1) =   T - TPR
      PHI(2) =   RPP
      PHI(3) =   Y(1)*Y(4) + Y(2)*Y(5) + Y(3)*Y(6)
C-----------------------------------------------------------------------
      PHIP(1) =   1.0D0
      PHIP(2) =   RPPP
      PHIP(3) =   Y(1)*YP(4) + Y(4)*Y(4) + Y(2)*YP(5) + Y(5)*Y(5)
     1         + Y(3)*YP(6) + Y(6)*Y(6)
C-----------------------------------------------------------------------
      TSAVE = T
      RETURN
  100 CONTINUE
C
C-----------------------------------------------------------------------
C     UPDATE BLOCK
C-----------------------------------------------------------------------
C
  110 CONTINUE
C-----------------------------------------------------------------------
C     IF THE PHI COMPONENT HAS VANISHED THROUGHOUT THE STEP--RETURN
C     (NO PRINTS OR UPDATES--INTEGRATION STEP SIZE IS SMALL AFTER
C                           TRAPPING ON PREVIOUS STEP
C-----------------------------------------------------------------------
      IF (IVAN)  PRINT 506
      IF (IVAN)  RETURN
C-----------------------------------------------------------------------
C     PRINT STATEMENTS FOR ALL UPDATES
C-----------------------------------------------------------------------
      IF (INDEX .EQ. 3)  GO TO 120
      PRINT 515,HEAD(INDEX),PHI(INDEX),T
      PRINT 516,Y(1),Y(2),Y(3),Y(4),Y(5),Y(6)
      IF (INDEX .EQ. 2)  RETURN
C-----------------------------------------------------------------------
C     TIME STOP--MULTIPLE TRAP MODE  (PHI UPDATED NOT NECESSARY)
C-----------------------------------------------------------------------
      TPR = TPR + TINCR
      RETURN
  120 CONTINUE
C-----------------------------------------------------------------------
C     PERIGEE OR APOGEE STOP--SINGLE TRAP MODE
C-----------------------------------------------------------------------
C
      IF (RPP .GT. 0.0D0)  GO TO 50
C     CONDITIONS AT APOGEE
C
```

```
      PRINT 515,HEAD(INDEX),PHI(INDEX),T
      PRINT 516,Y(1),Y(2),Y(3),Y(4),Y(5),Y(6)
      PRINT 525,HEAD(4),R
C
      RETURN
   50 CONTINUE
C-----------------------------------------------------------------------
C     PERIGEE STOP--SINGLE TRAP MODE  (PHI UPDATE NECESSARY IF SYSTEM
C                                      IS ALTERED)
C-----------------------------------------------------------------------
      PRINT 515,HEAD(INDEX),PHI(INDEX),T
      PRINT 516,Y(1),Y(2),Y(3),Y(4),Y(5),Y(6)
      PRINT 525,HEAD(5),R
      IPER = IPER + 1
      IF (IPER .NE. 2) RETURN
C-----------------------------------------------------------------------
C     TRANSFER ORBIT--"STRETCH" VELOCITY VECTOR, EVALUATE F, EVALUATE
C                     PHI(3) AND PHIP(3)
C-----------------------------------------------------------------------
      PRINT 1523
C
      CONST = 1.0500D0
      Y(4) =  Y(4)*CONST
      Y(5) =  Y(5)*CONST
      Y(6) =  Y(6)*CONST
      PRINT 516,Y(1),Y(2),Y(3),Y(4),Y(5),Y(6)
      CALL F(T,Y,YP)
      PHI(3) =   Y(1)*Y(4) + Y(2)*Y(5) + Y(3)*Y(6)
      PHIP(3) =   Y(1)*YP(4) + Y(4)*Y(4) + Y(2)*YP(5) + Y(5)*Y(5)
     1         + Y(3)*YP(6) + Y(6)*Y(6)
      IPRINT = 1
C
  506 FORMAT(' PHI(INDEX) HAS VANISHED OVER THE ENTIRE STEP',/)
  515 FORMAT(/,' UPDATE ',3X,1A8,2X,D15.7,3X,' AT T = ',D15.7)
  516 FORMAT(' POSITION=',3(1X,D15.7),/,' VELOCITY=',3(1X,D15.7))
C
  518 FORMAT(//,' TWO REVOLUTIONS HAVE OCCURRED--INCREMENT VELOCITY')
  519 FORMAT(' NEW ECCENTRICITY = ',D15.7,/,' Y-COMP = ',4(2X,D15.7),//)
  525 FORMAT(' MAGNITUDE OF RADIUS AT ',1A8,' = ',D15.7)
 1523 FORMAT(/,' TIME TO UPDATE THE VELOCITY FOR TRANSFER ORBIT')
      RETURN
      END
```

```
 TWO BODY PROBLEM: EARTH-SATELLITE--TRANSFER ORBIT
                   (TRANSFER AFTER 2 REVOLUTIONS)

 OPERATION PARAMETERS AND BOUNDARY CONDITIONS

 RELERR =   0.1000000D-05   ABSERR =   0.1000000D-05   IFLAG =   15
 T      =   0.0000000D 00   TF     =   0.8000000D 06
 POSITION  -0.7195613D 04  0.1546026D 04 -0.9839836D 03
 VELOCITY  -0.4201003D 01 -0.8358974D 01 -0.2073556D 01
```

```
UPDATE:   R" =       -0.3612253D-07    AT T =   0.1284196D 04
POSITION=  -0.8501289D 04  -0.8587487D 04  -0.2848147D 04
VELOCITY=   0.9868303D 00  -0.6842308D 01  -0.9382287D 00

UPDATE:   V . R =    -0.1162170D-04    AT T =   0.1997027D 05
POSITION=   0.2931916D 05  -0.3349531D 05  -0.3925918D 03
VELOCITY=   0.1128553D 01   0.9837115D 00   0.3527977D 00
MAGNITUDE OF RADIUS AT APOGEE   =   0.4424773D 05

