205
8/24/79

Dr. 3055

ORNL/CSD/TM-64

Lecture Notes on Ordinary Differential Equations Software, User's Guides for ODE, RKF45, GEAR and EPISODE

J. A. Wenzel
R. E. Funderlic

OAK RIDGE NATIONAL LABORATORY
OPERATED BY UNION CARBIDE CORPORATION · FOR THE DEPARTMENT OF ENERGY

Printed in the United States of America. Available from
National Technical Information Service
U.S. Department of Commerce
5285 Port Royal Road, Springfield, Virginia 22161
Price: Printed Copy $6.00; Microfiche $3.00

ORNL/CSD/TM-64
Distribution Category UC-32

Contract No. W-7405-eng-26

COMPUTER SCIENCES DIVISION

(Sponsor: D. A. Gardiner; Originator: R. E. Funderlic)

LECTURE NOTES ON ORDINARY DIFFERENTIAL EQUATIONS SOFTWARE, USER'S GUIDES FOR ODE, RKF45, GEAR AND EPISODE

J. A. Wenzel and R. E. Funderlic

Date Published: August 1979

UNION CARBIDE CORPORATION, NUCLEAR DIVISION
operating the
Oak Ridge Gaseous Diffusion Plant . Oak Ridge National Laboratory
Oak Ridge Y-12 Plant . Paducah Gaseous Diffusion Plant
for the
DEPARTMENT OF ENERGY

f29

TABLE OF CONTENTS

TABLE OF CONTENTS (continued)

LIST OF FIGURES

LIST OF FIGURES (continued)

ACKNOWLEDGMENTS

These user's guides were taken from lecture notes of the course "Software for Ordinary Differential Equations" offered under the auspices of the Technical Continuous Education Program in the winter term of 1978. The authors are grateful to P. Gaffney and M. T. Heath for suggestions and criticism. The first author was supported by stipends from the Oak Ridge Associated Universities, the Great Lakes Colleges Association/ Oak Ridge Science Program and Albion College while he worked in the Computing Applications Department of the Computer Sciences Division.

LECTURE NOTES ON ORDINARY DIFFERENTIAL EQUATIONS SOFTWARE, USER'S GUIDES FOR ODE, RKF45, GEAR AND EPISODE

J. A. Wenzel and R. E. Funderlic

ABSTRACT

Four software packages for the numerical solution of initial value problems for systems of ordinary differential equations are demonstrated with sample problems and solutions. These packages are contained in the Core Library of Numerical Software of the Computer Sciences Division.

I. INTRODUCTION

A. Definition of the Problem

Ordinary differential equations are encountered whenever the rate of a dependent variable, call it y, with respect to an independent variable, say t, is a function of t and y, that is, $dy/dt = f(t,y)$. A system of ordinary differential equations is a collection of differential equations the rate of change of each of the dependent variables $y_1, y_2, \dots, y_n$ depends on one or more of the dependent variables as well as on the independent variable:

$$dy_1/dt = f_1(t, y_1(t), y_2(t), \dots, y_n(t))$$

$$dy_2/dt = f_2(t, y_1(t), y_2(t), \dots, y_n(t))$$

$$\cdots$$

$$dy_n/dt = f_n(t, y_1(t), y_2(t), \dots, y_n(t)) \quad .$$

If we let $Y(t)$ be the vector $(y_1(t), y_2(t),\ldots,y_n(t))$, then the system may be written as $dY/dt = F(t,Y(t))$ where F is a vector-valued function. When the value of Y is desired at a point t and it is known that at $t = t_0$, $y(t_0) = y_0$, then the problem is an initial value problem.

A system is called stiff if the components of the solution are decaying at greatly differing rates. A classic example comes from Gear's text [7]:

solve
$$y_1' = 998y_1 + 1998y_2$$
$$y_2' = 999y_1 - 1999y_2$$

subject to the initial condition

$$y_1(0) = 1 \text{ and } y_2(0) = 0 .$$

The solution in closed form can be found to be

$$y_1(t) = 2e^{-t} - e^{-1000t}$$
$$y_2(t) = -e^{-t} + e^{-1000t}$$

and the graph is shown in Figure 1.

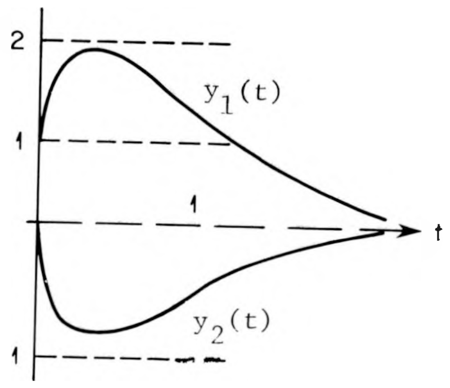


Fig. 1. A Stiff System

The lecture notes from which this report was taken originally contained much background material and we originally planned to include this. Since then an excellent course on numerical methods was given by M. T. Heath (CSD) from a book by Forsythe, Malcolm and Moler [6]. This book is available from many sources at the Oak Ridge facilities and we recommend reading Chapter 6 for a clear discussion of methods for solving ordinary differential equations.

B. The Four Packages

There are many techniques which can be used to find a numerical solution of an initial value problem. Four of the most efficient and accurate software packages for doing this are contained in the Core Library of Numerical Software which is maintained by the Computing Applications Department of the Computer Sciences Division. Each package is a set of FORTRAN subroutines which employ double-precision floating point variables. While the packages were designed for different types of problems and use different techniques, all are well-documented, easily accessible, as well as reliable and efficient.

The first software package that is demonstrated is ODE, which was written by L. F. Shampine and M. K. Gordon at Sandia Laboratories. It uses a modified divided-differences version of the Adams-Bashforth-Moulton formulas in a predictor-corrector mode. Adjusting the order of the formula and the step size as it moves across the interval, it will, in the interest of efficiency, go beyond the terminal value for t and then interpolate to find the desired value for $Y(t) = (y_1(t),\dots,y_n(t))$. The package is thoroughly examined in [11] and additional examples may be found in [8], [12] and [13]. This package is a good choice when high accuracy is required and the system is not stiff.

If the system is not stiff, high accuracy is not essential or appropriate, and it is fairly inexpensive to evaluate the function, then the RKF45 package developed by H. A. Watts and L. F. Shampine at Sandia Laboratories is called for. It uses a modification by Fehlberg, [4] and [5], of the classical Runge-Kutta formulas. The order of the method, 4 or 5, is determined before the integration advances a step towards the terminal value of t. The step size is also controlled. See [6] for a full discussion of the program.

A. C. Hindmarsh at Lawrence Livermore Laboratory wrote the computer code for GEAR based on the technique developed by C. W. Gear [7] for solving systems which are stiff but otherwise well-behaved. It uses the backward differentiation formulas for the predictor and the chord method with the Jacobian matrix of partial derivatives to calculate the corrector. See [9] for further details.

G. D. Byrne and A. C. Hindmarsh reworked the ideas in GEAR and produced EPISODE to treat stiff systems which are oscillatory or highly nonlinear. The algorithm is discussed in [1], the package is examined in [10] and some interesting comparisons between GEAR and EPISODE appear in [2] and [3].

A survey of several methods for solving non-stiff systems is found in [14]. The uses outlined above exploit the strengths of each of the packages although it is possible to use ODE on moderately stiff systems and to use GEAR on non-stiff systems. Of the four packages, ODE is the most versatile and perhaps the easiest to use. If you experience problems or have questions which aren't answered by these introductory guides, call a numerical consultant in the Computer Applications Department. Their names are given in the HELP file CORLIB. Listings of the codes and card decks are available from the Computer Librarian, ext. 4-5317, although normally these should not be needed since both the codes and documentation are available on the computers.

II. ODE

A. Description of the Subroutines

This software package consists of four FORTRAN double-precision subroutines, ODE, DE, STEP, and INTRP, which integrate dY/dt = F(t,Y) from t = T to TOUT. It is possible to advance TOUT and call ODE again as the variables are returned with all the information necessary to continue the integration.

The subroutine ODE allocates storage in two auxilliary arrays, WORK and IWORK, and calls the subroutine DE. DE controls the process by calling STEP repeatedly until TOUT has been reached or exceeded and checks for conditions which would cause termination: too many function evaluations, possible stiffness, or demands for excessive accuracy. If the integration procedes beyond TOUT, DE calls INTRP to interpolate for Y at TOUT. The subroutine STEP advances the integration one step at a time using an Adams-Bashforth formula to predict and an Adams-Moulton formula to correct. Based on tests it adjusts the order of the method and the step-size to control local error by using a divided difference method and local interpolation. Since the code chooses the step size to be as large as possible while still meeting the error tolerance specified by the user, the integration usually advances beyond TOUT. In which case, INTRP is called. It uses a polynomial whose degree is based on the order of the method last used and interpolates to find the value of Y at TOUT. This package was written by L. F. Shampine and M. K. Gordon at the Sandia Laboratories. The computer code was obtained from the National Energy Software Center at Argonne National Laboratory.

B. The Calling Program

The user needs to supply a calling program using double precision FORTRAN that

1) supplies the initial conditions,
2) sets the values of an input variable,
3) calls ODE, and optionally
4) writes out the results

and a double-precision subroutine F(X, Y, YP) that defines the system of first order ordinary differential equations. The call list for ODE consists of:

F	the name of the subroutine which defines the system of differential equations. (Note: F must also be declared in an EXTERNAL statement).
NEQN	the number of equations in the system, $1 \leq$ NEQN.
Y	the array that contains the values of Y at t = T when you call ODE and that contains the values at t = TOUT upon a successful completion of ODE.
T	the value of t at the beginning of the integration.
TOUT	the value of t at the end of the integration. This can be less than T but it can never equal T.
RELERR and ABSERR	the relative and absolute local error tolerance. At each internal step going from T to TOUT, the code tries to control each component of the local error so that $\lvert \text{local error}_L \rvert < \text{RELERR} * \lvert Y(L) \rvert + \text{ABSERR}$ for $1 \leq L \leq \text{NEQN}$. Neither value can be negative and at least one must be positive.
IFLAG	on the first call this is normally set equal to 1. Set IFLAG equal to -1, only if it is known that it is impossible to integrate beyond TOUT. On subsequent calls, IFLAG may be left as it was returned.
WORK and IWORK	two storage arrays controlled by ODE. The dimension of WORK is 100 + 21*NEQN and that of IWORK is 5.

C. System Dependent Procedures

The most efficient and convenient way to access ODE is to use the Core Library of Numerical Software. On the IBM 360 computers it resides in LOGLIB in load module form. This is automatically available to all FORTRAN jobs without additional JCL. The Core Library resides on the DEC-10 in the SYS area as a REL file library and is accessed when executing a FORTRAN program by typing

```
EX  MYPROG,MYSUB,SYS:CORLIB/SEA
```

If you have questions or problems, please consult a numerical consultant listed in the HELP file for the Core Library, or in case of a related system problem call Programming Assistance.

D. Values Returned by ODE

The subroutine ODE will return to your calling program these values

T	the value to which T has advanced. If there was no problem, then T will equal TOUT.
Y	the values of Y(1), Y(2),...,Y(NEQN) at T.
RELERR and ABSERR	the original values unless this degree of accuracy is untainable on the computer with this program. In that case, they will be increased and IFLAG will be set to 3. You can call ODE with these revised values and try again to integrate.

The value of IFLAG reports the success or failure of ODE.

Value	Interpretation
2	a normal return. The integration reached TOUT, T has been set to TOUT, and Y contains the solution. If you

change TOUT, you may call ODE again and continue to integrate.

3 the integration did not reach TOUT because the error tolerances were too small for the computer being used. T is set to the point closest to TOUT that has been reached and Y to the solution at that point. RELERR and ABSERR have been increased so that ODE may be called again and another attempt made to integrate to TOUT.

4 the integration did not reach TOUT because more than MAXNUM(= 500) steps were needed. T is set to the point closest to TOUT that was reached and Y to the answers at that point. If you want to continue, call ODE again.

5 more than MAXNUM(= 500) steps were required to reach TOUT and the equations appear to be stiff. T is set to the point closest to TOUT that was reached and Y to the answers at that point. You probably should switch to another software package (*e.g.*, GEAR. See Chapter III.), but you often can get accurate results with ODE if you are willing to pay the costs (more function evaluations which mean more time).

6 the integration did not begin because at least one of the input parameters is not valid. See Section B above.

If IFLAG had been set equal to -1, and if the integration did not reach TOUT, then IFLAG will return as -3, -4, or -5.

E. Examples

There are three examples (problem, sample program, output) in this section. The first solves a second order differential equation and the calling program is copiously commented. The second example demonstrates the ease with which ODE can handle a jump discontinuity in the first derivative. The third uses ODE to create a table although it could just as easily be combined with a plotting routine to create a graph.

1. ODE-1 A second order differential equation.

Solve $y'' + y = 0$

subject to $y(0) = 0,\ y'(0) = 1$.

We transform this second order differential equation to a system of two first order equations by introducing the variables $y_1 = y$ and $y_2 = y' = y_1'$. The equivalent problem is

Solve
$$y_1' = y_2$$
$$y_2' = -y_1\ ,$$
subject to $y_1(0) = 0,\ y_2(0) = 1$.

Analytically, the solution may be found in closed form to be

$$y_1(t) = \sin t \text{ and } y_2(t) = \cos t\ .$$

The double precision calling program which sets the initial conditions and calls ODE is shown in Figure 2. [Note: The same problem appears as the example RKF45-2 in Chapter III].

ORNL-DWG 78-10767

```
C
C     MAKE ALL NON-INTEGER VARIABLES DOUBLE-PRECISION VARIABLES
C
        IMPLICIT REAL*8(A-H,O-Z)
C
C     SET THE DIMENSION FOR Y TO BE THE NUMBER OF EQUATIONS IN THE SYSTEM
C     SET THE DIMENSION FOR WORK TO BE  100 + 21*NEQN
C     SET THE DIMENSION FOR IWORK TO BE 5
C
        DIMENSION Y(2), WORK(142), IWORK(5)
C
C     DECLARE  F  TO BE AN EXTERNAL SUBROUTINE
C
        EXTERNAL F
C
C     SET  NEQN  AND THE INITIAL CONDITIONS
C
        NEQN = 2
        Y(1) = 0.0D0
        Y(2) = 1.0D0
C
C     SET THE INITIAL AND TERMINAL VALUES OF THE INDEPENDENT VARIABLE  T
C
        T = 0.0D0
        TOUT = 3.14159265350D0
C
C     SET THE ERROR BOUNDS
C
        RELERR = 1.0D-8
        ABSERR = 1.0D-8
C
C     SET  IFLAG = 1  FOR THE FIRST CALL TO  ODE
C
        IFLAG = 1
```

Fig. 2. Calling Program for Example ODE-1

ORNL-DWG 78-10768

```
C
C   WRITE THE VALUES USED TO CALL  ODE
C
      WRITE(6,100) NEQN, Y(1), Y(2), T, TOUT, RELERR, ABSERR, IFLAG
C
C   CALL ODE
C
      CALL ODE(F, NEQN, Y, T, TOUT, RELERR, ABSERR, IFLAG, WORK, IWORK)
C
C   CHECK  IFLAG
C
      IF (IFLAG .NE. 6) GO TO 10
      WRITE(6,101)
      STOP
   10 IF (IFLAG .EQ. 2) GO TO 20
      WRITE(6,102) IFLAG, T, Y(1), Y(2), RELERR, ABSERR
      STOP
C
C   WRITE OUT THE RESULTS
C
   20 WRITE(6,103) T, Y(1), Y(2)
C
C   FORMATS
C
  100 FORMAT(//,5X,38HTHESE ARE THE VALUES USED TO CALL ODE:,//,5X,
     1 8HNEQN    =,I6,/,5X,8HY(1)    =,D20.10,/,5X,8HY(2)    =,D20.10,/,5X
     2  8HT       =,D20.10,/,5X,8HTOUT    =,D20.10,/,5X,8HRELERR =,D20.10,
     3  /,5X,8HABSERR =,D20.10,/,5X,8HIFLAG  =,I6,//)
  101 FORMAT(5X,67HIFLAG  = 6.  AT LEAST ONE OF THE INPUT VALUES IS WRON
     1G.  TRY AGAIN.)
  102 FORMAT(5X,8HIFLAG  =,I3,27H.  TAKE APPROPRIATE ACTION.,//,5X,
     1  8HY(1)    =,D20.10,/,5X,8HY(2)    =,D20.10,/,5X,8HRELERR   D20.10,
     2  /,5X,8HABSERR =,D20.10)
  103 FORMAT(5X,23HTHIS IS A NORMAL RETURN,/,5X,8HT       =,D20.10,/,5X,
     1  8HY(1)    =,D20.10,/,5X,8HY(2)    =,D20.10)
      STOP
      END
```

Fig. 2. (Cont'd)

The subroutine F(T,Y,YP) is shown in Fig. 3.

