ODEPAKK: AN ORDINARY DIFFERENTIAL EQUATIONS PACKAGE

by

JOHN W. SHELLENBERGER

B. S., Washburn University, 1973
B. S., Kansas State University, 1975

A MASTERS REPORT

Submitted in partial fulfillment of

the requirements for the degree

MASTER OF SCIENCE

Department of Computer Science

KANSAS STATE UNIVERSITY
Manhattan, Kansas
1976

Approved by:

Richard [Signature]
Major Professor
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABSTRACT</td>
<td>1</td>
</tr>
<tr>
<td>INTRODUCTION</td>
<td>2</td>
</tr>
<tr>
<td>CLASS OF PROBLEMS</td>
<td>3</td>
</tr>
<tr>
<td>METHODS USED</td>
<td>5</td>
</tr>
<tr>
<td>INFORMATION REQUIRED IN THE MAIN PROGRAM TO SOLVE THE O.D.E. SYSTEM</td>
<td>8</td>
</tr>
<tr>
<td>ROUTINES TO BE SUPPLIED BY THE USER</td>
<td></td>
</tr>
<tr>
<td>FORMAT OF MAIN PROGRAM</td>
<td>10</td>
</tr>
<tr>
<td>SUBROUTINES TO BE SUPPLIED BY THE USER</td>
<td>11</td>
</tr>
<tr>
<td>AMAT</td>
<td>12</td>
</tr>
<tr>
<td>GFUN</td>
<td>13</td>
</tr>
<tr>
<td>JMAT</td>
<td>14</td>
</tr>
<tr>
<td>INFORMATION RETURNED FROM ODEPAKK</td>
<td>15</td>
</tr>
<tr>
<td>SUMMARY OF ALL PACKAGE ROUTINES</td>
<td>16</td>
</tr>
<tr>
<td>STRUCTURE OF THE ODEPAKK PACKAGE</td>
<td>18</td>
</tr>
<tr>
<td>LIST OF ERROR MESSAGES</td>
<td>19</td>
</tr>
<tr>
<td>NOVICE INFORMATION AND TROUBLESHOOTING</td>
<td>21</td>
</tr>
<tr>
<td>ILLUSTRATIVE EXAMPLES</td>
<td>24</td>
</tr>
<tr>
<td>ACKNOWLEDGMENTS</td>
<td>46</td>
</tr>
<tr>
<td>REFERENCES</td>
<td>47</td>
</tr>
</tbody>
</table>
TABLE OF CONTENTS (continued)

APPENDIX: LISTINGS OF SUBROUTINES ....................... 48

ODEPAK
INTERP
STIFF
COSSET
RES
PMAT
DEC
SOL
DIFFUN
ADDA
ADDB
ADDBB
URND
ODEPAKK: An Ordinary Differential Equations Package

John W. Shellenberger

ABSTRACT

ODEPAKK is a FORTRAN software package for the numerical solution of the initial value problem for systems of ordinary differential equations. The package is a combination and modification of three previously developed ordinary differential equation solvers.

ODEPAKK allows the user to employ either a variable-stepsize, variable-order implicit Adams method or a variable-stepsize, variable-order backward differentiation method in solving a given system of ordinary differential equations.
INTRODUCTION

This report introduces ODEPAKK, a software package for the numerical solution of the initial value problem for systems of ordinary differential equations. The package combines most of the features of the GEAR$^2$, GEARB$^3$, and GEARIB$^4$, packages written by A.C. Hindmarsh, while modifying these somewhat to allow dynamic dimensioning of variables and including several general purpose subroutines to relieve the user of providing equivalent subroutines.

The ODEPAKK package consists of thirteen double precision subroutines written entirely in ANSI Standard FORTRAN. The package is readily convertible to single precision and transportable to any sizable computer with a standard FORTRAN compiler.
CLASS OF PROBLEMS

The ODEPAKK package numerically solves the initial value problem for systems of ordinary differential equations (O.D.E.'s) of the general form:

\[ A(y,t) \cdot \dot{y} = g(y,t) , \quad y(t_0) = y_0 , \quad (1) \]

where \( A \) is an \( N \) by \( N \) matrix, and \( y = (y_1, y_2, y_3, \ldots y_N) \), \( \dot{y} = dy/dt \), and \( g \) are vectors of length \( N \). Given the matrix \( A \), the initial value vector \( y_0 \), and the function \( g(y,t) \), the ODEPAKK package computes a numerical solution to the system at values of the independent variable \( t \). Several forms of Equation (1) are identified within ODEPAKK.

For the matrix \( A(y,t) \) the identity matrix, the resulting form of the ordinary differential equation is:

\[ \dot{y} = g(y,t) , \quad y(t_0) = y_0 . \]

Therefore, depending on the form of the Jacobian matrix,

\[ J = \frac{\partial g}{\partial y} = \left( \frac{\partial g}{\partial y} \right)_{i,j=1}^N , \]

ODEPAKK resembles the GEAR\(^2\) package when the Jacobian matrix is represented in full form as an \( N \) by \( N \) matrix (ICASE = 1), and resembles the GEARB\(^3\) package when the Jacobian matrix is represented in a banded structure (ICASE = 2).

ODEPAKK uses the variable ICASE to specify the form of the system of O.D.E.'s under consideration, with the following values.
indicating:

\[ I\text{CASE } 1, \quad \dot{y} = g(y,t), \text{ full Jacobian matrix;} \]
\[ = 2, \quad \dot{y} = g(y,t), \text{ banded Jacobian matrix;} \]
\[ = 3, \quad A \ast \dot{y} = g(y,t), \text{ full or banded Jacobian matrix.} \]

For a non-identity matrix \( A(y,t) \), the form of the ordinary differential equation to be solved is given by Equation (1). This form of O.D.E. system is referred to as \( I\text{CASE } 3 \). In the original GEARIB\(^4\), the matrix \( A \) and the Jacobian matrix are required to be of banded or nearly banded form. ODEPAKK eases the restrictions placed upon the input form of the matrix \( A \) and the Jacobian matrix (required under certain method options which are described in the next section), with modifications that allow both matrices to be input in either full or banded form, or any combination. For the Jacobian matrix and the matrix \( A \) in banded form, they need not be of the same bandwidth.

As do the GEAR\(^2\), GEARB\(^3\), and GEARIB\(^4\) packages, the ODEPAKK software package has incorporated within it an implicit linear multistep Adams method and an implicit linear multistep backward differentiation method. The generalized Adams method being useful in solving nonstiff problems whereas the generalized backward differentiation formula is useful in solving stiff problems. In the next section these methods are briefly described.
METHODS USED

The ODEPAKK package uses the linear multistep methods of the form:

\[ y_n = \sum_{j=1}^{K_1} \alpha_j y_{n-j} + h \sum_{j=0}^{K_2} \beta_j \dot{y}_{n-j}, \]

where \( y_i \) is an approximation to \( y(t_i) \), \( \dot{y}_i = f(y_i, t_i) \) is an approximation to \( \dot{y}(t_i) \), and \( h \) is a constant step size \( (h = t_{i+1} - t_i) \).

The two types of linear multistep methods available are the implicit Adams methods (to order 12), and the backward differentiation formula methods, or Gear's stiff methods (to order 5). For an order \( q \) Adams method \( K_1 = 1 \) and \( K_2 = q-1 \), and for an order \( q \) backward differentiation formula \( K_1 = q \) and \( K_2 = 0 \). The constants \( \alpha_j \) and \( \beta_j \) are constants associated with the method and for \( \beta_0 \neq 0 \), the above equation for \( y_n \) is in general, an implicit nonlinear algebraic system that must be solved on every step.

To solve this system, three corrector, iteration methods are available: functional iteration, Newton's chord method with an analytic Jacobian matrix, and Newton's chord method with the Jacobian matrix calculated internally by finite differences.

Due to the non-matrix treatment of the functional iteration method, it is less expensive than the other two methods. However, there is a loss in the speed of convergence and in general, a smaller value of \( h \) is required for convergence. Newton's chord method involves a matrix problem, therefore it is more expensive, while
at the same time it converges faster and for larger time step sizes than does the functional iteration method.

To compute the values of \( y_n \), using an integration formula of order \( q \) and step size \( h \), the following tasks are performed in taking a time step. Based on the computed solution values at previous times, an explicit prediction formula is used to estimate the values of \( y_n \). These predicted values are then used as the initial values for starting the nonlinear iteration process. The correction is then performed by using either the functional iteration or Newton's iteration method to solve the nonlinear equations of order \( q \) for the new solution values, \( y_n \). The step size is reduced and the prediction-correction process is terminated if the iteration process fails to converge.

For successfully computed values of \( y_n \), error estimates are calculated. The values of \( y_n \) are allowed if they are less than the error tolerance, EPS. However, if the errors do not conform to the error tolerance, the prediction-correction process is repeated with a different \( h \) and/or a lower order \( q \) until the error test is passed or a decision is made to terminate the attempt at taking the time step.

After successfully taking a time step, or after an unsuccessful attempt at taking a time step, a new step size \( h' \) and formula order \( q' \) is determined. If the last time step was successful \( q' \) may take on the values \( q-1 \), \( q \), or \( q+1 \); if not, \( q' \) may take on the values of \( q-1 \) or \( q \). Based on the integration formulas of orders
q-1, q, and q+1, estimates are made as to the respective maximum
time step sizes that may be used and still pass the error test.
The formula order that will allow the largest time step size is
then used, with that step size, for the next time step attempt.

A more complete and thorough discussion of these methods,
the control of error, the change of order and step size, and
procedure involved in taking a time step may be found in references
1, 2, 3, 5, and 6.
INFORMATION REQUIRED IN THE MAIN PROGRAM TO SOLVE THE O.D.E. SYSTEM

To define the system of ordinary differential equations to be solved, the user must produce three subroutines and a main program. The following is a list and a short description of the variables that must be defined in the main program. A more complete description of the variables is given in the ODEPAKK algorithm in the section describing the input parameters to ODEPAKK.

\[ N \] - the number of first-order ordinary differential equations;
\[ T0 \] - the initial value of the independent variable \( t \);
\[ HO \] - the initial step size to be attempted on the first step;
\[ Y0 \] - a vector of length \( N \), defining the initial values of \( y \) at \( T0 \). This vector must be dimensioned in the main program;
\[ TOUT \] - the next value of \( t \) at which output is desired;
\[ EPS \] - the relative error bound;
\[ MF \] - indicates the basic method to be used (Adams method or backward differentiation method) and the method of iterative solution (functional iteration, Newton's method with analytic Jacobian, or Newton's method with Jacobian calculated by finite differences);
\[ INDEX \] - an indicator of the type of call being made to ODEPAKK (first call for the initialization, continue integration to next value of \( TOUT \), user has reset \( N \) and/or \( EPS \), hit \( TOUT \) exactly, or return output after every step);
ICASE - the form of the O.D.E. to be solved, the allowed values are 1, 2, and 3;

ML, MU - the lower and upper bandwidths, respectively, of the band in the Jacobian matrix. Not required for ICASE = 1, or when functional iteration is requested (see MF). If the Jacobian matrix is full then ML and MU must be input as N-1;

MLA, MUA - the lower and upper bandwidths, respectively, of the band in the matrix A. Required only for ICASE = 3. If the matrix A is full then MLA and MUA must be input as N-1;

WORK - an array used for working storage by the integrator. It should be initially set to zero and the array should be dimensioned to have a size greater than or equal to that described in the ODEPAKK algorithm;

IWORK - an integer array which should normally be zero on input and dimensioned with a length greater than or equal to N+7.