UPDATE:   R" =        0.2128608D-07    AT T =   0.3865792D 05
POSITION=   0.9659848D 04   0.7254891D 04   0.2831174D 04
VELOCITY=  -0.6664918D 01   0.1893360D 01  -0.8367371D 00

UPDATE:   V . R =     0.1786375D-06    AT T =   0.4040718D 05
POSITION=  -0.4462832D 04   0.5098527D 04   0.5976175D 02
VELOCITY=  -0.7414175D 01  -0.6462593D 01  -0.2317745D 01
MAGNITUDE OF RADIUS AT PERIGEE  =   0.6776396D 04

UPDATE:   R" =       -0.7203513D-06    AT T =   0.4215722D 05
POSITION=  -0.8502086D 04  -0.8581981D 04  -0.2847392D 04
VELOCITY=   0.9854028D 00  -0.6843751D 01  -0.9387070D 00

UPDATE:   V . R =    -0.4580103D-04    AT T =   0.6084421D 05
POSITION=   0.2931931D 05  -0.3349543D 05  -0.3925838D 03
VELOCITY=   0.1128548D 01   0.9837090D 00   0.3527964D 00
MAGNITUDE OF RADIUS AT APOGEE   =   0.4430634D 05

UPDATE:   R" =        0.1246685D-07    AT T =   0.7953198D 05
POSITION=   0.9659784D 04   0.7254930D 04   0.2831169D 04
VELOCITY=  -0.6664940D 01   0.1893334D 01  -0.8367451D 00

UPDATE:   V . R =     0.3124829D-06    AT T =   0.8128124D 05
POSITION=  -0.4462840D 04   0.5098527D 04   0.5976027D 02
VELOCITY=  -0.7414167D 01  -0.6462599D 01  -0.2317745D 01
MAGNITUDE OF RADIUS AT PERIGEE  =   0.6776503D 04


TIME TO UPDATE: THE VELOCITY FOR TRANSFER ORBIT
POSITION=  -0.4462840D 04   0.5098527D 04   0.5976027D 02
VELOCITY=  -0.7784876D 01  -0.6785729D 01  -0.2433632D 01


UPDATE:   R" =       -0.1880129D-06    AT T =   0.8306752D 05
POSITION=  -0.9507222D 04  -0.9331483D 04  -0.3141112D 04
VELOCITY=   0.1625789D 00  -0.7200626D 01  -0.1137605D 01

UPDATE:   T-TPR =     0.0000000D 00    AT T =   0.1000000D 06
POSITION=   0.1745937D 05  -0.6672033D 05  -0.7804626D 04
VELOCITY=   0.1568620D 01  -0.1986538D 01  -0.5248399D-01

UPDATE:   V . R =    -0.1769465D-06    AT T =   0.1907712D 06
POSITION=   0.9897001D 05  -0.1130668D 06  -0.1325196D 04
VELOCITY=   0.3510427D 00   0.3059896D 00   0.1097398D 00
MAGNITUDE OF RADIUS AT APOGEE   =   0.1502666D 06
```

```
UPDATE:   T-TPR =     0.0000000D 00    AT T =   0.2000000D 06
POSITION=   0.1017085D 06  -0.1096812D 06  -0.3074815D 03
VELOCITY=   0.2416685D 00   0.4273841D 00   0.1106283D 00

UPDATE:   R" =        0.4227668D-07    AT T =   0.2984758D 06
POSITION=   0.1053792D 05   0.8147668D 04   0.3126289D 04
VELOCITY=  -0.7129841D 01   0.1127692D 01  -0.1040414D 01

UPDATE:   T-TPR =     0.0000000D 00    AT T =   0.3000000D 06
POSITION=  -0.2268374D 04   0.6628906D 04   0.6838715D 03
VELOCITY=  -0.8883153D 01  -0.4888772D 01  -0.2314973D 01

UPDATE:   V . R =     0.1618111D-05    AT T =   0.3002612D 06
POSITION=  -0.4462861D 04   0.5098542D 04   0.5975927D 02
VELOCITY=  -0.7784855D 01  -0.6785722D 01  -0.2433628D 01
MAGNITUDE OF RADIUS AT PERIGEE  =   0.6777237D 04

UPDATE:   R" =       -0.2216406D-06    AT T =   0.3020474D 06
POSITION=  -0.9507256D 04  -0.9331200D 04  -0.3141072D 04
VELOCITY=   0.1625186D 00  -0.7200677D 01  -0.1137624D 01

UPDATE:   T-TPR =     0.0000000D 00    AT T =   0.4000000D 06
POSITION=   0.9499987D 05  -0.1154126D 06  -0.2385862D 04
VELOCITY=   0.4628059D 00   0.1743329D 00   0.1076000D 00

UPDATE:   V . R =    -0.3895867D-06    AT T =   0.4097513D 06
POSITION=   0.9897021D 05  -0.1130669D 06  -0.1325171D 04
VELOCITY=   0.3510427D 00   0.3059901D 00   0.1097399D 00
MAGNITUDE OF RADIUS AT APOGEE   =   0.1502652D 06

UPDATE:   T-TPR =     0.0000000D 00    AT T =   0.5000000D 06
POSITION=   0.6481676D 05  -0.2718058D 05   0.6718212D 04
VELOCITY=  -0.1722661D 01   0.1801974D 01  -0.3811446D-02

UPDATE:   R" =        0.4248428D-07    AT T =   0.5174561D 06
POSITION=   0.1053795D 05   0.8147705D 04   0.3126300D 04
VELOCITY=  -0.7129828D 01   0.1127685D 01  -0.1040412D 01

UPDATE:   V . R =     0.1625319D-05    AT T =   0.5192415D 06
POSITION=  -0.4462881D 04   0.5098558D 04   0.5975828D 02
VELOCITY=  -0.7784835D 01  -0.6785716D 01  -0.2433623D 01
MAGNITUDE OF RADIUS AT PERIGEE  =   0.6777265D 04