ORNL-DWG 78-10769

```
      SUBROUTINE F(T, Y, YP)
C
C    MAKE ALL THE NON-INTEGER VARIABLES DOUBLE-PRECISION  ARIABLES
C
      IMPLICIT REAL*8(A-H,O-Z)
C
C    SET THE DIMENSION FOR  Y  AND  YP  TO BE THE NUMBER OF EQUATIONS
C
C    IN THE SYSTEM
      DIMENSION Y(2), YP(2)
C
C    DEFINE THE SYSTEM OF EQUATIONS
C
      YP(1) = Y(2)
      YP(2) = -Y(1)
      RETURN
      END
```

Fig. 3. Subroutine F for Example ODE-1

The results are shown in Figure 4.

ORNL-DWG 78-10770

```
     THESE ARE THE VALUES USED TO CALL ODE:

     NEQN    =      2
     Y(1)    =     0.0
     Y(2)    =     0.1000000000D 01
     T       =     0.0
     TOUT    =     0.3141592654D 01
     RELERR  =     0.1000000000D-07
     ABSERR  =     0.1000000000D-07
     IFLAG   =      1


     THIS IS A NORMAL RETURN
     T       =     0.3141592654D 01
     Y(1)    =     0.1369487599D-07
     Y(2)    =    -0.1000000013D 01

 IHC002I STOP       0
```

Fig. 4. Output from the Program for Example ODE-1

2. ODE-2 A jump discontinuity in the first derivative:

$$\text{Solve} \qquad y' = \begin{cases} y & \text{if } 0 \le X \le 1 \\ -y & \text{if } 1 \le X \le 2 \end{cases}$$

$$\text{subject to} \qquad y(0) = 1 \quad .$$

Analytically the closed form solution is:

$$y = \begin{cases} e^{X} & \text{if } 0 \le X \le 1 \\ e^{2-X} & \text{if } 1 \le X \le 2 \end{cases} \quad .$$

The calling program, without comments, and the subroutine F are shown in Fig. 5. The results are shown in Fig. 6.

ORNL-DWG 78-10772

```
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION Y(1), WORK(121), IWORK(5)
      EXTERNAL F
      NEQN = 1
      Y(1) = 1.0D0
      T = 0.0D0
      TOUT = 2.0D0
      RELERR = 0.0D0
      ABSERR = 1.0D-10
      IFLAG = 1
      WRITE(6,100) NEQN, Y(1), T, TOUT, RELERR, ABSERR, IFLAG
      CALL ODE(F, NEQN, Y, T, TOUT, RELERR, ABSERR, IFLAG, WORK, IWORK)
      WRITE(6,101) IFLAG, T, Y(1)
100   FORMAT(//,5X,38HTHESE ARE THE VALUES USED TO CALL ODE:,//,5X,
     1 8HNEQN   =,I6,/,5X,8HY(1)   =,D20.10,/,5X,8HT      =,D20.10,/,5X
     2  8HTOUT   =,D20.10,/,5X,8HRELERR =,D20.10,/,5X,8HABSERR =,D20.10,
     3  /,5X,8HIFLAG  =,I6,//)
101   FORMAT(/,5X,8HIFLAG  =,I6,/,5X,8HT      =,D20.10,/,5X,8HY(1)   =,
     1  D20.10)
      STOP
      END


      SUBROUTINE F(T, Y, YP)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION Y(1), YP(1)
      YP(1) = Y(1)
      IF (T .LE. 1.0D0) RETURN
      YP(1) = - Y(1)
      RETURN
      END
```

Fig. 5. Calling Program and Subroutine F for Example ODE-2

ORNL-DWG 78-10773

```
            THESE ARE THE VALUES USED TO CALL ODE:

            NEQN    =      1
            Y(1)    =     0.1000000000D 01
            T       =     0.0
            TOUT    =     0.2000000000D 01
            RELERR  =     0.0
            ABSERR  =     0.1000000000D-09
            IFLAG   =      1


            IFLAG   =      2
            T       =     0.2000000000D 01
            Y(1)    =     0.1000000000D 01

        IHC002I STOP        0
```

Fig. 6. Output from the Program for Example ODE-2

3. ODE-3 Creating a table.

Solve $y' = -y$

subject to $y(0) = 1$

The closed form solution is $y = e^{-t}$. This time a table will be printed for t = 0, 0.1, 0.2,..., 1.0. The calling program and subroutine F are shown in Fig. 7, and the table is shown in Figure 8. [Note: The same problem appears as the example GEAR-3 in Chapter IV.]

ORNL-DWG 78-10774

```
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION Y(1), WORK(121), IWORK(5)
      EXTERNAL F
      NEQN = 1
      Y(1) = 1.0D0
      T = 0.0D0
      TOUT = 0.1D0
      RELERR = 1.0D-10
      ABSERR = 1.0D-10
      IFLAG = 1
      WRITE(6,100) T, Y(1)
      DO  10  I = 1, 10
        CALL ODE(F,NEQN,Y,T,TOUT,RELERR,ABSERR,IFLAG,WORK,IWORK)
        WRITE(6,101) T, Y(1)
        TOUT = TOUT + 0.1D0
   10 CONTINUE
  100 FORMAT(5X,34HA TABLE OF VALUES FOR YP=-Y,Y(0)=1,//,10X,1HT,24X,
     1  1HY,/,5X,D20.10,5X,D20.10)
  101 FORMAT(5X,D20.10,5X,D20.10)
      STOP
      END


      SUBROUTINE F(T, Y, YP)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION Y(1), YP(1)
      YP(1) = -Y(1)
      RETURN
      END
```

Fig. 7. Calling Program and Subroutine F for Example ODE-3

ORNL-DWG 78-10775

```
     A TABLE OF VALUES FOR YP=-Y,Y(0)=1

          T                        Y
     0.0                      0.1000000000D 01
     0.1000000000D 00         0.9048374180D 00
     0.2000000000D 00         0.8187307531D 00
     0.3000000000D 00         0.7408182207D 00
     0.4000000000D 00         0.6703200460D 00
     0.5000000000D 00         0.6065306597D 00
     0.6000000000D 00         0.5488116361D 00
     0.7000000000D 00         0.4965853038D 00
     0.8000000000D 00         0.4493289641D 00
     0.9000000000D 00         0.4065696597D 00
     0.1000000000D 01         0.3678794411D 00

IHC002I STOP       0
```

Fig. 8 Output from the Program for Example ODE-3

F. ODERT

A frequently encountered problem is to locate the extreme values for one, say the ith, of the components of a solution to the system of differential equations,

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

subject to the initial conditions

$$y_1(a) = c_1, y_2(a) = c_2, \ldots, y_n(a) = c_n .$$

This can be done by first locating where the derivative, y_i', of the ith component, y_i, vanishes and then seeing if there is a local maximum or a local minimum (or possibly neither) at this point. The suite of codes ODERT/STEP, INTRP, DERT, ROOT, or ODERT for short, is a modification of ODE which integrates a system of first-order ordinary differential equations from T in the direction of TOUT until it locates the first root of some specified (nonlinear) equation,

$$G(t) = g(t,y_1(t),y_2(t),\ldots,y_n(t),y_1'(t),y_2'(t),\ldots,y_n'(t)) = 0 .$$

Upon finding a root, the code returns with all the parameters in the call list set for continuing the integration to the next root of G or to the first root of a new function G. If no roots are found, the integration proceeds to TOUT.

The routine ODERT is a supervisor that calls DERT, an adaptation of DE, which in turn calls STEP and INTRP. After each internal step, ODERT evaluates the function G and checks for a change in sign in the functional values from the previous step. If the sign has changed in going from B to C, then a zero is bracketed and ODERT calls ROOT which uses a combination of the secant method and the bisection method to determine the root of the desired accuracy. Two new parameters, REROOT and AEROOT, are added to the call list along with G. They set the relative and absolute error tolerances for computing the root of

The stopping criterion is

$$|G(B)-G(C)| \leq 2*(REROOT*|B|+ AEROOT) .$$

Example: Solve $y' = 2y/t + 5$,

subject to $y(1) = -4$,

finding all the extreme values of y over the interval from 1 to 7.

We need to supply the calling program and two subroutines, F and G, Here, F determines the system of equations and G sets the derivative equal to zero (Fig. 9).

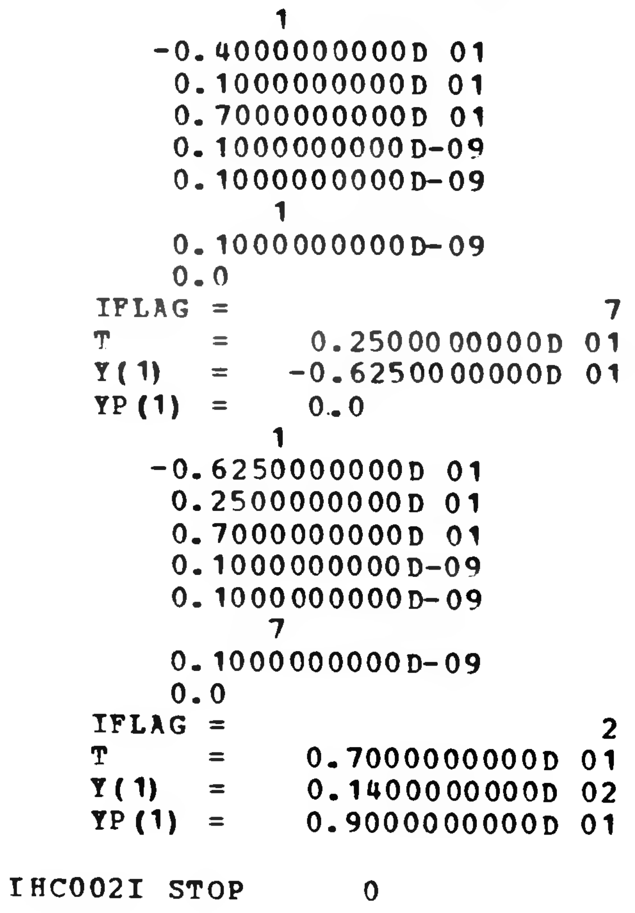
ORNL-DWG 78-10776

```
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION Y(1), YP(1), WORK(121), IWORK(5)
C
C   NOW WE MUST INCLUDE  G  IN THE EXTERNAL STATEMENT
C
      EXTERNAL  F, G
      NEQN = 1
      Y(1)  = -4.0D0
      T = 1.0D0
      TOUT = 7.0D0
      RELERR = 1.0D-10
      ABSERR = 1.0D-10
      IFLAG = 1
C
C   A PURE RELATIVE ERROR IS REASONABLE HERE
C
      REROOT = 1.0D-10
      AEROOT = 0.0D0
   10 WRITE(6,101)NEQN,Y(1),T,TOUT,RELERR,ABSERR,IFLAG,REROOT,AEROOT
      CALL ODERT(F,NEQN,Y,T,TOUT,RELERR,ABSERR,IFLAG,WORK,IWORK,G,
     1 REROOT,AEROOT)
      CALL F(T, Y, YP)
      WRITE(6,100) IFLAG, T, Y(1), YP(1)
      IF (IFLAG .EQ. 7) GO TO 10
  100 FORMAT(5X,7HIFLAG =,I20,/,5X,7HT      =,D20.10,/,5X,7HY(1)   =
     1  D20.10,/,5X,7HYP(1) =,D20.10)
  101 FORMAT(5X,I10,5(/,D25.10),/,I15,2(/,D25.10))
      STOP
      END


      SUBROUTINE F(T, Y, YP)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION Y(1), YP(1)
      YP(1)  = 2.0D0*Y(1)/T + 5.0D0
      RETURN
      END


      DOUBLE PRECISION FUNCTION G(T, Y, YP)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION Y(1), YP(1)
      G = YP(1)
      RETURN
      END
```

Fig. 9.   Calling Program and Subroutines F and G for ODERT.

The program yields(Fig. 10).

ORNL-DWG 78-10777



```
                1
      -0.4000000000D 01
       0.1000000000D 01
       0.7000000000D 01
       0.1000000000D-09
       0.1000000000D-09
                1
       0.1000000000D-09
       0.0
   IFLAG =                   7
   T     =     0.2500000000D 01
   Y(1)  =    -0.6250000000D 01
   YP(1) =     0.0
                1
      -0.6250000000D 01
       0.2500000000D 01
       0.7000000000D 01
       0.1000000000D-09
       0.1000000000D-09
                7
       0.1000000000D-09
       0.0
   IFLAG =                   2
   T     =     0.7000000000D 01
   Y(1)  =     0.1400000000D 02
   YP(1) =     0.9000000000D 01

IHC002I STOP        0
```

Fig. 10.   Output from the Program for ODERT.

In addition to the standard values for IFLAG (see Section D), we have:

Value	Interpretation
7	normal return, a root satisfying the criterion has been found.
8	abnormal return, an odd order pole of G was found.
9	abnormal return, over 500 evaluations of G were required to locate the root.

Although ODERT is not in the Core Library it is available from the Computer Librarian, ext. 4-5317.

III. RKF45

A. Description of the Subroutines

This software package consists of three FORTRAN double-precision subroutines, RKF45, RKFS and FEHL, which integrate dY/dt = F(t,Y) from t = T to TOUT. The user can then assign a new value to TOUT and call RKF45 again as on return the parameters in the call list are set for continuing the integration. It is also possible to use RKF45 as a one-step integrator to advance the solution a single step in the direction of TOUT.

