MaN 20

SC-M-70-724

RUNKUT: RUNGE-KUTTA INTEGRATOR OF SYSTEMS OF FIRST ORDER ORDINARY DIFFERENTIAL EQUATIONS

R. E. Jones
Mathematical Computing Services Division
Sandia Laboratories
Albuquerque, New Mexico
87115

Date published - November 1970

ABSTRACT

RUNKUT is a computer subroutine which integrates a system of n simultaneous first order differential equations, using a variable step size dual-mesh Runge-Kutta method. It is programmed to allow maximum flexibility of use, yet be reasonably easy to call.

Key words: routine, computing, mathematics

r

FOREWORD

The Sandia Laboratories Mathematical Program Library consists of a number of dependable, high-quality, general-purpose mathematical computing routines. The standards established for the library require that these routines be mathematically sound, effectively implemented, extensively tested, and thoroughly documented. This report documents one such routine.

The library emphasizes effective coverage of various distinct mathematical areas with a minimum number of routines. Nevertheless, it may contain other routines similar in nature but complementary to the one described here. Additional information on Sandia's mathematical program library, a description of the standard format for documenting these routines, and a guide to other routines in the library are contained in SC-M-69-337.

This report is also identified within Sandia Laboratories as Computing Publication ML0016/ALL. The routine was originally documented in April 1969. This report and its corresponding library routine are expected to be available from COSMIC shortly after publication.

ACKNOWLEDGMENT

The author wishes to thank Carl Bailey for his many helpful suggestions during the process of arriving at this version of RUNKUT and for his help in preparing this document. Edward Clark and William Gavin read all or parts of the document and proffered suggestions; Mr. Clark also provided a useful user's viewpoint of both this routine and its predecessor RKINTA. H. A. Watts provided advice also. Wendell Smith edited the final document.

CONTENTS

RUNKUT: RUNGE-KUTTA INTEGRATOR OF SYSTEMS OF FIRST ORDER ORDINARY DIFFERENTIAL EQUATIONS

1. Introduction

1.1 Background

RUNKUT is based on the Sandia Mathematical Program Library routine RKINTA which was written in April 1968. During the year following RKINTA's introduction into the library, it became clear that RKINTA lacked certain capabilities and desirable features. RUNKUT was written to include these features and to be as easy to use as practicable.

1.2 Applicable Programming Languages and Computer Systems

RUNKUT is written in a relatively bland subset of FORTRAN IV. It is thought to conform to ANSI* FORTRAN except for two statements which are noted in Appendix A. In particular, RUNKUT is written in Control Data 6600 FORTRAN.

The applicable computing system is the Control Data 6600 SCOPE.

In the Control Data 6600, RUNKUT is maintained for the convenience of the user in a library file. It is accessible by control cards which are described in Appendix C.

1.3 Consideration Regarding Use

RUNKUT is a flexible, general-purpose, variable step size, dual-mesh Runge-Kutta routine for integration of systems of first order ordinary differential equations with initial values. Most systems of higher order differential equations can be written as (larger) systems of first order equations and hence may be integrated with RUNKUT, although this may not be the best way to treat such systems. Also, RUNKUT will not produce satisfactory results on many systems of equations which are inherently unstable. For such unstable systems no classical method is likely to work. However, RUNKUT should be applicable for almost any system of equations for which one would expect a classical method to be appropriate. The version of RUNKUT that is maintained on a library file is restricted to no more than 25 equations. Card decks of RUNKUT modified for larger systems are available.

---

*American National Standards Institute, formerly USASI, originally ASA.

2. Usage

2.1 Entry

CALL RUNKUT (DERIV,YANS,N1D,NN,MP1,DELT,INIT,ERR,OPT,IERR,IP)

2.2 Description of Arguments

In the following discussion we will refer to an argument which must be defined in the calling program before the call to RUNKUT as an input argument. An argument which will be defined or redefined by RUNKUT will be called an output argument. The form of an argument which is input only can be either a variable name, an array element name, a constant, or an expression. All other arguments except DERIV must be variable names, array element names, or array names, as appropriate.

DERIV -- This argument must be the external name of the subroutine describing the system of equations to be integrated. The actual argument must appear in an EXTERNAL statement in the calling program. The form of the calling sequence of DERIV must be SUBROUTINE DERIV(Y,YP), where Y is an input array of variable values and YP is the resulting array of derivatives computed by DERIV. The order in which DERIV assumes the input variables to be in the array Y establishes the order which must be complied with in YP and in the array YANS. The only restriction is that the NN dependent variables must be first, followed by the independent variable (called "time") in Y(NN+1). The ordering of the variables used in DERIV will be called "standard," or "usual" order in the remainder of this document. DERIV must not alter the values in Y, and need not define YP(NN+1). See the examples in Section 2.4.

YANS -- (real, input/output) YANS is the initial condition and output array. It must be dimensioned exactly N1D by at least MM, where MM is the larger of 2 and MP1. On each call to RUNKUT, the initial conditions for the system of equations to be integrated must be in the first column of YANS in the standard order (see DERIV). Upon return to the calling program, YANS will contain the solution in one of two forms, depending on the value of the mode parameter MP1.

N1D -- (integer, input) N1D must be the actual size of the first dimension of YANS in the calling program. N1D must be at least NN+1. (This parameter is provided so that YANS need not be dimensioned exactly NN+1 but can be larger if desired.)

NN -- (integer, input) NN is the number of equations to be integrated. Thus, the total number of variables, including "time," is NN+1. (Hence N1D must be at least NN+1). NN must be no larger than 25. However, versions of RUNKUT with larger limits are easily made. See Appendix A.

MP1 -- (integer, input) MP1 is the output mode parameter, which may have any nonnegative value.

> If MP1 = 0, then one dual-mesh Runge-Kutta step (see Section 4) is done in the direction indicated by DELT, and of a size determined by RUNKUT according to the specifications in the array OPT. Then the new values of the variables are placed in column one of YANS in the usual order, and the corresponding derivatives are placed in column two, for the user's convenience. YANS(NN+1,2) is set equal to the new time value.

If MP1 = 1, then the integration will normally proceed until time = YANS(NN+1,1) + DELT. Then the new values of the variables are placed in column one of YANS in the usual order, and the corresponding derivatives are placed in column two. In this mode the user may optionally provide a subroutine called RKCHK(Y,K) which will be called at the end of each good dual-mesh Runge-Kutta step. The array Y will contain the current values of all the variables, in the usual order. If K is set to 1 in RKCHK, RUNKUT will immediately return control to the calling program with the current set of variable values (and derivatives) returned in YANS. (Thus RKCHK can be used for special printing purposes or to stop integration when some special condition occurs or to modify the variables dynamically to accomplish some special purpose.) If RKCHK modifies any element(s) of Y and if K is not set to 1, integration will proceed using these values.

If MP1 $\geq$ 2, then M = MP1-1 sets of answers are obtained and returned in columns 2 to MP1 of YANS, respectively. The i-th column of answers will correspond to time = YANS(NN+1,1) + (i-1)*DELT. Thus in this case the initial conditions remain in column one.

DELT -- (real, input) DELT is the desired time interval between output sets whenever MP1 is greater than or equal to 1. When MP1 = 0, the sign of DELT gives the direction of integration (see description of OPT(1) also). DELT may be positive or negative, but may not be zero.

INIT -- (integer, input/output) INIT should be set to a nonzero value for any call to RUNKUT which is not a continuation of integration. (For a continuation of integration the initial conditions would be the last set of answers obtained from the previous call to RUNKUT, INIT would be 0, and all other parameters would have the same values as they had upon exiting from RUNKUT.) RUNKUT will set INIT to zero if the input parameters are acceptable, i.e., if IERR$\neq$3. Whenever INIT is set nonzero, an initialization area of code is executed which checks for acceptable, consistent input values, and defines certain elements of OPT.

ERR -- (real, input/output) Normally ERR is the maximum relative error allowed for any single equation at any step of the Runge-Kutta procedure. See the description of OPT(4) to OPT(7). If the user has no particular knowledge on which to base the choice of the value of ERR, it may be set to a negative value, say -1.0, and RUNKUT will set it equal to 1.0×10^{-7}.

IERR -- (integer, output) IERR is the error status parameter. IERR = 1 is the normal value. IERR = 2 means the integration was not completed. IERR = 3 means the integration was not begun, due to invalid input. See Section 2.8.

IP -- (integer, output) IP gives the number of output sets, or columns of computed answers, actually found. (IP is for use primarily when MP1 is larger than 1 and IERR = 2.) Note that the value of IP does not count the initial conditions as a column of answers.

OPT -- (real, input/output) OPT must be an array of at least 23 locations. The first 16 elements of OPT are input values, described below. The remaining seven elements are defined by RUNKUT and are used for storage of values between calls to RUNKUT so that integration may be restarted most efficiently. If the user wishes, he may set each of the first 16 elements of OPT to a negative value, say -1.0, and RUNKUT will choose nominal values for them. Or, any subset of these 16 elements may be otherwise defined by the user and RUNKUT will choose those which are set negative. The "standard value" stated below for each element is the value RUNKUT will choose for each parameter if it is set negative by the user. These values are then actually placed in OPT. Do not use zeros to fill up OPT. For ease of reference later on, each element in OPT is here given a name.

(The reader may profitably skip to Section 2.4 for some examples of RUNKUT usage before finishing this section.)

OPT(1) -- HMAX, the maximum step size to be used in the Runge-Kutta process.

Restrictions: $0 < HMAX \leq 0.5 * ABS(DELT)$

Standard value: $HMAX = 0.5 * ABS(DELT)$

Note that this restriction must be satisfied even when MP1 = 0.

OPT(2) -- HMIN, the minimum step size.

Restrictions: $0 < HMIN \leq HMAX$

Standard value: $HMIN = 0.001 * HO$

OPT(3) -- HO, the initial, or starting, step size.

Restrictions: $HMIN \leq HO \leq HMAX$

Standard Value: $HO = 0.02 * HMAX$

Note: In RUNKUT, HO is chosen before HMIN is chosen by setting it to 0.02 * HMAX unless HMIN has been set by the user to a larger value than 0.02 * HMAX, in which case HO is set equal to HMIN.

OPT(4) -- RMAX. RMAX times ERR gives the maximum relative error allowed for any single equation at any step. See OPT(18).

Restrictions: $0 \leq RMAX$ (see Section 2.3 also)

Standard value: $RMAX = 1.0$

OPT(5) -- RMIN. RMIN times ERR gives the minimum relative error desired. This is used for determining when the step size should be increased.

Restrictions: $0 \leq RMIN$ (see Section 2.3 also)

Standard value: $RMIN = 0.01 * RMAX$

OPT(6) -- AMAX. The total allowed error for any equation at any step is RMAX times ERR times the value of the variable, plus AMAX times ERR. Thus if RMAX is set to zero, AMAX times ERR becomes the maximum absolute error allowed. See Section 3.5 also.

Restrictions: $0 \leq$ AMAX (see Section 2.3)

Standard value: AMAX = 0

OPT(7) -- AMIN, plays a role analogous to AMAX, but for the minimum absolute error desired. See Section 3.5.