A typical structure for the main program follows:
FORMAT OF MAIN PROGRAM

IMPLICIT REAL*8(A-H,O-Z)
DIMENSION Y0(******), WORK(******), IWORK(******)
COMMON /ODE3/ HUSED, NQUSED, NSTEP, NGE, NJE

The symbols ****** are used to denote numbers which are
specified in the ODEPAKK algorithm under the description of
the input parameters to allocate working storage for the
integrator.

The user must initialize the integrator by setting the
constants N, TO, HO, Y0, TOUT, EPS, MF, INDEX, ICASE, ML, MU,
MLA, and MUA. The user must also set the appropriate initial
conditions of the O.D.E. system into the array Y0.

CALL ODEPAK (N, TO, HO, Y0, TOUT, EPS, MF, ICASE, INDEX, ML,
* MU, MLA, MUA, WORK, IWORK)

The user should check the variable INDEX for errors after
the initial call to the integrator. If no errors have occurred,
the results found in the array Y0 for time t = TOUT should be
output. TOUT may then be reset to the next output time and
then control should be returned to the call to ODEPAK again
to continue the time integration, or the problem may be termin-
ated. In the event an error has occurred, the user must insert
the appropriate logic to handle this situation.

STOP
END
SUBROUTINES TO BE SUPPLIED BY THE USER

Three subroutines; AMAT, GFUN, and JMAT are to be supplied by the user. The primary function of each of these subroutines follows:

1. Subroutine AMAT defines the matrix $A(y,t)$ for a system of O.D.E.'s of the form $ICASE = 3$. For $ICASE = 1$ and 2, AMAT may be replaced by a dummy subroutine.

2. Subroutine GFUN computes the right-hand side, $g(y,t)$, of a system of O.D.E.'s.

3. Subroutine JMAT computes the Jacobian matrix of partial derivatives, $J = \frac{\partial g}{\partial y}$. Used only for $MITER = 1$; for $MITER = 0$ and 2, JMAT may be replaced by a dummy subroutine.

The form of the three user-supplied subroutines AMAT, GFUN, and JMAT follows:
SUBROUTINE AMAT (N, MLA, MUA, T, Y, A)
IMPLICIT REAL*8(A-H, O-Z)
DIMENSION Y(1), A(N, 1)

This routine defines the matrix A(y,t), used when
ICASE = 3 \( (A(y,t) * (dy/dt) = g(y,t)) \).

For ICASE = 3; the user defines matrix A in full form
(as an N by N array) if MLA + MUA = 2 * (N-1); otherwise,
matrix A is defined in banded form (as an N by (MLA + MUA + 1)
array where A(I, J) is stored in location A(I, J-I+MLA+1)).

The user provides, in the main program, the input
variables MLA and MUA.

Matrix A must not be defined as a diagonal matrix
(i.e. MLA = MUA = 0).

For MF = 10 or 20, matrix A must be a constant matrix
with respect to T and Y.

For ICASE = 1 and 2; AMAT may be replaced by a dummy
subroutine.

RETURN
END
SUBROUTINE GFUN (N, T, Y, G)
IMPLICIT REAL*8(A-H,O-Z)
DIMENSION Y(N),G(N)

This routine computes the function G(Y,T), the right-hand side of the ordinary differential equation. Here Y and G(Y,T) are vectors of length N.

GFUN is called in all cases.

RETURN
END
This routine computes the Jacobian matrix of partial derivatives of $G(Y,T) = (G(1), G(2), \ldots, G(N))$.

For ICASE = 1, the Jacobian is stored in PW as an N by N array; where PW(I,J) is set to be the partial derivative of $G(I)$ with respect to $Y(J)$.

For ICASE = 2, the Jacobian is stored as an N by (ML+MU+1) array; where PW(I,J-I+ML+1) is set to be the partial derivative of $G(I)$ with respect to $Y(J)$.

For ICASE = 3, the Jacobian is stored either in full or banded form as just described.

JMAT is called only if MITER = 1, otherwise, JMAT may be replaced by a dummy subroutine.
INFORMATION RETURNED FROM ODEPAKK

After control is returned to the user, the following values are available to the user's main program, from which ODEPAK was called:

HO - the last used step size h, whether the step was successful or not;
YO - the computed values at t = TOUT;
TOUT - the output value of t;
INDEX - an indicator of the success of the integration step.

In addition to the above values available after an integration step, common block ODE3, which the user may access in his main program, contains some statistical information of interest. Within this block are the following values:

HUSED - the step size h last successfully used;
NQUSED - the last successful time integration order used;
NSTEP - the cumulative number of steps taken;
NGE - the cumulative number of CFUN calls;
NJE - the cumulative number of Jacobian evaluations (also LU decompositions).
SUMMARY OF ALL PACKAGE Routines

The following thirteen routines are included in the ODEPAK package. Their calling sequence and basic functions are described below.

ODEPAK (N, TO, HO, YO, TOUT, EPS, MF, ICASE, INDEX, ML, MU, MLA, MUA, WORK, IWORK) -- Allocates storage to save areas, performs initialization of variables and arrays, performs input error checks, makes repeated calls to the core integrator, STIFF, to perform an integration to the next step.

INTERP (TOUT, Y, NO, YO) -- Computes interpolated values of y at the desired output time TOUT.

STIFF (NO, MSIZE, NPW, NA, NIPIV, Y, YMAX, ERROR, SAVE1, SAVE2, SAVE3, PW, A, IPIV) -- Core integrator to perform a single step of the integration using the Adams or GEAR (BDF) methods. Controls local error of a step by choosing appropriate step size and method order.

COSET (METH, NQ, EL, TQ, MAXDER) -- Establishes coefficients for the integration formulas and the error control.

RES (N, T, H, A, Y, V, R) -- Computes the residual vector
R = H * G(Y,T) - A(Y,T) * V, given Y and T, for V a given vector and H the step size.

PMAT (Y, A, NO, CON, MITER, IER, YMAX, ERROR, SAVE1, SAVE2, PW, IPIV) -- For MITER = 1 or 2, sets up the proper Jacobian matrix.
DEC (NO, N, ML, MU, B, IP, IER) -- Performs an LU decomposition of a full or banded matrix.

SOL (NO, N, ML, MU, B, Y, IP) -- For MITER = 1 or 2, perform forward and backward substitution to solve a linear algebraic system given the LU decomposition from DEC.

DIFFUN (N, T, Y, YDOT, A) -- Given Y and T, computes A**-1 * G(Y,T).

Called when ICASE = 3.

ADDA (N, T, Y, P, A, NO, ML, MU) -- Adds the matrix A(Y,T) to the matrix stored in P, with the assistance of routines ADDB and ADDBB.

ADDB (N, ILO, IUP, B, F) -- Adds the full matrix F to the banded matrix B. The resulting sum is placed in matrix F in full form.

ADDBB (N, NML, MLL, MAXNM, KL, KLL, CL, DLL) -- Adds the banded matrices CL and DLL. The resulting sum is placed in matrix DLL in banded form.

URND (U) -- Computes the relative machine precision.
Note: User-supplied routines are indicated by dashed boxes. A downward sloping line from one box to another indicates that the lower routine is called by the upper one.
LIST OF ERROR MESSAGES

ODEPAKK produces appropriate error messages to inform the user of an unnatural occurrence. Following is a list of the possible error messages that may be generated from ODEPAKK. The first message is a response given to the user to inform him as to the minimum dimensions of Y0, WORK, and IWORK to avoid storage overwrites.

1. TO AVOID POSSIBLE STORAGE OVERWRITES, USER MUST, (AT A MINIMUM), DIMENSION.. Y0(***), IWORK(***), AND WORK(***).

2. KFLAG = -1 FROM INTEGRATOR AT T = ________
   ERROR TEST FAILED WITH DABS(H) = HMIN

3. H HAS BEEN REDUCED TO ________ AND STEP WILL BE RETRIED

4. PROBLEM APPEARS UNSOLVABLE WITH GIVEN INPUT

5. KFLAG = -2 FROM INTEGRATOR AT T = ________ H = ________
   THE REQUESTED ERROR IS SMALLER THAN CAN BE HANDLED

6. INTEGRATION HALTED BY ODEPAKK AT T = ________
   EPS TOO SMALL TO BE ATTAINED FOR THE MACHINE PRECISION

7. KFLAG = -3 FROM INTEGRATOR AT T = ________
   CORRECTOR CONVERGENCE COULD NOT BE ACHIEVED

8. ILLEGAL INPUT.. INVALID VALUE OF MF

9. ILLEGAL INPUT.. ML AND MU DO NOT CONFORM WITH N

10. ILLEGAL INPUT.. MLA AND MUA DO NOT CONFORM WITH N

11. ILLEGAL INPUT.. EPS .LE. 0.

12. ILLEGAL INPUT.. N .LE. 0

13. ILLEGAL INPUT.. (TO-TOUT)*H .GE. 0.

14. ILLEGAL INPUT.. INDEX = ______
15. ILLEGAL FORM FOR MATRIX A.. MATRIX A IS A DIAGONAL MATRIX

16. INDEX = -1 ON INPUT WITH (T-TOUT)*H .GE. 0.
   T = ___________ TOUT = ___________ H = ___________
   INTERPOLATION WAS DONE AS ON NORMAL RETURN.
   DESIRED PARAMETER CHANGES WERE NOT MADE.
To effectively use the ODEPAKK package, several things should be considered: the proper choice of the method used (i.e. value of MF), the choice of the initial value of the step size H0, the choice of the local truncation error estimate EPS, and the consequences of the ODEPAKK package failing, returning a negative value of INDEX.

The selection of the method to be used depends on the nature of the problem and any storage restrictions that may be imposed by the environment. Since stiffness is the major factor in choosing a value of MF: if the problem is not stiff, one should choose MF = 10 (Adams methods with functional iteration). If the problem is stiff then a choice of MF = 22 (Gear’s stiff method with the Jacobian calculated internally by finite differences) should be considered.

If the stiffness properties of the problem are unknown, then choosing MF = 10 will give some indication as to whether the problem is a stiff one. For this value of MF; if the values of the step size h are small compared to the change in the solutions per unit interval, then the problem is probably stiff.

Regarding step size, the initial value of H0 should be chosen considerably smaller than the average value expected for the problem. A suggested starting value of H0 would be 1.0E-6. It should be noted that the STIFF routine tests H0, as well as later values of the step size, to see if it passes the error test (based on EPS), and if the step size is too large, it is automatically adjusted.
downward. Note; it is better to choose H0 too small since it will be increased rather rapidly by the algorithm, whereas for a value of H0 too large, the error recovery procedure will be followed.

The error associated with the solution is controlled by the value of the parameter EPS, and is an estimate of the local truncation error, the error committed on taking a single step with the method, starting with data that is regarded as exact. A reasonable starting value of EPS would be 1.0E-4, however since global error is what the user is most interested in, some experimentation may be required to satisfy the user's tolerance on the error.

An execution time of thirty (30) seconds should be attempted for an average sized system ran with a compiled version of ODEPAKK.

Due to the complex nature of the construction of ODEPAKK, it is impossible to enumerate all the possible difficulties the user may encounter in attempting to solve a system of O.D.E.'s using ODEPAKK. The following are some troubleshooting symptoms and possible causes.

1. Integration Will Not Start.
   a. O.D.E.'s do not satisfy the required form.
   b. Inconsistency of the O.D.E.'s.
   c. Incorrect user-supplied subroutine(s); AMAT, GFUN, or JMAT.
   d. Storage overwrites occurring.
   e. Wrong initialization, INDEX ≠ 1.
f. Initial step size, $h_0$, too large.
g. EPS too small for machine word length.

2. Integration Starts, Then The Step Size Approaches Zero.
   a. O.D.E.'s do not satisfy the required form.
   b. Incorrect user-supplied subroutine(s); AMAT, GFUN, or JMAT.
   c. Storage overwrites occurring.