The subroutine RKF45 serves as an interface between the users' calling program and the subroutine RKFS. This subroutine checks the input parameters for errors, determines the accuracy of the particular computer being used and determines if the next step can be taken. If it can, it sets the step size and advances the approximate solution by one step towards TOUT by calling FEHL where the integration actually is performed. It checks the error criteria and if the step was successful, continues to advance towards TOUT. Since the step size is changed, it maybe that the step just taken advances the solution almost to TOUT and that another step can not be made without exceeding the computer's accuracy. In that case, the solution at TOUT is determined by extrapolation.

The classical fourth-order Runge-Kutta method is given by

$$y_{n+1} = y_n + (k_0 + 2k_1 + 2k_2 + k_3)/6$$

where

$$k_0 = hf(t_n,y_n)$$

$$k_1 = hf(t_n + h/2,y_n + k_0/2)$$

$$k_2 = hf(t_n + h/2,y_n + k_1/2)$$

$$k_3 = hf(t_n + h,y_n + k_2) \ .$$

Fehlberg discovered a set of values for the coefficients α_i, β_{ij}, γ_i for

$$k_i = hf(t_n + \alpha_i h_n,y_n + \sum_{j=1}^{i-1} \beta_{ij}k_j) \ , \ i = 1,\dots,6$$

$$y_{n+1} = y_n + \sum_{i=1}^{6} \gamma_i k_i$$

which result in a fifth-order method for a predictor and a fourth-order method for a corrector. The difference between the predictor and the corrector is used to compute the local error estimate in controlling the step size. This package was written H. A. Watts and L. F. Shampine [6].

B. The Calling Program

The user needs to write a calling program that

1) supplies the initial conditions,

2) sets the values of the imput variables,

3) invokes RFK45, and optionally

4) writes out the results,

as well as a subroutine F(T,Y,YP) that defines the system of first order ordinary differential equations. The input variables for RFK45 are

F	the name of the subroutine which defines the system of differential equations. (Note: F must be declared in an EXTERNAL statement.)
NEQN	the number of equations in the system $1 \le NEQN$.
Y	the array that contains the initial values of Y(1), Y(2), ..., Y(NEQN) at t = T when you call RKF45 and that contains the values of Y(1),Y(2), ..., Y(NEQN) at t = TOUT upon a successful completion of RFK45.
T	the starting point of integration.
TOUT	the output point at which the solution is desired. This value can be less than T.
RELERR and ABSERR	the relative and absolute error tolerances. At each internal step going from T to TOUT the code requires that $$\mid \text{local error} \mid \le \text{RELERR}^* \mid Y \mid + \text{ABSERR}$$ for each component of Y. Neither value can be negative, at least one must be positive, and ABSERR must be positive if the solution vanishes.
IFLAG	set this equal to 1 the first time RKF45 is called, unless one-step integrator control is necessary. Then use -1. On the return from RKF45, it will have the value 2 if the integration was completed successfully.
WORK and IWORK	two arrays to hold information internal to RKF45 which is necessary for subsequent calls. The dimension of WORK is 3 + 6*NEQN and that of IWORK is 5.

C. System Dependent Procedures

The most efficient and convenient way to access RKF45 is to use the Core Library of Numerical Software. On the IBM 360 computer it resides in LOGLIB in load module form. This is automatically available to all FORTRAN jobs without additional JCL. The Core Library resides on the DEC-10 in the SYS area as a REL file library and is accessed when executing a FORTRAN program by typing

```
.EX MYPROG,MYSUB,SYS: CORLIB/SEA
```

If you have questions or problems, please consult a numerical consultant or Programming Assistance.

D. Values Returned by RKF45

The subroutine RKF45 will return to your calling program these values:

T	the value to which T advanced. If there was no trouble, then T will equal TOUT.
Y	the values of Y(1),Y(2),..., Y(NEQN) at the current value of T.
RELERR and ABSERR	the same values as when the program began unless RELERR is too small. In this case, IFLAG will equal 3 and RELERR will be increased so that you can call RKF45 again and continue from the current value for T.

The value of IFLAG reports the success or failure of RKF45.

Value	Interpretation
= 2	a normal return. The integration reached TOUT. The user may reset TOUT and call RKF45 again to continue integration.
=-2	a single step has been successfully taken in the direction

of TOUT. The users may reset TOUT and call RFK45 again to continue integration.

=3 the integration did not reach TOUT because the relative error tolerance was too small. RELERR has been increased appropriately for continuing.

=4 the integration was not completed because more than 3000 derivative evaluations were needed. This is approximately 500 steps. The user may continue to integrate from the current value of T by calling RKF45 again.

=5 the integration was not completed because the solution vanished which made a pure relative error test impossible. Make ABSERR nonzero if you wish to continue.

=6 the integration was not completed because the requested accuracy could not be achieved using the smallest allowable stepsize. You must increase the error tolerance before continued integration can be attempted and reset IFLAG to 2 or -2.

=7 it is likely that RKF45 is inefficient for solving this problem. Too much output is restricting the natural stepsize choice. Switch to the one-step mode or try ODE.

=8 at least one of the input parameters is incorrect. One or more of the following errors has occurred:

NEQN is less than 1.

T = TOUT and IFLAG is not equal to 1.

RELERR or ABSERR is less than 0.

IFLAG is equal to 0, is less than -1, or is greater than 8.

E. Examples

1. RKF45-1 A second order differential equation.

Solve $y'' + y = 0$

subject to $y(0) = 0, y'(0) = 1$

We transform this second order differential equation to a system of two first order equations by introducing the variables $y_1 = y$ and $y_2 = y' = y_1'$. The equivalent problem is:

Solve
$$y_1' = y_2$$
$$y_2' = -y_1$$

subject to $y_1(0) = 0, y_2(0) = 1$.

Analytically, the solution may be found in closed form to be

$$y_1(t) = \sin t \quad \text{and} \quad y_2(t) = \cos t \ .$$

The main program which sets the initial conditions and calls RKF45 is shown in Fig. 11 and the subroutine F(T,Y,YP) is shown in Fig. 12.

ORNL-DWG 78-10791

```
C
C    SET THE DIMENSION FOR Y TO BE THE NUMBER OF EQUATIONS IN THE SYSTEM
C    SET THE DIMENSION FOR WORK TO BE  3 + 6*NEQN
C    SET THE DIMENSION FOR IWORK TO BE 5
C
      DIMENSION Y(2), WORK(15), IWORK(5)
C
C    DECLARE  F  TO BE AN EXTERNAL SUBROUTINE
C
      EXTERNAL F
C
C    SET  NEQN  AND THE INITIAL CONDITIONS
C
      NEQN = 2
      Y(1) = 0.0
      Y(2) = 1.0
C
C    SET THE INITIAL AND TERMINAL VALUES OF THE INDEPENDENT VARIABLE  T
C
      T = 0.0
      TOUT = 3.1415926535
C
C    SET THE ERROR BOUNDS
C
      RELERR = 1.0E-9
      ABSERR = 1.0E-9
C
C    SET  IFLAG = 1  FOR THE FIRST CALL TO RKF45
C
      IFLAG = 1
C
C    WRITE THE VALUES USED TO CALL  RKF45
C
      WRITE(6,100) NEQN, Y(1), Y(2), T, TOUT, RELERR, ABSERR, IFLAG
C
C    CALL  RKF45
C
      CALL RKF45(F,NEQN,Y,T,TOUT,RELERR,ABSERR,IFLAG,WORK,IWORK)
C
C    CHECK IFLAG
C
      IF (IFLAG .NE. 8) GO TO 10
      WRITE(6,101)
      STOP
  10  IF (IFLAG .EQ. 2) GO TO 20
      WRITE(6,102) IFLAG, T, Y(1), Y(2), RELERR, ABSERR
      STOP
C
C    WRITE OUT THE RESULTS
C
  20  WRITE(6,103) T, Y(1), Y(2)
C
C    FORMATS
C
 100  FORMAT(//,5X,40HTHESE ARE THE VALUES USED TO CALL RKF45:,//,5X,
     1 8HNEQN    =,I6,/,5X,8HY(1)    =,E20.10,/,5X,8HY(2)    =,E20.10,/,5X
     2  8HT       =,E20.10,/,5X,8HTOUT    =,E20.10,/,5X,8HRELERR =,E20.10,
     3  /,5X,8HABSERR =,E20.10,/,5X,8HIFLAG  =,I6,//)
 101  FORMAT(5X,67HIFLAG  = 8.  AT LEAST ONE OF THE INPUT VALUES IS WRON
     1G.  TRY AGAIN.)
 102  FORMAT(5X,8HIFLAG  =,I3,27H.  TAKE APPROPRIATE ACTION.,//,5X,
     1  8HY(1)    =,E20.10,/,5X,8HY(2)    =,E20.10,/,5X,8HRELERR   E20.10,
     2  /,5X,8HABSERR =,E20.10)
 103  FORMAT(5X,23HTHIS IS A NORMAL RETURN,/,5X,8HT       =,E20.10,/,5X,
     1  8HY(1)    =,E20.10,/,5X,8HY(2)    =,E20.10)
      STOP
      END
```

Fig. 11 Calling Program for Example RKF45-1

ORNL-DWG 78-10792

```
      SUBROUTINE F(T, Y, YP)
C
C    SET THE DIMENSION FOR  Y  AND  YP  TO BE THE NUMBER OF EQUATIONS
C    IN THE SYSTEM
C
      DIMENSION Y(2), YP(2)
C
C    DEFINE THE SYSTEM OF EQUATIONS
C
      YP(1) = Y(2)
      YP(2) = -Y(1)
      RETURN
      END
```

Fig. 12 Subroutine F for Example RKF45-1

When your job runs, you will get the results shown in Figure 13

ORNL-DWG 78-10793

```
     THESE ARE THE VALUES USED TO CALL RKF45:

     NEQN    =      2
     Y(1)    =     0.0
     Y(2)    =     0.1000000000E 01
     T       =     0.0
     TOUT    =     0.3141592026E 01
     RELERR  =     0.9999998607E-09
     ABSERR  =     0.9999998607E-09
     IFLAG   =      1


     THIS IS A NORMAL RETURN
     T       =     0.3141592026E 01
     Y(1)    =    -0.1518428326E-04
     Y(2)    =    -0.9999980330E 00

 IHC002I STOP       0
```

Fig. 13 Output from the Program for Example RKF45-1

[Note: This problem appears in Example ODE-1 in Chapter II.]

2. RKF45-2 The motion of two bodies under mutual gravitational attraction.

The following example comes from [6, pp. 122-133].

"Let y(t) denote the position of one body in a coordinate system with the origin fixed in the other body. The differential equations derived from Newton's laws of motion are

$$x''(t) = \frac{-a^2 x(t)}{R(t)}$$

$$y''(t) = \frac{-a^2 y(t)}{R(t)}$$

where

$$R(t) = [x(t)^2 + y(t)^2]^{3/2}$$

and α is a constant involving the gravitational constant, the masses of the two bodies, and the units of measurement.

If the initial conditions are chosen as

$$x(0) = 1 - e, \qquad x'(0) = 0,$$

$$y(0) = 0, \qquad y'(0) = a\left(\frac{1+e}{1-e}\right)^{1/2},$$

for some parameter e with $0 \leq e < 1$, the solution turns out to be periodic with period $2\Pi/a$. The orbit is an ellipse with eccentricity e and with one focus at the origin.

To write this as a system of four first-order equations, we introduce

$$y_1 \equiv x \quad y_2 = y, \quad y_3 = x', \quad y_4 = y'.$$

The equations and initial conditions then become

$$R = \frac{(y_1^2 + y_2^2)^{3/2}}{\alpha^2}$$

$$y_1' = y_3, \qquad y_1(0) = 1 - e,$$

$$y_2' = y_4, \qquad y_2(0) = 0,$$

$$y_3' = -\frac{y_1}{R}, \qquad y_3(0) = 0,$$

$$y_4' = -\frac{y_2}{R}, \qquad y_4(0) = \alpha\left(\frac{1 + e}{1 - e}\right)^{1/2}$$

By rescaling the time variable, it is possible to eliminate α, but we have not done this because we wish to illustrate the use of FORTRAN Common to pass parameters such as α from the main program to the subroutine defining the equations.

The parameter IFLAG is an important control variable. It should be set to 1 for the first entry to RKF45. Ordinarily, RKF45 will reset it to 2, and it should be left at 2 for subsequent entries. Values other than 2 returned by RKF45 signal various warning and error conditions described in detail in the comments. IFLAG = 4 and IFLAG = 7 are warnings that RKF45 must work very hard to obtain the requested accuracy. It is possible to continue, but the user may want to consider increasing the error tolerances or changing to a subroutine which uses a multistep method. IFLAG = 3 indicates that too much relative accuracy is being requested, and IFLAG = 5 or 6 indicates that the error tolerances must be changed before continuing. IFLAG = 8 indicates that RKF45 is being called incorrectly. The user is strongly advised to include a check on IFLAG in his main program.

In this sample run, we have taken $e = 0.25$ and $\alpha = \Pi/4$ and have printed the position for $0 \leq t \leq 12$ in steps of 0.5. The output is in Table 6.2. Notice that the orbit is periodic with a period of $t = 8$."

Table 6.2 OUTPUT FROM SAMPLE PROGRAM, [6]

```
 0.0      0.750000000     0.000000000
 0.5      0.619769032     0.477791373
 1.0      0.294417538     0.812178519
 1.5     -0.105176382     0.958038092
 2.0     -0.490299793     0.939874996
 2.5     -0.813942832     0.799590802
 3.0     -1.054031517     0.575706078
 3.5     -1.200735042     0.300160708
 4.0     -1.250000001    -0.000000001
 4.5     -1.200735042    -0.300160709
 5.0     -1.054031517    -0.575706079
 5.5     -0.813942932    -0.799590803
 6.0     -0.490299793    -0.939874996
 6.5     -0.105176383    -0.958038092
 7.0      0.294417537    -0.812178518
 7.5      0.619768031    -0.477791370
 8.0      0.749999996     0.000000006
 8.5      0.619768024     0.477791379
 9.0      0.294417526     0.812178522
 9.5     -0.105176395     0.958038091
10.0     -0.490299806     0.939874991
10.5     -0.813942843     0.799590794
11.0     -1.054031524     0.575706068
11.5     -1.200735047     0.300160697
12.0     -1.250000002    -0.000000011
```

The main program is shown in Fig. 14.