UPDATE:   R" =       -0.2216161D-06    AT T =   0.5210277D 06
POSITION=  -0.9507282D 04  -0.9331242D 04  -0.3141083D 04
VELOCITY=   0.1625249D 00  -0.7200664D 01  -0.1137620D 01

UPDATE:   T-TPR =     0.0000000D 00    AT T =   0.5500000D 06
POSITION=   0.3492053D 05  -0.8605259D 05  -0.7938726D 04
VELOCITY=   0.1339494D 01  -0.1296990D 01   0.1982543D-01

UPDATE:   T-TPR =     0.0000000D 00    AT T =   0.6000000D 06
POSITION=   0.8418488D 05  -0.1161495D 06  -0.4360165D 04
VELOCITY=   0.6768570D 00  -0.1026458D 00   0.9948326D-01

UPDATE:   V . R =    -0.3888030D-06    AT T =   0.6287318D 06
POSITION=   0.9897041D 05  -0.1130669D 06  -0.1325146D 04
```

```
VELOCITY=   0.3510427D 00   0.3059906D 00   0.1097399D 00
MAGNITUDE OF RADIUS AT APOGEE   =   0.1502654D 06

UPDATE:   T-TPR =     0.0000000D 00    AT T =   0.7000000D 06
POSITION=   0.8801541D 05  -0.5668672D 05   0.5921491D 04
VELOCITY=  -0.8490028D 00   0.1341839D 01   0.7156505D-01

UPDATE:   R" =        0.4247250D-07    AT T =   0.7364368D 06
POSITION=   0.1053798D 05   0.8147742D 04   0.3126312D 04
VELOCITY=  -0.7129815D 01   0.1127678D 01  -0.1040411D 01

UPDATE:   V . R =     0.1625384D-05    AT T =   0.7382222D 06
POSITION=  -0.4462901D 04   0.5098573D 04   0.5975729D 02
VELOCITY=  -0.7784815D 01  -0.6785709D 01  -0.2433618D 01
MAGNITUDE OF RADIUS AT PERIGEE  =   0.6777290D 04
```

Example 3, restricted problem of three bodies

```
C
      SUBROUTINE RP3B1(NPHI,INDEX,NEQN,T,Y,YP,PHI,PHIP,KOUNTR,UPDATE,
     1                 IVAN,RELER,ABSER)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION Y(4),YP(4),PHI(NPHI),PHIP(NPHI),YYP(4)
      DOUBLE PRECISION MU,MUSTAR,JAKOB0,JAKOB
      DIMENSION HEAD1(8)
      COMMON/CONST/MU,MUSTAR,JAKOB0
      DATA ZAPP1/0.60D-04/
      LOGICAL UPDATE,IVAN,ACROSS
      DATA HEAD1/8H V . R =,8H X =     ,8H Y =     ,8H VX =    ,
     1           8H VY =    ,8H X-0.5 =,8H Y+0.6 =,8H VX-1.0=/
C
      IF (UPDATE)  GO TO 100
C-----------------------------------------------------------------------
C     PHI COMPONENTS AND DERIVATIVES
C-----------------------------------------------------------------------
      PHI(1) =   Y(1)*Y(3) + Y(2)*Y(4) - 0.0D0
      PHI(2) =   Y(1) - 0.0D0
      PHI(3) =   Y(2) - 0.0D0
      PHI(4) =   Y(3) - 0.0D0
      PHI(5) =   Y(4) - 0.0D0
      PHI(6) =   Y(1) - 0.5D0
      PHI(7) =   Y(2) + 0.6D0
      PHI(8) =   Y(3) - 1.0D0
      PHIP(1) =   Y(1)*YP(3) + YP(1)*Y(3) + Y(2)*YP(4) + YP(2)*Y(4)
      PHIP(2) =   Y(3)
      PHIP(3) =   Y(4)
      PHIP(4) =   YP(3)
      PHIP(5) =   YP(4)
      PHIP(6) =   YP(1)
```

```
      PHIP(7) =   YP(2)
      PHIP(8) =   YP(3)
C-----------------------------------------------------------------------
      IF (INDEX .GT. 0)  RETURN
C-----------------------------------------------------------------------
C     EVALUATE JACOBIAN INTEGRAL--IF NOT IN TRAPPING ITERATION
C-----------------------------------------------------------------------
      R1 = ( (Y(1)+MU)**2 + Y(2)**2 )**0.5
      R2 = ( (Y(1)-MUSTAR)**2 + Y(2)**2 )**0.5
      JAKOB = 0.50D0*(Y(3)*Y(3) + Y(4)*Y(4) - Y(1)*Y(1) -Y(2)*Y(2))
     1        - MUSTAR/R1 - MU/R2
      ERR = DABS(JAKOB - JAKOB0)
      IF (ERR .GT. ZAPP1)  PRINT 1500,T,ERR
 1500 FORMAT(' AT T=',D15.7,/,' DEVIATION FROM INITIAL JACOB. INTEGRAL='
     1       ,D15.7)
      RETURN
C-----------------------------------------------------------------------
  100 CONTINUE
C-----------------------------------------------------------------------
C     UPDATE PORTION
C-----------------------------------------------------------------------
      PRINT 515,HEAD1(INDEX),PHI(INDEX),T
      IF (IVAN)  PRINT 506,INDEX
C-----------------------------------------------------------------------
C
  506 FORMAT(' PHI(',I2,') HAS VANISHED OVER THE ENTIRE STEP',/)
  515 FORMAT(/,' UPDATE:',3X,1A8,2X,D15.7,3X,' AT T = ',D23.16)
  516 FORMAT(/,' UPDATE:',3X,
     1       ' JACOBIAN INTEGRAL DEVIATES FROM INITIAL VALUE',/,
     2       ' ERROR = ',D23.16,3X,'AT T = ',D23.16)
C
      RETURN
      END
```