3. Integration Starts But The Step Size Remains Too Small and/or Cycles.
   a. O.D.E.'s do not satisfy the required form.
   b. Incorrect user-supplied Jacobian in subroutine JMAT.
   c. Using METH = 1 option and equations are "stiff".
   d. Using MITER = 0 option and equations are "stiff".
   e. Integration is "stuck" at higher order.

4. Incorrect Solution Values.
   a. O.D.E.'s do not satisfy the required form.
   b. Solving wrong problem -- bad initialization or definition.
   c. Incorrect user-supplied subroutine(s); AMAT, GFUN, or JMAT.
   d. EPS too large.
   e. Storage overwrites occurring.
ILLUSTRATIVE EXAMPLES

The following example problem is provided to illustrate the use of the ODEPAKK package in solving initial value problems. After describing the example problem, the required FORTRAN user-supplied main routine and subroutines are given along with the output generated from the package. All examples provided were produced on an IBM 370/158 machine at Kansas State University using the FORTRAN IV G Level 21 compiler.

One problem, with various configurations, will be used as a test problem. Thereby, the consistency of solutions may be seen between the various configurations.

The basic form of the problem may be described by the following system of first-order ordinary differential equations:

ORBITAL PROBLEM...

\[
\begin{align*}
\dot{y}_1 &= y_2 \\
\dot{y}_2 &= (-16y_1) / (y_1^2 + y_3^2)^{**1.5} \\
\dot{y}_3 &= y_4 \\
\dot{y}_4 &= (-16y_3) / (y_1^2 + y_3^2)^{**1.5}
\end{align*}
\]

where the initial conditions are given by:

\[y_1(0) = 2, \ y_2(0) = 0, \ y_3(0) = 0, \ y_4(0) = 2 \times \text{SQRT}(3)\ .\]

On page 26 is the main program required for the examples given. Note: at statement number 20, and where ICASE, ML, MU, MLA, and MUA are defined, additional comments will be made in each example problem to completely specify these statements.
The dimension of the Y0 vector is N (in these examples, N = 4), the dimension of the IWORK vector is N+7 (in these examples, N+7 = 11), and the dimension of the WORK vector has been determined as prescribed in the ODEPAKK algorithm.
MAIN PROGRAM FOR EXAMPLE PROBLEMS

IMPLICIT REAL*8(A-H,C-Z)
DIMENSION Y0(4), WORK(*), ICN, N
COMMON /ODE3/ HUSED, NQSED, NSTEP, NGE, NJE
DATA LCUT, 0/7

C

C DEMONSTRATION MAIN PROGRAM FOR THE ODEPAK PK PACKAGE.

C

KNT = 0
WRITE(LCUT, 10)
10 FORMAT(23H DEMONSTRATION PROGRAM..., 15X, 18HORBITAL PROBLEM..., //,
*13H Y10CT = Y2, /43H Y200T = (-16 * Y1) / (Y1 * Y1 + Y3 * Y3),
*5H**1.5, /13H Y300T = Y4, /33H Y400T = (-16 * Y3) / (Y1 * Y1 ,
*15H+ Y3 * Y3)**1.5, //, 39H INITIAL VALUES... Y1(0) = 2, Y2(0) = 0 ,
*/18X, 31HY3(0) = 0, Y4(0) = 2 * SQRT(3), //)
WRITE(LCUT, 20)
20 FORMAT(84H)
N = 4
TO = 0.000
HO = 1.00-5
Y0(1) = 2.000
Y0(2) = 0.000
Y0(3) = 0.000
Y0(4) = 2.000 * DSRQRT(3.000)
TOUT = 1.000
EPS = 1.00-5
MF = 21
ICASE = 1
MLA = 1
MLA = 1
MUA = 1
WRITE(LCUT, 30) MF, ICASE, N, TO, HO, EPS, ML, MU, MLA, MUA
30 FORMAT(4H MM =, I3, 15X, 7H ICASE =, I3, 7H, TO = ,
*11H, 5HMLA =, I3, 5X, 5HMLA =, I3, //)

C

CALL CDDEPAK(N, TO, HO, Y0, TCUT, EPS, MF, ICASE, INDEX, ML, MU, 
* MLA, MUA, WCRK, IWORK)

C

IF (KNT .EQ. 0) WRITE(LCUT, 50)
50 FORMAT(5X, 8H, 8H, 2HY1, 5X, 2HY2, 5X, 2HY3, 9X, 2HY4, 6X, 5HCRDER, //)
KNT = 1
WRITE(LCUT, 60) TCLT, YC(1), Y0(2), Y0(3), YC(4), NCUSED
60 FCPMAT(DS.1, 4D1P1L.13, 15)
IF (INDEX .EQ. 0) GO TO 80
WRITE(LCUT, 70) INDEX
70 FORMAT(4H, 26H ERROR RETURN WITH INDEX =, I3, //)
GO TO 90
80 TCLT = TCUT + 1.000
IF (TOLT .LE. 13.000) GC TO 40
90 WRITE(LCUT, 100) NSTEP, NGE, NJE
100 FORMAT(4H, 21H PROBLEM COMPLETE IN, I5, 6H STEPS, //, 21X, I5, 
*14H-G EVALUATIONS, //, 21X, I5, I4H J EVALUATIONS, //)
STOP
END
-26-
EXAMPLE 1

To solve the basic problem (2) with ODEPAKK using the backward differentiation formulas (METH = 2) and the chord iteration method with the user-supplied Jacobian (MTER = 1), MF = 21. Further; for an initial stepsize (H0) of 1.0E-5, a relative error bound (EPS) of 1.0E-5, and the form of the system of O.D.E.'s being ICASE = 1 (full Jacobian), the following insertions must be made in the main program previously defined:

1. The dimension of WORK should be at least 65.
2. Statement number 20 should read:

   20 FORMAT(/,35H JACOBIAN MATRIX FULL, NO MATRIX A.,//)

3. Set ICASE = 1. ML, MU, MLA, and MUA may be omitted since they are not required for ICASE = 1.

The following page contains the three user-supplied subroutines required to solve Example 1. Note that subroutine AMAT is a dummy subroutine since there is no matrix A required in this example problem. For MF = 10, 20, 12, or 22; JMAT may be replaced by a dummy subroutine. If MF = 11, then the user supplied subroutines are identical to those shown on page 28.
EXAMPLE 1 USER-SUPPLIED SUBROUTINES

SUBROUTINE AMAT (N, MLA, MUA, T, Y, A)
IMPLICIT REAL*8(A-F, C-Z)
DIMENSION Y(N), A(N,1)
RETURN
END

SUBROUTINE GFUN (N, T, Y, G)
IMPLICIT REAL*8(A-F, C-Z)
DIMENSION Y(N), G(N)
DENOM = (Y(1) * Y(1) + Y(3) * Y(3))**1.5
G(1) = Y(2)
G(2) = (-16.0D0 * Y(1)) / DENOM
G(3) = Y(4)
G(4) = (-16.0D0 * Y(3)) / DENOM
RETURN
END

SUBROUTINE JMAT (N, ML, MU, T, Y, PW)
IMPLICIT REAL*8(A-F, C-Z)
DIMENSION PW(N,1), Y(N)
A = Y(1) * Y(1) + Y(3) * Y(3)
AA = A**1.5
BB = A**0.5
CC = AA * AA
PW(1,1) = 0.0D0
PW(1,2) = 1.0D0
PW(1,3) = 0.0D0
PW(1,4) = 0.0D0
PW(2,1) = (AA*(-16.0D0) - (16.0D0*Y(1) + 1.5D0*BB*2.0D0*Y(1))) / CC
PW(2,2) = 0.0D0
PW(2,3) = (16.0D0*Y(1) + 1.5D0*BB*2.0D0*Y(1)) / CC
PW(2,4) = 0.0D0
PW(3,1) = 0.0D0
PW(3,2) = 0.0D0
PW(3,3) = 0.0D0
PW(3,4) = 1.0D0
PW(4,1) = (16.0D0*Y(3) + 1.5D0*BB*2.0D0*Y(1)) / CC
PW(4,2) = 0.0D0
PW(4,3) = (AA*(-16.0D0) - (16.0D0*Y(3) + 1.5D0*BB*2.0D0*Y(3))) / CC
PW(4,4) = C.0D0
RETURN
END
EXAMPLE 2

Solve the same problem as in Example 1 except treat the form of the system of O.D.E.'s as ICASE = 2 (banded Jacobian).
The following insertions must be made in the main program previously defined to run this example.

1. The dimension of WORK should be at least 81.
2. Statement number 20 should read:
   
   20  FORMAT(/,37H JACOBIAN MATRIX BANDED, NO MATRIX A.,/)  

3. Set ICASE = 2, ML = 3 and MU = 1. MLA and MUA may be omitted since they are not required for ICASE = 2.

The following page contains the three user-supplied subroutines required to solve Example 2. Note that subroutine AMAT is a dummy subroutine since there is no matrix A required in this example problem. For MF = 10, 20, 12, or 22; JMAT may be replaced by a dummy subroutine. If MF = 11, then the user supplied subroutines are identical to those shown on page 30.
EXAMPLE 2 USER-SUPPLIED SUBROUTINES

SUBROUTINE AMAT (N, MLA, MLA, T, Y, A)
IMPLICIT REAL*8(A-T, C-Z)
DIMENSION Y(N), A(N, 1)
RETURN
END

SUBROUTINE GFUN (N, T, Y, G)
IMPLICIT REAL*8(A-T, C-Z)
DIMENSION Y(N), G(N)
DENOM = (Y(1) * Y(1) + Y(3) * Y(3))**1.5
C(1) = Y(2)
G(2) = (-16.0DO * Y(1)) / DENOM
G(3) = Y(4)
G(4) = (-16.0DO * Y(3)) / DENOM
RETURN
END

SUBROUTINE JMAT (N, M, MU, T, Y, PH)
IMPLICIT REAL*8(A-T, C-Z)
DIMENSION PH(N, 1), Y(N)
A = Y(1) * Y(1) + Y(3) * Y(3)
AA = A**1.5
BB = A**C.5
CC = AA * AA
PW(1, 1) = 0.0DO
PW(1, 2) = C.CDC
PW(1, 3) = 0.0DO
PW(1, 4) = C.CDC
PW(1, 5) = 1.0DO
PW(2, 1) = 0.0DO
PW(2, 2) = C.CDC
PW(2, 3) = (AA*(-16.0DO)-(-16.0DO*Y(1)*1.5DC*BB*2.0DC*Y(1)))/CC
PW(2, 4) = C.CDC
PW(2, 5) = (16.0DC*Y(1)*1.5DC*BB*2.0DC*Y(1))/CC
PW(3, 1) = 0.0DO
PW(3, 2) = 0.0DC
PW(3, 3) = 0.0DC
PW(3, 4) = 0.0DO
PW(3, 5) = 1.0DC
PW(4, 1) = (16.0DC*Y(3)*1.5DC*BB*2.0DC*Y(1))/CC
PW(4, 2) = C.CDC
PW(4, 3) = (AA*(-16.0DC)-(-16.0DC*Y(3)*1.5DC*BB*2.0DC*Y(3)))/CC
PW(4, 4) = 0.0DO
PW(4, 5) = 0.0DO
RETURN
END
EXAMPLE 3

Example problem 3 solves the general example problem (2) when the form of the system is \( A(y,t) \times (dy/dt) = g(y,t) \). In order to compare results produced with other example problems, first multiply the right-hand side of the system of O.D.E.'s by the same matrix \( A(y,t) \). Therefore, the actual form of the system being solved is:

\[
A(y,t) \times (dy/dt) = A(y,t) \times g(y,t) ,
\]

(3)

and the solution produced should agree favorably with the other example problems.