Restrictions: $0 \leq$ AMIN (see Section 2.3)

Standard value: AMIN = 0

Note: If RMAX and RMIN are both set to zero, then AMAX and AMIN should both be set positive by the calling program.

OPT(8) -- CSNFAC. When RUNKUT determines that the step size should be increased, the new step size used is CSNFAC times the current step size (or HMAX, if HMAX is smaller than that).

Restrictions: CSNFAC > 1.0

Standard value: CSNFAC = 2.0

OPT(9) -- REFFAC. When RUNKUT determines that the step size should be decreased, the new step size used is REFFAC times the current step size (or HMIN, if HMIN is larger than that).

Restrictions: $0.0 <$ REFFAC < 1.0

Standard value: REFFAC = 0.5

OPT(10) -- HMINB. When RUNKUT determines that it needs a smaller step size, but the step size is already equal to HMIN, then HMINB is consulted to see whether execution should be terminated (if HMINB $\neq 0.0$) or if integration should be continued at the minimum step size (if HMINB = 0.0).

Restrictions: None

Standard value: HMINB = 1.0

OPT(11) -- EXTRAP. After each dual-mesh Runge-Kutta step is performed, a Richardson extrapolation can be performed, if desired, as a "mop-up" procedure which will normally produce more accurate answers if ERR is reasonably small. If EXTRAP is positive, the extrapolation is performed. Otherwise, the extrapolation is not performed, and the method remains the classical Runge-Kutta.

Restrictions: None

Standard value: Negative (no extrapolation)

OPT(12) -- FIXHO. If FIXHO is set positive by the calling program, the step size is fixed at HO and no error checking is done. Interruption of the step size in order to hit desired output points exactly will still be performed if OPT(15) is left at its standard value, but immediately after each output point the step size will be set back to HO. (Note that if FIXHO is positive, the value of EXTRAP is ignored and no extrapolation is done, since the double step answers which are needed in the extrapolation are not computed.)

Restrictions: None

Standard value: Negative

OPT(13) -- NTG. If NTG consecutive steps are "too good," the step size will be increased on the next step (see CSNFAC). A step is considered "too good" if the estimated error for each equation is less than the minimum desired error for that equation. See Section 3.5. (NTG = 0 is allowed but RUNKUT performs exactly as if NTG = 1 in that case.)

Restrictions: NTG > 0

Standard value: NTG = 3

OPT(14) -- HTOLF. If RUNKUT determines that the next dual-mesh step will integrate to at least T-H*HTOLF (where T is the desired output point and H is the current step size) then the step size for the next dual-mesh step is adjusted so that it will actually integrate exactly to that output point. This happens only when OPT(15) $\neq$ 0. (If HTOLF is set larger than 1.0, RUNKUT will reset it to 1.0.)

Restrictions: $0 \leq$ HTOLF ≤ 1.0

Standard value: HTOLF = 0.02

OPT(15) -- HINTR. When RUNKUT determines that the next dual-mesh step will integrate nearly to (as defined by HTOLF) or beyond an output point, HINTR is consulted to see what to do. If HINTR is equal to zero, the step is performed with no interruption and the computed set of answers is stored in its appropriate column of YANS. If HINTR is positive, the step size for the next dual-mesh step is altered so the independent variable will hit the desired output point exactly, and this set of answers is stored in its appropriate column of YANS. The step size is then set back to the value it had before this last step and integration continues. Errors are checked as usual in either case.

Restrictions: None

Standard value: HINTR = 1.0

OPT(16) -- NEQCHK. This is the number of equations to be checked for error. Equations numbered 1 through NEQCHK are checked, while the remainder are not. In almost all applications this quantity should be left at its standard value. (If NEQCHK is set larger than NN, RUNKUT will reset it to NN.)

Restrictions: $1 \leq NEQCHK \leq NN$.

Standard value: NEQCHK = NN

Note: The remaining quantities are defined by RUNKUT and need not be defined by the calling program.

OPT(17) -- ITG. This quantity is the current number of consecutive steps for which all equations have had less than the minimum desired error. It is needed for step-size coarsening when MP1 = 0.

OPT(18) -- RELMAX. RELMAX = RMAX * ERR.

OPT(19) -- RELMIN. RELMIN = RMIN * ERR.

OPT(20) -- ABSMAX. ABSMAX = AMAX * ERR.

OPT(21) -- ABSMIN. ABSMIN = AMIN * ERR.

OPT(22) -- IEQB. Whenever RUNKUT returns with IERR = 2, this quantity will be the index of the earliest equation which caused the difficulty. Otherwise IEQB will equal 0.

OPT(23) -- H. H is the value of the step size that was in use just before RUNKUT returned to the calling program. If HINTR is positive, H will not be the value used in the last dual-mesh step but the one used before the step-size interruption occurred.

2.3 Restrictions Between Arguments

Each argument which is defined or redefined by RUNKUT (i.e., YANS, INIT, ERR, OPT, IERR, and IP) must be distinct from all other elements of the calling sequence.

YANS must be dimensioned exactly N1D by at least the larger of MP1 and 2.

N1D must be at least NN+1.

The inequalities listed below relating elements of OPT must be satisfied after any requested standard values are set by RUNKUT. If the user uses nonstandard values for any of the elements of OPT, he must not make them such that they conflict with the standard values of the other elements. However, such conflicts are not likely if the values are set at all reasonably (see Section 2.7).

$0.0 < HMIN < HO \leq HMAX \leq 0.5 * ABS\ (DELT)$
(ABS (DELT) means absolute value of DELT.)

$RMIN \leq RMAX$ if $AMAX = AMIN = 0.0$

$AMIN \leq AMAX$ if $RMAX = RMIN = 0.0$

$RMAX + RMIN + AMAX + AMIN > 0.0$

$NEQCHK \leq NN$.

2.4 Principle Uses with Examples

Suppose we wish to integrate this system of differential equations,

$$y_1'(t) = 2\ t\ y_2(t)$$

$$y_2'(t) = -2\ t\ y_1(t)\ ,$$

with initial conditions of

$$y_1(0) = 0$$

$$y_2(0) = 1\ .$$

First we would write a subroutine, say DERIVA, "describing" the system:

```
SUBROUTINE DERIVA (Y,YP)
DIMENSION Y(3),YP(3)
T = Y(3)
YP(1) = 2.0 * T * Y(2)
YP(2) = -2.0 * T * Y(1)
RETURN
END
```

We must set up a driving program to call RUNKUT. The major factor in the design of this program is the choice of the value of MP1. If we

wish to print out the solution to the system at a number of points, say to the right of zero, we can use MP1 equal to some fairly large number and let RUNKUT do the whole integration in one call. For example (for the CDC 6600):

```
      PROGRAM ODE1(OUTPUT,TAPE6=OUTPUT)
      DIMENSION YANS(3,105),OPT(23)
      EXTERNAL DERIVA
      YANS(1,1) = 0.0
      YANS(2,1) = 1.0
      YANS(3,1) = 0.0
      INIT = -1
      ERR = -1.0
      DO 10 I = 1, 16
   10 OPT(I) = -1.0
      OPT(6) = 0.01
      CALL RUNKUT(DERIVA,YANS,3,2,101,0.1,INIT,ERR,
     1OPT,IERR,IP)
      WRITE (6,20) ((YANS(I,J),I=1,3),J=1,101)
   20 FORMAT (1X,3HY1=,E20.10,5X,3HY2=,E20.10,5X,2HT=,E20.10)
      END
```

If it is not desirable to retain all computed values until the end of the computation, we can use MP1=1 as in the following example, which would produce exactly the same printout as in the previous example.

```
      PROGRAM ODE2(OUTPUT,TAPE6=OUTPUT)
      DIMENSION YANS(3,2), OPT(23)
      EXTERNAL DERIVA
      YANS(1,1) = 0.0
      YANS(2,1) = 1.0
      YANS(3,1) = 0.0
      INIT = -1
      ERR = -1.0
      DO 10 I = 1, 16
   10 OPT(I) = -1.0
      OPT(6) = 0.01
      WRITE (6,20) (YANS(I,1),I=1,3)
   20 FORMAT (1X,3HY1=,E20.10,5X,3HY2=,E20.10,5X,2HT=,E20.10)
      DO 30 K = 1, 100
      CALL RUNKUT(DERIVA,YANS,3,2,1,0.1,INIT,ERR,OPT,IERR,IP)
      PRINT 20,(YANS(I,1),I=1,3)
   30 CONTINUE
      END
```

Note that in both of the above examples ERR and all the elements of OPT but OPT(6) were left to their standard values. OPT(6) was set to a small positive value because the solutions will repeatedly pass through zero, and using purelative error in such cases will usually cause difficulties. Setting OPT(6) to some positive value will probably be one of the most common nonstandard usages in most applications.

The case of MP1 = 0 would not usually be used when moderate to large numbers of steps are required to produce the desired answers. Rather, the MP1 = 0 option is meant for certain situations where it is desirable to monitor the integrated values continuously, or to get the solution to a system of equations very near to a pole or other discontinuity in one or more of the equations. For example, suppose

$$y'(t) = y^2(t) + 1,\ y(0) = 0\ ,$$

and we want the solution at $t = \pi/2$. The true solution is $y(t) = \tan t$, so $\pi/2$ is a pole, and the following program would allow integration up to very near $\pi/2$. Note that OPT(2),HMIN, is set to a very small value, and integration continues until acceptable error cannot be obtained with this step size.

```
      PROGRAM ODE3(OUTPUT,TAPE6=OUTPUT)
      DIMENSION YANS(3,2), OPT(23)
      EXTERNAL DERIV
      YANS(1,1) = 0.0
      YANS(2,1) = 0.0
      INIT = -1
      ERR = -1.0
      DO 10 I = 1,16
   10 OPT(I) = -1.0
      OPT(2) = 1.0E-10
   15 CALL RUNKUT (DERIV,YANS,3,1,0,1.0,INIT,ERR,
     1OPT,IERR,IP)
      IF (IERR.EQ.2) GO TO 50
      WRITE (6,20) (YANS(I,1),I=1,2)
   20 FORMAT (1X,2HY=,E20.10,5X,2HT=,E20.10)
      GO TO 15
   50 CONTINUE
      END
```

One can also do this type of operation or monitor the integrated values continuously by using MP1 = 1 (see description in Section 2.2) and supplying a special subroutine named RKCHK(Y,K). Normally a dummy version of RKCHK is provided along with RUNKUT, but the user may supply his own if desired. When MP1 = 1, RKCHK is called at the end of each good (accepted) dual-mesh Runge-Kutta step. At that time the user may examine the values of all the variables, including the independent variable, and may decide to terminate integration. The following example program, using the same subroutine DERIV as in the previous example ($y'(t) = y^2(t) + 1$, $y(0) = 0$), would print one line giving the t and y values at the end of the first dual-mesh Runge-Kutta step at which y becomes larger than 10^6.

```
      PROGRAM ODE4(OUTPUT,TAPE6=OUTPUT)
      DIMENSION YANS(2,2),OPT(23)
      EXTERNAL DERIV
      YANS(1,1) = 0.0
      YANS(2,1) = 0.0
      INIT = 1
      ERR = -1.0
      DO 10 I=1,16
   10 OPT(I) = -1.0
      OPT(2) = 1.0E-10
      CALL RUNKUT (DERIV,YANS,2,1,1,2.0,INIT,ERR,
     1OPT,IERR,IP)
      WRITE (6,20) (YANS(I,1),I=1,2)
   20 FORMAT (1X,2HY=E20.10,5X,2HT=,E20.10)
      END

      SUBROUTINE RKCHK(Y,K)
      DIMENSION Y(2)
      IF (Y(1).GT.1.0E6) K=1
      RETURN
      END
```

RKCHK can also be used for other purposes such as printing results periodically or when certain conditions occur, or imposing special conditions on the variables. For example, one may know a priori that the dependent variables being integrated must, say, add to 1.0. But due to cumulative integration errors the variables may drift away from this condition. RKCHK could be used to readjust slightly the values of one or more of these variables at the end of each step to enforce their sum remaining at 1.0. After each such alteration RUNKUT will continue integration with the altered values (if K is not set to 1). Note also that RKCHK could communicate with DERIV through COMMON, if desired.