ORNL-DWG 78-10794

```
      EXTERNAL ORBIT
      REAL T, Y(4), TOUT, RELERR, ABSERR
      REAL TFINAL, TPRINT, ECC, ALFA, ALFASQ, WORK(27)
      INTEGER IWORK(5), IFLAG, NEQN
      COMMON ALFASQ
      ECC = 0.25
      ALFA = 3.141592653589/4.0
      ALFASQ = ALFA*ALFA
      NEQN = 4
      T = 0.0
      Y(1) = 1.0 - ECC
      Y(2) = 0.0
      Y(3) = 0.0
      Y(4) = ALFA*SQRT((1.0 + ECC)/(1.0 - ECC))
      RELERR = 1.0E-9
      ABSERR = 0.0
      TFINAL = 12.0
      TPRINT = 0.5
      IFLAG = 1
      TOUT = T
   10 CALL RKF45(ORBIT,NEQN,Y,T,TOUT,RELERR,ABSERR,IFLAG,WORK,IWORK)
      WRITE(6,11) T, Y(1), Y(2)
      GO TO (80, 20, 30, 40, 50, 60, 70, 80), IFLAG
   20 TOUT = T + TPRINT
      IF (T .LT. TFINAL) GO TO 10
      WRITE(6,25)
   25 FORMAT(///,5X,39HTHIS IS A NORMAL COMPLETION OF  RKF45 .,///)
      STOP
   30 WRITE(6,31) RELERR, ABSERR
      GO TO 10
   40 WRITE(6,41)
      GO TO 10
   50 ABSERR = 1.0D-9
      WRITE(6,31) RELERR, ABSERR
      GO TO 10
   60 RELERR = 10.0*RELERR
      WRITE(6,31) RELERR, ABSERR
      IFLAG = 2
      GO TO 10
   70 WRITE(6,71)
      IFLAG = 2
      GO TO 10
   80 WRITE(6,81)
      STOP
C
   11 FORMAT(F5.1,2F15.9)
   31 FORMAT(17H TOLERANCES RESET, 2E12.3)
   41 FORMAT(11H MANY STEPS)
   71 FORMAT(12H MUCH OUTPUT)
   81 FORMAT(14H IMPROPER CALL)
      END
```

Fig. 14. Calling Program for Example RKF45-2

The subroutine ORBIT is shown in Fig. 15.

```
SUBROUTINE ORBIT(T, Y, YP)
REAL T, Y(4), YP(4), R, ALFASQ
COMMON ALFASQ
R = Y(1)*Y(1) + Y(2)*Y(2)
R = R*SQRT(R)/ALFASQ
YP(1) = Y(3)
YP(2) = Y(4)
YP(3) = -Y(1)/R
YP(4) = -Y(2)/R
RETURN
END
```

Fig. 15. Subroutine ORBIT for Example RKF45-2

The results are shown in Fig. 16.

The following statement by Watts and Shampine occurs in the listing of RKF45:

> "RKF45 is primarily designed to solve non-stiff, and mildly stiff differential equations when derivative evaluations are inexpensive. RKF45 should generally not be used when the user is demanding high accuracy."

```
 0.0     0.750000000    0.0
 0.0     0.750000000    0.0
TOLERANCES RESET   0.100E-08   0.100E-08
 0.5     0.619767427    0.477788806
 1.0     0.294415772    0.812172532
 1.5    -0.105182588    0.958026409
 2.0    -0.490307987    0.939852118
 2.5    -0.813949347    0.799551964
 3.0    -1.054026604    0.575650275
 3.5    -1.200708389    0.300088167
 4.0    -1.249939919   -0.000086818
 4.5    -1.200632095   -0.300254643
 5.0    -1.053875923   -0.575793087
 5.5    -0.813728869   -0.799649894
 6.0    -0.490025043   -0.939874530
 6.5    -0.104854643   -0.957933247
 7.0     0.294736445   -0.811916292
 7.5     0.619973838   -0.477347434
 8.0     0.749956071    0.000545219
 8.2     0.725498140    0.212828636
MANY STEPS
 8.5     0.619443297    0.478255749
 9.0     0.293920815    0.812429488
 9.5    -0.105715394    0.958052158
10.0    -0.490790486    0.939681590
10.5    -0.814329267    0.799231887
11.0    -1.054275513    0.575223446
11.5    -1.200806618    0.299594939
12.0    -1.249875069   -0.000607297


    THIS IS A NORMAL CCMPLETION OF  RKF45
```

Fig. 16, Output from the Program for Example RKF45-2

IV. GEAR

A. Description of the Subroutines

This software package consists of seven FORTRAN double-precision subroutines, GEAR, INTERP, STIFF, COSET, PSET, DEL, and SOL, which integrate dY/dt = F(t,Y) from t = T to TOUT. The user can repeatedly reset TOUT and integrate again or the user can specify that control is to be returned after one step in the direction of TOUT. While there are many options, the chief virtue of GEAR is that it works well in solving stiff systems by using the "stiffly stable" technique of Gear [7], a modification of the backward differentiation formulas.

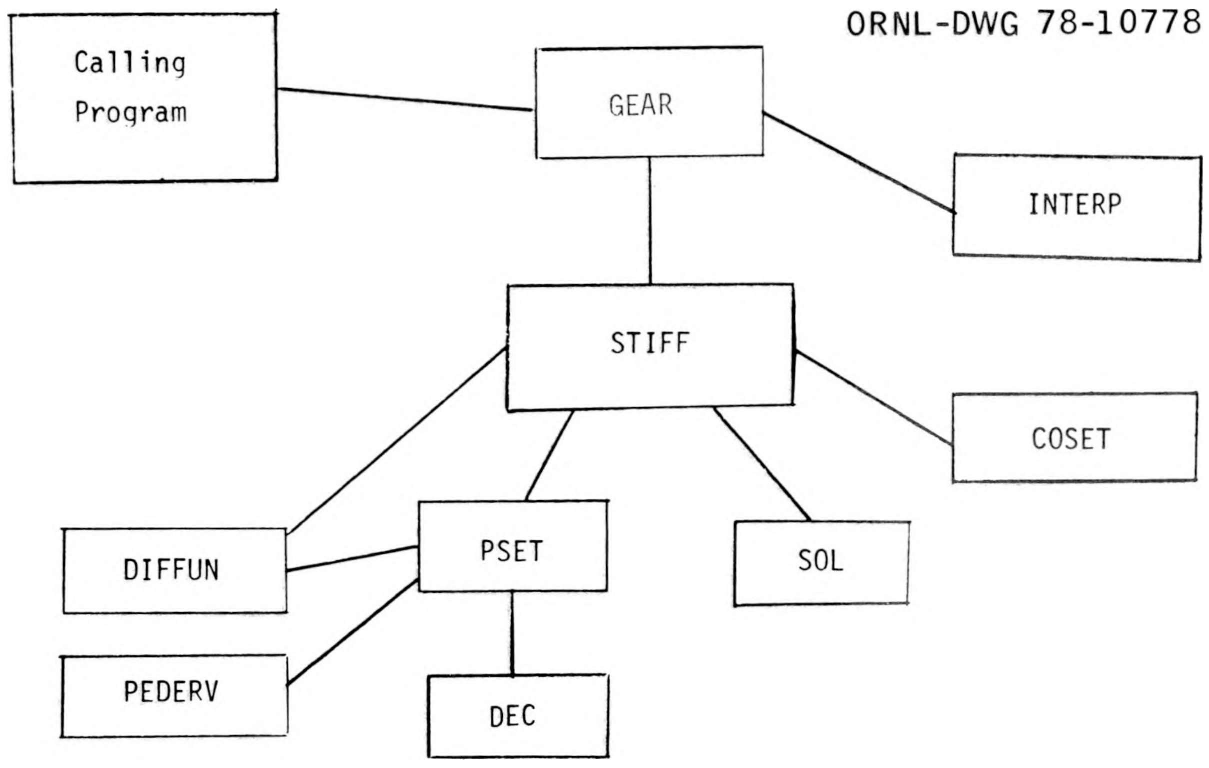


Fig. 17 Block Diagram for GEAR.

The subroutine GEAR controls the calls to the other subroutines and returns the solution and messages to the main program. STIFF performs a single step of the integration and tries to control the local error by selecting the step size and the order of the method. Since the last step may go beyond TOUT, INTERP computes the interpolated values for $y_1, \ldots, y_n$ at TOUT. The subroutines COSET and PSET set various constants and DEC and SOL are used in solving linear algebra problems associated with the differential equations.

B. The Calling Program

The user needs to write a calling program using double-precision floating-point variables and two subroutines also in double precision. The calling program

1) supplies the initial values and sets the parameters in the call list,
2) calls the routine DRIVE, and optionally
3) writes out the results.

The parameters in the call list are:

DIFFUN	the name of the user supplied subroutine DIFFUN(N,T,Y,DOT) which computes the vector function YDOT = F(T,Y).
PEDERV	the name of the user supplied subroutine PEDERV(N,T,Y,PD,NO) which computes the N by N Jacobian matrix of partial derivatives and stores it in PD as an NO by NO array. [Note: PD(I,J) is the partial derivative of YDOT(I) with respect to Y(J)]. This subroutine is called only if MITER (See below) is set equal to 1. In all other cases, PEDERV will

be a dummy subroutine.

N	the number of equations, $1 \leqslant N \leqslant 20$.
TO	the starting value for the independent variable t.
HO	the initial value for the step size, h. It should start out low, and if it is not low enough to pass the error test based on EPS, the program reduces h automatically.
YO	the array that contains the values for Y_1, Y_2, ... Y_n at TOUT.
TOUT	the terminal value for t. The interval of integration goes to TO to TOUT.
EPS	the local error bound. Estimates of the single step error δ_1 in $Y_i(t)$ divided by $YMAX_i$, the previous maximum absolute value of Y_i, are kept less than EPS in the following sense:

$$\{[(\delta_1/YMAX_1)^2 + \dots + (\delta_n/YMAX_n)^2]/n\}^{1/2} \leqslant EPS \ .$$

MF the method flag. You have a choice of two methods, each with four types of iterations for the corrector formula. The selection is determined by the input variable MF,

$$MF = 10*METH + MITER ,$$

where METH indicates the method

1 implicit Adams methods.

2 Gear's "stiffly stable" method.

and MITER indicates the corrector iteration technique

0 functional iteration - no partial derivatives are needed.

1	the chord method with the Jacobian supplied by the user supplied subroutine PEDERV.
2	the chord method with the Jacobian calculated internally by finite differences.
3	the chord method with the Jacobian replaced by a diagonal approximation based on a directional derivative.

If the problem is not stiff then you should use ODE instead. If the problem is stiff, use MF = 21 or 22 for best results.

INDEX a flag used for input and output.

Value	Interpretation
1	this is the first call for this integration problem.
0	this is not the first call for the problem and integration is to continue.
-1	this is not the first call for the problem and the user has reset at least one of N,EPS, or MF.
2	the same as 0 except that TOUT is to be attained without interpolation. This assumes that TOUT is greater than or equal to the current value of T.
3	the same as 0 except that control is returned to the calling program after one step without regard to TOUT.

The subroutine DIFFUN(N, T, Y, YDOT) defines the system of equations while the subroutine PEDERV(N, T, Y, PD, NO) is used if MF = 11 or 21 and applies the Jacobian matrix (i.e., the matrix of the partial derivatives) $\partial f_i/\partial y_j$. In all other cases a dummy routine

```
SUBROUTINE PEDERV (N, T, Y, PD, NO)
RETURN
END
```

must be supplied. Both subroutines must be declared to be EXTERNAL.

C. System Dependent Procedures

The most efficient and convenient way to access GEAR is to use the Core Library of Numerical Software. On the IBM 360 computers, it resides in LOGLIB in load module form. This is automatically available to all FORTRAN jobs without additional JCL. The Core Library resides on the DEC-10 in the SYS area as a REL file library and is accessed when executing a FORTRAN program by typing

```
.EX  MYPROG,MYSUB,SYS:CORLIB/SEA
```

If you have questions or problems, please consult a numerical consultant or Programming Assistance.

D. Values Returned by GEAR

The subroutine GEAR will return these values to your calling program:

HO	the step size last used in STIFF whether it was successful or not.
YO	the values of $y_1,\dots,y_n$ at t = TOUT.
TOUT	if the integration was successful and INDEX was not set to 3 on input, then TOUT is unchanged from its input value. Otherwise, TOUT is the farthest value of t for which integration has been completed.

INDEX indicates the results of the last call.

0 the integration was successfully completed.

–1 the integration was stopped after failing to pass the error test even after reducing the step size by a factor of 10^{10} from its initial value.

–2 the integration was stopped after some success because EPS was too small.

–3 the integration was stopped after failing to achieve corrector convergence even after reducing the step size by a factor of 10^{10}.

–4 at least one input value was illegal, that is,

$$EPS \leqslant 0,\ N \leqslant 0,\ (TO-TOUT)*HO < 0,$$

or index was illegal.

–5 INDEX was –1 but TOUT was not beyond the current value for t.

E. Examples

GEAR-1 A second-order differential equation.

Solve $y'' = 11y' + 10y = 0$

subject to $y(0) = 1,\ y'(0) = -1$

The equation may be transformed into a system of two first order equations by introducing the variables $y_1 = y$ and $y_2 = y' = y_1'$.

$$\text{Solve} \quad y_1' = y_2$$

$$y_2' = -11y_2 - 10y ,$$

$$\text{subject to} \quad y_1(0) = 1,\ y_2(0) = -1.$$

The closed form solution is $y = e^{-t}$. The double precision calling program which sets the initial conditions and calls GEAR is shown in Fig. 18. All eight combinations of the two methods and the four iteration techniques are used. The common block

```
COMMON/GEAR9/HUSED,NQUSED,NSTEP,NFE,NJE
```

has been accessed in the calling program so that the number of steps taken (NSTEP), the number of function evaluations (NFE), and the number of times the Jacobian was evaluated (NJE) could be printed out. [Note: The same equation appears in example EPISODE-1 in Chapter V.]

```
C
C     MAKE ALL NONINTEGER VARIABLE DOUBLE PRECISION VARIABLES
C
        IMPLICIT REAL*8(A-H,O-Z)
C
C     DECLARE THE VARIABLES HELD IN COMMON
C
        COMMON /GEAR9/HUSED, NQUSED, NSTEP, NFE, NJE
C
C     SET THE DIMENSION OF Y0
C
        DIMENSION Y0(2)
C
C     DECLARE  DIFFUN AND PEDERV  TO BE EXTERNAL SUBROUTINES
C
        EXTERNAL DIFFUN, PEDERV
C
C     TEST THE FOUR TECHNIQUES ON BOTH METHODS.
C
        DO  10  METH = 1, 2
          DO  20  MITER1 = 1, 4
            MITER = MITER1 - 1
C
C     SET  N (THE NUMBER OF EQUATIONS) AND THE INITAL CONDITICNS.
C
            N = 2
            Y0(1) = 1.0D0
            Y0(2) = -1.0D0
C
C     SET THE INITIAL AND TERMINAL VALUES OF T.
C
            T0 = 0.0D0
            TOUT = 100.0D0
C
C     SET THE INITIAL STEP SIZE AND THE ERROR BOUND
C
            H0 = 1.0D-10
            EPS = 1.0D-10
C
C      SET INDEX = 1 FOR THE FIRST CALL TO GEAR .
C
            INDEX = 1
C
C     SELECT THE METHOD AND TECHNIQUE
C
            MF = 10*METH + MITER
C
C      WRITE THE VALUES USED TO CALL GEAR.
C
            WRITE(6,100) N,T0,H0,Y0(1),Y0(2),TOUT,IERROR,MF,INDEX
C
C     CALL GEAR
C
            CALL GEAR(DIFFUN,PEDERV,N,T0,H0,Y0,TOUT,EPS,MF,INDEX)
C
C     CHECK INDEX
```