This same manipulation of the system will be employed for the remainder of the example problems. This example, and the remaining, will exploit the various forms in which the matrix \( A \) and the Jacobian matrix may be specified (either of full or of banded form) and combined in the system under consideration.

The matrix \( A \) considered in the remaining examples is of the form:

\[
A = \begin{bmatrix}
1 & 2 & 0 & 0 \\
1 & 3 & 1 & 0 \\
0 & 3 & 2 & 3 \\
0 & 0 & 2 & 1
\end{bmatrix}
\]

In this example, ICASE = 3, the Jacobian of the right-hand side of Equation (3) and the matrix \( A \) will be considered as if
they were in full form.

The following insertions must be made in the main program previously defined to run this example:

1. The dimension of WORK should be at least 80.
2. Statement 20 should read:
   
   20 FORMAT(/, 37H JACOBIAN MATRIX FULL, MATRIX A FULL., //)
   

The following page contains the three user-supplied subroutines required to solve Example 3. For MF = 10, 20, 12, or 22; JMAT may be replaced by a dummy subroutine. If MF = 11, then the user supplied subroutines are identical to those on page 33.
EXAMPLE 3 USER-SUPPLIED SUBROUTINES

SUBROUTINE AMAT (N, ML, MUA, T, Y, A)
IMPLICIT REAL*8(A-H,C-Z)
DIMENSION Y(1), A(N,1)
DO 10 I = 1, N
   DO 10 J = 1, 4
      A(I, J) = 0.000
   10 A(1,1) = 1.000
      A(2,1) = 1.000
      A(1,2) = 2.000
      A(2,2) = 3.000
      A(3,2) = 3.000
      A(2,3) = 1.000
      A(1,3) = 2.000
      A(4,3) = 2.000
      A(3,4) = 3.000
      A(4,4) = 1.000
RETURN
END

SUBROUTINE GFUN (N, T, Y, G)
IMPLICIT REAL*8(A-H,C-Z)
DIMENSION Y(N), G(N)
AA = (Y(1) * Y(1) + Y(3) * Y(3))**1.5
BB = Y(2) + Y(4)
CC = Y(1) + Y(3)
G(1) = Y(2) - (32.000 * Y(1) / AA)
G(2) = BB - (48.000 * Y(1) / AA)
G(3) = 2.000 * Y(4) - ((48.000 / AA) * CC)
G(4) = 2.000 * Y(4) - (16.000 * Y(3) / AA)
RETURN
END

SUBROUTINE JMAT (N, ML, MUA, T, Y, P)
IMPLICIT REAL*8(A-H,C-Z)
DIMENSION P(N,1), Y(N)
AA = (Y(1) * Y(1) + Y(3) * Y(3))**0.5
BB = (Y(1) * Y(1) + Y(3) * Y(3))**0.5
CC = AA * AA
PN(1,1) = (AA*(-32.000)+32.000*Y(1)*1.500*BB*2.000*Y(1))/CC
PN(1,2) = 1.000
PN(1,3) = 32.000*Y(1)*1.500*BB*2.000*Y(3)/CC
PN(1,4) = 0.000
PN(2,1) = (AA*(-48.000)+48.000*Y(1)*1.500*BB*2.000*Y(1))/CC
PN(2,2) = 1.000
PN(2,3) = 48.000*Y(1)*1.500*BB*2.000*Y(3)/CC
PN(2,4) = 1.000
PN(3,1) = (AA*(-48.000)+48.000*Y(1)*1.500*BB*2.000*Y(1))/CC+
* (48.000*Y(3)*1.500*BB*2.000*Y(1))/CC
PN(3,2) = 0.000
PN(3,3) = (48.000*Y(3)*1.500*BB*2.000*Y(3))/CC+(AA*(-48.000)+
*48.000*Y(3)*1.500*BB*2.000*Y(3))/CC
PN(3,4) = 2.000
PN(4,1) = (16.000*Y(3)*1.500*BB*2.000*Y(1))/CC
PN(4,2) = 0.000
PN(4,3) = (AA*(-16.000)+16.000*Y(3)*1.500*BB*2.000*Y(3))/CC
PN(4,4) = 2.000
RETURN
END
EXAMPLE 4

Referring to the modified problem (3); this example problem employs the Jacobian matrix in banded form while the matrix A is treated in full form.

The following insertions must be made in the main program previously defined to run this example:

1. The dimension of WORK should be at least 88.
2. Statement 20 should read:
   
   20 FORMAT(/,39H JACOBIAN MATRIX BANDED, MATRIX A FULL,,//)


The following page contains the three user-supplied subroutines required to solve Example 4. For MF = 10, 20, 12, or 22; JMAT may be replaced by a dummy subroutine. If MF = 11, then the user supplied subroutines are identical to those on page 35.
ILLEGIBLE

THE FOLLOWING DOCUMENT (S) IS ILLEGIBLE DUE TO THE PRINTING ON THE ORIGINAL BEING CUT OFF

ILLEGIBLE
EXAMPLE 4 USER-SUPPLIED SUBROUTINES

SUBROUTINE AMAT (N, MLA, MUA, T, Y, A)
IMPLICIT REAL*8(A-H,C-Z)
DIMENSION Y(1), A(N,1)
CO 10 I = 1,4
DO 10 J = 1,4

10 A(I,J) = C.000
A(1,1) = 1.000
A(2,1) = 1.000
A(1,2) = 2.000
A(2,2) = 3.000
A(3,2) = 3.000
A(2,3) = 1.000
A(3,3) = 2.000
A(4,3) = 2.000
A(3,4) = 3.000
A(4,4) = 1.000
RETURN
END

SUBROUTINE GFUN (N, T, Y, G)
IMPLICIT REAL*8(A-H,C-Z)
DIMENSION Y(N), G(N)
AA = (Y(1) * Y(1) + Y(3) * Y(3))^1.5
BB = Y(2) + Y(4)
CC = Y(1) + Y(3)
G(1) = Y(2) - (32.000 * Y(1) / AA)
G(2) = BB - (48.000 * Y(1) / AA)
G(3) = 2.000 * Y(4) - ((48.000 / AA) * CC)
G(4) = 2.000 * Y(4) - (16.000 * Y(3) / AA)
RETURN
END

SUBROUTINE JMAT (N, MLA, MUA, T, Y, PW)
IMPLICIT REAL*8(A-H,C-Z)
DIMENSION PW(N,1), Y(N)
AA = (Y(1) * Y(1) + Y(3) * Y(3))^0.5
BB = (Y(1) * Y(1) + Y(3) * Y(3))^0.5
CC = AA * AA
CC 10 J = 1,6
DO 10 I = 1,4

10 PW(I,J) = 0.000
PW(1,4) = (AA*(-32.000) + 32.000 * Y(1) * 1.5*C0*BB*2.000 * Y(1)) / CC
PW(1,5) = 1.000
PW(1,6) = 32.000 * Y(1) * 1.5*C0*BB*2.000 * Y(3) / CC
PW(2,3) = (AA*(-48.000) + 48.000 * Y(1) * 1.5*C0*BB*2.000 * Y(1)) / CC
PW(2,4) = 1.000
PW(2,5) = (48.000 * Y(1) * 1.5*C0*BB*2.000 * Y(3)) / CC
PW(2,6) = 1.000
PW(3,4) = (AA*(-48.000) + 48.000 * Y(1) * 1.5*C0*BB*2.000 * Y(1)) / CC + 
* (48.000 * Y(3) * 1.5*C0*BB*2.000 * Y(3)) / CC
PW(3,5) = 0.000
PW(3,6) = (48.000 * Y(1) * 1.5*C0*BB*2.000 * Y(3)) / CC + (AA*(-48.000) + 
* 48.000 * Y(3) * 1.5*C0*BB*2.000 * Y(3)) / CC
PW(4,5) = 2.000
PW(4,6) = (16.000 * Y(3) * 1.5*C0*BB*2.000 * Y(1)) / CC
PW(4,7) = 0.000
PW(4,8) = (AA*(-16.000) + 16.000 * Y(3) * 1.5*C0*BB*2.000 * Y(3)) / CC
PW(4,9) = 2.000
RETURN
END
EXAMPLE 5

Referring to the modified problem (3), this example problem employs the Jacobian matrix in full form while the matrix $A$ is treated in banded form.

The following insertions must be made in the main program previously defined to run this example:

1. The dimension of WORK should be at least 76.
2. Statement 20 should read:
   \begin{verbatim}
   20 FORMAT(/'JACOBIAN MATRIX FULL, MATRIX A BANDED./)
   \end{verbatim}
3. Set ICASE = 3, ML = 3, MU = 3, MLA = 1, and MUA = 1.

The following page contains the three user-supplied subroutines required to solve Example 5. For $MF = 10, 20, 12,$ or $22$; JMAT may be replaced by a dummy subroutine. If $MF = 11$, then the user supplied subroutines are identical to those on page 37.
EXAMPLE 6 USER-SUPPLIED SUBROUTINES

SUBROUTINE AMAT (N, ML, MUA, T, Y, A)
IMPLICIT REAL*8(A-H,C-Z)
DIMENSION Y(1),A(N,1)
A(1,1) = 0.000
A(2,1) = 1.000
A(3,1) = 3.000
A(4,1) = 2.000
A(1,2) = 1.000
A(2,2) = 3.000
A(3,2) = 2.000
A(4,2) = 1.000
A(1,3) = 2.000
A(2,3) = 1.000
A(3,3) = 3.000
A(4,3) = 0.000
RETURN
END

SUBROUTINE GFUN (N, T, Y, G)
IMPLICIT REAL*8(A-H,C-Z)
DIMENSION Y(N),G(N)
AA = (Y(1) * Y(1) + Y(3) * Y(3))**1.5
BB = Y(2) + Y(4)
CC = Y(1) + Y(3)
G(1) = Y(2) - (32.000 * Y(1) / AA)
G(2) = BB - (48.000 * Y(1) / AA)
G(3) = 2.000 * Y(4) - ((48.000 / AA) * CC)
G(4) = 2.000 * Y(4) - (16.000 * Y(3) / AA)
RETURN
END

SUBROUTINE JMAT (N, ML, MUA, T, Y, Ph)
IMPLICIT REAL*8(A-H,C-Z)
DIMENSION Ph(N,1),Y(N)
AA = (Y(1) * Y(1) + Y(3) * Y(3))**1.5
BB = Y(1) + Y(3)
CC = AA * AA
Ph(1,1) = (AA*(-32.000)+32.000*y(1)*1.500*BB*2.000*y(1))/CC
Ph(1,2) = 1.000
Ph(1,3) = 32.000*y(1)*1.500*BB*2.000*y(3)/CC
Ph(1,4) = 0.000
Ph(2,1) = (AA*(-48.000)+48.000*y(1)*1.500*BB*2.000*y(1))/CC
Ph(2,2) = 1.000
Ph(2,3) = (48.000*y(1)*1.500*BB*2.000*y(3))/CC
Ph(2,4) = 1.000
Ph(3,1) = (AA*(-48.000)+48.000*y(1)*1.500*BB*2.000*y(1))/CC*
(48.000*y(3)*1.500*BB*2.000*y(3))/CC+(AA*(-48.000)+
*48.000*y(3)*1.500*BB*2.000*y(3))/CC
Ph(3,2) = 0.000
Ph(3,3) = (48.000*y(1)*1.500*BB*2.000*y(3))/CC+(AA*(-48.000)+
*48.000*y(3)*1.500*BB*2.000*y(3))/CC
Ph(3,4) = 2.000
Ph(4,1) = (16.000*y(3)*1.500*BB*2.000*y(1))/CC
Ph(4,2) = 0.000
Ph(4,3) = (AA*(-16.000)+16.000*y(3)*1.500*BB*2.000*y(3))/CC
Ph(4,4) = 2.000
RETURN
END

-37-
EXAMPLE 6

Referring to the modified problem (3), this example problem considers both the Jacobian matrix and the matrix A as being of banded form.