2.5 Library Routines Explicitly Required

The standard FORTRAN routines ABS, AMAX1, and SIGN are required, also the Sandia Mathematical Program Library routine ERRCHK.[1]

2.6 User-Supplied Subroutines Required

The user must supply the subroutine DERIV, as described in Section 2.2. A given program would have at least one routine used as DERIV, and might have several. It might be desirable to have one subroutine describe several systems of equations by providing a switch through COMMON to tell which system is currently in use. The name(s) of the routine(s) used must appear in an EXTERNAL statement in the calling program. Also the user may optionally provide the subroutine RKCHK.

2.7 Cautions and Restrictions

The version of RUNKUT which is maintained on a library file in each system is limited to a maximum of 25 equations. Only two statements (three lines) needs to be changed in RUNKUT to increase this limit.

Probably the most common error made in using such routines as RUNKUT is the failure to include the name of the subroutine supplying the derivatives in an EXTERNAL statement in the calling program. Beware.

The user should be aware that many systems of equations are unstable and ordinary methods of integrating are not appropriate for their solution. (In fact, no numerical method may be appropriate.) When RUNKUT appears not to be producing good answers (and if programming errors have been ruled out as a cause) one should first check to see if OPT(1) to OPT(7) have reasonable values for the problem at hand--the standard values are not always reasonable--and should then try a smaller value for ERR. If this does not help, then RUNKUT may not be appropriate for solving that particular system of equations.

2.8 Error Conditions, Messages, and Codes

If execution is completed normally, IERR is set to 1; IP is set to MP1-1 if MP1 > 1, or to 1 if MP1 = 1 or 0.

If at some point in the integration process RUNKUT cannot satisfy the specified error bounds with the minimum step size, and if HMINB is positive (standard), then IERR is set to 2, IP is set to the number of sets of answers thus far computed (not including the initial conditions), and control is returned to the calling program.

If RUNKUT determines that one of the restrictions on argument values stated in Sections 2.2 or 2.3 has been violated, then IERR is set to 3, IP is set to 0, and ERRCHK is called to print the error message:

AN INPUT PARAMETER HAD AN ILLEGAL VALUE.

Normally, ERRCHK will cause program execution to terminate. To make this condition nonfatal, ERRSET[1] (an entry point in ERRCHK) must be called before RUNKUT is called. For example, to make errors nonfatal and set a maximum of 10 messages to be printed, the following call should be made:

CALL ERRSET(10,0)

If ERRSET is called in this way before a call to RUNKUT which detects an IERR = 3 type error, then RUNKUT will return control to the user with no integration done. Any values in ERR and OPT which RUNKUT had defined before detecting the error will remain so defined.

3. Mathematical Methods

3.1 Statement of Problem

The problem is to integrate a system of n simultaneous differential equations given in the "canonical" form

$$\begin{aligned} y_1'(t) &= f_1(y_1, y_2, \ldots, y_n, t) \\ y_2'(t) &= f_2(y_1, y_2, \ldots, y_n, t) \\ &\vdots \\ y_n'(t) &= f_n(y_1, y_2, \ldots, y_n, t) \end{aligned} \qquad (1)$$

with the initial conditions

$$y_i(t_0) = a_i, \quad i = 1, 2, \ldots, n.$$

Equations written in the "resolved" form, i.e., the form

$$y^{(n)}(t) = f(y, y', y'', y^{(n-1)}, t)$$

can be rewritten in the canonical (see Reference 2).

For example, the system

$$y^{(2)}(t) = f(y, y', t)$$

can be written as

$$y_1'(t) = y_2(t)$$

$$y_2'(t) = f(y_1, y_2, t)$$

3.2 Methods Used

The integration method used in RUNKUT is the classical fourth order Runge-Kutta, with error control based on estimates obtained from a dual-step size technique.

3.3 Mathematical Range and Domain

Mathematically, the domain is any set of equations which can be written in the canonical form (see Section 3.1) and which have a solution over the interval of interest. (Numerically, many such systems are impossible or excessively difficult to integrate with RUNKUT.) The range of the solutions is the real numbers.

3.4 Equations and Discussion

For convenience in dealing with a system of equations, we will use vector notation. In particular, let

$$\vec{y}(t) = (y_1(t), y_2(t), \ldots, y_n(t))$$

and

$$\vec{f}(t) = (f_1(\vec{y}(t), t), f_2(\vec{y}(t), t), \ldots, f_n(\vec{y}(t), t)) .$$

Then the classical fourth order Runge-Kutta integration step can be stated as (Reference 2, p. 66)

$$\vec{y}(t + h) = \vec{y}(t) + h(\vec{K}_1 + 2\vec{K}_2 + 2\vec{K}_3 + \vec{K}_4)/6 \qquad (2)$$

where

$$\begin{aligned} \vec{K}_1 &= \vec{f}(\vec{y}(t), t) \\ \vec{K}_2 &= \vec{f}(\vec{y}(t) + 0.5\,\vec{K}_1, t + 0.5\,h) \\ \vec{K}_3 &= \vec{f}(\vec{y}(t) + 0.5\,\vec{K}_2, t + 0.5\,h) \\ \vec{K}_4 &= \vec{f}(\vec{y}(t) + \vec{K}_3, t + h) . \end{aligned} \qquad (3)$$

Integration of a system of equations by this method from $t = t_0$ to $t = t_0 + \Delta t$ proceeds step by step, with the "t + h" on the left-hand side of Equation 2 at one step becoming the "t" on the right-hand side at the next step. The step size h is increased or decreased at appropriate times as determined by examining the error estimates. These estimates are obtained by comparing the answers obtained by integrating from t to t+2h by two steps of size h with those obtained by using one step of size 2h.

3.5 Error Analysis, Bounds, and Estimates

Assume that integration of a system of n equations has been carried out to some time t, and h is the step size most recently used. RUNKUT then proceeds by first integrating from t to t+2h with one Runge-Kutta

step of size 2h, obtaining an estimate $\vec{y}_1(t+2h)$ of the true solution at t+2h. Next, RUNKUT integrates from t to t+2h again, but with two Runge-Kutta steps of size h, thereby obtaining theoretically a better estimate, $\vec{y}_2(t+2h)$, of the true solution at t+2h. The estimate of the error incurred in each of the two steps of size h is given by (Reference 2, pp. 242-243)

$$\vec{E}(t, h) = \frac{1}{30}\left[\vec{y}_1(t + 2h) - \vec{y}_2(t + 2h)\right] . \qquad (4)$$

Given reasonable assumptions, Equation 4 follows from the fact that the Runge-Kutta technique used is fourth order. That is, the ratio of the errors in the "2h" answers to the errors in the "h" answers should be $(2h/h)^4 = 16$. Thus the difference between the two answers for any of the n equations should be 15 times as large as the error in the better answer, or 30 times the error due to either one of the two steps used in computing the better answer.

The error checking performed at the end of each dual-mesh step (as described in the preceding paragraph) is done as follows. If for any one of the n equations being integrated it is the case that

$$E_i(t, h) > \text{ERR } (\text{RMAX} \cdot y_{2_i}(t + 2h) + \text{AMAX}) \qquad (5)$$

then h is replaced by REFFAC·h (usually 0.5 h) and the integration starts over again from t with this smaller step size. If Equation 5 does not hold for any equation in the system, then $\vec{y}_2(t+2h)$ is accepted as the true solution at t+2h. If for at least one equation it is the case that

$$E_i(t, h) \geq \text{ERR } (\text{RMIN} \cdot y_{2_i}(t + 2h) + \text{AMIN}) \qquad (6)$$

then a parameter called ITG is set to zero and the integrations proceed forward. Otherwise (i.e., if all equations have less than the minimum desired error) ITG is incremented by 1, and if ITG is then at least three (or whatever nonstandard value for NTG the user sets) then h is replaced by CSNFAC·h (usually 2.0 h) and integration proceeds with this new step size.

4. Programming Methods

The main considerations in the coding of RUNKUT were straightforwardness (to simplify debugging and allow easy modification) and internal efficiency of execution. Where these two objectives sometimes conflicted, the problem was dealt with if necessary by increasing the length of the code.

Internal efficiency in RUNKUT is quite good; in an actual test RUNKUT and a simple fixed step size Runge-Kutta-Gill routine were made to perform equivalent integrations of a simple set of equations ($y_1'(t) = y_2(t)$, $y_2'(t) = -y_1(t)$, $y_1(0) = 0$, $y_2(0) = 1$) for 10,000 steps. RUNKUT was the faster even though it had to perform more derivative evaluations plus error checking. (RUNKUT required more derivative evaluations because

the fixed step size used was not obtained by using the FIXHO > 0.0 option, but by setting HMAX = HMIN = HO. When FIXHO > 0.0 is used RUNKUT is of course even faster.)

One measure of the straightforwardness achieved in RUNKUT is that, when using MP1 = 0, there are never any "branch backs" unless a decrease in the step size is required.

5. Space, Time, and Accuracy Considerations

Precise core requirements are given in the machine-dependent appendices. The amount of core required by RUNKUT (on a particular machine) is dependent only on the maximum number of equations allowed. The basic core requirement given in the appendices accounts for up to 25 equations. Larger allowables will require 10 (NMAX-25) additional core locations, where NMAX is the new maximum allowable number of equations.

Timing is of course extremely dependent on the number of equations to be integrated; the values of ERR, RMAX, RMIN, AMAX, and AMIN; the length of time over which the integration is to be performed; and the difficulty presented by the particular set of equations. The tables given in Appendix B give a rough idea of how some of these variables affect execution time.

Accuracy is generally closely related to the values of ERR, RMAX, RMIN, AMAX, AMIN, and the length of integration. In cases where many steps are performed in the integration, round-off error will limit the obtainable accuracy. In most cases, though, the truncation error, which is controlled by ERR, etc., will be the major determining factor in the actual error obtained.