Fig. 18. Driver Program for Example GEAR-1

```
C
          IF(INDEX .EQ. 0) GO TO 30
C
C   WRITE THE RESULTS.
C
          WRITE(6,101) INDEX,H0,Y0(1),Y0(2),TOUT,EPS,MF
          GO TO 20
  30      WRITE(6,102) MF, TOUT, Y0(1), Y0(2), NSTEP, NFE, NJE
      IF(MITER1.EQ.2.OR.MITER1.EQ.4)WRITE(6,104)
  20    CONTINUE
  10  CONTINUE
C
C   FORMATS
C
 100  FORMAT(//,5X,39HTHESE ARE THE VALUES USED TO CALL GEAR:,//,5X,
     1 8HN       =,I7,/,5X,8HT0      =,D20.10,/,5X,8HH0      =,D20.10,/,5X,
     2 8HY0(1)   =,D20.10,/5X,8HY0(2)   =,D20.10,/,5X,8HTOUT    =,D20.10,/,
     3 5X,8HEPS     =,D20.10,/,5X,8HMF      =,I7,/,5X,8HINDEX   =,I7)
 101  FORMAT(//,5X16HWARNING  INDEX =,I3,/,5X,8HH0      =,D20.10,/,5X,
     1 8HY0(1)   =,D20.10,/,5X,8HY0(2)   =,D20.10,/,5X,8HTOUT    =,D20.10,
     2 /,5X,8HEPS     =,D20.10,/,5X,8HMF      =,I7)
 102  FORMAT(//,5X,13HMF WAS SET TO,I3,//,5X,8HTOUT    =,D20.10,/,5X,
     1 8HY0(1)   =,D20.10,/,5X,8HY0(2)   =,D20.10,/5X,28HTHE PROBLEM WAS C
     2COMPLETED IN,I5,7H STEPS.,/,5X,10HTHERE WERE,I5,21H CALLS TO DIFFUN
     3 AND,/,5X,10HTHERE WERE,I5,17H CALLS TO PEDERV.)
 103  FORMAT(//,5X,29HTHIS IS A NORMAL TERMINATION.)
  104 FORMAT(1H1)
      STOP
      END
```

Fig. 18. (Cont'd)

Note that all four modifications of both methods will be used. The two subroutines are shown in Fig. 19.

```
      SUBROUTINE DIFFUN(N, T, Y, YDOT)
C
C   MAKE ALL NONINTEGER VARIABLE DOUBLE PRECISION VARIABLES
C
      IMPLICIT REAL*8(A-H,O-Z)
C
C   DIMENSION  Y  AND  YDOT  FOR THE SYSTEM OF EQUATIONS.
C
      DIMENSION Y(2), YDOT(2)
C
C   DEFINE THE SYSTEM OF EQUATIONS.
C
      YDOT(1) = Y(2)
      YDOT(2) = -10.0D0*Y(1) - 11.0D0*Y(2)
      RETURN
      END

      SUBROUTINE PEDERV(N, T, Y, PD, NO)
C
C   MAKE ALL NONINTEGER VARIABLE DOUBLE PRECISION VARIABLES
C
      IMPLICIT REAL*8(A-H,O-Z)
C
C   SET THE DIMENSION FOR  PD , THE JACOBIAN MATRIX OF PARTIAL
C   DERIVATIVES.
C
      DIMENSION PD(NO, NO)
C
C   DEFINE THE  NO BY NO  MATRIX  PD .
C
      PD(1,1) = 0.0D0
      PD(1,2) = 1.0D0
      PD(2,1) = -10.0D0
      PD(2,2) = -11.0D0
      RETURN
      END
```

Fig. 19. Subroutines PEDERV and DIFFUN for Example GEAR-1

When your job is executed you will have the results displayed in Fig. 20.

```
THESE ARE THE VALUES USED TO CALL GEAR:

N         =       2
TO        =     0.0
HO        =     C.1000000000D-09
YO (1)    =     0.1000000000D 01
YO (2)    =    -0.1000000000D 01
TOUT      =     C.1000000000D 03
EPS       =     0.7863317309E 66
MF        =      10
INDEX     =       1


MF WAS SET TC 10

TOUT      =     0.1000000000D 03
YO (1)    =     0.1165759789D-12
YO (2)    =    -C.1165759789D-11
THE PRCBIEM WAS COMPLETED IN 1332 STEPS.
THERE WERE 2213 CALLS TO DIFFUN AND,
THERE WERE    0 CALLS TO PEDERV.


THESE ARE THE VALUES USED TO CALL GEAR:

N         =       2
TO        =     0.0
HO        =     C.1000000000D-09
YO (1)    =     0.1000000000D 01
YO (2)    =    -0.1000000000D 01
TOUT      =     C.1000000000D 03
EPS       =     0.7863317309E 66
MF        =      11
INDEX     =       1


MF WAS SET TC 11

TOUT      =     0.1000000000D 03
YO (1)    =     0.3578126631D-18
YO (2)    =    -0.3625970288D-18
THE PRCBIEM WAS COMPLETED IN  203 STEPS.
THERE WERE  277 CALLS TO DIFFUN AND,
THERE WERE   25 CALLS TO PEDERV.
```

Fig. 20. Output for Example GEAR-1

```
THESE ARE THE VALUES USED TO CALL GEAR:

N        =       2
TO       =     0.0
HO       =     C.10C000C000D-09
YO (1)   =     0.1000000000D 01
YO (2)   =    -0.1000000000D 01
TOUT     =     0.10C00C0000D 03
EPS      =     0.7863317309E 66
MF       =      12
INDEX    =       1


MF WAS SET TO 12

TOUT     =     0.1000000000D 03
YO (1)   =     0.6608692204D-19
YO (2)   =    -0.6631521927D-19
THE PRCBIEM WAS COMPLETED IN  222 STEPS.
THERE WERE  375 CALLS TO DIFFUN AND,
THERE WERE   29 CALLS TO PEDERV.


THESE ARE THE VAIUES USED TO CALL GEAR:

N        =       2
TO       =     0.0
HO       =     C.10C00CC000D-09
YO (1)   =     0.1000000000D 01
YO (2)   =    -0.1000000000D 01
TOUT     =     C.10C0CC0000D 03
EPS      =     0.7863317309E 66
MF       =      13
INDEX    =       1


MF WAS SET TO 13

TOUT     =     0.1000000000D 03
YO (1)   =    -0.5391672358D-10
YO (2)   =     C.4927872230D-10
THE PRCBIEM WAS COMPLETED IN  297 STEPS.
THERE WERE  633 CALLS TO DIFFUN AND,
THERE WERE   70 CALLS TO PEDERV.
```

Fig. 20. (Cont'd)

```
THESE ARE THE VALUES USED TO CALL GEAR:

N       =      2
TO      =     0.0
HO      =     C.10C0000000D-09
YO (1)  =     0.1000000000D 01
YO (2)  =    -0.1000000000D 01
TOUT    =     C.1000000000D 03
EPS     =     0.7863317309E 66
MF      =     20
INDEX   =      1


MF WAS SET TC 20

TOUT    =     0.1000000000D 03
YO (1)  =    -0.7597327242D-12
YO (2)  =     C.7597327242D-11
THE PRCBLEM WAS COMPLETED IN 1313 STEPS.
THERE WERE 2116 CALLS TO DIFFUN AND,
THERE WERE    0 CALLS TO PEDERV.


THESE ARE THE VALUES USED TO CALL GEAR:

N       =      2
TO      =     0.0
HO      =     C.1000000000D-09
YO (1)  =     0.1000000000D 01
YO (2)  =    -0.1000000000D 01
TOUT    =     C.10C0CC0000D 03
EPS     =     0.7863317309E 66
MF      =     21
INDEX   =      1


MF WAS SET TC 21

TOUT    =     0.1000000000D 03
YO (1)  =     0.7112814795D-13
YO (2)  =    -C.7112814795D-13
THE PRCBLEM WAS COMPLETED IN  325 STEPS.
THERE WERE  362 CALLS TO DIFFUN AND,
THERE WERE   27 CALLS TO PEDERV.
```

Fig. 20. (Cont'd)

```
THESE ARE THE VALUES USED TO CALL GEAR:

N        =       2
TO       =    0.0
HO       =    0.1000000000D-09
YO (1)   =    0.1000000000D 01
YO (2)   =   -0.1000000000D 01
TOUT     =    0.1000000000D 03
EPS      =    0.7863317309E 66
MF       =     22
INDEX    =      1


MF WAS SET TO 22

TOUT     =    0.1000000000D 03
YO (1)   =    0.7112814797D-13
YO (2)   =   -0.7112814797D-13
THE PROBLEM WAS COMPLETED IN  325 STEPS.
THERE WERE  416 CALLS TO DIFFUN AND,
THERE WERE   27 CALLS TO PEDERV.


THESE ARE THE VALUES USED TO CALL GEAR:

N        =       2
TO       =    0.0
HO       =    0.1000000000D-09
YO (1)   =    0.1000000000D 01
YO (2)   =   -0.1000000000D 01
TOUT     =    0.1000000000D 03
EPS      =    0.7863317309E 66
MF       =     23
INDEX    =      1


MF WAS SET TO 23

TOUT     =    0.1000000000D 03
YO (1)   =   -0.2176168699D-10
YO (2)   =    0.1990345713D-10
THE PROBLEM WAS COMPLETED IN  364 STEPS.
THERE WERE  588 CALLS TO DIFFUN AND,
THERE WERE   53 CALLS TO PEDERV.

IHC002I STOP       0
```

Fig. 20. (Cont'd)

2. GEAR-2 A pair of stiff equations

Solve
$$y_1' = 998y_1 + 1998y_2$$
$$y_2' = -999y_1 - 1999y_2$$
subject to $y_1(0) = 1, y_2(0) = 1.$

Analytically, the closed form solution is:
$$y_1(t) = 4e^{-t} - 3e^{-1000t}$$
$$y_2(t) = -2e^{-t} + 3e^{-1000t}$$

The calling program and the two subroutines are shown in Fig. 21 and the printout for this program is shown in Fig. 22.

```
      IMPLICIT REAL*8(A-H,O-Z)
      COMMON /GEAR9/HUSED, NQUSED, NSTEP, NFE, NJE
      DIMENSION Y0(2)
      EXTERNAL DIFFUN, PEDERV
      N = 2
      Y0(1)  = 1.0D0
      Y0(2)  = 1.0D0
      T0 = 0.0D0
      TOUT =  1.0D0
      H0 = 1.CD-10
      EPS =  1.0D-10
      MF = 21
      INDEX = 1
      WRITE(6,100) N,T0,H0,Y0(1),Y0(2),TOUT,EPS,MF,INDEX
      CALL GEAR(DIFFUN,PEDERV,N,T0,H0,Y0,TOUT,EPS,MF,INDEX)
      WRITE(6,101) MF, INDEX, TOUT, Y0(1), Y0(2), NSTEP, NFE, NJE
      WRITE(6,102)
100   FORMAT(//,5X,39HTHESE ARE THE VALUES USED TO CALL GEAR:,//,5X,
     1 8HN       =,I7,/,5X,8HT0      =,D20.10,/,5X,8HH0      =,D20.10,/,5X,
     2 8HY0(1)   =,D20.10,/5X,8HY0(2)   =,D20.10,/,5X,8HTOUT    =,D20.10,/,
     3 5X,8HEPS     =,D20.10,/,5X,8HMF      =,I7,/,5X,8HINDEX   =,I7)
101   FORMAT(//,5X,13HMF WAS SET TO,I3,//,5X,8HTOUT    =,D20.10,/,5X,
     1 8HY0(1)   =,D20.10,/,5X,8HY0(2)   =,D20.10,/5X,28HTHE PRCBLEM WAS C
     2OMPLETED IN,I5,7H STEPS.,/,5X,10HTHERE WERE,I5,21H CALLS TO DIFFUN
     3 AND,/,5X,10HTHERE WERE,I5,17H CALLS TO PEDERV.)
102   FORMAT(//,5X,29HTHIS IS A NORMAL TERMINATION.)
      STOP
      END
```

```
      SUBROUTINE PEDERV(N, T, Y, PD, N0)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION PD(N0, N0)
      PD(1,1) =   998.0D0
      PD(1,2) =  1998.0D0
      PD(2,1) =  -999.0D0
      PD(2,2) = -1999.0D0
      RETURN
      END
```

```
      SUBROUTINE DIFFUN(N, T, Y, YDOT)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION Y(2), YDOT(2)
      YDOT(1) = 998.0D0*Y(1) + 1998.0*Y(2)
      YDOT(2) = -999.0D0*Y(1) -1999.0D0*Y(2)
      RETURN
      END
```

Fig. 21. Calling Program and Subroutines PEDERV and DIFFUN for Example GEAR-2

```
    THESE ARE THE VALUES USED TO CALL GEAR:

    N       =      2
    TO      =     0.0
    HO      =     C.10COCCCOOOD-09
    YO (1)  =     0.1000000000D 01
    YO (2)  =     0.1000000000D 01
    TOUT    =     C.10COCCCOOOD 01
    EPS     =     0.1000000000D-09
    MF      =     21
    INDEX   =      1


    MF WAS SET TC 21

    TOUT    =     0.0
    YO (1)  =     0.1000000000D 01
    YO (2)  =     C.1471517766D 01
    THE PRCBIEM WAS CCMPLETED IN***** STEPS.
    THERE WERE  389 CALLS TO DIFFUN AND,
    THERE WERE  427 CALLS TO PEDERV.


    MF WAS SET TC 32

    TOUT    =


    THIS IS A NORMAL TERMINATION.

IHC002I STCP      0
```

Fig. 22. Output from the Program for Example GEAR-2

3. GEAR-3 Creating a table:

Solve $y' = -y$

subject to $y(0) = 1$.

The closed form solution is $y = e^{-t}$. A table of values for y will be printed for t = 0, 0.1, 0.2, ..., 1.0. The main program and two subroutines are shown in Fig. 23. The printout for this program is shown in Fig. 24. Since MF has been set to 10, the subroutine PEDERV is never called so a dummy subroutine has been used. [Note: The same problem appears in Example ODE-3 in Chapter II.]

```
      IMPLICIT REAL*8(A-H,O-Z)
      COMMON /GEAR9/HUSED, NQUSED, NSTEP, NFE, NJE
      DIMENSION Y0(1)
      EXTERNAL DIFFUN, PEDERV
      N = 1
      Y0(1)  = 1.0D0
      T0 = 0.0D0
      TOUT = 0.1D0
      H0 = 1.0D-10
      EPS = 1.0D-10
      MF = 10
      INDEX = 1
      WRITE(6,100) T0, Y0(1)
C
C    SET UP THE LOOP TO FORM THE TABLE.
C
      DO  10  I = 1, 10
        CALL GEAR(DIFFUN,PEDERV,N,T0,H0,Y0,TOUT,EPS,MF,INDEX)
        WRITE(6,101) TOUT, Y0(1)
        TOUT = TOUT + 0.1D0
   10 CONTINUE
  100 FORMAT(5X,39HA TABLE OF VALUES FOR YP = -Y, Y(0) = 1,//,10X,1HT,
     1 24X,1HY,/,5X,D20.10,5X,D20.10)
  101 FORMAT(5X,D20.10,5X,D20.10)
      STOP
      END
```

Fig. 23. Calling Program and Subroutines DIFFUN and PEDERV for Example GEAR-3

```
SUBROUTINE DIFFUN(N, T, Y, YDOT)
IMPLICIT REAL*8(A-H,O-Z)
DIMENSION Y(1), YDOT(1)
YDOT(1) = -Y(1)
RETURN
END

SUBROUTINE PEDERV(N, T, Y, PD, N0)
RETURN
END
```

Fig. 23. (Cont'd)

```
   A TABLE OF VALUES FOR YP = -Y, Y(0) = 1

       T                              Y
      0.0                            0.1000000000D 01
      0.1000000000D 00               0.9048374179D 00
      0.2000000000D 00               0.8187307531D 00
      0.3000000000D 00               0.7408182207D 00
      0.4000000000D 00               0.6703200460D 00
      0.5000000000D 00               0.6065306597D 00
      0.6000000000D 00               0.5488116360D 00
      0.7000000000D 00               0.4965853037D 00
      0.8000000000D 00               0.4493289640D 00
      0.9000000000D 00               0.4065696596D 00
      0.1000000000D 01               0.3678794410D 00

IEC002I STOP      0
```

Fig. 24. Output from the Program for Example GEAR-3

V. EPISODE

A. Description of the Subroutines

This Software package consists of eight FORTRAN double-precision subroutines, EPSODE, INTERP, TSTEP, COSET, ADJUST, PSET, DEC, and SOL, which integrate dY/dt = F(t,Y) from t = T to TOUT. The user can repeatedly reset TOUT and call EPSODE to continue the integration or the user can specify that control is to be returned after one step in the direction of TOUT. EPISODE was designed to solve a typical set of problems from chemical engineering and works well on stiff systems that are oscillatory or highly nonlinear. The user is advised to select the option METH = 2, which uses a variable-order, variable-step size backward differentiation method.