```
 RESTRICTED PROBLEM OF THREE BODIES

 OPERATION PARAMETERS AND BOUNDARY CONDITIONS

 RELERR =   0.1000000D-05   ABSERR =   0.1000000D-05   IFLAG =   15
 T      =   0.0000000D 00   TF     =   0.6192169D 01
 Y(1)   =   0.1200000D 01   Y(2)   =   0.0000000D 00
 Y(3)   =   0.0000000D 00   Y(4)   =  -0.1049358D 01
 JACOBIAN INTEGRAL (AT T=0)  =  -0.1041589D 01

 UPDATE:    V . R =   -0.5720000D-14    AT T =  0.0000000000000000D 00

 UPDATE:    Y =       -0.5720000D-14    AT T =  0.0000000000000000D 00

 UPDATE:    VX =      -0.5720000D-14    AT T =  0.0000000000000000D 00

 UPDATE:    VY =       0.2863387D-10    AT T =  0.8613208344927214D 00

 UPDATE:    X-0.5 =   -0.2070436D-09    AT T =  0.1046283795762994D 01

 UPDATE:    Y =        0.2214422D-06    AT T =  0.1448084325660553D 01

 UPDATE:    V . R =    0.2089552D-06    AT T =  0.1458571434491113D 01
```

```
UPDATE:    X =        -0.3681892D-05    AT T =  0.1459781007315835D 01

UPDATE:    VY =       -0.4598140D-07    AT T =  0.1461133846568575D 01

UPDATE:    Y =        -0.2195288D-05    AT T =  0.1472952988035424D 01

UPDATE:    Y+0.6 =    -0.1556487D-06    AT T =  0.1878139441039302D 01

UPDATE:    VY =        0.2477135D-10    AT T =  0.2126492806452552D 01

UPDATE:    Y+0.6 =     0.1346683D-07    AT T =  0.2396228631771846D 01

UPDATE:    Y =         0.7046048D-09    AT T =  0.3096069058097068D 01

UPDATE:    VX =        0.4738180D-06    AT T =  0.3096106917047610D 01

UPDATE:    V . R =    -0.8186590D-07    AT T =  0.3096161443936035D 01

UPDATE:    VY =       -0.8476735D-10    AT T =  0.4065756069599825D 01

UPDATE:    VX-1.0=     0.1300828D-06    AT T =  0.4670806136154190D 01

UPDATE:    Y =        -0.9581854D-07    AT T =  0.4719374046932356D 01

UPDATE:    VY =        0.2842921D-03    AT T =  0.4731191718141558D 01

UPDATE:    X =         0.1546022D-09    AT T =  0.4732544685928599D 01

UPDATE:    V . R =     0.2026057D-06    AT T =  0.4733753099474543D 01

AT T=  0.4743581D 01
DEVIATION FROM INITIAL JACOB. INTEGRAL=  0.6069291D-04

AT T=  0.4746211D 01
DEVIATION FROM INITIAL JACOB. INTEGRAL=  0.6172761D-04

UPDATE:    Y =         0.6909008D-06    AT T =  0.4744244036815875D 01

AT T=  0.4744244D 01
DEVIATION FROM INITIAL JACOB. INTEGRAL=  0.7270484D-04

AT T=  0.4746211D 01
DEVIATION FROM INITIAL JACOB. INTEGRAL=  0.6172761D-04

AT T=  0.4749214D 01
DEVIATION FROM INITIAL JACOB. INTEGRAL=  0.6186404D-04

AT T=  0.4752682D 01
DEVIATION FROM INITIAL JACOB. INTEGRAL=  0.6150855D-04

AT T=  0.4756721D 01
DEVIATION FROM INITIAL JACOB. INTEGRAL=  0.6088231D-04

AT T=  0.4761458D 01
DEVIATION FROM INITIAL JACOB. INTEGRAL=  0.6011234D-04

UPDATE:    VX-1.0=    -0.2864002D-09    AT T =  0.4847312724886170D 01
```

```
 UPDATE:     X-0.5 =     0.2669629D-08     AT T =  0.5145936167581279D 01

 UPDATE:     VY =       -0.6669679D-11     AT T =  0.5330930517574152D 01

 UPDATE:     Y =        -0.4183865D-11     AT T =  0.6192102468600831D 01

 T =  0.6192169331319640D 01

 SOLUTION:
 0.1200064D 01    -0.7016685D-04     0.1322904D-03    -0.1049417D 01


 ERRORS:
 0.6353576D-04     0.7016685D-04     0.1322904D-03     0.5919274D-04


 JACOBIAN INTEGRAL:

    INITIAL VALUE =  -0.1041589D 01
    CURRENT VALUE =  -0.1041543D 01
    DIFFERENCE    =   0.4571855D-04


 NFE =    1206      NEXTRA =      67
```

```
C
      SUBROUTINE RP3B2(NPHI,INDEX,NEQN,T,Y,YP,PHI,PHIP,KOUNTR,UPDATE,
     1                 IVAN,RELER,ABSER)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION Y(4),YP(4),PHI(NPHI),PHIP(NPHI),YYP(4)
      DOUBLE PRECISION MU,MUSTAR,JAKOB0,JAKOB
      DIMENSION HEAD1(8)
      COMMON/CONST/MU,MUSTAR,JAKOB0
      DATA ZAPP1/0.60D-04/
      LOGICAL UPDATE,IVAN,ACROSS
      DATA HEAD1/8H V . R =,8H X =     ,8H Y =     ,8H VX =    ,
     1           8H VY =    ,8H X-0.5 =,8H Y+0.6 =,8H VX-1.0=/
C
      IF (UPDATE)  GO TO 100
C-----------------------------------------------------------------------
C     INITIALIZATION OF JACOBI INTEGRAL ANALYSIS
C-----------------------------------------------------------------------
      IF (KOUNTR .GT. 0)   GO TO 22
      ERR0 = 0.0D0
      ACROSS = .FALSE.
   22 CONTINUE
C-----------------------------------------------------------------------
C     PHI COMPONENTS AND DERIVATIVES
C-----------------------------------------------------------------------
      PHI(1) =   Y(1)*Y(3) + Y(2)*Y(4) - 0.0D0
      PHI(2) =   Y(1) - 0.0D0
      PHI(3) =   Y(2) - 0.0D0
      PHI(4) =   Y(3) - 0.0D0
      PHI(5) =   Y(4) - 0.0D0
      PHI(6) =   Y(1) - 0.5D0
```