6. Testing Methods

6.1 General Discussion

The usual method of testing an ordinary differential equation integrator is to have it integrate several systems of equations whose solutions are known. The computed solutions are then compared to the known solutions to determine the behavior of the computational errors. The main difficulty in evaluating an integration routine in this way is in the choice of a set of test problems (sets of equations) and an appropriate way of compiling the test results. These test problems should include "normal" or nonpathological systems as well as a variety of "difficult" or pathological systems.

6.2 Kinds of Tests Used

Seven systems of equations were chosen for testing RUNKUT. For each system a single value of the independent variable was chosen at which errors were to be tabulated. (This value was chosen so as to be as typical as possible; in particular, such points as maxima or zeroes of the dependent variables were avoided.) Each system of equations was then

integrated with various values of ERR, RMAX, and AMAX, and both with and without the optional extrapolation. RMIN and AMIN were always set to RMAX/100 and AMAX/100, respectively.

Sample tables of the results of these tests are included in the appendices. The system of equations to which each table corresponds is given by a number assigned in Sections 6.3 and 6.4. Each table gives the value of the independent variable to which the errors correspond, the values of ERR, RMAX, and AMAX which were used, and the actual execution time for the integration. When extrapolation was used, this fact is indicated at the bottom of the table.

Besides the tests which produced these tables, separate small-scale tests were made using nonstandard values of HMIN, HO, CSNFAC, REFFAC, HMINB, FIXHO, NTG, HTOLF, HINTR, and NEQCHK to check that RUNKUT performed predictably to changes in these quantities.

6.3 Normal Cases Tested

System No. 1

$$y_1'(t) = -y_1(t) \quad , \quad y_1(0) = 1.0$$

$$y_2'(t) = y_2(t) \quad , \quad y_2(0) = 1.0$$

The solution is

$$y_1(t) = e^{-t}$$

$$y_2(t) = e^{t}$$

System No. 2

$$y'(t) = -2ty(t) \quad , \quad y(0) = 1.0$$

The solution is

$$y(t) = e^{-t^2}$$

System No. 3

$$y_1'(t) = y_2(t) \quad , \quad y_1(0) = 0$$

$$y_2'(t) = -y_1(t) \quad , \quad y_2(0) = 1$$

The solution is

$$y_1(t) = \sin(t)$$

$$y_2(t) = \cos(t)$$

System No. 4

$$y'(t) = -y(t)^2 \quad , \quad y(0) = 1.0$$

The solution is

$$y(t) = \frac{1}{1 + t}$$

6.4 Difficult Cases Tested

System No. 5

$$y_1'(t) = 2ty_2(t) \quad , \quad y_1(0) = 0$$

$$y_2'(t) = -2ty_1(t) \quad , \quad y_2(0) = 1$$

The solution is

$$y_1(t) = \sin(t^2)$$

$$y_2(t) = \cos(t^2)$$

System No. 6

$$y'(t) = \begin{cases} 1 & \text{if } [t] \text{ is even} \\ -1 & \text{if } [t] \text{ is odd} \end{cases} \quad , \quad y(0) = 0$$

where [t] is the "greatest integer not greater than t."

The solution is

$$y(t) = \begin{cases} t - [t] & \text{if } [t] \text{ is even} \\ 1 - (t - [t]) & \text{if } [t] \text{ is odd} \end{cases}$$

System No. 7

$$y'(t) = 100\, y(t) - t^2), y(0) = 0.0002$$

The solution is

$$y(t) = 0.0002 + 0.02t + t^2$$

System No. 5 becomes very oscillatory as t gets large. System No. 6 is a triangular function oscillating between +1 and -1, which will require rapid step size refining and coarsening. System No. 7 is extremely unstable numerically.

6.5 Range, Error, and Fault Checks Tested

Special tests were done violating the restrictions on argument values and restrictions between arguments given in Sections 2.2 and 2.3 in order to determine whether the proper error message (Section 2.8) was indeed printed.

7. Remarks

In order to answer some questions which are likely to occur about the use of RUNKUT, we present the following typical questions with their answers.

Q1. When I called RUNKUT my program execution stopped. Why?

A. You probably forgot to load RUNKUT into memory from its resident disk or tape file. Or, on some systems this might happen if you failed to put the name of the subroutine which supplies the derivatives to RUNKUT in an EXTERNAL statement.

Q2. When I call RUNKUT my program time limits. What's wrong?

A. If this program has not successfully run before, you probably forgot to put the name of the derivative subroutine in an EXTERNAL statement. If this program has run before, you are apparently asking for too much accuracy or too long an integration in each call. Try using MP1 = 1 with a fairly small DELT so you can see how far RUNKUT is getting, instead of asking for integration over a long interval in one call.

Q3. When I called RUNKUT it printed the message indicating that I had given it an invalid argument. Where should I look for my error?

A. If you used nonstandard values for more than one of the step size criteria, OPT(1) to OPT(3), or more than one of the error criteria, OPT(4) to OPT(7), your problem is probably in having these in the wrong order or having them incompatible. Or, if you are using MP1 = 0, the problem may be with the parameter DELT, which should in this case be set to a value whose magnitude is twice the maximum step size you wish to use, and whose sign is positive for integration forward and negative for integration backwards. Also, if any one (or more) of N1D, NN, DELT, HMAX, HMIN, HO, or RMAX is zero, that could cause the problem.

Q4. I checked all those things and they all look good. What now?

A. Check to be sure that the calling sequence is in the right order and that all arguments are present. A simple transposition of say, N1D and NN could be the problem. If the calling sequence is all right, try calling ERRSET to make the error nonfatal (see Section 2.8) and print out the contents of the array OPT when RUNKUT returns to your program. Examining the contents of OPT should let you see how far RUNKUT had gotten in setting up the standard values, and this in turn will help pin down more nearly where the problem is.

Q5. RUNKUT returns with IERR = 2. How do I get it to go ahead and finish the integration?

A. RUNKUT may have found an actual singularity in the solution, in which case the routine cannot legitimately be made to complete the integration. If this is not thought to be the case, the problem may well be that your error criteria are unreasonable for this problem. In particular, if the solutions for one or more of the equations either start at zero or later cross through zero, you need to set AMAX to some positive value since one cannot preserve good relative error as a variable passes through a zero. As a general rule set AMAX to a value that is "small" relative to the nominal value of all the dependent variables. You may also wish to relax the value of ERR somewhat, or try using a smaller value of HMIN. Setting HMINB to zero should be a last resort.

Q6. I want to stop the integration in RUNKUT whenever one of the dependent variables reaches a certain value. How do I do this?

A. You should use a subroutine RKCHK(Y,K), as shown in the fourth example in Section 2.4. (Alternatively, you may use MP1 = 0, though with a slight loss in efficiency.) You may find that the integration is proceeding at such a large step size that the value you are looking for is passed by an appreciable margin in a single dual-mesh step. In this event, after RUNKUT returns control to the calling program you may wish to reverse the direction of the integration and proceed, using a smaller value of HMAX (and probably using MP1 = 0), until you cross back over the critical point. This process can be repeated, using successively smaller values of HMAX, until the desired value of the dependent variable is found to the desired accuracy.

8. Certification

This routine was subjected to a wide variety of tests. The performance of the routine throughout the tests was checked carefully. The nature of the tests, the reliability of the routine, the error analyses conducted, and the observed variation in accuracy are reported in this document. While it is believed that the facts recorded and the judgments expressed regarding accuracy and reliability are strong indications of the general quality and validity of the routine, the tests should not be considered to be exhaustive. The use of this routine outside of the stated range of application or in violation of stated restrictions may produce unspecified results. The statements made in this document are intended to apply only to those versions of the indicated routine which are released by the Sandia Laboratories Mathematical Program Library Project.

The author wrote RUNKUT, tested it on the CDC 6600, and prepared this document.

APPENDIX A

The RUNKUT Listing

APPENDIX A

The RUNKUT Listing

The listing of the CDC 6600 version of RUNKUT at the time of publication is given in this appendix. This version is limited to no more than 25 equations.

In order to modify RUNKUT for a larger maximum number of equations, lines RNKT0790, RNKT0800, and RNKT0920 must be changed. For example, to change the maximum to 50 equations, these three lines would become

```
      DIMENSION Y(51,),YP(51,A1(51),A2(51),A3(51),A4(51)
     1,DY1(51),DYA(51),DY2(51),YY1(51),YYA(51),YY2(51),ERRX(51)
```