The following insertions must be made in the main program previously defined to run this example:

1. The dimension of WORK should be at least 108.
2. Statement 20 should read:
   
   20 FORMAT(//,41H JACOBIAN MATRIX BANDED, MATRIX A BANDED.,//)


The following page contains the three user-supplied subroutines required to solve Example 6. For MF = 10, 20, 12, or 22; JMAT may be replaced by a dummy subroutine. If MF = 11, then the user supplied subroutines are identical to those on page 39.
EXAMPLE 6 USER-SUPPLIED SUBROUTINES

SUBROUTINE AMAT (N, MLA, MUA, T, Y, A)
  IMPLICIT REAL*8(A-H,C-Z)
  DIMENSION Y(1), A(N,1)
  A(1,1) = 0.000
  A(2,1) = 1.000
  A(3,1) = 2.000
  A(4,1) = 3.000
  A(5,1) = 4.000
  A(2,2) = 1.000
  A(3,2) = 2.000
  A(4,2) = 3.000
  A(5,2) = 4.000
  A(3,3) = 1.000
  A(4,3) = 2.000
  A(5,3) = 3.000
  A(4,4) = 1.000
  A(5,4) = 2.000
  A(5,5) = 3.000
  RETURN
END

SUBROUTINE GFUN (N, T, Y, G)
  IMPLICIT REAL*8(A-H,C-Z)
  DIMENSION Y(N), G(N)
  AA = (Y(1) * Y(1) + Y(3) * Y(3))**1.5
  BB = Y(2) + Y(4)
  CC = Y(1) + Y(3)
  G(1) = Y(2) - (32.000 * Y(1) / AA)
  G(2) = BB - (48.000 * Y(1) / AA)
  G(3) = 2.000 * Y(4) - ((48.000 / AA) * CC)
  G(4) = 2.000 * Y(4) - (16.000 * Y(3) / AA)
  RETURN
END

SUBROUTINE JMAT (N, ML, MU, T, Y, PW)
  IMPLICIT REAL*8(A-H,C-Z)
  DIMENSION PW(N,1), Y(N)
  AA = (Y(1) * Y(1) + Y(3) * Y(3))**1.5
  BB = Y(1) + Y(3)
  CC = AA * BB
  DO 10 J = 1, N
    DO 10 I = 1, N
      PW(I,J) = 0.000
      PW(I,1) = (AA*(-32.000) + 32.000*Y(1)*Y(1) + 1.500*BB*2.000*Y(1))/CC
      PW(I,2) = 32.000*Y(1)*Y(1) + 1.500*BB*2.000*Y(3)/CC
      PW(I,3) = (AA*(-48.000) + 48.000*Y(1)*Y(1) + 1.500*BB*2.000*Y(1))/CC
      PW(I,4) = 48.000*Y(1)*Y(1) + 1.500*BB*2.000*Y(3)/CC
      PW(I,5) = (AA*(-48.000) + 48.000*Y(1)*Y(1) + 1.500*BB*2.000*Y(1))/CC
      PW(I,6) = (AA*(-48.000) + 48.000*Y(1)*Y(1) + 1.500*BB*2.000*Y(3))/CC
      PW(I,7) = (AA*(-48.000) + 48.000*Y(1)*Y(1) + 1.500*BB*2.000*Y(1))/CC
      PW(I,8) = (AA*(-48.000) + 48.000*Y(1)*Y(1) + 1.500*BB*2.000*Y(3))/CC
      PW(I,9) = (AA*(-48.000) + 48.000*Y(1)*Y(1) + 1.500*BB*2.000*Y(1))/CC
      PW(I,10) = (AA*(-48.000) + 48.000*Y(1)*Y(1) + 1.500*BB*2.000*Y(3))/CC
    10 CONTINUE
  RETURN
END
OUTPUT PRODUCED FROM EXAMPLE PROBLEM 1

DEMONSTRATION PROGRAM.

CRBITAL PROBLEM...

\[ \begin{align*}
Y1DCT &= Y2 \\
Y2DCT &= (-16 \times Y1) / (Y1 \times Y1 + Y3 \times Y3)^{1.5} \\
Y3DCT &= Y4 \\
Y4DCT &= (-16 \times Y3) / (Y1 \times Y1 + Y3 \times Y3)^{1.5}
\end{align*} \]

INITIAL VALUES: \( Y1(0) = 2, \quad Y2(0) = 0, \)
\( Y3(0) = 0, \quad Y4(0) = 2 \times \text{SQRT}(3) \)

JACOBIAN MATRIX FULL, NO MATRIX A.

\[
\begin{align*}
MF &= 21 \\
ICASE &= 1 \\
N &= 4 \\
T0 &= 0.0 \\
HO &= 0.10D-04 \\
EPS &= 0.10D-04 \\
ML &= 0 \\
MU &= 0 \\
MLA &= 0 \\
MLA &= 0
\end{align*}
\]

TO AVOID POSSIBLE STORAGE OVERWRITES, USER MUST (AT A MINIMUM)
DIMENSION: \( Y0( \quad 4), \text{WORK( 65), AND IWORK( 11)}. \)

<table>
<thead>
<tr>
<th>( T )</th>
<th>( Y1 )</th>
<th>( Y2 )</th>
<th>( Y3 )</th>
<th>( Y4 )</th>
<th>ORDER</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.00D 00</td>
<td>5.243D-01</td>
<td>-2.267D 00</td>
<td>2.687D 00</td>
<td>1.557D 00</td>
<td>4</td>
</tr>
<tr>
<td>2.00D 00</td>
<td>-1.712D 00</td>
<td>-2.069D 00</td>
<td>3.455D 00</td>
<td>1.296D-01</td>
<td>5</td>
</tr>
<tr>
<td>3.00D 00</td>
<td>-3.526D 00</td>
<td>-1.553D 00</td>
<td>3.202D 00</td>
<td>-5.547D-01</td>
<td>5</td>
</tr>
<tr>
<td>4.00D 00</td>
<td>-4.823D 00</td>
<td>-1.047D 00</td>
<td>2.455D 00</td>
<td>-9.034D-01</td>
<td>5</td>
</tr>
<tr>
<td>5.00D 00</td>
<td>-5.632D 00</td>
<td>-5.761D-01</td>
<td>1.452D 00</td>
<td>-1.082D 00</td>
<td>5</td>
</tr>
<tr>
<td>6.00D 00</td>
<td>-5.982D 00</td>
<td>-1.259D-01</td>
<td>3.274D-01</td>
<td>-1.151D 00</td>
<td>5</td>
</tr>
<tr>
<td>7.00D 00</td>
<td>-5.885D 00</td>
<td>3.197D-01</td>
<td>-8.217D-01</td>
<td>-1.132D 00</td>
<td>5</td>
</tr>
<tr>
<td>8.00D 00</td>
<td>-5.339D 00</td>
<td>7.771D-01</td>
<td>-1.507D 00</td>
<td>-1.020D 00</td>
<td>5</td>
</tr>
<tr>
<td>9.00D 00</td>
<td>-4.322D 00</td>
<td>1.262D 00</td>
<td>-2.819D 00</td>
<td>-7.753D-01</td>
<td>5</td>
</tr>
<tr>
<td>1.00D 01</td>
<td>-2.802D 00</td>
<td>1.781D 00</td>
<td>-3.392D 00</td>
<td>-3.175D-01</td>
<td>5</td>
</tr>
<tr>
<td>1.10D 01</td>
<td>-7.717D-01</td>
<td>2.249D 00</td>
<td>-3.295D 00</td>
<td>6.289D-01</td>
<td>5</td>
</tr>
<tr>
<td>1.20D 01</td>
<td>1.431D 00</td>
<td>1.799D 00</td>
<td>-1.777D 00</td>
<td>2.605D 00</td>
<td>5</td>
</tr>
<tr>
<td>1.30D 01</td>
<td>1.645D 00</td>
<td>-1.512D 00</td>
<td>1.423D 00</td>
<td>2.903D 00</td>
<td>5</td>
</tr>
</tbody>
</table>

PROBLEM COMPLETED IN 103 STEPS
168 G EVALUATIONS
12 J EVALUATIONS

-40-
OUTPUT PRODUCED FROM EXAMPLE PROBLEM 2

DEMONSTRATION PROGRAM. CRBITAL PROBLEM...

\[
\begin{align*}
Y1DCT &= Y2 \\
Y2DOT &= (-16*Y1)/(Y1*Y1 + Y3*Y3)\times1.5 \\
Y3DOT &= Y4 \\
Y4DCT &= (-16*Y3)/(Y1*Y1 + Y3*Y3)\times1.5
\end{align*}
\]

INITIAL VALUES. \(Y1(0) = 2,\ Y2(0) = 0,\ Y3(0) = 0,\ Y4(0) = 2*SQRT(3)\)

JACOBIAN MATRIX EENDED, NO MATRIX A.

\[
\begin{align*}
MF &= 21 \\
ICASE &= 2 \\
N &= 4 \\
T0 &= 0.0 \\
HO &= 0.10E-04 \\
EPS &= 0.10E-04 \\
ML &= 3 \\
MU &= 1 \\
MLA &= 0 \\
MUA &= 0
\end{align*}
\]

TO AVOID POSSIBLE STORAGE OVERWITRES, USER MUST (AT A MINIMUM) DIMENSION. \(Y0(4), WORK(81), AND IWORK(11)\).

<table>
<thead>
<tr>
<th>T</th>
<th>Y1</th>
<th>Y2</th>
<th>Y3</th>
<th>Y4</th>
<th>CRDER</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.00D 00</td>
<td>5.243D-01</td>
<td>-2.267D 00</td>
<td>2.687D 00</td>
<td>1.597D 00</td>
<td>4</td>
</tr>
<tr>
<td>2.00D 00</td>
<td>-1.712D 00</td>
<td>-2.669D 00</td>
<td>3.455D 00</td>
<td>1.256D-01</td>
<td>5</td>
</tr>
<tr>
<td>3.00D 00</td>
<td>-3.526D 00</td>
<td>-1.553D 00</td>
<td>3.202D 00</td>
<td>-5.547D-01</td>
<td>5</td>
</tr>
<tr>
<td>4.00D 00</td>
<td>-4.823D 00</td>
<td>-1.047D 00</td>
<td>2.455D 00</td>
<td>-9.034D-01</td>
<td>5</td>
</tr>
<tr>
<td>5.00D 00</td>
<td>-5.632D 00</td>
<td>-5.761D-01</td>
<td>1.452D 00</td>
<td>-1.082D 00</td>
<td>5</td>
</tr>
<tr>
<td>6.00D 00</td>
<td>-5.982D 00</td>
<td>-1.259D-01</td>
<td>3.274D-01</td>
<td>-1.151D 00</td>
<td>5</td>
</tr>
<tr>
<td>7.00D 00</td>
<td>-5.856D 00</td>
<td>3.197D-01</td>
<td>8.217D-01</td>
<td>-1.132D 00</td>
<td>5</td>
</tr>
<tr>
<td>8.00D 00</td>
<td>-5.339D 00</td>
<td>7.771D-01</td>
<td>-1.907D 00</td>
<td>-1.020D 00</td>
<td>5</td>
</tr>
<tr>
<td>9.00D 00</td>
<td>-4.322D 00</td>
<td>1.262D 00</td>
<td>-2.819D 00</td>
<td>-7.793D-01</td>
<td>5</td>
</tr>
<tr>
<td>1.00D 01</td>
<td>-2.802D 00</td>
<td>1.781D 00</td>
<td>-3.392D 00</td>
<td>-3.157D-01</td>
<td>5</td>
</tr>
<tr>
<td>1.10D 01</td>
<td>-7.717D-01</td>
<td>2.249D 00</td>
<td>-3.295D 00</td>
<td>6.289D-01</td>
<td>5</td>
</tr>
<tr>
<td>1.20D 01</td>
<td>1.431D 00</td>
<td>1.799D 00</td>
<td>-1.777D 00</td>
<td>2.605D 00</td>
<td>5</td>
</tr>
<tr>
<td>1.30D 01</td>
<td>1.645D 00</td>
<td>-1.512D 00</td>
<td>1.423D 00</td>
<td>2.903D 00</td>
<td>5</td>
</tr>
</tbody>
</table>