ORNL-DWG 78-10796

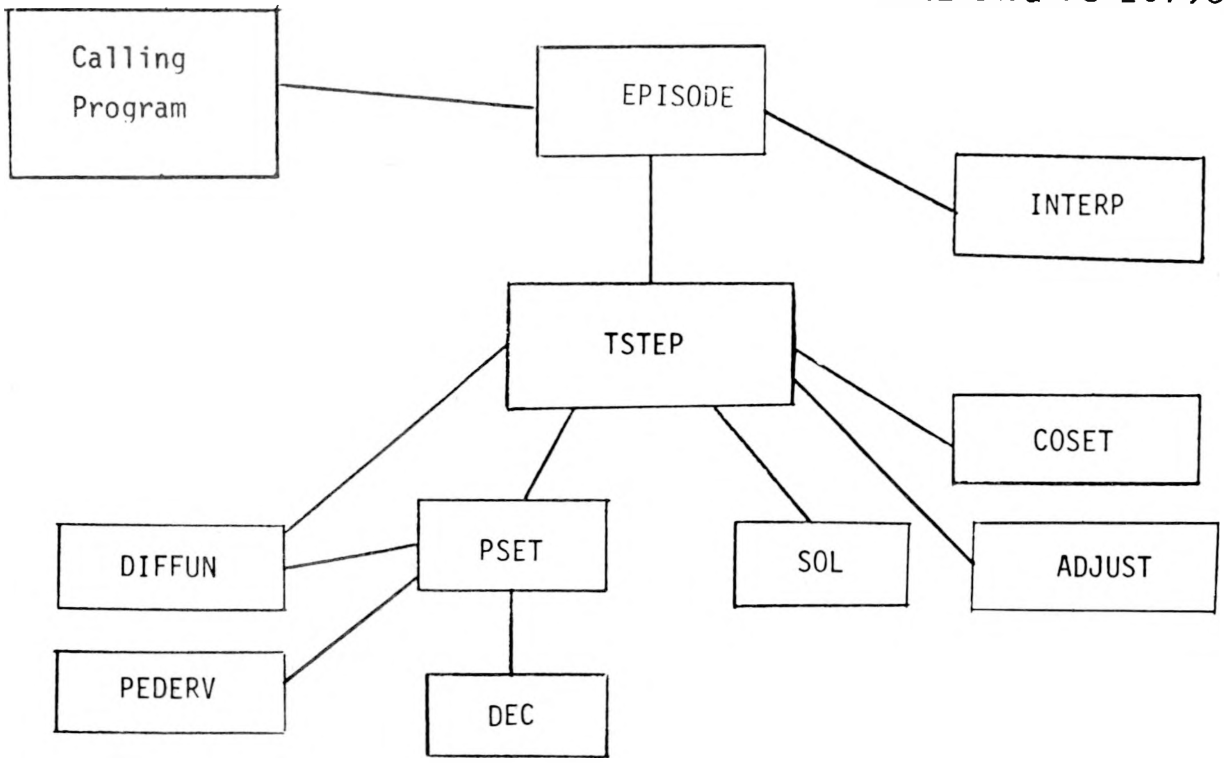


Fig. 25. Block Diagram for EPSODE

The Subroutine EPSODE is called once for each output value of T. It then makes repeated calls to TSTEP and one call to INTERP. It returns the solution and messages. TSTEP is the integration subroutine and controls the error by selecting the step size and the order of the method. Since the last step may go beyond TOUT, INTERP computes the interpolated values for Y(1), Y(2), ..., Y(N) at TOUT. COSET sets coefficients used in TSTEP and ADJUST adjusts the history array on reduction of order. Only when MITER equals 1 or 2 are PSET, which computes and processes the Jacobian matrix, DEC, which performs the LU decomposition of a matrix, and SOL, which solves a linear system AX = B where A has been processed by DEC, called.

B. The Calling Program

The user needs to write a calling program using double-precision floating point variables and two subroutines which also use double-precision. The main program:

1) supplies the initial values,
2) sets the variables in the call list,
3) calls the routine EPSODE, and optionally
4) writes out the results.

The variables in the call list are

DIFFUN	the name of the user supplied subroutine DIFFUN(N,T,Y, YDOT) which computes the vector function YDOT = F(T,Y).
PEDERV	the name of the user supplied subroutine PEDERV(N,T,Y, PD, NO) which computes the N by N Jacobian matrix of partial derivatives and stores in PD as an NO by NO

array. [Note: PD(I,J) is the partial derivative of YDOT(I) with respect to Y(J).] This subroutine is called only if MITER is set equal to 1 (See below). In all other cases, PEDERV will be a dummy subroutine.

N	the number of equations, $1 \leq N \leq 20$ (it is possible to increase this upper bound of 20).
TO	the starting value for the independent variable t.
HO	the initial value for step size, h. It should start out small; and if it is not small enough to pass the error test based on EPS, the program automatically reduces h by up to a factor of 10^{-3} before stopping.
YO	the array that contains the values for Y(1), Y(2), ..., Y(N) at t = TO. After EPSODE returns control to the main program, YO contains the values of Y(1), Y(2), ..., Y(N) at TOUT.
TOUT	the terminal value for t at the end of this call to DRIVE.
EPS	the relative error bound used on the first step. Let R(I) denote the estimated relative local error in Y(I) (i.e., the error relative to YMAX(I)). Then EPS is a bound on the root mean square norm of the vector R, that is,

$$[\{(R(1))^2 + \dots + (R(N))^2\}/N]^{1/2} \leq EPS .$$

IERROR	the error flag.	
	Value	Meaning
	1	absolute error control, YMAX(I) = 1.
	2	error relative to ABS(Y) is controlled. If the initial value of Y(I) is 0, then YMAX(I) is equal to 1 initially.
	3	error relative to the largest value of YMAX(I) seen so far is controlled. If the initial value of Y(I) is 0, then YMAX(I) is set to 1 initially.
MF	the method flag. It is a two-digit decimal integer.	
	METH = 1	indicates a variable step size, variable order Adams method suitable for nonstiff problems.
	METH = 2	indicates a variable step size, variable order backward differentiation method suitable for stiff problems.
MITER	indicates the method of interative correction.	
	MITER = 0	indicates functional iteration and no partial derivatives are needed.
	MITER = 1	indicates a chord or semi-stationary Newton method with a Jacobian matrix of partial derivatives supplied by the user-written subroutine PEDERV.
	MITER = 2	indicates a chord Newton method in an internally computed Jacobian.

	MITER = 3	indicates a chord Newton method with an internally computed diagonal matrix approximation to the Jacobian.
INDEX	a flag used for input and output	

Value	Interpretation
1	this is the first call for this integration problem.
0	this is not the first call for the problem and integration is to continue.
-1	this is not the first call for the problem and the user has reset at least one of N,EPS, or MF.
2	the same as 0 except that TOUT is to be attained without interpolation. This assumes that TOUT is greater than or equal to the current value of T.
3	the same as 0 except that control is returned to the calling program after one step without regard to TOUT.

The subroutine DIFFUN(N,T,Y,YDOT) defines the system of differential equations while the subroutine PEDERV(N,T,Y,PD,NO) provides the Jacobian matrix of partial derivatives. If you do not use MF = 11 or 21, you can use the dummy routine

```
SUBROUTINE PEDERV(N,T,Y,PD,NO)
RETURN
END
```

Both subroutines must be declared to be EXTERNAL.

C. System Dependent Procedures

The most efficient and convenient way to access GEAR is to use the Core Library of Numerical Software. On the IBM 360 computers, it resides in LOGLIB in load module form. This is automatically available to all FORTRAN jobs without additional JCL. The Core Library resides on the DEC-10 in the SYS area as a REL file library is accessed when executing a FORTRAN program by typing

```
.EX MYPROG,MYSUB,SYS:CORLIB/SEA
```

If you have questions or problems, please consult a numerical consultant or Programming Assistance.

D. Values Returned by EPSODE

The subroutine EPSODE will return to your calling program these values:

HO	the step size used last whether or not the step was successful.
YO	the values of Y(1),Y(2),..., Y(N) at TO.
TO	the last value of t reached successfully. It is TOUT in most cases.

	Value	Interpretation
INDEX =	0	the integration was completed to TOUT or beyond.
	1	the integration was halted because the error test was failed--even after reducing h by a factor of 10^{10} from its initial value.
	-2	after some initial success, the integration

was halted by repeated error test failures. Perhaps too much accuracy is being requested or a bad choice of MF was made.

-3	the integration was halted because the corrector failed to converge even after reducing h by a factor of 10^{10} from its initial value.
-4	an error was made in the values of the input parameters.
-5	INDEX was -1 on input but TOUT was not beyond T. Interpolation to t = TOUT was performed. The user may reset INDEX to -1, assign a new value to TOUT and call EPSODE again.
-6	INDEX was 2 on input but TOUT was not beyond T. No action was taken.

The following statement is taken from the Abstract to [3]:

"We conclude that EPISODE is generally faster than GEAR for problems involving wave fronts or transients on the interior of the interval of integration. For linear or simply decaying problems, these roles are usually reversed."

E. Examples

1. EPISODE-1 A second-order differential equation.

Solve $$y'' + 11y' + 10y = 0$$

subject to $$y(0) = 1,\ y'(0) = -1 .$$

The equation may be transformed into a system of two first-order equations by introducing the variables $y_1 = y$ and $y_2 = y' = y_1'$. The equivalent problem is

Solve $y_1' = y_2$

$y_2' = -11y_2 - 10y_1$

subject to $y_1(0) = 1,\ y_2(0) = -1$.

The closed form solution is $y = e^{-t}$. The double-precision main program which sets the initial conditions and calls EPSODE is shown in Fig. 26.

```
C
C    MAKE ALL NONINTEGER VARIABLE DOUBLE PRECISION VARIABLES
C
       IMPLICIT REAL*8(A-H,O-Z)
C
C    DECLARE THE VARIABLES HELD IN COMMON
C
       COMMON /EPCOM9/ HUSED, NQUSED, NSTEP, NFE, NJE
C
C    SET THE DIMENSION OF Y0
C
       DIMENSION Y0(2)
C
C    DECLARE  DIFFUN AND PEDERV  TO BE EXTERNAL SUBROUTINES
C
       EXTERNAL DIFFUN,PEDERV
C
C    TEST THE FOUR TECHNIQUES ON BOTH METHODS.
C
       DO  10  METH = 1, 2
         DO  20  MITER1 = 1, 4
           MITER = MITER1 - 1
C
C    SET  N (THE NUMBER OF EQUATIONS) AND THE INITAL CONDITICNS.
C
           N = 2
           Y0(1) = 1.0D0
           Y0(2) = -1.0D0
C
C    SET THE INITIAL AND TERMINAL VALUES OF T.
C
           T0 = 0.0D0
           TOUT = 100.0D0
C
C    SET THE INITIAL STEP SIZE AND THE ERROR BOUND
C
           H0 = 1.0D-10
           EPS = 1.0D-10
C
C     SET INDEX = 1 FOR THE FIRST CALL TO EPSODE.
C
           INDEX = 1
C
C    SELECT THE METHOD AND TECHNIQUE
C
           MF = 10*METH + MITER
```

Fig. 26. Calling Program for Example EPISODE-1

```
C
C   SET IERROR = 3
C
          IERROR = 3
C
C    WRITE THE VALUES USED TO CALL EPSODE.
C
          WRITE(6,100) N,T0,H0,Y0(1),Y0(2),TOUT,EPS,IERROR,MF,INDEX
C
C   CALL EPSODE


C
          CALL EPSODE(DIFFUN,PEDERV,N,T0,H0,Y0,TOUT,EPS,IERROR,MF,INDEX)
C
C   CHECK INDEX
C
       IF(INDEX .EQ. 0) GO TO 30
C
C   WRITE THE RESULTS.
C
          WRITE(6,101) INDEX,H0,Y0(1),Y0(2),TOUT,EPS,MF
          GO TO 20
  30      WRITE(6,102) MF, IERROR, TOUT, Y0(1), Y0(2), NSTEP, NFE, NJE
       IF(MITER1.EQ.2.OR.MITER1.EQ.4)WRITE(6,104)
  20     CONTINUE
  10   CONTINUE.
C
C   FORMATS
C
 100   FORMAT(//,5X,41HTHESE ARE THE VALUES USED TO CALL EPSODE:,//,5X,
     1 8HN      =,I6,/,5X,8HT0      =,D20.10,/,5X,8HH0      =,D20.10,/,5X,
     2 8HY0(1)  =,D20.10,/5X,8HY0(2)  =,D20.10,/,5X,8HTOUT   =,D20.10,/,
     3 5X,8HEPS    =,D20.10,/,5X,8HIERROR =,I6,/,5X,8HMF     =,I6,/,
     4 5X,8HINDEX  =,I6)
 101   FORMAT(//,5X16HWARNING  INDEX =,I3,/,5X,8HH0     =,D20.10,/,5X,
     1 8HY0(1)  =,D20.10,/,5X,8HY0(2)  =,D20.10,/,5X,8HTOUT   =,D20.10,
     2 /,5X,8HEPS    =,D20.10,/,5X,8HIERROR=,I6,/,5X,8HMF     =,I6)
 102   FORMAT(//,5X,13HMF WAS SET TO,I7,/,5X,17HIERROR WAS SET TO,I3,//
     1 5X,8HTOUT   =,D20.10,/,5X,8HY0(1)  =,D20.10,/,5X,8HY0(2)  =,
     2 D20.10,/,5X,28HTHE PROBLEM WAS COMPLETED IN,I5,7H STEPS.,/,5X,
     3 10HTHERE WERE,I5,20H CALLS TO DIFFUN AND,/,5X,10HTHERE WERE,I5,
     4 17H CALLS TO PEDERV.)
 103   FORMAT(//,5X,29HTHIS IS A NORMAL TERMINATION.)
  104  FORMAT(1H1)
       STOP
       END
```

Fig. 26. (Cont'd)