and

```
      IF (NN.GT.50) GO TO 520
```

Statements RNKT0740 and RNKT0830 are known not to conform to ANSI Standard FORTRAN. The first is nonstandard because, when MP1 > 2, more columns of YANS are referenced than are dimensioned. The second is nonstandard because of the long Hollerith string. With the exception of these two statements, RUNKUT is believed to conform to ANSI standards.

```
SUBROUTINE RUNKUT(DERIV,YANS,N1D,NN,MP1,DELT,INIT,ERR,OPT,IERR,IP)RNKT0010
                                                                RNKT0020
SANDIA MATHEMATICAL PROGRAM LIBRARY                             RNKT0030
MATHEMATICAL COMPUTING SERVICES DIVISION 9422                   RNKT0040
SANDIA LABORATORIES                                             RNKT0050
P. O. BOX 5800                                                  RNKT0060
ALBUQUERQUE, NEW MEXICO  87115                                  RNKT0070
                                                                RNKT0080
WRITTEN BY RONDALL E JONES   DIV 9422                           RNKT0090
                                                                RNKT0100
CONTROL DATA 6600 VERSION                                       RNKT0110
                                                                RNKT0120
ABSTRACT                                                        RNKT0130
   RUNKUT INTEGRATES A SYSTEM OF N SIMULTANEOUS FIRST ORDER     RNKT0140
   DIFFERENTIAL EQUATIONS USING A VARIABLE STEP SIZE DUAL-MESH  RNKT0150
   RUNGE-KUTTA METHOD.  THREE OUTPUT MODES ARE PROVIDED.        RNKT0160
                                                                RNKT0170
DESCRIPTION OF PARAMETERS                                       RNKT0180
DERIV, N1D, NN, MP1 AND DELT ARE INPUT ONLY                     RNKT0190
IERR AND IP ARE OUTPUT ONLY                                     RNKT0200
YANS, INIT, ERR AND OPT ARE INPUT AND OUTPUT                    RNKT0210
NOTATION  - LET NP1 = NN+1                                      RNKT0220
   DERIV  - EXTERNAL NAME OF SUBROUTINE PROVIDING DERIVATIVES.  RNKT0230
            FORM MUST BE SUBROUTINE DERIV(Y,YP), WHERE Y IS THE RNKT0240
            ARRAY OF VARIABLES, WITH THE INDEPENDENT VARIABLE   RNKT0250
            IN Y(NP1), AND YP IS THE RESULTING ARRAY OF         RNKT0260
            DERIVATIVES.  YP(NP1) NEED NOT BE DEFINED.          RNKT0270
   YANS   - INITIAL CONDITION AND OUTPUT ARRAY.                 RNKT0280
            INITIAL CONDITIONS MUST BE IN COLUMN ONE WHEN RUNKUT RNKT0290
            IS CALLED, AND IN THE ORDER ESTABLISHED IN THE      RNKT0300
            SUBROUTINE DERIV.  THE INITIAL VALUE OF THE         RNKT0310
            INDEPENDENT VARIABLE MUST BE IN YANS(NP1,1).        RNKT0320
            ON RETURN, THE VALUES IN EACH COLUMN OF YANS        RNKT0330
            WILL BE IN SIMILAR ORDER.                           RNKT0340
   N1D    - THE NUMBER OF ROWS IN THE ACTUAL ARRAY YANS IN THE  RNKT0350
            CALLING PROGRAM. (INTEGER)                          RNKT0360
   NN     - THE NUMBER OF EQUATIONS BEING INTEGRATED. (INTEGER) RNKT0370
   MP1    - OUTPUT MODE PARAMETER (INTEGER)                     RNKT0380
            =0 MEANS DO TWO RUNGE-KUTTA STEPS (OF EQUAL SIZE,   RNKT0390
               WITH THE SIZE BEING DETERMINED BY RUNKUT),       RNKT0400
               RETURN ANSWERS IN COLUMN 1, DERIVATIVES IN       RNKT0410
               COLUMN 2.                                        RNKT0420
            =1 MEANS INTEGRATE UNTIL INDEPENDENT VARIABLE =     RNKT0430
               YANS(NP1,1) + DELT, RETURN ANSWERS AS IN MP1=0.  RNKT0440
               IF THE USER SUPPLIES THE ROUTINE RKCHK(Y,K), THEN RNKT0450
               INTEGRATION MAY BE TERMINATED AT ANY TIME (UP TO RNKT0460
               YANS(NP1,1)+DELT) BY SETTING K EQUAL TO 1.       RNKT0470
            .GE.2 MEANS RETURN M SETS (COLUMNS) OF ANSWERS,     RNKT0480
               WHERE MP1=M+1.  THE INDEPENDENT VARIABLE         RNKT0490
               INCREMENT BETWEEN SETS IS DELT.  INITIAL         RNKT0500
               CONDITIONS REMAIN IN COLUMN ONE WITH THE OUTPUT  RNKT0510
               IN COLUMNS 2 TO MP1.                             RNKT0520
   DELT   - INDEPENDENT VARIABLE INCREMENT BETWEEN OUTPUT SETS, RNKT0530
            WHENEVER MP1.GE.1.  DELT MAY BE NEGATIVE, IN WHICH  RNKT0540
```

```
C                CASE THE INTEGRATION PROCEDES TO SMALLER VALUES OF      RNKT0550
C                THE INDEPENDENT VARIABLE.  THUS, FOR ALL VALUES OF      RNKT0560
C                MP1, THE SIGN OF DELT GIVES THE DIRECTION OF            RNKT0570
C                INTEGRATION.                                            RNKT0580
C      INIT    - SHOULD BE .NE. 0  FOR ALL CALLS EXCEPT THOSE FOR        RNKT0590
C                WHICH INTEGRATION IS MERELY PROCEEDING FURTHER.         RNKT0600
C                INIT IS SET = 0 BY RUNKUT. (INTEGER)                    RNKT0610
C      ERR     - MAXIMUM RELATIVE ERROR, NORMALLY.                       RNKT0620
C      OPT     - ARRAY OF STEP SIZE CONDITIONS, ERROR DISTRIBUTIONS,     RNKT0630
C                OPTIONS, AND STORAGE.  OPT MUST CONTAIN AT LEAST 23     RNKT0640
C                LOCATIONS.  SEE THE OFFICIAL WRITE-UP ON RUNKUT.        RNKT0650
C      IERR    - ERROR FLAG (INTEGER)                                    RNKT0660
C                =1 IS NORMAL.                                           RNKT0670
C                =2 MEANS A SMALLER STEP SIZE THAN HMIN WAS NEEDED.      RNKT0680
C                   IP WILL CONTAIN THE NUMBER OF OUTPUT SETS OBTAINED.  RNKT0690
C                =3 MEANS AN INPUT PARAMETER WAS IMPROPER.               RNKT0700
C      IP      - NUMBER OF OUTPUT SETS SUCCESSFULLY FOUND.               RNKT0710
C                                                                        RNKT0720
      DOUBLE PRECISION DXSTRT,DDELT,DIP                                  RNKT0730
      DIMENSION YANS(N1D,2),OPT(23)                                      RNKT0740
C                                                                        RNKT0750
C     TO USE N EQUATIONS, CHANGE 26 IN THE FOLLOWING STATEMENT TO N+1    RNKT0760
C     AND CHANGE THE THIRD STATEMENT AFTER STATEMENT 1 ACCORDINGLY       RNKT0770
C                                                                        RNKT0780
      DIMENSION Y(26),YP(26),A1(26),A2(26),A3(26),A4(26)                 RNKT0790
     1,DY1(26),DYA(26),DY2(26),YY1(26),YYA(26),YY2(26),ERRX(26)          RNKT0800
      DIMENSION MES3(6)                                                  RNKT0810
      EQUIVALENCE (A1,YY1),(A2,DY2),(A3,ERRX)                            RNKT0820
      DATA (MES3(I),I=1,6)/3,41H AN INPUT PARAMETER HAD AN ILLEGAL VALUE RNKT0830
     1./                                                                 RNKT0840
C                                                                        RNKT0850
C     INITIALIZATION AREA                                                RNKT0860
C                                                                        RNKT0870
      IF (INIT) 1,10,1                                                   RNKT0880
    1 CONTINUE                                                           RNKT0890
      IF (N1D.LT.2)          GO TO 520                                   RNKT0900
      IF (NN.LT.1)           GO TO 520                                   RNKT0910
      IF (NN.GT.25)          GO TO 520                                   RNKT0920
      IF (N1D.LT.(NN+1))     GO TO 520                                   RNKT0930
      IF (MP1.LT.0)          GO TO 520                                   RNKT0940
      IF (DELT.EQ.0.0)       GO TO 520                                   RNKT0950
      IF (ERR.LE.0.0)        ERR = 1.0E-7                                RNKT0960
      PHDELT = 0.5*ABS(DELT)                                             RNKT0970
      IF (OPT(1).LE.0.0)     OPT(1) = PHDELT                             RNKT0980
      IF (OPT(3).LE.0.0)     OPT(3) = AMAX1(0.02*OPT(1),OPT(2))          RNKT0990
      IF (OPT(2).LE.0.0)     OPT(2) = 0.001*OPT(3)                       RNKT1000
      IF (OPT(1).GT.PHDELT) GO TO 520                                    RNKT1010
      IF (OPT(3).GT.OPT(1)) GO TO 520                                    RNKT1020
      IF (OPT(2).GT.OPT(3)) GO TO 520                                    RNKT1030
      IF (OPT(4).LT.0.0)     OPT(4) = 1.0                                RNKT1040
      IF (OPT(5).LT.0.0)     OPT(5) = 0.01*OPT(4)                        RNKT1050
      IF (OPT(6).LT.0.0)     OPT(6) = 0.0                                RNKT1060
      IF (OPT(7).LT.0.0)     OPT(7) = 0.0                                RNKT1070
      IF ((OPT(4)+OPT(5)+OPT(6)+OPT(7)).LE.0.0) GO TO 520                RNKT1080
```

```
      IF ((OPT(4).EQ.0.0).AND.(OPT(5).EQ.0.0).AND.(OPT(7).GT.OPT(6)))   RNKT1090
     1 GO TO 520                                                        RNKT1100
      IF ((OPT(6).EQ.0.0).AND.(OPT(7).EQ.0.0).AND.(OPT(5).GT.OPT(4)))   RNKT1110
     1 GO TO 520                                                        RNKT1120
      IF (OPT(8).LE.0.0)     OPT(8) = 2.0                               RNKT1130
      IF (OPT(9).LE.0.0)     OPT(9) = 0.5                               RNKT1140
      IF (OPT(8).LE.1.0)     GO TO 520                                  RNKT1150
      IF (OPT(9).GE.1.0)     GO TO 520                                  RNKT1160
      IF (OPT(10).LT.0.0)    OPT(10) = 1.0                              RNKT1170
      IF (OPT(13).LT.0.0)    OPT(13) = 3.0                              RNKT1180
      IF (OPT(14).LT.0.0)    OPT(14) = 0.02                             RNKT1190
      IF (OPT(14).GT.1.0)    OPT(14) = 1.0                              RNKT1200
      IF (OPT(15).LT.0.0)    OPT(15) = 1.0                              RNKT1210
      IF (OPT(16).LT.1.0)    OPT(16) = NN                               RNKT1220
      IF (OPT(16).GT.NN)     OPT(16) = NN                               RNKT1230
      OPT(17) = 0.0                                                     RNKT1240
      OPT(18) = OPT(4)*ERR                                              RNKT1250
      OPT(19) = OPT(5)*ERR                                              RNKT1260
      OPT(20) = OPT(6)*ERR                                              RNKT1270
      OPT(21) = OPT(7)*ERR                                              RNKT1280
      OPT(22) = 0.0                                                     RNKT1290
      OPT(23) = SIGN(OPT(3),DELT)                                       RNKT1300
      INIT    = 0                                                       RNKT1310
C                                                                       RNKT1320
C     NON-INITIALIZATION ENTRY                                          RNKT1330
C                                                                       RNKT1340
   10 N       = NN                                                      RNKT1350
      NP1     = NN+1                                                    RNKT1360
C                                                                       RNKT1370
C     FETCH CONSTANTS                                                   RNKT1380
C                                                                       RNKT1390
      HMAX    = OPT(1)                                                  RNKT1400
      HMIN    = OPT(2)                                                  RNKT1410
      CSNFAC= OPT(8)                                                    RNKT1420
      REFFAC= OPT(9)                                                    RNKT1430
      HMINB  = OPT(10)                                                  RNKT1440
      EXTRAP= OPT(11)                                                   RNKT1450
      FIXHO  = OPT(12)                                                  RNKT1460
      NTG     = OPT(13)                                                 RNKT1470
      HTOLF  = OPT(14)                                                  RNKT1480
      HINTR  = OPT(15)                                                  RNKT1490
      NEQCHK= OPT(16)                                                   RNKT1500
      ITG     = OPT(17)                                                 RNKT1510
      RELMAX= OPT(18)                                                   RNKT1520
      RELMIN= OPT(19)                                                   RNKT1530
      ABSMAX= OPT(20)                                                   RNKT1540
      ABSMIN= OPT(21)                                                   RNKT1550
      H       = OPT(23)                                                 RNKT1560
C                                                                       RNKT1570
      HPH     = H+H                                                     RNKT1580
      HLFH    = 0.5*H                                                   RNKT1590
      PH      = ABS(H)                                                  RNKT1600
      HTOL    = HTOLF*PH                                                RNKT1610
      HGO     = PH+PH+HTOL                                              RNKT1620
```

```
      IP     = 0                                                        RNKT1630
      IPRINT= 0                                                         RNKT1640
C                                                                       RNKT1650
C     FETCH INITIAL CONDITIONS                                          RNKT1660
C                                                                       RNKT1670
      DO 102 IY=1,NP1                                                   RNKT1680
  102 Y(IY) = YANS(IY,1)                                                RNKT1690
      IF (MP1.EQ.0) GO TO 125                                           RNKT1700
      M      = MP1-1                                                    RNKT1710
      DXSTRT= Y(NP1)                                                    RNKT1720
      DDELT = DELT                                                      RNKT1730
C                                                                       RNKT1740
C     COMPUTE NEXT *PRINT* POSITION, ETC.                               RNKT1750
C                                                                       RNKT1760
  110 IPRINT= 0                                                         RNKT1770
      IP     = IP+1                                                     RNKT1780
      DIP    = IP                                                       RNKT1790
      XPRINT= DXSTRT + DIP*DDELT                                        RNKT1800
C                                                                       RNKT1810
C     PRINT CHECK                                                       RNKT1820
C                                                                       RNKT1830
  115 IF (HGO.LT.ABS(XPRINT-Y(NP1))) GO TO 125                          RNKT1840
      IPRINT= 1                                                         RNKT1850
      IF (HINTR) 125,125,117                                            RNKT1860
  117 HKEEP = H                                                         RNKT1870
      HPH    = XPRINT-Y(NP1)                                            RNKT1880
      H      = 0.5*HPH                                                  RNKT1890