```
      PHI(7) =   Y(2) + 0.6D0
      PHI(8) =   Y(3) - 1.0D0
      PHIP(1) =   Y(1)*YP(3) + YP(1)*Y(3) + Y(2)*YP(4) + YP(2)*Y(4)
      PHIP(2) =   Y(3)
      PHIP(3) =   Y(4)
      PHIP(4) =   YP(3)
      PHIP(5) =   YP(4)
      PHIP(6) =   YP(1)
      PHIP(7) =   YP(2)
      PHIP(8) =   YP(3)
C-----------------------------------------------------------------------
      IF (INDEX .GT. 0)  RETURN
C-----------------------------------------------------------------------
C     EVALUATE JACOBIAN INTEGRAL--IF NOT IN TRAPPING ITERATION
C-----------------------------------------------------------------------
      R1 = ( (Y(1)+MU)**2 + Y(2)**2 )**0.5
      R2 = ( (Y(1)-MUSTAR)**2 + Y(2)**2 )**0.5
      JAKOB = 0.50D0*(Y(3)*Y(3) + Y(4)*Y(4) - Y(1)*Y(1) -Y(2)*Y(2))
     1        - MUSTAR/R1 - MU/R2
      ERR = DABS(JAKOB - JAKOB0)
      IF (ERR .GT. ZAPP1)  GO TO 35
      IF (ACROSS) PRINT 1501,T,ERR,ZAPP1,CRJD,TCR
      ACROSS = .FALSE.
      ERR0 = ERR
      RETURN
   35 CONTINUE
C
C     JACOBI INTEGRAL PROBLEMS--FURTHER ANALYSIS
C
      IF (.NOT. ACROSS)  PRINT 1503,T,ERR,ZAPP1
      ACROSS = .TRUE.
      IF (ERR .LT. ERR0)  GO TO 38
      TCR = T
      CRJD = ERR
   38 CONTINUE
      ERR0 = ERR
 1501 FORMAT(/,' AT T=',D15.7,'  THE J.I. DEVIATION = ',D15.7,
     1       /,' WHICH IS AGAIN LESS THAN ',D15.7,/,
     2  ' MAX DEVIATION=',D15.7,'   OCCURRED AT T=',D15.7,/)
 1503 FORMAT(/,' AT T=',D15.7,' THE J.I. DEVIATION = ',D15.7,
     1       /,' WHICH IS GREATER THAN ',D15.7,/)
C
      RETURN
  100 CONTINUE
C
C-----------------------------------------------------------------------
C     UPDATE PORTION
C-----------------------------------------------------------------------
      PRINT 515,HEAD1(INDEX),PHI(INDEX),T
      IF (IVAN)  PRINT 506,INDEX
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
C
  506 FORMAT(' PHI(',I2,') HAS VANISHED OVER THE ENTIRE STEP',/)
  515 FORMAT(/,' UPDATE:',3X,1A8,2X,D15.7,3X,' AT T = ',D23.16)
  516 FORMAT(/,' UPDATE:',3X,
     1       ' JACOBIAN INTEGRAL DEVIATES FROM INITIAL VALUE',/,
     2       ' ERROR = ',D23.16,3X,'AT T = ',D23.16)
```

```
C
      RETURN
      END
```

```
 RESTRICTED PROBLEM OF THREE BODIES

 OPERATION PARAMETERS AND BOUNDARY CONDITIONS

 RELERR =   0.1000000D-05   ABSERR =   0.1000000D-05   IFLAG =   15
 T      =   0.0000000D 00   TF     =   0.6192169D 01
 Y(1)   =   0.1200000D 01   Y(2)   =   0.0000000D 00
 Y(3)   =   0.0000000D 00   Y(4)   =  -0.1049358D 01
 JACOBIAN INTEGRAL (AT T=0)  =  -0.1041589D 01

 UPDATE:    V . R =   -0.5720000D-14    AT T =  0.0000000000000000D 00

 UPDATE:    Y =       -0.5720000D-14    AT T =  0.0000000000000000D 00

 UPDATE:    VX =      -0.5720000D-14    AT T =  0.0000000000000000D 00

 (Output deleted.  See output from RP3B1)

 UPDATE:    X =        0.1546022D-09    AT T =  0.4732544685928599D 01

 UPDATE:    V . R =    0.2026057D-06    AT T =  0.4733753099474543D 01

 AT T=  0.4743581D 01 THE J.I. DEVIATION =   0.6069291D-04
 WHICH IS GREATER THAN   0.6000000D-04

 UPDATE:    Y =        0.6909008D-06    AT T =  0.4744244036815875D 01

 AT T=  0.4767043D 01  THE J.I. DEVIATION =   0.5927332D-04
 WHICH IS AGAIN LESS THAN   0.6000000D-04
 MAX DEVIATION=  0.6186404D-04   OCCURRED AT T=  0.4749214D 01

 UPDATE:    VX-1.0=   -0.2864002D-09    AT T =  0.4847312724886170D 01

 UPDATE:    X-0.5 =    0.2669629D-08    AT T =  0.5145936167581279D 01

 UPDATE:    VY =      -0.6669679D-11    AT T =  0.5330930517574152D 01

 UPDATE:    Y =       -0.4183865D-11    AT T =  0.6192102468600831D 01


 Final conditions, accuracy, and Jacobian integral information
 given in  SUBPHI=RP3B2 results.
```

Example 4, derivative evaluation dependent upon tabular data