PROBLEM COMPLETED IN 103 STEPS
168 G EVALUATIONS
12 J EVALUATIONS

-41-
OUTPUT PRODUCED FROM EXAMPLE PROBLEM 3

DEMONSTRATION PROGRAM, CRITICAL PROBLEM...

\[
\begin{align*}
Y1DOT &= Y2 \\
Y2DOT &= (-16 \times Y1) / (Y1 \times Y1 + Y3 \times Y3)^{1.5} \\
Y3DOT &= Y4 \\
Y4DOT &= (-16 \times Y3) / (Y1 \times Y1 + Y3 \times Y3)^{1.5}
\end{align*}
\]

INITIAL VALUES: \(Y1(0) = 2, \ Y2(0) = 0, \ Y3(0) = 0, \ Y4(0) = 2 \times \sqrt{3}\)

JACOBIAN MATRIX FULL, MATRIX A FULL.

\[
\begin{align*}
MF &= 21, & ICASE &= 3 \\
N &= 4, & T0 &= 0.0, & H0 &= 0.10D-04, & EPS &= 0.10D-04 \\
ML &= 3, & MU &= 3, & MLA &= 3, & NUA &= 3
\end{align*}
\]

TO AVOID POSSIBLE STORAGE OVERWITRES, USER MUST (AT A MINIMUM) DIMENSION: \(Y0(4), \ \text{WORK}(80), \ \text{AND IWORK}(11)\).

<table>
<thead>
<tr>
<th>T</th>
<th>Y1</th>
<th>Y2</th>
<th>Y3</th>
<th>Y4</th>
<th>CRDER</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.00D 00</td>
<td>5.243D-01</td>
<td>-2.267D 00</td>
<td>2.687D 00</td>
<td>1.597D 00</td>
<td>4</td>
</tr>
<tr>
<td>2.00D 00</td>
<td>-1.712D 00</td>
<td>-2.069D 00</td>
<td>3.455D 00</td>
<td>1.296D-01</td>
<td>5</td>
</tr>
<tr>
<td>3.00D 00</td>
<td>-3.526D 00</td>
<td>-1.553D 00</td>
<td>3.202D 00</td>
<td>-5.547D-01</td>
<td>5</td>
</tr>
<tr>
<td>4.00D 00</td>
<td>-4.823D 00</td>
<td>-1.047D 00</td>
<td>2.455D 00</td>
<td>-9.034D-01</td>
<td>5</td>
</tr>
<tr>
<td>5.00D 00</td>
<td>-5.632D 00</td>
<td>-5.761D-01</td>
<td>1.452D 00</td>
<td>-1.062D 00</td>
<td>5</td>
</tr>
<tr>
<td>6.00D 00</td>
<td>-5.582D 00</td>
<td>-1.259D-01</td>
<td>3.274D-01</td>
<td>-1.151D 00</td>
<td>5</td>
</tr>
<tr>
<td>7.00D 00</td>
<td>-5.885D 00</td>
<td>3.197D-01</td>
<td>-8.217D-01</td>
<td>-1.132D 00</td>
<td>5</td>
</tr>
<tr>
<td>8.00D 00</td>
<td>-5.339D 00</td>
<td>7.771D-01</td>
<td>-1.907D 00</td>
<td>-1.020D 00</td>
<td>5</td>
</tr>
<tr>
<td>9.00D 00</td>
<td>-4.322D 00</td>
<td>1.262D 00</td>
<td>-2.819D 00</td>
<td>-7.793D-01</td>
<td>5</td>
</tr>
<tr>
<td>1.00D 01</td>
<td>-2.802D 00</td>
<td>1.781D 00</td>
<td>-3.392D 00</td>
<td>-3.157D-01</td>
<td>5</td>
</tr>
<tr>
<td>1.10D 01</td>
<td>-7.717D-01</td>
<td>2.249D 00</td>
<td>-3.295D 00</td>
<td>6.289D-01</td>
<td>5</td>
</tr>
<tr>
<td>1.20D 01</td>
<td>1.431D 00</td>
<td>1.799D 00</td>
<td>-1.777D 00</td>
<td>2.650D 00</td>
<td>5</td>
</tr>
<tr>
<td>1.30D 01</td>
<td>1.645D 00</td>
<td>-1.512D 00</td>
<td>1.423D 00</td>
<td>2.903D 00</td>
<td>5</td>
</tr>
</tbody>
</table>

PROBLEM COMPLETED IN 103 STEPS
168 G EVALUATIONS
12 J EVALUATIONS

-42-
OUTPUT PRODUCED FROM EXAMPLE PROBLEM 4

DEMONSTRATION PROGRAM

\[ Y_1 \text{DOT} = Y_2 \]
\[ Y_2 \text{DOT} = (-16 \times Y_1) / (Y_1 \times Y_1 + Y_3 \times Y_3)^{1.5} \]
\[ Y_3 \text{DOT} = Y_4 \]
\[ Y_4 \text{DOT} = (-16 \times Y_3) / (Y_1 \times Y_1 + Y_3 \times Y_3)^{1.5} \]

INITIAL VALUES: \( Y_1(0) = 2 \), \( Y_2(0) = 0 \), \( Y_3(0) = 0 \), \( Y_4(0) = 2 \times \text{SQRT}(3) \)

JACOBIAN MATRIX BANDED, MATRIX A FULL.

\( MF = 21 \)
\( ICASE = 3 \)

\( N = 4 \)
\( TO = 0.0 \)
\( HO = 0.10 \times 10^{-4} \)
\( EPS = 0.10 \times 10^{-4} \)

\( ML = 3 \)
\( MU = 2 \)
\( MLA = 3 \)
\( MUA = 3 \)

TO AVOID POSSIBLE STORAGE OVERWRITES, USER MUST (AT A MINIMUM) DIMENSION... \( Y(4) \), \( WORK(88) \), AND \( IWCRK(11) \).

<table>
<thead>
<tr>
<th>( T )</th>
<th>( Y_1 )</th>
<th>( Y_2 )</th>
<th>( Y_3 )</th>
<th>( Y_4 )</th>
<th>CRDER</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.00D 00</td>
<td>5.243E-01</td>
<td>-2.267D 00</td>
<td>2.687D 00</td>
<td>1.557D 00</td>
<td>4</td>
</tr>
<tr>
<td>2.00D 00</td>
<td>-1.712D 00</td>
<td>-2.069D 00</td>
<td>3.455D 00</td>
<td>1.296D-01</td>
<td>5</td>
</tr>
<tr>
<td>3.00D 00</td>
<td>-3.526D 00</td>
<td>-1.553D 00</td>
<td>3.202D 00</td>
<td>-5.547D-01</td>
<td>5</td>
</tr>
<tr>
<td>4.00D 00</td>
<td>-4.823D 00</td>
<td>-1.047D 00</td>
<td>2.455D 00</td>
<td>-9.034D-01</td>
<td>5</td>
</tr>
<tr>
<td>5.00D 00</td>
<td>-5.632D 00</td>
<td>-5.761D-01</td>
<td>1.452D 00</td>
<td>-1.082D 00</td>
<td>5</td>
</tr>
<tr>
<td>6.00D 00</td>
<td>-5.982D 00</td>
<td>-1.259D-01</td>
<td>3.274D-01</td>
<td>-1.151D 00</td>
<td>5</td>
</tr>
<tr>
<td>7.00D 00</td>
<td>-5.885D 00</td>
<td>3.197D-01</td>
<td>-8.217D-01</td>
<td>-1.132D 00</td>
<td>5</td>
</tr>
<tr>
<td>8.00D 00</td>
<td>-5.339D 00</td>
<td>7.771D-01</td>
<td>-1.967D 00</td>
<td>-1.020D 00</td>
<td>5</td>
</tr>
<tr>
<td>9.00D 00</td>
<td>-4.322D 00</td>
<td>1.262D 00</td>
<td>-2.819D 00</td>
<td>-7.793D-01</td>
<td>5</td>
</tr>
<tr>
<td>1.00D 01</td>
<td>-2.802D 00</td>
<td>1.781D 00</td>
<td>-3.352D 00</td>
<td>-3.157D-01</td>
<td>5</td>
</tr>
<tr>
<td>1.10D 01</td>
<td>-7.717D-01</td>
<td>2.249D 00</td>
<td>-3.255D 00</td>
<td>6.289D-01</td>
<td>5</td>
</tr>
<tr>
<td>1.20D 01</td>
<td>1.431D 00</td>
<td>1.799D 00</td>
<td>-1.777D 00</td>
<td>2.605D 00</td>
<td>5</td>
</tr>
<tr>
<td>1.30D 01</td>
<td>1.645D 00</td>
<td>-1.512D 00</td>
<td>1.423D 00</td>
<td>2.503D 00</td>
<td>5</td>
</tr>
</tbody>
</table>

PROBLEM COMPLETED IN 103 STEPS
168 G EVALUATIONS
12 J EVALUATIONS

-43-
OUTPUT PRODUCED FROM EXAMPLE PROBLEM 5

DEMONSTRATION PROGRAM, CRBITAL PROBLEM...

\[
\begin{align*}
Y1DOT &= Y2 \\
Y2DOT &= (-16 \times Y1) / (Y1 \times Y1 + Y3 \times Y3)^{**1.5} \\
Y3DOT &= Y4 \\
Y4DOT &= (-16 \times Y3) / (Y1 \times Y1 + Y3 \times Y3)^{**1.5}
\end{align*}
\]

INITIAL VALUES. \( Y1(0) = 2, \quad Y2(0) = 0, \)
\( Y3(0) = 0, \quad Y4(0) = 2 \times \text{SQRT}(3) \)

JACOBIAN MATRIX FULL, MATRIX A BANDED.

\[MF = 21 \quad \text{ICASE} = 3\]
\[N = 4 \quad T0 = 0.0 \quad H0 = 6.10E-04 \quad EPS = 0.10E-04\]
\[ML = 3 \quad MU = 3 \quad MLA = 1 \quad MLA = 1\]

TO AVOID POSSIBLE STORAGE OVERWRITES, USER MUST (AT A MINIMUM)
DIMENSION... \( Y0(4), \text{WORK}(76), \text{AND} \text{IWORK}(11). \)