      HLFH   = 0.5*H                                                    RNKT1900
      PH     = ABS(H)                                                   RNKT1910
      HTOL   = HTOLF*PH                                                 RNKT1920
      HGO    = PH+PH+HTOL                                               RNKT1930
C                                                                       RNKT1940
C     ************************                                          RNKT1950
C     START OF INTEGRATION STEP                                         RNKT1960
C                                                                       RNKT1970
C     FIRST DO DOUBLE-SIZE STEP                                         RNKT1980
C                                                                       RNKT1990
  125 CALL DERIV (Y,YP)                                                 RNKT2000
      IF (FIXHO) 132,132,155                                            RNKT2010
  132 DO 135 I=1,N                                                      RNKT2020
  135 YY1(I)   = Y(I) + H*YP(I)                                         RNKT2030
      YY1(NP1)= Y(NP1) + H                                              RNKT2040
      CALL DERIV (YY1,A2)                                               RNKT2050
      DO 140 I=1,N                                                      RNKT2060
  140 YY1(I)   = Y(I) + H*A2(I)                                         RNKT2070
      CALL DERIV (YY1,A3)                                               RNKT2080
      DO 145 I=1,N                                                      RNKT2090
  145 YY1(I)   = Y(I) + HPH*A3(I)                                       RNKT2100
      YY1(NP1)= Y(NP1) + HPH                                            RNKT2110
      CALL DERIV (YY1,A4)                                               RNKT2120
      DO 150 I=1,N                                                      RNKT2130
  150 DY1(I)   = HPH*(YP(I) + 2.0*(A2(I)+A3(I)) + A4(I))/6.0            RNKT2140
C                                                                       RNKT2150
C     DO TWO STEPS OF SIZE H                                            RNKT2160
```

```
C                                                                       RNKT2170
  155 DO 160 I=1,N                                                      RNKT2180
  160 YYA(I)   = Y(I) + HLFH*YP(I)                                      RNKT2190
      YYA(NP1)= Y(NP1) + HLFH                                           RNKT2200
      CALL DERIV (YYA,A2)                                               RNKT2210
      DO 165 I=1,N                                                      RNKT2220
  165 YYA(I)   = Y(I) + HLFH*A2(I)                                      RNKT2230
      CALL DERIV (YYA,A3)                                               RNKT2240
      DO 170 I=1,N                                                      RNKT2250
  170 YYA(I)   = Y(I) + H*A3(I)                                         RNKT2260
      YYA(NP1)= Y(NP1) + H                                              RNKT2270
      CALL DERIV (YYA,A4)                                               RNKT2280
      DO 175 I=1,N                                                      RNKT2290
      DYA(I)   = H*(YP(I) + 2.0*(A2(I)+A3(I)) + A4(I))/6.0              RNKT2300
  175 YYA(I)   = Y(I) + DYA(I)                                          RNKT2310
C                                                                       RNKT2320
      CALL DERIV (YYA,A1)                                               RNKT2330
      DO 180 I=1,N                                                      RNKT2340
  180 YY2(I)   = YYA(I) + HLFH*A1(I)                                    RNKT2350
      YY2(NP1)= YYA(NP1) + HLFH                                         RNKT2360
      CALL DERIV (YY2,A2)                                               RNKT2370
      DO 185 I=1,N                                                      RNKT2380
  185 YY2(I)   = YYA(I) + HLFH*A2(I)                                    RNKT2390
      CALL DERIV (YY2,A3)                                               RNKT2400
      DO 190 I=1,N                                                      RNKT2410
  190 YY2(I)   = YYA(I) + H*A3(I)                                       RNKT2420
      YY2(NP1)= Y(NP1) + HPH                                            RNKT2430
      CALL DERIV (YY2,A4)                                               RNKT2440
      DO 195 I=1,N                                                      RNKT2450
      DY2(I)   = DYA(I) + H*(A1(I) + 2.0*(A2(I)+A3(I)) + A4(I))/6.0     RNKT2460
  195 YY2(I)   = Y(I) + DY2(I)                                          RNKT2470
C                                                                       RNKT2480
C     ************************                                          RNKT2490
C     STEP SIZE CONTROL AREA                                            RNKT2500
C                                                                       RNKT2510
C     ESTIMATE ERRORS AND COMPARE WITH ALLOWED ERRORS                   RNKT2520
C                                                                       RNKT2530
      IF (FIXHO) 199,199,255                                            RNKT2540
  199 DO 205 I=1,NEQCHK                                                 RNKT2550
      ERRX(I) = ABS(DY1(I)-DY2(I))/30.0                                 RNKT2560
      IEQB    = I                                                       RNKT2570
      IF (ERRX(I).GT.(RELMAX*ABS(YY2(I))+ABSMAX)) GO TO 225             RNKT2580
  205 CONTINUE                                                          RNKT2590
      IF (PH.GE.HMAX) GO TO 248                                         RNKT2600
      DO 210 I=1,NEQCHK                                                 RNKT2610
      IF (ERRX(I).GE.(RELMIN*ABS(YY2(I))+ABSMIN)) GO TO 248             RNKT2620
  210 CONTINUE                                                          RNKT2630
      ITG    = ITG+1                                                    RNKT2640
      IF (ITG.LT.NTG) GO TO 250                                         RNKT2650
      IF (IPRINT) 215,215,250                                           RNKT2660
C                                                                       RNKT2670
C     COARSEN                                                           RNKT2680
C                                                                       RNKT2690
  215 ITG    = 0                                                        RNKT2700
```

```
      H      = H*CSNFAC                                                    RNKT2710
      IF (ABS(H).GT.HMAX) H = SIGN(HMAX,DELT)                              RNKT2720
      HLFH   = 0.5*H                                                       RNKT2730
      HPH    = H+H                                                         RNKT2740
      PH     = ABS(H)                                                      RNKT2750
      HTOL   = HTOLF*PH                                                    RNKT2760
      HGO    = PH+PH+HTOL                                                  RNKT2770
      GO TO 250                                                            RNKT2780
C                                                                          RNKT2790
C     REFINE                                                               RNKT2800
C                                                                          RNKT2810
  225 ITG    = 0                                                           RNKT2820
      IF(PH-HMIN) 226,226,227                                              RNKT2830
  226 IF (HMINB) 250,250,510                                               RNKT2840
  227 PH = PH*REFFAC                                                       RNKT2850
      IF (PH.LT.HMIN) PH = HMIN                                            RNKT2860
      H      = SIGN(PH,DELT)                                               RNKT2870
      HLFH   = 0.5*H                                                       RNKT2880
      HPH    = H+H                                                         RNKT2890
      HTOL   = HTOLF*PH                                                    RNKT2900
      HGO    = PH+PH+HTOL                                                  RNKT2910
      IPRINT= 0                                                            RNKT2920
      IF (REFFAC.NE.0.5) GO TO 230                                         RNKT2930
      DO 228 I=1,N                                                         RNKT2940
  228 DY1(I)= DYA(I)                                                       RNKT2950
      IF (HINTR.GT.0.0) GO TO 155                                          RNKT2960
      IF (MP1.EQ.0) GO TO 155                                              RNKT2970
      IF (HGO.GE.ABS(XPRINT-Y(NP1))) IPRINT = 1                            RNKT2980
      GO TO 155                                                            RNKT2990
C                                                                          RNKT3000
  230 IF (HINTR.GT.0.0) GO TO 132                                          RNKT3010
      IF (MP1.EQ.0) GO TO 132                                              RNKT3020
      IF (HGO.GE.ABS(XPRINT-Y(NP1))) IPRINT = 1                            RNKT3030
      GO TO 132                                                            RNKT3040
C                                                                          RNKT3050
C     ************************                                             RNKT3060
C                                                                          RNKT3070
C     IF ERRORS ARE O.K., FINISH THE BOOKEEPING, ETC., FOR THIS STEP.      RNKT3080
C                                                                          RNKT3090
  248 ITG    = 0                                                           RNKT3100
  250 IF (EXTRAP) 255,255,265                                              RNKT3110
  255 DO 260 I=1,NP1                                                       RNKT3120
  260 Y(I)   = YY2(I)                                                      RNKT3130
      GO TO 275                                                            RNKT3140
  265 DO 270 I=1,N                                                         RNKT3150
  270 Y(I)   = YY2(I) + (DY2(I)-DY1(I))/15.0                               RNKT3160
      Y(NP1)= YY2(NP1)                                                     RNKT3170
C                                                                          RNKT3180
C     CHECK FOR OUTPUT CONDITIONS, ETC.                                    RNKT3190
C                                                                          RNKT3200
  275 IF (MP1-1) 300,325,350                                               RNKT3210
C                                                                          RNKT3220
  300 DO 310 IY=1,NP1                                                      RNKT3230
  310 YANS(IY,1) = Y(IY)                                                   RNKT3240
```

```
      CALL DERIV (Y,YANS(1,2))                                          RNKT3250
      YANS(NP1,2)= Y(NP1)                                               RNKT3260
      IP = 1                                                            RNKT3270
      GO TO 600                                                         RNKT3280
C                                                                       RNKT3290
  325 K = 0                                                             RNKT3300
      CALL RKCHK(Y,K)                                                   RNKT3310
      IF (K.EQ.1) IPRINT=1                                              RNKT3320
      IF (IPRINT) 115,115,330                                           RNKT3330
  330 DO 335 IY=1,NP1                                                   RNKT3340
  335 YANS(IY,1) = Y(IY)                                                RNKT3350
      CALL DERIV (Y,YANS(1,2))                                          RNKT3360
      YANS(NP1,2)= Y(NP1)                                               RNKT3370
      IF (HINTR.GT.0.0) H = HKEEP                                       RNKT3380
      GO TO 600                                                         RNKT3390
C                                                                       RNKT3400
  350 IF (IPRINT) 115,115,355                                           RNKT3410
  355 IPP1 = IP+1                                                       RNKT3420
      DO 360 IY=1,NP1                                                   RNKT3430
  360 YANS(IY,IPP1) = Y(IY)                                             RNKT3440
      IF (HINTR) 370,370,365                                            RNKT3450
  365 H    = HKEEP                                                      RNKT3460
      HLFH = 0.5*H                                                      RNKT3470
      HPH  = H+H                                                        RNKT3480
      PH   = ABS(H)                                                     RNKT3490
      HTOL = HTOLF*PH                                                   RNKT3500
      HGO  = PH+PH+HTOL                                                 RNKT3510
  370 IF (IP-M) 110,600,600                                             RNKT3520
C                                                                       RNKT3530
C     SET OUTPUT CODES,ETC., CALL ERRCHK IF NEEDED, AND EXIT.           RNKT3540
C                                                                       RNKT3550
  510 IERR = 2                                                          RNKT3560
      IF (IP.GT.0) IP = IP-1                                            RNKT3570
      OPT(17) = 0.0                                                     RNKT3580
      OPT(22) = IEQB                                                    RNKT3590
      OPT(23) = H                                                       RNKT3600
      RETURN                                                            RNKT3610
C                                                                       RNKT3620
  520 IERR = 3                                                          RNKT3630
      IP   = 0                                                          RNKT3640
      CALL ERRCHK(41,MES3)                                              RNKT3650
      RETURN                                                            RNKT3660
C                                                                       RNKT3670
  600 IERR = 1                                                          RNKT3680
      OPT(17) = ITG                                                     RNKT3690
      OPT(22) = 0.0                                                     RNKT3700
      OPT(23) = H                                                       RNKT3710
      RETURN                                                            RNKT3720
      END                                                               RNKT3730
      SUBROUTINE RKCHK(Y,K)                                             RNKT3740
      J=0                                                               RNKT3750
      RETURN                                                            RNKT3760
      END                                                               RNKT3770
```