```
      SUBROUTINE TABLE(NPHI,INDEX,NEQN,T,Y,YP,PHI,PHIP,KOUNTR,UPDATE,
     1                 IVAN,RELER,ABSER)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION Y(2),YP(2),PHI(NPHI),PHIP(NPHI)
      COMMON/GRID/AA(10,10),XX(10),YY(10),IX,IY
      COMMON/FVAL/AA1
      LOGICAL UPDATE,IVAN
C
      IF (UPDATE)  GO TO 100
      IF (KOUNTR .GT. 0)   GO TO 20
C-----------------------------------------------------------------------
C     INITIALIZATION BLOCK
C-----------------------------------------------------------------------
      AA1 = AA(IX,IY)
      X1 = XX(IX+1)
      Y1 = YY(IY+1)
      X0 = XX(IX)
      Y0 = YY(IX)
   20 CONTINUE
C-----------------------------------------------------------------------
C     PHI AND PHIP
C-----------------------------------------------------------------------
      PHI(1) =   (X1-Y(1))*(Y(1)-X0)
      PHI(2) =   (Y1-Y(2))*(Y(2)-Y0)
      PHIP(1) = ((X0+X1) -2.0D0*Y(1))*YP(1)
      PHIP(2) = ((Y0+Y1) -2.0D0*Y(2))*YP(2)
C-----------------------------------------------------------------------
      RETURN
  100 CONTINUE
C-----------------------------------------------------------------------
C     UPDATE:
C-----------------------------------------------------------------------
      IF (IVAN) RETURN
      PRINT 1500,T,Y(1),Y(2)
 1500 FORMAT(' T = ',D15.7,2X,'Y(1) = ',D15.7,2X,'Y(2) = ',D15.7)
      GO TO (110,120),INDEX
C
  110 CONTINUE
      IF (PHIP(1) .GT. 0.0D0)  GO TO 115
C     X1 CROSSING
      IX = IX + 1
      X0 = Y(1)
      X1 = XX(IX+1)
      GO TO 130
C
  115 CONTINUE
C     X0 CROSSING
      IX = IX -1
      X0 = XX(IX)
      X1 = Y(1)
      GO TO 130
C
  120 CONTINUE
      IF (PHIP(2) .GT. 0.0D0)   GO TO 125
C     Y1 CROSSING
      IY = IY + 1
      Y0 = Y(2)
```

```
      Y1 = YY(IY+1)
      GO TO 130
C
  125 CONTINUE
C     Y0 CROSSING
      IY = IY - 1
      Y0 = YY(IY)
      Y1 = Y(2)
  130 CONTINUE
C
      AA1 = AA(IX,IY)
      CALL F(T,Y,YP)
C-----------------------------------------------------------------------
C     UPDATE PHI FUNCTIONS
C-----------------------------------------------------------------------
      IF (INDEX .EQ. 2) GO TO 150
      PHI(1) =   +0.0D0
      PHIP(1) = ((X0+X1) -2.0D0*Y(1))*YP(1)
      RETURN
  150 CONTINUE
      PHI(2) =   +0.0D0
      PHIP(2) = ((Y0+Y1) -2.0D0*Y(2))*YP(2)
      RETURN
      END
C
      SUBROUTINE F(T,Y,YP)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION Y(2),YP(2)
      COMMON/FVAL/AA1
c
      YP(1) = AA1*T
      YP(2) = AA1
      RETURN
      END
C
```

```
 F EVALUATIONS DEPEND UPON TABULAR DATA

 OPERATION PARAMETERS AND BOUNDARY CONDITIONS
 RELERR =   0.1000000D-05   ABSERR =   0.1000000D-05   IFLAG =   10
 T      =   0.0000000D 00   TF     =   0.1000000D 01
 Y(1)   =   0.5000000D 00   Y(2)   =   0.5000000D 00


 T =   0.1250002D 00  Y(1) =   0.5312501D 00  Y(2) =   0.1000001D 01
 T =   0.3250000D 00  Y(1) =   0.7562501D 00  Y(2) =   0.2000000D 01
 T =   0.4322904D 00  Y(1) =   0.1000000D 01  Y(2) =   0.2643742D 01
 T =   0.4768227D 00  Y(1) =   0.1161940D 01  Y(2) =   0.3000001D 01
 T =   0.5879338D 00  Y(1) =   0.1694318D 01  Y(2) =   0.4000000D 01
 T =   0.6378108D 00  Y(1) =   0.2000000D 01  Y(2) =   0.4498771D 01
 T =   0.6795800D 00  Y(1) =   0.2330158D 01  Y(2) =   0.5000001D 01
 T =   0.7515860D 00  Y(1) =   0.3000000D 01  Y(2) =   0.5936079D 01
 T =   0.7558474D 00  Y(1) =   0.3048178D 01  Y(2) =   0.6000000D 01
 T =   0.8183474D 00  Y(1) =   0.3835276D 01  Y(2) =   0.7000000D 01
 T =   0.8301035D 00  Y(1) =   0.4000000D 01  Y(2) =   0.7199854D 01
 T =   0.8912547D 00  Y(1) =   0.5000000D 01  Y(2) =   0.8361727D 01
```

```
 T =   0.9431718D 00  Y(1) =   0.6000000D 01  Y(2) =   0.9451986D 01
 T =   0.9881951D 00  Y(1) =   0.7000000D 01  Y(2) =   0.1048752D 02


 T  =  0.10000000000000000D 01
 Y1 =  0.7293379684390654D 01  Y2 = 0.1078264334598878D 02

 NFE =    192     NEXTRA =     95


Exact stopping conditions:

T =  0.10000000000000000D 01
Y1 = 0.7293380335138080D 01   Y2 = 0.1078264444194078D 02

NFE =    103     NEXTRA =      0


"Natural" stopping iteration with RKF45:

T =  0.10000000000000000D 01
Y1 = 0.7295142710565982D 01   Y2 = 0.1078492347330573D 02
NFE =   1206     NEXTRA =      0
```

Example 5, highly oscillatory problem

```
C
      SUBROUTINE OSCIL8(NPHI,INDEX,NEQN,T,Y,YP,PHI,PHIP,KOUNTR,UPDATE,
     1                  IVAN,RELER,ABSER)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION Y(1),YP(1),PHI( 1),PHIP( 1)
      DATA PI/3.141592653589793D0/
      COMMON/NOSC/OSC
      LOGICAL UPDATE,IVAN
C
      IF (UPDATE)  GO TO 100
C---------------------------------------------------------------------
      OSCPIT = OSC*PI*T
      PHI(1) = DSIN(OSCPIT)
      PHIP(1) =   OSC*PI*DCOS(CSCPIT)
C---------------------------------------------------------------------
      RETURN
  100 CONTINUE
C---------------------------------------------------------------------
C     UPDATE PORTION:
C---------------------------------------------------------------------
      PRINT 528,T,PHI(1),Y(1)
      IF (IVAN)  PRINT 506
```

```
  506 FORMAT(' PHI HAS VANISHED OVER THE ENTIRE STEP',/)
  528 FORMAT(' T = ',D15.7,3X,' PHI = ',D15.7,3X,' Y = ',D15.7)
C
      RETURN
      END
```

```
  HIGHLY OSCILLATORY PHI FUNCTION

  OPERATION PARAMETERS AND BOUNDARY CONDITIONS

  RELERR =   0.1000000D-05   ABSERR =   0.1000000D-05   IFLAG =   10
  T      =   0.0000000D 00   TF     =   0.1000000D 01
  Y(1)   =   0.1000000D 01


  T =   0.0000000D 00    PHI =   0.5720000D-14    Y =   0.1000000D 01

  THE PANIC OPTION IS BEING USED AT EACH STEP IN RKF45T
  TO CHECK FOR MULTIPLE ZEROS OF A PHI COMPONENT WITHIN A GIVEN STEP
  THIS IS VERY INEFFICIENT.


  T =   0.2000000D 00    PHI =  -0.8719671D-15    Y =   0.1049600D 01
  T =   0.4000000D 00    PHI =   0.1062963D-10    Y =   0.1249600D 01
  T =   0.6000000D 00    PHI =  -0.4153248D-06    Y =   0.1705600D 01
  T =   0.8000000D 00    PHI =   0.2076624D-06    Y =   0.2561600D 01
  T =   0.1000000D 01    PHI =  -0.2790295D-14    Y =   0.4000000D 01


  NFE =       104    NEXTRA =        14
```

Example 6, a large convergence region (a flat PHI component)

```
C
C
      SUBROUTINE FLAT(NPHI,INDEX,NEQN,T,Y,YP,PHI,PHIP, KNTR,UPDATE,
     1                IVAN,RELER,ABSER)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION Y(NEQN),YP(NEQN),PHI(NPHI),PHIP(NPHI)
      LOGICAL UPDATE,IVAN
      COMMON/EXPO/POWER,IPOW,IPOWM1
C
      IF (UPDATE)  GO TO 100
C-----------------------------------------------------------------------
      PHI(1) = (T-2.0D0)**IPOW
      PHIP(1) =  POWER*(T-2.0D0)**IPOWM1
C-----------------------------------------------------------------------
      RETURN
C
```

```
  100 CONTINUE
C-----------------------------------------------------------------------
C     UPDATE:
C-----------------------------------------------------------------------
      PRINT 1500,T,PHI(INDEX),PHIP(INDEX)
 1500 FORMAT(' T = ',D15.7,2X,' PHI = ',D15.7,2X,' PHIP = ',D15.7)
C
      RETURN
      END
```

LARGE CONVERGENCE REGION (IPOW=3):

OPERATION PARAMETERS AND BOUNDARY CONDITIONS

RELERR	=	0.1000000D-05	ABSERR	=	0.1000000D-05	IFLAG	= 10
T	=	0.0000000D 00	TF	=	0.6283185D 01		
Y(1)	=	0.1000000D 01	Y(2)	=	0.0000000D 00		
Y(3)	=	0.0000000D 00	Y(4)	=	0.1000000D 01		

THE PANIC OPTION IS BEING USED AT EACH STEP IN RKF45T
TO CHECK FOR MULTIPLE ZEROS OF A PHI COMPONENT WITHIN A GIVEN STEP
THIS IS VERY INEFFICIENT.

T = 0.1999863D 01 PHI = 0.2548717D-11 PHIP = 0.5597606D-07

RESULTS WITHOUT PANIC:
OPERATION PARAMETERS AND BOUNDARY CONDITIONS (UNCHANGED)

T = 0.1991936D 01 PHI = 0.5242923D-06 PHIP = 0.1950609D-03

LARGE CONVERGENCE REGION (IPOW=5):

OPERATION PARAMETERS AND BOUNDARY CONDITIONS (UNCHANGED)

THE PANIC OPTION IS BEING USED AT EACH STEP IN RKF45T
TO CHECK FOR MULTIPLE ZEROS OF A PHI COMPONENT WITHIN A GIVEN STEP
THIS IS VERY INEFFICIENT.

T = 0.1976935D 01 PHI = 0.6527201D-08 PHIP = 0.1414984D-05
T = 0.1999863D 01 PHI = 0.4755571D-19 PHIP = 0.1740733D-14
T = 0.2022791D 01 PHI = 0.6149686D-08 PHIP = 0.1349126D-05

RESULTS WITHOUT PANIC:
OPERATION PARAMETERS AND BOUNDARY CONDITIONS (UNCHANGED)

T = 0.1941217D 01 PHI = 0.7018756D-06 PHIP = 0.5970052D-04

LARGE CONVERGENCE REGION (IPOW=9):

```
 OPERATION PARAMETERS AND BOUNDARY CONDITIONS (UNCHANGED)

 THE PANIC OPTION IS BEING USED AT EACH STEP IN RKF45T
 TO CHECK FOR MULTIPLE ZEROS OF A PHI COMPONENT WITHIN A GIVEN STEP
 THIS IS VERY INEFFICIENT.