<table>
<thead>
<tr>
<th>T</th>
<th>Y1</th>
<th>Y2</th>
<th>Y3</th>
<th>Y4</th>
<th>ORDER</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0000</td>
<td>5.243D-01</td>
<td>-2.267D CC</td>
<td>2.687D CC</td>
<td>1.557D 00</td>
<td>4</td>
</tr>
<tr>
<td>2.0000</td>
<td>-1.712D 00</td>
<td>-2.069D 00</td>
<td>3.455D 00</td>
<td>1.296D-01</td>
<td>5</td>
</tr>
<tr>
<td>3.0000</td>
<td>-3.526D 00</td>
<td>-1.553D 00</td>
<td>3.202D 00</td>
<td>-5.547D-01</td>
<td>5</td>
</tr>
<tr>
<td>4.0000</td>
<td>-4.823D 00</td>
<td>-1.047D 00</td>
<td>2.455D 00</td>
<td>-9.334D-01</td>
<td>5</td>
</tr>
<tr>
<td>5.0000</td>
<td>-5.632D 00</td>
<td>-5.761D-01</td>
<td>1.452D 00</td>
<td>-1.082D 00</td>
<td>5</td>
</tr>
<tr>
<td>6.0000</td>
<td>-5.982D 00</td>
<td>-1.259D-01</td>
<td>3.274D-01</td>
<td>-1.151D 00</td>
<td>5</td>
</tr>
<tr>
<td>7.0000</td>
<td>-5.885D 00</td>
<td>3.197D-01</td>
<td>-8.217D-01</td>
<td>-1.122D 00</td>
<td>5</td>
</tr>
<tr>
<td>8.0000</td>
<td>-5.339D 00</td>
<td>7.771D-01</td>
<td>-1.967D 00</td>
<td>-1.020D 00</td>
<td>5</td>
</tr>
<tr>
<td>9.0000</td>
<td>-4.322D 00</td>
<td>1.262D 00</td>
<td>-2.819D 00</td>
<td>-7.753D-01</td>
<td>5</td>
</tr>
<tr>
<td>1.0001</td>
<td>-2.802D 00</td>
<td>1.781D 00</td>
<td>-3.392D 00</td>
<td>-3.157D-01</td>
<td>5</td>
</tr>
<tr>
<td>1.2001</td>
<td>1.431D 00</td>
<td>1.799D CC</td>
<td>-1.777D CC</td>
<td>2.655D 00</td>
<td>5</td>
</tr>
<tr>
<td>1.3001</td>
<td>1.645D 00</td>
<td>-1.512D CC</td>
<td>1.423D 00</td>
<td>2.903D 00</td>
<td>5</td>
</tr>
</tbody>
</table>

PROBLEM COMPLETED IN 103 STEPS
168 G EVALUATIONS
12 J EVALUATIONS
PROBLEM 6

DEMONSTRATION PROGRAM.

Y1DOT = Y2
Y2DOT = (-16 * Y1) / (Y1 * Y1 + Y3 * Y3)**1.5
Y3DOT = Y4
Y4DOT = (-16 * Y3) / (Y1 * Y1 + Y3 * Y3)**1.5

INITIAL VALUES: Y1(0) = 2, Y2(0) = 0,
Y3(0) = 0, Y4(0) = 2 * SQRT(3)

JACOBIAN MATRIX BANDED, MATRIX A BANDED.

MF = 21
ICASE = 3

N = 4
TC = 0.0
HO = 0.1D-04
EPS = 0.1D-04

ML = 3
MU = 2
MLA = 1
MUA = 1

TO AVOID POSSIBLE STORAGE OVERWRITES, USER MUST (AT A MINIMUM) DIMENSION.
Y0( 4), WORK( 108), AND IWORK( 11).

<table>
<thead>
<tr>
<th>T</th>
<th>Y1</th>
<th>Y2</th>
<th>Y3</th>
<th>Y4</th>
<th>CRDER</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.00D 00</td>
<td>5.243D-01</td>
<td>-2.267D 00</td>
<td>2.687D 00</td>
<td>1.597D 00</td>
<td>4</td>
</tr>
<tr>
<td>2.00D 00</td>
<td>-1.712D 00</td>
<td>-2.669D 00</td>
<td>3.455D 00</td>
<td>1.296D-01</td>
<td>5</td>
</tr>
<tr>
<td>3.00D 00</td>
<td>-3.526D 00</td>
<td>-1.553D 00</td>
<td>3.202D 00</td>
<td>-5.547D-01</td>
<td>5</td>
</tr>
<tr>
<td>4.00D 00</td>
<td>-4.823D 00</td>
<td>-1.047D 00</td>
<td>2.455D 00</td>
<td>-9.034D-01</td>
<td>5</td>
</tr>
<tr>
<td>5.00D 00</td>
<td>-5.632D 00</td>
<td>-5.761D-01</td>
<td>1.452D 00</td>
<td>-1.082D 00</td>
<td>5</td>
</tr>
<tr>
<td>6.00D 00</td>
<td>-5.982D 00</td>
<td>-1.259D-01</td>
<td>3.274D-01</td>
<td>-1.151D 00</td>
<td>5</td>
</tr>
<tr>
<td>7.00D 00</td>
<td>-5.885D 00</td>
<td>3.197D-01</td>
<td>-8.217D-01</td>
<td>-1.122D 00</td>
<td>5</td>
</tr>
<tr>
<td>8.00D 00</td>
<td>-5.329D 00</td>
<td>7.771D-01</td>
<td>-1.907D 00</td>
<td>-1.020D 00</td>
<td>5</td>
</tr>
<tr>
<td>5.00D 00</td>
<td>-4.322D 00</td>
<td>1.262D 00</td>
<td>-2.819D 00</td>
<td>-7.793D-01</td>
<td>5</td>
</tr>
<tr>
<td>1.00D 01</td>
<td>-2.802D 00</td>
<td>1.781D 00</td>
<td>-3.392D 00</td>
<td>-3.157D-01</td>
<td>5</td>
</tr>
<tr>
<td>1.10D 01</td>
<td>-7.717D-01</td>
<td>2.249D 00</td>
<td>-3.295D 00</td>
<td>6.289D-01</td>
<td>5</td>
</tr>
<tr>
<td>1.20D 01</td>
<td>1.431D 00</td>
<td>1.799D CC</td>
<td>-1.777D 00</td>
<td>2.650D 00</td>
<td>5</td>
</tr>
<tr>
<td>1.30D 01</td>
<td>1.645D 00</td>
<td>-1.512D 00</td>
<td>1.423D 00</td>
<td>2.503D 00</td>
<td>5</td>
</tr>
</tbody>
</table>

PROBLEM COMPLETED IN 103 STEPS
168 G EVALUATIONS
12 J EVALUATIONS

-45-
ACKNOWLEDGMENTS

I wish to dedicate this work to my wife, Janet, and to express both love and gratitude to her for her devotion. It was only through her support that this work was possible.

I would like to express my gratitude to my major professor, Richard Sincovec, for his interest, encouragement, and guidance in the preparation of this report.

I also extend thanks to my committee members, William Hankley and Virgil Wallentine, for their suggestions and comments concerning my work.
REFERENCES


Listing of the ODEPAKK subroutines:

ODEPAK
INTERP
STIFF
COSSET
RES
PMAT
DEC
SOL
DIFFUN
ADDA
ADDB
ADDBB
URNND
SUBROUTINE ODEPAK (N, TO, HO, YO, TCUT, EPS, MF, ICASE, INDEX,  
* ML, MU, MLA, MUA, WORK, IWORK)  
IMPLICIT REAL*8(A-H,O-Z)  

C-----------------------------------------------
C
C ODEPAKK.... A PACKAGE FOR THE SOLUTION OF THE INITIAL VALUE  
C PROBLEMS FOR SYSTEMS OF ORDINARY DIFFERENTIAL EQUATIONS,  
C A(Y,T) * DY/DT = G(Y,T), Y = (Y(1),Y(2),...,Y(N)).  
C A(Y,T) IS A MATRIX; EITHER THE IDENTITY MATRIX, A FULL  
C MATRIX, OR A BANDED MATRIX. THE JACOBIAN MATRIX IS  
C REPRESENTED AS EITHER A FULL OR A BANDED MATRIX.  
C THE DEPENDENCE OF A(Y,T) ON Y IS ASSUMED TO BE WEAK.
C
C SUBROUTINE ODEPAK IS A DRIVER ROUTINE FOR THE ODEPAKK PACKAGE.
C
C THIS PACKAGE IS A MERGED VERSION OF THE GEAR, GEARB, AND THE  
C GEARIB PACKAGES WRITTEN BY A.C. HINDMARCH OF THE LAWRENCE  
C LIVERMORE LABORATORY FOR CDC COMPUTERS. THE CDC VERSIONS  
C WERE MODIFIED FOR USE ON IBM COMPUTERS IN DOUBLE PRECISION
C AT ARGONNE NATIONAL LABORATORY.
C
C-----------------------------------------------
C
C REQUIRED USER SUPPLIED SUBRUTINES...
C
C THE USER MUST SUPPLY A MAIN PROGRAM AND THREE SUBPROGRAMS WHICH  
C DEFINE ORDINARY DIFFERENTIAL EQUATIONS WHOSE SOLUTION IS TO BE  
C ATTEMPTED. THE THREE USER-SUPPLIED SUBPROGRAMS ARE...
C
C 1) SUBROUTINE AMAT (N, MLA, MUA, T, Y, A)  
IMPLICIT REAL*8(A-H,O-Z)  
DIMENSION Y(1), A(N,1)  

    THIS ROUTINE DEFINES THE MATRIX A, USED WHEN ICASE = 3  
    (A(Y,T) * (DY/DT) = G(Y,T)).

    FOR ICASE = 3: THE USER DEFINES MATRIX A IN FULL FORM  
    (AS AN N BY N ARRAY) IF MLA=MUA=N-1; OTHERWISE,  
    MATRIX A IS DEFINED IN BANDED FORM (AS AN N BY (MLA+MUA+1)  
    ARRAY WHERE A(I,J) IS STORED IN LOCATION A(I,J-I+MLA+1)).

    THE USER PROVIDES, IN THE MAIN ROUTINE, THE INPUT VARIABLES  
    MLA AND MUA.

    MATRIX A MUST NOT BE DEFINED AS A DIAGNOL MATRIX (I.E.  
    MLA=MUA=0).

    FOR MF = 10 OR 20, MATRIX A MUST BE A CONSTANT MATRIX  
    WITH RESPECT TO T AND Y.

    FOR ICASE = 1,2; AMAT MAY BE REPLACED BY A CUMMY SUBROUTINE.

    RETURN
    END

C 2) SUBROUTINE GFUN (N, T, Y, G)  
IMPLICIT REAL*8(A-H,O-Z)  
DIMENSION Y(N), G(N)
THIS ROUTINE COMPUTES THE FUNCTION \( g(y, t) \), THE RIGHT-HAND
SIDE OF THE C.C.E. HERE \( y \) AND \( g(y, t) \) ARE VECTORS OF LENGTH
\( n \). GFUN IS CALLED IN ALL CASES.

RETURN
END

3) SUBROUTINE JMAT (A, ML, MU, T, Y, PW)
IMPLICIT REAL*8(A-H, O-Z)
DIMENSION PW(N,1), Y(N)

THIS ROUTINE COMPUTES THE JACOBIAN MATRIX OF PARTIAL
DERIVATIVES OF \( g(y, t) = (g(1), g(2), \ldots, g(n)) \).

FOR ICASE = 1, THE JACOBIAN IS STORED IN PW AS AN N BY N
ARRAY, WHERE PW(I,J) IS SET TO BE THE PARTIAL DERIVATIVE OF
\( g(I) \) WITH RESPECT TO \( y(J) \).

FOR ICASE = 2, THE JACOBIAN IS STORED AS AN N BY
\( (ML+MU+1) \) ARRAY, WHERE PW(I,J-1:ML+1) IS SET TO BE THE
PARTIAL DERIVATIVE OF \( g(I) \) WITH RESPECT TO \( y(J) \).

FOR ICASE = 3, THE JACOBIAN IS STORED EITHER IN FULL OR
BANDED FORM AS JUST DESCRIBED. NOTE: FOR THE JACOBIAN
FULL, \( ML = MU = N-1 \); FOR THE JACOBIAN BANDED, \( (ML+MU)
\LT (2*N-1) \).
FULL, \( (ML+MU) = (2*NM1) \); FOR THE JACOBIAN BANDED, \( (ML+MU)
\LT (2*NM1) \).