APPENDIX B

Test Results for CDC 6600

APPENDIX B

Test Results for CDC 6600

On the CDC 6600 RUNKUT requires 2400_8 words of storage (when set up for 25 equations).

Sample tables of the tests described in Section 6 are included in this appendix. The value MP1 = 2 was used in obtaining these tables, so the values of DELT can be obtained from the starting and ending time (independent variable) values given at the beginning of each table. The step size criteria were left at their standard values except for System No. 4 of Section 6 (for which HMIN = 10^{-6} and HO = 0.01) and System No. 6 (for which HMIN = 10^{-10} in the top half of its table). The first half of each of these tables is for a nominal value of ERR(10^{-8}) with various values of RMAX and AMAX. The second half is for fixed values of RMAX and AMAX with various values of ERR. Blank lines indicate that RUNKUT could not finish the integration (i.e., RUNKUT returned with IERR = 2).

The first seven tables are for each of the seven systems of equations given in Sections 6.3 and 6.4 using pure relative error (AMAX = 0.0) in the second half of each table.

Tables III and V are good examples of the fact that, when integrating systems of equations which have zero crossings in some of their variables, a significant amount of execution time can be saved with little loss of accuracy by using a positive value for AMAX. Table IV illustrates how RUNKUT can greatly coarsen the integration step size when the system being integrated is very stable. Table VI shows the difficulty with using pure relative error control with an equation which sometimes lingers near zero. Table VII demonstrates that RUNKUT is simply incapable of accurately integrating some systems of equations. This innocent-looking equation, $y'(t) = 100(y(t) - t^2)$, is actually extremely unstable mathematically as well as being particularly subject to numerical inaccuracy, so that no classical integration technique will solve it.

Table VIII is like Table III except that the optional Richardson extrapolation was used. Note that for larger values of ERR the extrapolation did not consistently help the accuracy and in some cases actually hurt the accuracy. For small values of ERR, however, the extrapolation definitely did help, giving up to one full extra correct significant figure.

Table IX is for System No. 5 with a positive value of AMAX. This Table is given to show that the behavior of the error with decreasing value of ERR when a combination of relative and absolute error is used is similar in character to the behavior when just relative error is used.

We note here that the ratio AMAX/RMAX gives the ordinate value at which the error control process switches from being essentially relative error (for large ordinates) to being essentially absolute error (for smaller ordinates). Thus, when using a positive value for AMAX to avoid excessive refinements near zero crossings, this ratio should be picked so as to be small relative to all the nominal variable values. This choice will make the error control process use essentially relative error except at the zero crossing.

TABLE I

Data for System of Equations Number 1

INTEGRATION BEGAN AT T= -1.0
VALUES OF THE ERROR AT T= 9.0 ARE GIVEN.

ERR = 1.000E-08

RMAX	AMAX	Y1 REL ERR	Y1 ABS ERR	Y2 REL ERR	Y2 ABS ERR		
1.00	0.00	5.43E-07	6.70E-11	-5.00E-07	-4.05E-03	TIME=	.068 SEC.
.99	.01	5.43E-07	6.70E-11	-5.00E-07	-4.05E-03	TIME=	.068 SEC.
.90	.10	5.43E-07	6.70E-11	-5.00E-07	-4.05E-03	TIME=	.067 SEC.
.50	.50	5.43E-07	6.70E-11	-5.00E-07	-4.05E-03	TIME=	.068 SEC.

IN THE FOLLOWING RMAX=1.00 , AND AMAX=0.00

ERR	Y1 REL ERR	Y1 ABS ERR	Y2 REL ERR	Y2 ABS ERR		
1.00E-02	2.41E-03	2.97E-07	-1.24E-03	-1.00E+01	TIME=	.012 SEC.
1.00E-03	2.41E-03	2.97E-07	-1.24E-03	-1.00E+01	TIME=	.013 SEC.
1.00E-04	1.46E-04	1.80E-08	-1.04E-04	-8.47E-01	TIME=	.018 SEC.
1.00E-05	1.46E-04	1.80E-08	-1.04E-04	-8.47E-01	TIME=	.020 SEC.
1.00E-06	9.06E-06	1.12E-09	-7.67E-06	-6.21E-02	TIME=	.034 SEC.
1.00E-07	9.06E-06	1.12E-09	-7.67E-06	-6.21E-02	TIME=	.033 SEC.
1.00E-08	5.43E-07	6.70E-11	-5.00E-07	-4.05E-03	TIME=	.067 SEC.
1.00E-09	3.32E-08	4.10E-12	-3.19E-08	-2.58E-04	TIME=	.133 SEC.
1.00E-10	3.32E-08	4.10E-12	-3.19E-08	-2.58E-04	TIME=	.133 SEC.

TABLE II

Data for System of Equations Number 2

INTEGRATION BEGAN AT T= 0.0
VALUES OF THE ERROR AT T= 5.0 ARE GIVEN.

ERR = 1.000E-08

RMAX	AMAX	Y1 REL ERR	Y1 ABS ERR		
1.00	0.00	1.51E-06	2.09E-17	TIME=	.147 SEC.
.99	.01	3.52E-02	4.88E-13	TIME=	.069 SEC.
.90	.10	8.48E-02	1.18E-12	TIME=	.058 SEC.
.50	.50	9.33E-02	1.30E-12	TIME=	.041 SEC.

IN THE FOLLOWING RMAX=1.00 , AND AMAX=0.00

ERR	Y1 REL ERR	Y1 ABS ERR		
1.00E-02	4.34E-02	6.03E-13	TIME=	.018 SEC.
1.00E-03	9.02E-03	1.25E-13	TIME=	.021 SEC.
1.00E-04	1.32E-03	1.83E-14	TIME=	.033 SEC.
1.00E-05	3.38E-04	4.70E-15	TIME=	.044 SEC.
1.00E-06	4.16E-05	5.78E-16	TIME=	.068 SEC.
1.00E-07	7.31E-06	1.01E-16	TIME=	.105 SEC.
1.00E-08	1.51E-06	2.09E-17	TIME=	.147 SEC.
1.00E-09	1.72E-07	2.39E-18	TIME=	.252 SEC.
1.00E-10	3.08E-08	4.28E-19	TIME=	.399 SEC.

TABLE III

Data for System of Equations Number 3

INTEGRATION BEGAN AT T= 2.0
VALUES OF THE ERROR AT T= -5.0 ARE GIVEN.

ERR = 1.000E-08

RMAX	AMAX	Y1 REL ERR	Y1 ABS ERR	Y2 REL ERR	Y2 ABS ERR		
1.00	0.00	1.83E-08	1.76E-08	-2.34E-07	-6.65E-08	TIME=	.088 SEC.
.99	.01	1.98E-08	1.90E-08	-2.54E-07	-7.19E-08	TIME=	.081 SEC.
.90	.10	2.33E-08	2.24E-08	-2.98E-07	-8.46E-08	TIME=	.068 SEC.
.50	.50	2.33E-08	2.24E-08	-2.98E-07	-8.46E-08	TIME=	.069 SEC.

IN THE FOLLOWING RMAX=1.00 , AND AMAX=0.00

ERR	Y1 REL ERR	Y1 ABS ERR	Y2 REL ERR	Y2 ABS ERR		
1.00E-02	-5.08E-04	-4.87E-04	-1.01E-02	-2.88E-03	TIME=	.010 SEC.
1.00E-03	1.67E-05	1.60E-05	-1.02E-03	-2.89E-04	TIME=	.013 SEC.
1.00E-04	8.74E-06	8.38E-06	-4.48E-04	-1.27E-04	TIME=	.017 SEC.
1.00E-05	2.81E-06	2.69E-06	-5.45E-05	-1.54E-05	TIME=	.024 SEC.
1.00E-06	2.83E-07	2.71E-07	-4.09E-06	-1.16E-06	TIME=	.042 SEC.
1.00E-07	1.70E-07	1.63E-07	-2.45E-06	-6.95E-07	TIME=	.058 SEC.
1.00E-08	1.83E-08	1.76E-08	-2.34E-07	-6.65E-08	TIME=	.088 SEC.
1.00E-09	1.46E-09	1.40E-09	-1.76E-08	-5.00E-09	TIME=	.145 SEC.
1.00E-10	9.53E-10	9.14E-10	-1.15E-08	-3.26E-09	TIME=	.199 SEC.

TABLE IV

Data for System of Equations Number 4

INTEGRATION BEGAN AT T= 0.0
VALUES OF THE ERROR AT T= 1000000.0 ARE GIVEN.

ERR = 1.000E-08

RMAX	AMAX	Y1 REL ERR	Y1 ABS ERR		
1.00	0.00	1.81E-08	1.81E-14	TIME=	.147 SEC.
.99	.01	2.67E-06	2.67E-12	TIME=	.099 SEC.
.90	.10	7.15E-06	7.15E-12	TIME=	.081 SEC.
.50	.50	8.20E-06	8.20E-12	TIME=	.068 SEC.

IN THE FOLLOWING RMAX=1.00 , AND AMAX=0.00

ERR	Y1 REL ERR	Y1 ABS ERR		
1.00E-02	8.77E-06	8.77E-12	TIME=	.036 SEC.
1.00E-03	8.77E-06	8.77E-12	TIME=	.036 SEC.
1.00E-04	5.11E-06	5.11E-12	TIME=	.041 SEC.
1.00E-05	1.72E-06	1.72E-12	TIME=	.053 SEC.
1.00E-06	4.06E-07	4.06E-13	TIME=	.071 SEC.
1.00E-07	8.98E-08	8.98E-14	TIME=	.102 SEC.
1.00E-08	1.81E-09	1.81E-14	TIME=	.147 SEC.
1.00E-09	3.20E-09	3.20E-15	TIME=	.225 SEC.
1.00E-10	6.32E-10	6.32E-16	TIME=	.344 SEC.