 T =   0.1841841D 01   PHI =   0.6192059D-07   PHIP =   0.3523585D-05
 T =   0.1863945D 01   PHI =   0.1597486D-07   PHIP =   0.1056731D-05
 T =   0.1886048D 01   PHI =   0.3239611D-08   PHIP =   0.2558668D-06
 T =   0.1908152D 01   PHI =   0.4652094D-09   PHIP =   0.4558469D-07
 T =   0.1930255D 01   PHI =   0.3905037D-10   PHIP =   0.5039110D-08
 T =   0.1952358D 01   PHI =   0.1264419D-11   PHIP =   0.2388612D-09
 T =   0.1974462D 01   PHI =   0.4621086D-14   PHIP =   0.1628519D-11
 T =   0.1996565D 01   PHI =   0.6658683D-22   PHIP =   0.1744612D-18
 T =   0.2018668D 01   PHI =   0.2753896D-15   PHIP =   0.1327654D-12
 T =   0.2040772D 01   PHI =   0.3113351D-12   PHIP =   0.6872455D-10
 T =   0.2062875D 01   PHI =   0.1535695D-10   PHIP =   0.2198210D-08
 T =   0.2084978D 01   PHI =   0.2310881D-09   PHIP =   0.2447437D-07
 T =   0.2107082D 01   PHI =   0.1851144D-08   PHIP =   0.1555848D-06
 T =   0.2129185D 01   PHI =   0.1002105D-07   PHIP =   0.6981408D-06
 T =   0.2151289D 01   PHI =   0.4151962D-07   PHIP =   0.2469960D-05

 RESULTS WITHOUT PANIC:
 OPERATION PARAMETERS AND BOUNDARY CONDITIONS (UNCHANGED)

 T =   0.1908152D 01   PHI =   0.4652094D-09   PHIP =   0.4558469D-07
 T =   0.2137431D 01   PHI =   0.1748919D-07   PHIP =   0.1145319D-05
```

Example 7, a bouncing PHI function

```
      SUBROUTINE BOUNCE(NPHI,INDEX,NEQN,T,Y,YP,PHI,PHIP,KOUNTR,UPDATE,
     1                  IVAN,RELER,ABSER)
C
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION Y(NEQN),YP(NEQN),PHI(NPHI),PHIP(NPHI)
      DATA A0,A1/3.0D0,6.0D0/
      LOGICAL UPDATE,IVAN
C
      IF (UPDATE)  GO TO 100
C-----------------------------------------------------------------------
      TMA0 = T - A0
      TMA1 = T - A1
      PHI(1)= (TMA0)**4 * (TMA1)**2
      PHI(2)= 4.0D0*TMA0**3 * TMA1**2 + 2.0D0*TMA0**4 * TMA1
      PHI(3) = (T-10.0D0)**6
      PHI(4) = 6.0D0*(T-10.D0)**5
      PHI(5) = (T-12.0D0)**8
```

```
      PHI(6) = 8.0D0*(T-12.D0)**7
C-----------------------------------------------------------------------
      PHIP(1) = PHI(2)
      PHIP(2) = 12.0D0*TMA0**2 * TMA1**2 + 8.0D0*TMA0**3 * TMA1
     1         + 8.0D0*TMA0**3 * TMA1 + 2.0D0*TMA0**4
      PHIP(3) = PHI(4)
      PHIP(4) = 30.0D0*(T-10.0D0)**4
      PHIP(5) = PHI(6)
      PHIP(6) = 56.0D0*(T-12.0D0)**6
C-----------------------------------------------------------------------
C
      RETURN
C
  100 CONTINUE
C-----------------------------------------------------------------------
C     UPDATE:  IF INDEX=2,4,6) RETURN. THESE ARE THE DERIVATIVE UPDATES.
C-----------------------------------------------------------------------
C
      IF (INDEX/2*2 .EQ. INDEX)  RETURN
      PRINT 527
  527 FORMAT(/,' UPDATE:')
      IF (IVAN)  PRINT 529,INDEX,T
C
      PRINT 528, T,INDEX,PHI(INDEX)
C
  528 FORMAT(' T = ',D15.7,3X,'PHI(',I2,') = ',D15.7)
  529 FORMAT(' PHI(',I3,') VANISHED THROUGHOUT THE STEP AT T = ',D15.7)
  530 FORMAT(' T=',D23.16)
C
      RETURN
      END
```

```
 BOUNCING PHI COMPONENTS:

 OPERATION PARAMETERS AND BOUNDARY CONDITIONS

 RELERR =   0.1000000D-05   ABSERR =   0.1000000D-05   IFLAG =   10
 T      =   0.0000000D 00   TF     =   0.1884956D 02
 Y(1)   =   0.1000000D 01


 UPDATE:
 T =   0.3002370D 01   PHI( 1) =  -0.2836618D-09

 UPDATE:
 T =   0.6000000D 01   PHI( 1) =  -0.4969498D-15

 UPDATE:
 T =   0.9924322D 01   PHI( 3) =  -0.1878504D-06

 UPDATE:
 PHI(  3) VANISHED THROUGHOUT THE STEP AT T =   0.9961253D 01
 T =   0.9961253D 01   PHI( 3) =   0.3384011D-08

 UPDATE:
 T =   0.1184198D 02   PHI( 5) =  -0.3888253D-06
```

```
UPDATE:
PHI(  5) VANISHED THROUGHOUT THE STEP AT T =   0.1209020D 02
T =    0.1209020D 02    PHI( 5) =  -0.4380999D-08
```