Note that all four types of iteration techniques on the corrector will be used with each of the two methods so that there will be eight solutions. The two subroutines, DIFFUN and PEDERV, are shown in Figures 27 and 28. [Note: The same equation appears in Example GEAR-1 in Chapter IV.]

```
      SUBROUTINE DIFFUN(N, T, Y, YDOT)
C
C   MAKE ALL NONINTEGER VARIABLE DOUBLE PRECISION VARIABLES
C
      IMPLICIT REAL*8(A-H,O-Z)
C
C   DIMENSION  Y  AND  YDOT  FOR THE SYSTEM OF EQUATIONS.
C
      DIMENSION Y(2), YDOT(2)
C
C   DEFINE THE SYSTEM OF EQUATIONS.
C
      YDOT(1) = Y(2)
      YDOT(2) = -10.0D0*Y(1) - 11.0D0*Y(2)
      RETURN
      END
```

Fig. 27. Subroutine DIFFUN for Example EPISODE-1

```
      SUBROUTINE PEDERV(N, T, Y, PD, NO)
C
C   MAKE ALL NONINTEGER VARIABLES DOUBLE PRECISION VARIABLES.
C
      IMPLICIT REAL*8(A-H,O-Z)
C
C   SET THE DIMENSION FOR  PD , THE JACOBIAN MATRIX OF PARTIAL
C   DERIVATIVES.
C
      DIMENSION PD(NO, NO)
C
C   DEFINE THE  NO BY NO  MATRIX  PD .
C
      PD(1,1) = 0.0D0
      PD(1,2) = 1.0D0
      PD(2,1) = -10.0D0
      PD(2,2) = -11.0D0
      RETURN
      END
```

Fig. 28. Subroutine PEDERV for Example EPISODE-1

When your program runs you will get the results displayed in Figure 29.

```
THESE ARE THE VALUES USED TO CALL EPSODE:

N       =      2
TO      =     0.0
HO      =     C.10COCOC000D-09
YO (1)  =     0.1000000000D 01
YO (2)  =    -0.1000000000D 01
TOUT    =     C.10COCCC000D 03
EPS     =     0.1000000000D-09
IERROR  =      3
MF      =     10
INDEX   =      1


MF WAS SET TO      10
IERROR WAS SET TC   3

TOUT    =     0.1000000000D 03
YO (1)  =     0.1040599335D-10
YO (2)  =    -C.104C599335D-09
THE PROBLEM WAS COMPLETED IN 2912 STEPS.
THERE WERE 3412 CALLS TO DIFFUN AND
THERE WERE    0 CALLS TO PEDERV.

THESE ARE THE VALUES USED TO CALL EPSODE:

N       =      2
TO      =     0.0
HO      =     C.10C0000000D-09
YO (1)  =     0.1000000000D 01
YO (2)  =    -0.1000000000D 01
TOUT    =     C.10COCCC000D 03
EPS     =     0.1000000000D-09
IERROR  =      3
MF      =     11
INDEX   =      1


MF WAS SET TO      11
IERROR WAS SET TC   3

TOUT    =     0.1000000000D 03
YO (1)  =    -0.6644411195D-12
YO (2)  =     C.6644411195D-11
THE PROBLEM WAS COMPLETED IN  363 STEPS.
THERE WERE  564 CALLS TO DIFFUN AND
THERE WERE   61 CALLS TO PEDERV.
```

Fig. 29. Output from the Program for Example EPISODE-1

```
THESE ARE THE VALUES USED TO CALL EPSODE:

N       =      2
TO      =     0.0
HO      =     0.1000000000D-09
YO (1)  =     0.1000000000D 01
YO (2)  =    -0.1000000000D 01
TOUT    =     0.1000000000D 03
EPS     =     0.1000000000D-09
IERROR  =      3
MF      =     20
INDEX   =      1


MF WAS SET TO      20
IERROR WAS SET TO   3

TOUT    =     0.1000000000D 03
YO (1)  =     0.1508244560D-12
YO (2)  =    -0.1508244560D-11
THE PROBLEM WAS COMPLETED IN 1443 STEPS.
THERE WERE 2375 CALLS TO DIFFUN AND
THERE WERE    0 CALLS TO PEDERV.


THESE ARE THE VALUES USED TO CALL EPSODE:

N       =      2
TO      =     0.0
HO      =     0.1000000000D-09
YO (1)  =     0.1000000000D 01
YO (2)  =    -0.1000000000D 01
TOUT    =     0.1000000000D 03
EPS     =     0.1000000000D-09
IERROR  =      3
MF      =     21
INDEX   =      1


MF WAS SET TO      21
IERROR WAS SET TO   3

TOUT    =     0.1000000000D 03
YO (1)  =     0.5898197618D-11
YO (2)  =    -0.5898197618D-11
THE PROBLEM WAS COMPLETED IN  304 STEPS.
THERE WERE  379 CALLS TO DIFFUN AND
THERE WERE   44 CALLS TO PEDERV.
```

Fig. 29. (Cont'd)

```
THESE ARE THE VALUES USED TO CALL EPSODE:

N       =      2
TO      =     0.0
HO      =     C.10COCCC000D-09
YO (1)  =     0.1000000000D 01
YO (2)  =    -0.1000000000D 01
TOUT    =     C.10COCCCOCOD 03
EPS     =     0.1000000000D-09
IERROR  =      3
MF      =     12
INDEX   =      1


MF WAS SET TO      12
IERRCR WAS SET TC   3

TOUT    =     0.1000000000D 03
YO (1)  =     0.1696466502D-19
YO (2)  =    -C.1696467336D-19
THE PRCBIEM WAS COMPLETED IN  407 STEPS.
THERE WERE  783 CALLS TO DIFFUN AND
THERE WERE   80 CALLS TO PEDERV.


THESE ARE THE VALUES USED TO CALL EPSODE:

N       =      2
TO      =     0.0
HO      =     C.10COCOC000D-09
YO (1)  =     0.1000000000D 01
YO (2)  =    -0.1000000000D 01
TOUT    =     C.1COCOCC000D 03
EPS     =     0.1000000000D-09
IERROR  =      3
MF      =     13
INDEX   =      1


MF WAS SET TO      13
IERROR WAS SET TC   3

TOUT    =     0.1000000000D 03
YO (1)  =    -0.4982889154D-09
YO (2)  =     C.4077953845D-09
THE PRCBIEM WAS CCMPLETED IN  307 STEPS.
THERE WERE  635 CALLS TO DIFFUN AND
THERE WERE   84 CALLS TO PEDERV.
```

Fig. 29. (Cont'd)

```
THESE ARE THE VALUES USED TO CALL EPSODE:

N       =      2
TO      =     0.0
HO      =     C.10COCCC000D-09
YO (1)  =     0.1000000000D 01
YO (2)  =    -0.1000000000D 01
TOUT    =     C.10COCOC000D 03
EPS     =     0.1000000000D-09
IERROR  =      3
MF      =     22
INDEX   =      1


MF WAS SET TO      22
IERROR WAS SET TC   3

TOUT    =     0.1000000000D 03
YO (1)  =     0.5898199076D-11
YO (2)  =    -0.5898199076D-11
THE PRCBIEE WAS COEFLETEC IN  304 STEPS.
THERE WERE  467 CALLS TO DIFFUN AND
THERE WERE   44 CALLS TO PEDERV.


THESE ARE THE VAIUES USED TO CALL EPSODE:

N       =      2
TO      =     0.0
HO      =     C.10COCCC000D-09
YO (1). =     0.1000000000D 01
YO (2)  =    -0.1000000000D 01
TOUT    =     C.10COCCC000D 03
EPS     =     0.1000000000D-09
IERROR  =      3
MF      =     23
INDEX   =      1


MF WAS SET TO      23
IERRCR WAS SET TC   3

TOUT    =     0.1000000000D 03
YO (1)  =    -0.1571789488D-10
YO (2)  =     C.26C6899619D-10
THE PRCBIEE WAS COEFLETEC IN  414 STEPS.
THERE WERE  775 CALLS TO DIFFUN AND
THERE WERE  115 CALLS TO PEDERV.


 IEC002I STOP      0
```

Fig. 29 (Cont'd)

2. EPISODE-2 A pair of stiff equations

Solve

$$y_1' = 998y_1 + 1998y_2$$

$$y_2' = -999y_1 - 1999y_2$$

subject to $y_1(1) = 1,\ y_2(0) = 1$.

Analytically, the closed form solution is:

$$y_1(t) = 4e^{-t} - 3e^{-1000t}$$

$$y_2(t) = -2e^{-t} + 3e^{-1000t}$$

The calling program and the two subroutines are shown in Fig. 30.

```
      IMPLICIT REAL*8(A-H,O-Z)
      COMMON /EPCOM9/ HUSED, NQUSED, NSTEP, NFE, NJE
      DIMENSION Y0(2)
      EXTERNAL DIFFUN, PEDERV
C
C    ALL THREE METHODS OF ERROR CONTROL WILL BE USED.
C
      DO  10  I = 1, 3
        N = 2
        Y0(1) = 1.0D0
        Y0(2) = 1.0D0
        T0 = 0.0D0
        TOUT = 1.0D0
        H0 = 1.0D-10
        EPS = 1.0D-10
        MF = 21
C
C    SET IERROR.
C
        IERROR = I
        INDEX = 1
        WRITE(6,100) N,T0,H0,Y0(1),Y0(2),TOUT,EPS,IERROR,MF,INDEX
        CALL EPSODE(DIFFUN,PEDERV,N,T0,H0,Y0,TOUT,EPS,IERROR,MF,INDEX)
        IF (INDEX .NE. 0) WRITE(6,101) INDEX,H0,Y0(1),Y0(2),TOUT,EPS,MF
        IF (INDEX .EQ. 0) WRITE(6,102) MF,IERROR,TOUT,Y0(1),Y0(2),NSTEP,
     1                                 NFE,NJE
      IF(I.EQ.2) WRITE(6,104)
   10 CONTINUE
      WRITE(6,103)
  100 FORMAT(//,5X,41HTHESE ARE THE VALUES USED TO CALL EPSODE:,//,5X,
     1 8HN       =,I6,/,5X,8HT0      =,D20.10,/,5X,8HH0      =,D20.10,/,5X,
     2 8HY0(1)   =,D20.10,/5X,8HY0(2)   =,D20.10,/,5X,8HTOUT    =,D20.10,/,
     3 5X,8HEPS     =,D20.10,/,5X,8HIERROR =,I6,/,5X,8HMF      =,I6,/,
     4 5X,8HINDEX   =,I6)
  101 FORMAT(//,5X 16HWARNING  INDEX =,I3,/,5X,8HH0      =,D20.10,/,5X,
     1 8HY0(1)   =,D20.10,/,5X,8HY0(2)   =,D20.10,/,5X,8HTOUT    =,D20.10,
     2 /,5X,8HEPS     =,D20.10,/,5X,8HIERROR=,I6,/,5X,8HMF      =,I6)
  102 FORMAT(//,5X,13HMF WAS SET TO,I7,/,5X,17HIERROR WAS SET TO,I3,//
     1 5X,8HTOUT    =,D20.10,/,5X,8HY0(1)   =,D20.10,/,5X,8HY0(2)   =,
     2 D20.10,/,5X,28HTHE PROBLEM WAS COMPLETED IN,I5,7H STEPS.,/,5X,
     3 10HTHERE WERE,I5,20H CALLS TO DIFFUN AND,/,5X,10HTHERE WERE,I5,
     4 17H CALLS TO PEDERV.)
  103 FORMAT(//,5X,29HTHIS IS A NORMAL TERMINATION.)
  104 FORMAT(1H1)
      STOP
      END

      SUBROUTINE PEDERV(N, T, Y, PD, N0)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION PD(N0, N0)
      PD(1,1) = 998.0D0
      PD(1,2) = 1998.0D0
      PD(2,1) = -999.0D0
      PD(2,2) = -1999.0D0
      RETURN
      END

      SUBROUTINE DIFFUN(N, T, Y, YDOT)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION Y(2), YDOT(2)
      YDOT(1) = 998.0D0*Y(1) + 1998.0D0*Y(2)
      YDOT(2) = -999.0D0*Y(1) - 1999.0*Y(2)
      RETURN
      END
```

Fig. 30. Calling Program and Subroutines DIFFUN and PEDERV for Example EPISODE-2

The printout for this program is shown in Fig. 31.

```
THESE ARE THE VALUES USED TO CALL EPSODE:

N        =     2
TO       =    0.0
HO       =    C.10C000C000D-09
YO (1)   =    0.1000000000D 01
YO (2)   =    0.1000000000D 01
TOUT     =    C.1CC0CCC000D 01
EPS      =    0.1000000000D-09
IERROR   =     1
MF       =    21
INDEX    =     1

MF WAS SET TO      21
IERROR WAS SET TC   1

TOUT     =    0.1000000000D 01
YO (1)   =    0.1471517767D 01
YO (2)   =   -C.7357588835D 00
THE PRCBIEM WAS CCMPLETEC IN  403 STEPS.
THERE WERE  507 CALLS TO DIFFUN AND
THERE WERE   51 CALLS TO PEDERV.

THESE ARE THE VALUES USED TO CALL EPSODE:

N        =     2
TO       =    0.0
HO       =    C.10C000C000D-09
YO (1)   =    0.1000000000D 01
YO (2)   =    0.1000000000D 01
TOUT     =    C.1CC0CCC000D 01
EPS      =    0.1000000000D-09
IERROR   =     2
MF       =    21
INDEX    =     1
```

Fig. 31. Output from the Program for Example EPISODE-2

```
MF WAS SET TO      21
IERRCR WAS SET TC  2

TOUT    =     0.1000000000D 01
Y0(1)   =     0.1471517766D 01
Y0(2)   =    -C.7357588832D 00
THE PRCBIEM WAS CCMPLETED IN  382 STEPS.
THERE WERE  534 CALLS TO DIFFUN AND
THERE WERE   62 CALLS TO PEDERV.



 THESE ARE THE VALUES USED TO CALL EPSODE:

 N       =      2
 TO      =     0.0
 HO      =     C.10C000C000D-09
 Y0(1)   =     0.1000000000D 01
 Y0(2)   =     0.1000000000D 01
 TOUT    =     C.10C00CC000D 01
 EPS     =     0.1000000000D-09
 IERROR  =      3
 MF      =     21
 INDEX   =      1

 MF WAS SET TO      21
 IERRCR WAS SET TC  3

 TOUT    =     0.1000000000D 01
 Y0(1)   =     0.1471517767D 01
 Y0(2)   =    -C.7357588834D 00
 THE PRCBIEM WAS COMPLETED IN  365 STEPS.
 THERE WERE  492 CALLS TO DIFFUN AND
 THERE WERE   51 CALLS TO PEDERV.

 THIS IS A NCRMAL TERMINATION.

IHC002I STOP      0
```

Fig. 31. (Cont'd)

3. EPISODE-3 Creating a table

The system of differential equations comes from [2, pp. 141-142] and [3, pp. 34-41]. "This test problem was motivated by a study of concentrations of minor chemical species in the earth's atmosphere. Some of these concentrations are governed by photochemical reactions which vary diurnally (with the sunlight present), as a square wave with a 24-hour period." [3, 34].