TABLE V

Data for System of Equations Number 5

INTEGRATION BEGAN AT T= 0.0
VALUES OF THE ERROR AT T= 10.0 ARE GIVEN.

ERR = 1.000E-08

RMAX	AMAX	Y1 REL ERR	Y1 ABS ERR	Y2 REL ERR	Y2 ABS ERR		
1.00	0.00	1.86E-06	-9.40E-07	-6.99E-07	-6.03E-07	TIME=	1.320 SEC.
.99	.01	1.91E-06	-9.66E-07	-7.18E-07	-6.19E-07	TIME=	1.261 SEC.
.90	.10	2.26E-06	-1.15E-06	-8.51E-07	-7.34E-07	TIME=	1.164 SEC.
.50	.50	5.99E-06	-3.03E-06	-2.28E-06	-1.97E-06	TIME=	.890 SEC.

IN THE FOLLOWING RMAX=1.00 , AND AMAX=0.00

ERR	Y1 REL ERR	Y1 ABS ERR	Y2 REL ERR	Y2 ABS ERR		
1.00E-02	1.10E-01	-5.57E-02	-1.91E-01	-1.65E-01	TIME=	.066 SEC.
1.00E-03	1.46E-02	-7.40E-03	-1.00E-02	-8.64E-03	TIME=	.133 SEC.
1.00E-04	2.14E-03	-1.08E-03	-1.13E-03	-9.77E-04	TIME=	.217 SEC.
1.00E-05	3.37E-04	-1.71E-04	-1.51E-04	-1.30E-04	TIME=	.336 SEC.
1.00E-06	7.01E-05	-3.55E-05	-2.94E-05	-2.53E-05	TIME=	.540 SEC.
1.00E-07	1.10E-05	-5.55E-06	-4.30E-06	-3.71E-06	TIME=	.838 SEC.
1.00E-08	1.86E-06	-9.40E-07	-6.99E-07	-6.03E-07	TIME=	1.320 SEC.
1.00E-09	3.77E-07	-1.91E-07	-1.37E-07	-1.18E-07	TIME=	1.993 SEC.
1.00E-10	4.97E-08	-2.51E-08	-1.77E-08	-1.53E-08	TIME=	3.154 SEC.

TABLE VI

Data for System of Equations Number 6

INTEGRATION BEGAN AT T= 0.0
VALUES OF THE ERROR AT T= 3.0 ARE GIVEN.

ERR = 1.000E-08

RMAX	AMAX	Y1 REL ERR	Y1 ABS ERR		
1.00	0.00				
.99	.01	-1.28E-07	-1.28E-07	TIME=	.138 SEC.
.90	.10	-1.37E-07	-1.37E-07	TIME=	.128 SEC.
.50	.50	-1.65E-07	-1.65E-07	TIME=	.123 SEC.

IN THE FOLLOWING RMAX=1.00 , AND AMAX=0.00

ERR	Y1 REL ERR	Y1 ABS ERR		
1.00E-02	-1.00E-02	-1.00E-02	TIME=	.011 SEC.
1.00E-03	1.28E-02	1.28E-02	TIME=	.032 SEC.
1.00E-04				
1.00E-05				
1.00E-06				
1.00E-07				
1.00E-08				
1.00E-09				
1.00E-10				

TABLE VII

Data for System of Equations Number 7

INTEGRATION BEGAN AT T= 0.0
VALUES OF THE ERROR AT T= 1.0 ARE GIVEN.

ERR = 1.000E-08

		Y1		
RMAX	AMAX	REL ERR	ABS ERR	
1.00	0.00	-8.12E+32	-8.28E+32	TIME= .850 SEC.
.99	.01	-1.26E+34	-1.28E+34	TIME= .836 SEC.
.90	.10	-1.26E+34	-1.28E+34	TIME= .441 SEC.
.50	.50	-1.89E+35	-1.92E+35	TIME= .811 SEC.

IN THE FOLLOWING RMAX=1.00 , AND AMAX=0.00

	Y1		
ERR	REL ERR	ABS ERR	
1.00E-02	-2.23E+37	-2.27E+37	TIME= .030 SEC.
1.00E-03	-2.54E+36	-2.60E+36	TIME= .063 SEC.
1.00E-04	-2.15E+35	-2.19E+35	TIME= .118 SEC.
1.00E-05	-1.89E+35	-1.92E+35	TIME= .222 SEC.
1.00E-06	-1.27E+34	-1.29E+34	TIME= .236 SEC.
1.00E-07	-1.13E+33	-1.15E+33	TIME= .444 SEC.
1.00E-08	-8.12E+32	-8.28E+32	TIME= .852 SEC.
1.00E-09	-5.23E+31	-5.33E+31	TIME= .915 SEC.
1.00E-10	-4.90E+30	-5.00E+30	TIME= 1.736 SEC.

TABLE VIII

Data for System of Equations Number 3

INTEGRATION BEGAN AT T= 2.0
VALUES OF THE ERROR AT T= -5.0 ARE GIVEN.

ERR = 1.000E-09

		Y1		Y2			
RMAX	AMAX	REL ERR	ABS ERR	REL ERR	ABS ERR		
1.00	0.00	2.14E-09	2.06E-09	1.80E-09	5.10E-10	TIME=	.089 SEC.
.99	.01	2.33E-09	2.23E-09	1.95E-09	5.53E-10	TIME=	.082 SEC.
.90	.10	2.76E-09	2.64E-09	2.31E-09	6.54E-10	TIME=	.069 SEC.
.50	.50	2.76E-09	2.64E-09	2.31E-09	6.54E-10	TIME=	.069 SEC.

IN THE FOLLOWING RMAX=1.00 , AND AMAX=0.00

	Y1		Y2			
ERR	REL ERR	ABS ERR	REL ERR	ABS ERR		
1.00E-02	1.33E-03	1.27E-03	-2.02E-03	-5.72E-04	TIME=	.009 SEC.
1.00E-03	7.43E-05	7.13E-05	-1.80E-05	-5.11E-06	TIME=	.014 SEC.
1.00E-04	3.12E-05	2.99E-05	-6.66E-06	-1.89E-06	TIME=	.019 SEC.
1.00E-05	1.99E-06	1.91E-06	7.37E-07	2.09E-07	TIME=	.026 SEC.
1.00E-06	7.53E-08	7.22E-08	5.10E-08	1.45E-08	TIME=	.044 SEC.
1.00E-07	4.41E-08	4.23E-08	3.00E-08	8.52E-09	TIME=	.062 SEC.
1.00E-08	2.14E-09	2.06E-09	1.80E-09	5.10E-10	TIME=	.092 SEC.
1.00E-09	8.01E-11	7.68E-11	7.47E-11	2.12E-11	TIME=	.155 SEC.
1.00E-10	5.08E-11	4.87E-11	4.83E-11	1.37E-11	TIME=	.201 SEC.

EXTRAPOLATION WAS USED IN ALL THE ABOVE.

TABLE IX

Data for System of Equations Number 5

INTEGRATION BEGAN AT T= 0.0
VALUES OF THE ERROR AT T= 10.0 ARE GIVEN.

ERR = 1.000E-08

RMAX	AMAX	Y1 REL ERR	Y1 ABS ERR	Y2 REL ERR	Y2 ABS ERR		
1.00	0.00	1.86E-06	-9.40E-07	-6.99E-07	-6.03E-07	TIME=	1.321 SEC.
.99	.01	1.91E-06	-9.66E-07	-7.18E-07	-6.19E-07	TIME=	1.263 SEC.
.90	.10	2.26E-06	-1.15E-06	-8.51E-07	-7.34E-07	TIME=	1.164 SEC.
.50	.50	5.99E-06	-3.03E-06	-2.28E-06	-1.97E-06	TIME=	.889 SEC.

IN THE FOLLOWING RMAX= .90 , AND AMAX= .10

ERR	Y1 REL ERR	Y1 ABS ERR	Y2 REL ERR	Y2 ABS ERR		
1.00E-02	1.15E-01	-5.84E-02	-3.79E-01	-3.27E-01	TIME=	.053 SEC.
1.00E-03	3.03E-02	-1.53E-02	-2.32E-02	-2.00E-02	TIME=	.098 SEC.
1.00E-04	3.57E-03	-1.81E-03	-1.95E-03	-1.68E-03	TIME=	.175 SEC.
1.00E-05	5.98E-04	-3.03E-04	-2.79E-04	-2.40E-04	TIME=	.287 SEC.
1.00E-06	1.11E-04	-5.60E-05	-4.66E-05	-4.02E-05	TIME=	.434 SEC.
1.00E-07	1.39E-05	-7.06E-06	-5.46E-06	-4.71E-06	TIME=	.707 SEC.
1.00E-08	2.26E-06	-1.15E-06	-8.51E-07	-7.34E-07	TIME=	1.164 SEC.
1.00E-09	4.52E-07	-2.29E-07	-1.65E-07	-1.42E-07	TIME=	1.735 SEC.
1.00E-10	6.03E-08	-3.05E-08	-2.16E-08	-1.86E-08	TIME=	2.746 SEC.

APPENDIX C

Control Cards for Using RUNKUT on the CDC 6600

APPENDIX C

Control Cards for Using RUNKUT on the CDC 6600

RUNKUT is maintained in a library file for the convenience of the Control Data 6600 users at Sandia Laboratories, Albuquerque, New Mexico. The name of the file is MATHLIB. Questions concerning the availability of RUNKUT on the Control Data 6600 at Sandia Laboratories, Livermore, California, should be directed to the Numerical Applications Division 8321.

Two control cards, ATTACH and COLLECT, are required for using the mathematical library file. The ATTACH card is required prior to using COLLECT. The COLLECT processor operates on one relocatable binary file and from one to six library files. The library files are searched for routines which contain entry points matching unsatisfied external references in the relocatable binary file. Such routines are added to the relocatable binary file.

A complete typical example follows:

```
JOB CARD
ACCOUNT CARD
FUN,S.
ATTACH,MATHLIB,MATHLIB.
COLLECT,LGO,MATHLIB.
LGO.
7/8/9 punch in column 1
Program
7/8/9 punch in column 1
Data
6/7/8/9 punch in column 1
```

In the above example, external references in LGO are satisfied, if possible, by selectively adding routines to LGO from MATHLIB. Additional information on the COLLECT processor with examples is contained in UR0004/6600.[3]

References

1. C. B. Bailey, A Guide to the Sandia Mathematical Program Library, SC-M-69-337, Sandia Laboratories, Albuquerque, New Mexico, January 1970.

2. F. Ceschino and J. Kuntzman, Numerical Solution of Initial Value Problems, Prentice-Hall, Englewood Cliffs, New Jersey, 1966.

3. P. A. Lemke, Auxiliary Library Routines PREP and COLLECT, Sandia Computing Publication UR0004/6600, Sandia Laboratories, Albuquerque, New Mexico, June 1969.