A one-dimensional model mockup of such a process is given by

$$\text{Solve} \qquad y'(t) = H'(t) - B[y(t) - H(t)]$$

$$\text{subject to} \qquad y(0) = H(0) \text{ for } 0 \leqslant t \leqslant 432{,}000$$

where

$$H(t) = [D + A*E(t)]/B,\ A = 10^{-18},\ B = 10^{8},\ D = 10^{-19}$$

and

$$E(t) = \begin{cases} \exp\,[-C\omega/\sin\,\omega\, t], & \sin\,\omega t > 0 \\ 0, & \sin\,\omega t \leqslant 0 \end{cases}$$

$$C = 4,\ \omega = \pi/43200$$

The solution can be seen to be $y(t) = H(t)$. This solution is represented by what is nearly a square wave of period 86,400 seconds (24 hours), which starts at 0 at $t = 0$, abruptly attains its maximum, holds it for almost 12 hours (corresponding to the twelve daylight hours) and abruptly drops to its minimum, which is held for almost 12 hours (the nighttime). Output will be taken every 43,200 seconds.

ORNL-DWG 78-10806

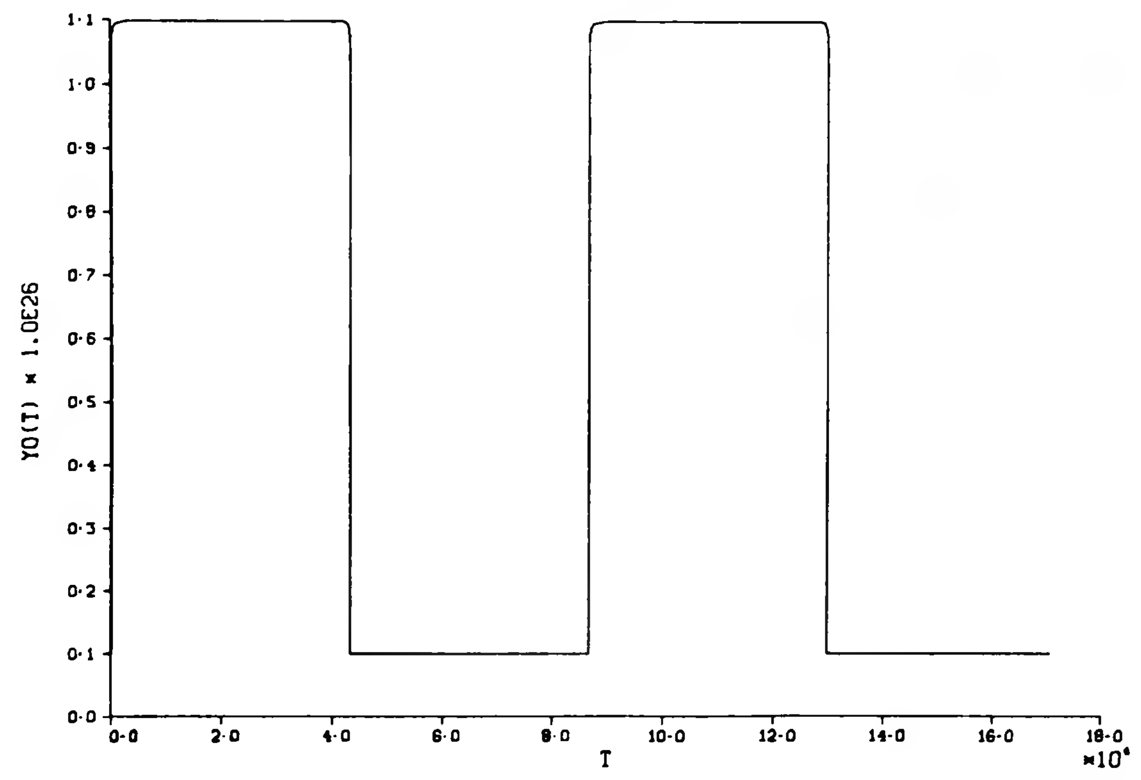


Fig. 32. Solution for Example EPISODE-3

The time constant $\tau = 1/B = 10^{-8}$ is very small in comparison with the length of the interval of integration and the problem is very stiff.

We shall use MF = 23 and EPS = 1.0D - 6 because they worked well in [3]. A subprogram has been written for the function H(t).

The calling program is shown in Fig. 33.

```
      IMPLICIT REAL*8(A-H,O-Z)
      COMMON /EPCOM9/ HUSED, NQUSED, NSTEP, NFE, NJE
      DIMENSION Y0(1)
      EXTERNAL DIFFUN, PEDERV
      N = 1
      Y0(1) = H(0.0D0)
      T0 = 0.0D0
      TOUT = 4.32D4
      H0 = 1.0D-6
      EPS = 1.0D-6
      MF = 23
      IERROR = 3
      INDEX = 1
      WRITE(6,100) N,T0,H0,Y0(1),TOUT,EPS,IERROR,MF,INDEX
      WRITE(6,101) TOUT, Y0(1)
      DO  10  I = 1, 10
        CALL EPSODE(DIFFUN,PEDERV,N,T0,H0,Y0,TOUT,EPS,IERROR,MF,INDEX)
        HT = H(TOUT)
        AE = HT - Y0(1)
        RE = AE/Y0(1)
        WRITE(6,102) TOUT, Y0(1) , HT, AE, RE
        TOUT = TOUT + 4.32D4
 10   CONTINUE
      WRITE(6,103)
100   FORMAT(//,5X,41HTHESE ARE THE VALUES USED TO CALL EPSODE:,//,5X,
     1 8HN       =,I6,/,5X,8HT0      =,D20.10,/,5X,8HH0      =,D20.10,/,5X,
     2 8HY0(1)   =,D20.10,/5X,                          8HTOUT    =,D20.10,/,
     3 5X,8HEPS     =,D20.10,/,5X,8HIERROR =,I6,/,5X,8HMF      =,I6,/,
     4 5X,8HINDEX   =,I6)
101   FORMAT(//,5X,39HA TABLE OF VALUES FOR EXAMPLE EPISODE-3,//,
     1  10X,4HTIME,21X,
     2  4HY(T),21X,4HH(T),16X,8HABSOLUTE,7X,8HRELATIVE,/,10X,
     3  10HIN SECONDS,60X,5HERROR,10X,5HERROR,/,2(5X,D20.10))
102   FORMAT(3(5X,D20.10),2(5X,D10.5))
103   FORMAT(//,5X,29HTHIS IS A NCRMAL TERMINATION.)
      STOP
      STOP
      END
```

Fig. 33. Calling Program for Example EPISODE-3

The subroutine DIFFUN is shown in Fig. 34.

```
      SUBROUTINE DIFFUN(N, T, Y, YDOT)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION Y(1), YDOT(1)
      A = 1.0D-18
      B = 1.0D8
      C = 4.0D0
      D = 1.0D-19
      OMEGA = 3.1415926535D0/4.32D4
      SOT = DSIN(OMEGA*T)
      EDT = 0.0D0
      IF (SOT .GT. 0.0D0) EDT = C*OMEGA*OMEGA*DCOS(OMEGA*T)*DEXP(-C*
     1                          OMEGA/SOT)
      HDT = A*EDT
      YDOT(1) = HDT - B*(Y(1) - H(T))
      RETURN
      END
```

Fig. 34. Subroutine DIFFUN for Example EPISODE-3

The double-precision function H is shown in Fig. 35.

```
      DOUBLE PRECISION FUNCTION H(T)
      IMPLICIT REAL*8(A-H,O-Z)
      A = 1.0D-18
      B = 1.0D8
      C = 4.0D0
      D = 1.0D-19
      OMEGA = 3.1415926535D0/4.32D4
      SOT = DSIN(OMEGA*T)
      ET = 0.0D0
      IF (SOT .GT. 0.0D0) ET = DEXP(-C*OMEGA/SOT)
      H = (D + A*ET)/B
      RETURN
      END
```

Fig. 35. Double-Precision Function H for Example EPISODE-3

The dummy subroutine PEDERV is shown in Fig. 36.

```
SUBROUTINE PEDERV(N, T, Y, PD, NO)
RETURN
END
```

Fig. 36. Subroutine PEDERV for Example EPISODE-3

The output you get is shown in Fig. 37.

```
THESE ARE THE VALUES USED TO CALL EPSODE:

N       =      1
TO      =     0.0
HO      =     0.1000000000D-05
YO(1)   =     0.1000000000D-26
TOUT    =     0.4320000000D 05
EPS     =     0.1000000000D-05
IERROR  =      3
MF      =     23
INDEX   =      1

A TABLE OF VALUES FOR EXAMPLE EPISODE-3

     TIME                       Y(T)                      H(T)                ABSOLUTE      RELATIVE
     IN SECONDS                                                               ERROR         ERROR
    0.4320000000D 05         0.1000000000D-26
    0.4320000000D 05         0.9999999998D-27          0.1000000000D-26      .17878D-36    .17878D-09
    0.8640000000D 05         0.1000001522D-26          0.1000000000D-26      **********    **********
    0.1296000000D 06         0.1000000000D-26          0.1000000000D-26      .40564D-39    .40564D-12
    0.1728000000D 06         0.1000000000D-26          0.1000000000D-26      .0            .0
    0.2160000000D 06         0.1000000013D-26          0.1000000000D-26      **********    **********
    0.2592000000D 06         0.1000000000D-26          0.1000000000D-26      .0            .0
    0.3024000000D 06         0.1000000000D-26          0.1000000000D-26      **********    **********
    0.3456000000D 06         0.1000000000D-26          0.1000000000D-26      .0            .0
    0.3888000000D 06         0.9999999955D-27          0.1000000000D-26      .44511D-35    .44511D-08
    0.4320000000D 06         0.1000000028D-26          0.1000000000D-26      **********    **********

THIS IS A NORMAL TERMINATION.

IHC002I STOP      0
```

Fig. 37. Output from the Program for Example EPISODE-3

VI. REFERENCES

1. G. D. Byrne and A. C. Hindmarsh, "A Polyalgorithm for the Numerical Solution of Ordinary Differential Equations," ACM *Transactions on Mathematical Software* 1, pp. 71-96, 1975.

2. G. D. Byrne, A. C. Hindmarsh, K. R. Jackson, and H. G. Brown, "A Comparison of Two ODE Codes: GEAR and EPISODE," *Computers and Chemical Engineering* Vol. 1, pp. 133-147, 1977.

3. G. D. Byrne, A. C. Hindmarsh, K. R. Jackson, and H. G. Brown, *Comparative Test Results for Two ODE Solvers--EPISODE and GEAR,* Report ANL-77-19, Argonne National Laboratory, Argonne, Illinois (1977).

4. E. Fehlberg, *Low-Order Classical Runge-Kutta Formulas with Stepsize Control,* NASA TR R-315.

5. E. Fehlberg, "Klassiche Runge-Kutta-formeln vierter and niedregerer ordnung mit schrittweitenkontrolle und ihre anwendung auf warmeleitungs-probleme," *Computing* Vol. 6, pp. 61-71, 1970.

6. George E. Forsythe, Michael A. Malcolm, and Cleve B. Moler, *Computer Methods for Mathematical Computations*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1977.

7. C. W. Gear, *Numerical Initial Value Problems in Ordinary Differential Equations*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1971.

8 Marilyn K. Gordon, *Using DEROOT/STEP,INTRP to Solve Ordinary Differential Equations*, Report SAND-76-0211, Sandia Laboratories, (1975).

REFERENCES (Cont'd)

9. A. C. Hindmarsh, *GEAR: Ordinary Differential Equation System Solver*, Report UCID-30001, Rev. 3, Lawrence Livermore Laboratory, Lawrence, California (1974).

10. A. C. Hindmarsh and G. D. Byrne, *EPISODE: An Effective Package for the Integration of Systems of Ordinary Differential Equations*, Report UCID-30112, Rev. 1, Lawrence Livermore Laboratory, Lawrence, California (1974).

11. L. F. Shampine and M. K. Gordon, *Computer Solution of Ordinary Differential Equations, The Initial Value Problem*, W. H. Freeman and Company, San Francisco, California, 1975.

12. L. F. Shampine and M. K. Gordon, *Solving Ordinary Differential Equations with ODE, STEP, and INTRP*, Report SLA-73-1060, Sandia Laboratories (1973).

13. L. F. Shampine and M. K. Gordon, *Using DE/STEP, INTRP to Solve Ordinary Differential Equations*, Abstract 640, Argonne Code Center, Argonne National Laboratory (1976).

14. L. F. Shampine, H. A. Watts, and S. M. Davenport, "Solving Non-Stiff Ordinary Differential Equations-The State of the Art," *SIAM Review* Vol. 18, No. 3, pp. 376-411, July 1976.

ORNL/CSD/TM-64
Distribution Category UC-32

INTERNAL DISTRIBUTION

1-2.	Central Research Library	51.	G. M. Maxwell
3.	Patent Office	52.	G. S. McNeilly
4.	ORNL Technical Library, Document Reference Section	53.	J. K. Munro
		54.	C. W. Nestor
5.	Laboratory Records, ORNL R. C.	55.	D. W. Noid
6-8.	Laboratory Records Department	56.	H. Postma
9.	H. P. Carter/A. A. Brooks/CSD X-10 Library	57.	B. W. Rust
10.	S. J. Chang	58.	G. Sanders
11.	E. L. Compere	59.	C. A. Serbin
12.	J. E. Cope	60.	R. D. Sharp
13.	O. H. Crawford	61.	D. F. Starr
14.	P. M. DiZillo-Benoit	62.	F. E. Stooksbury
15.	C. W. Forsberg	63.	D. J. Strickler
16-41.	R. E. Funderlic	64.	M. L. Tobias
42.	P. W. Gaffney	65.	J. S. Tolliver
43.	D. A. Gardiner	66.	R. Triolo
44.	R. H. Greene	67.	V. R. R. Uppuluri
45.	M. T. Heath	68.	W. I. Van Rij
46.	L. J. Holloway	69.	R. C. Ward
47.	R. A. Just	70.	G. W. Westley
48.	M. E. LaVerne	71.	J. R White
49.	E. Leach	72.	R. M. Wieland
50.	G. E. Liepins	73.	A. Wohlpart

EXTERNAL DISTRIBUTION

74. Professor C. W. Gear, Department of Computer Sciences, University of Illinois, Urbana, IL 61803

75. Professor Gene H. Golub, Department of Computer Science, Stanford University, Stanford, CA 94305

76. Dr. A. Hindmarsh, Lawrence Livermore Laboratory, P.O. Box 808, Livermore, CA 94550

77. Dr. Robert E. Huddleston, Applied Mathematics Division, 8332, Sandia Laboratories, Livermore, CA 94550

78. Dr. James C. T. Pool, Office of Basic Energy Sciences, ER-17, J-309, GTN, U. S. Department of Energy, Washington, D. C. 20545

79. Dr. Lawrence F. Shampine, Numerical Mathematics Division, 5642, Sandia Laboratories, P. O. Box 5800, Albuquerque, NM 87115

80-89. Dr. J. A. Wenzel, Mathematics Department, Albion College, Albion, MI 49224

90. Assistant Manager, office of Energy Research and Development, DOE, Oak Ridge

91-295. Given distribution as shown in TID-4500 under Mathematics and Computers category (25 copies-NTIS)

☆U.S. GOVERNMENT PRINTING OFFICE: 1979-640-079-152