JMAT IS CALLED ONLY IF MITE = 1, OTHERWISE, JMAT MAY BE
REPLACED BY A DUMMY SUBROUTINE.

RETURN
END

METHODS...

THE ODEPACK PACKAGE USES THE LINEAR MULTISTEP METHODS OF THE FCRM:
\[
K1 \quad y(n) = \sum (\alpha(j) \ast y(n-j)) + h \ast \sum (\beta(j) \ast ydct(n-j)) , \\
J=1 \quad K2
\]

WHERE \( y(i) \) IS AN APPROXIMATION TO \( y(t(i)) \), \( ydct(i) = f(y(i), t(i)) \)
IS AN APPROXIMATION TO \( ydct(t(i)) \), AND \( h \) IS A CONSTANT STEP SIZE
\( h = t(i+1) - t(i) \).

THE TWO TYPES OF LINEAR MULTISTEP METHODS AVAILABLE ARE THE
IMPLICIT ADAMS METHODS (TO ORDER 12), AND THE BACKWARD DIFFERENTIA-
TION FORMULA METHODS, OR GEAR'S STIFF METHODS (TO ORDER 5).

AN ORDER Q METHOD IS IDENTIFIED AS BEING ONE IN WHICH, WHEN THE
EQUATION FOR \( y(n) \) IS SOLVED WITH ALL PRECEDING \( y \) VALUES EXACT,
THEN FOR SMALL \( h \), \( y(n) \) WILL DIFFER FROM THE CORRECT SOLUTION OF
THE O.D.E. BY A LOCAL TRUNCATION ERROR THAT IS OF THE ORDER
\( h^{q+1} \).

FCR AN ORDER Q ADAMS METHOD \( K1 = 1 \) AND \( K2 = Q-1 \), AND FCR AN ORDER Q
Backward Differentiation Formula $k_1 = C$ and $k_2 = 0$. The constants 
alpha(j) and beta(j) are constants associated with the method 
and since beta(0) is greater than 0, the above equation for $y(n)$ 
is, in general, an implicit nonlinear algebraic system that must 
be solved on every step.

To solve this system, three correct CF, iteration methods are 
available: functional iteration, Newton's chord method with an 
analytic Jacobian matrix, and Newton's chord method with the 
Jacobian matrix calculated internally by finite differences. 
Due to the non-matrix treatment of the functional iteration 
method, it is less expensive than the other methods. However, 
there is a loss in the speed of convergence and in general, a 
smaller value of $h$ is required for convergence. Newton's chord 
method involves a matrix problem, therefore it is more expensive, 
while at the same time it converges faster and for larger times 
step sizes than does the functional iteration method.

To compute the values of $y(n)$, using an integration formula of 
order $q$ and step size $h$, the following tasks are performed in taking 
a time step. Based on the computed solution values at previous 
times, an explicit prediction formula is used to estimate the values 
of $y(n)$. These predicted values are then used as the initial 
values for starting the nonlinear iteration process. The correction 
is then performed by using either the functional iteration or 
Newton's iteration method to solve the nonlinear equation of order 
$q$ for the new solution values, $y(n)$. The step size is reduced and 
the prediction-correction process is terminated if the iteration 
process fails to converge.

For successfully computed values of $y(n)$, error estimates are 
calculated. The values of $y(n)$ are allowed if they are less than 
the error tolerance, $\epsilon$. However, if the errors do not conform 
to the error tolerance, the prediction-correction process is 
repeated with a different $h$ and/or a lower order $q$ until the error 
test is passed or a decision is made to terminate the attempt at 
taking the time step.

After successfully taking a time step, or after an unsuccessful 
attempt at taking a time step, a new step size $h'$-prime and formula 
order $q'$-prime is determined. If the last time step was successful, 
$q'$-prime may take on the values $q-1$, $q$, or $q+1$; if not, $q'$-prime may 
take on the values $q-1$ or $q$. Based on the integration formulas 
of orders $q-1$, $q$, and $q+1$, estimates are made as to the respective 
maximum time step sizes that may be used and still pass the error 
test. The formula order that will allow the largest time step size 
is then used, with that step size, for the next time step attempt.

A more complete and thorough discussion of these methods, the 
control of error, the change of order and step size, and procedure 
involves in taking a time step may be found in references 1, 2, 3, 
5, and 6.

References...

1. C. W. Gear, Numerical Initial Value Problems in Ordinary 
   Differential Equations, Prentice-Hall, Englewood Cliffs, 
   1971.
STORAGE ALLOCATION...

THE STORAGE FOR ODEPAK IS ALLOCATED DYNAMICALLY BASED ON THE INPUT PARAMETERS SUPPLIED TO SUBROUTINE ODEPAK FROM THE USER'S MAIN PROGRAM.

THE ARRAY WORK IS A STORAGE POOL FOR THE WORKING ARRAYS; Y, YMAX, ERRCR, SAVE1, SAVE2, SAVE3, PW, AND A. THE ARRAY IWORK IS A LIST OF POINTERS INTO WORK INDICATING THE STARTING LOCATIONS OF THE VARIOUS WORKING ARRAYS, AND IWORK(8) THRU IWORK(N+7) HOLD THE IPIV ARRAY. THE FOLLOWING TABLE ASSOCIATES THE WORK AND IWORK ARRAYS.

WORK(1) IS THE BEGINNING OF THE ARRAY Y
IWORK(1) POINTS TO THE BEGINNING LOCATION OF THE ARRAY YMAX IN WORK
IWORK(2) POINTS TO THE BEGINNING LOCATION OF THE ARRAY ERRCR IN WORK
IWORK(3) POINTS TO THE BEGINNING LOCATION OF THE ARRAY SAVE1 IN WORK
IWORK(4) POINTS TO THE BEGINNING LOCATION OF THE ARRAY SAVE2 IN WORK
IWORK(5) POINTS TO THE BEGINNING LOCATION OF THE ARRAY SAVE3 IN WORK
IWORK(6) POINTS TO THE BEGINNING LOCATION OF THE ARRAY PW IN WORK
IWORK(7) POINTS TO THE BEGINNING LOCATION OF THE ARRAY A IN WORK
IWORK(8) IS THE BEGINNING OF THE ARRAY IPIV.

THE MINIMUM SIZE OF THE VARIOUS WORKING ARRAYS IS DEFINED AS FOLLOWS.

<table>
<thead>
<tr>
<th>ARRAY</th>
<th>FOR ICASE = 1</th>
<th>FOR ICASE = 2</th>
<th>FOR ICASE = 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y</td>
<td>N * MSIZE</td>
<td>N * MSIZE</td>
<td>N * MSIZE</td>
</tr>
<tr>
<td>YMAX</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>ERRCR</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
</tbody>
</table>
SAVE1  N  N  N  N
SAVE2  N  N  N
SAVE3  N  N  N

PW
FOR MITER = 0;  1  1  N*(2*MLA+MUA+1).
FOR MITER = 1,2;  N * N  N*(2*ML+MU+1)
AND PW FULL;
AND PW IS FULL
AND MATRIX A
IS BANDED;
AND PW IS BANDED
AND MATRIX A
IS FULL;
AND MATRICES PW
AND A BANDED;

A
FOR MITER = 0;  1  1  N*(MLA+MUA+1)
FOR MITER=1,2;  1  1
AND MATRIX A
FULL;
AND MATRIX A
BANDED AND PW
FULL;
AND MATRICES PW
AND A BANDED;

IPIV
FOR MITER = 0;  1  1  1
FOR MITER = 1,2;  N  N  N

WHERE...
MAXDER = MAXIMUM ORDER OF METHOD USED...
5 FOR BCF METHODS (METH = 2),
12 FOR ADAMS METHODS (METH = 1).
MSIZE = MAXDER + 1

AN OUTPUT MESSAGE IS PRODUCED FROM ODEPAK TO INFORM THE USER OF THE
MINIMUM STORAGE REQUIREMENTS NEEDED FOR ARRAYS Y0, WCRK, AND IWORK.

ODEPAK IS TO BE CALLED CNCE FCR EACH OUTPUT VALUE OF T, AND
-53-
IN TURN MAKES REPEATED CALLS TO THE CORE INTEGRATOR, STIFF.

THE GENERAL FCRM OF THE CALL TO CDEPAK FCLLCKS:

CDEPAK(N,TO,YO,TCUT,eps,MF,ICASE,INDEX,ML,MU,MLA,MUA,WORK,IWORK)

THE INPUT PARAMETERS ARE...

N = THE NUMBER OF FIRST-ORDER DIFFERENTIAL EQUATIONS.
N MAY BE REDUCED, BUT NEVER INCREASED, CURING A PROBLEM.

TO = THE INITIAL VALUE OF T, THE INDEPENDENT VARIABLE
(USED ONLY ON FIRST CALL).

HO = THE NEXT STEP SIZE IN T (USED FOR INPUT ONLY ON THE
FIRST CALL).

YCO = A VECTOR OF LENGTH N CONTAINING THE INITIAL VALUES OF
Y (USED FOR INPUT ONLY ON FIRST CALL).

TCUT = THE VALUE OF T AT WHICH OUTPUT IS DESIRED NEXT.
INTEGRATION WILL NORMALLY BE SLIGHTLY BEYOND TCUT
AND THE PACKAGE WILL INTERPOLATE TC T = TCUT.

EPS = THE RELATIVE BOUND (USED ONLY ON THE
FIRST CALL, UNLESS INDEX = -1). SINGLE STEP ERROR
ESTIMATES DIVIDED BY YMAX(I) WILL BE KEPT LESS THAN
EPS IN ROCT-MEAN-SQUARE NORM (I.E. EUCLIDEAN NORM
DIVIDED BY SCRT(N)). THE VECTOR YMAX OF
WEIGHTS IS COMPUTED IN CDEPAK. INITIALLY YMAX(I) IS
ABS(Y(I)), WITH A DEFAULT VALUE OF 1 IF Y(I) = 0
INITIALLY. THEREAFTER, YMAX(I) IS THE LARGEST VALUE
OF ABS(Y(I)) SEEN SO FAR, OR THE INITIAL YMAX(I) IF
THAT IS LARGER. TO ALTER EITHER OF THESE, CHANGE THE
APPROPRIATE STATEMENTS IN THE DO-LOCPS ENDING AT
STATEMENTS 10 AND 7C BELOW.

MF = THE METHOD FLAG (USED ONLY ON FIRST CALL, UNLESS
INDEX = -1). MF HAS TWX DECIMAL DIGITS, MTH AND
MTER (MF = 0*MTH + MTER).
ALLOWED VALUES: 10, 11, 12, 20, 21, AND 22.
FOR ICASE = 3 WITH MF = 10 OR 20, MATRIX A MUST BE
A CONSTANT MATRIX WITH RESPECT TO T AND Y.

METH IS THE BASIC METHOD INDICATOR...
METH = 1 MEANS THE ADAMS METHODS.
METH = 2 MEANS THE STIFF METHODS OF GEAR, CR THE
BACKWARD DIFFERENTIATION FORMULAS (EDF).

MTER IS THE ITERATION METHOD INDICATOR AND DETERMINES
HOW THE JACOBIAN MATRIX DC/DT IS TO BE COMPUTED.
MTER = 0 MEANS FUNCTIONAL ITERATION (NC PARTIAL
DERIVATIVES NEEDED).
MTER = 1 MEANS CHORD METHOD WITH ANALYTIC JACOBIAN.
FOR THIS OPTION, USER SUPPLIES SUCREDUCE
MAT (SEE DESCRIPTION BELOW).
MTER = 2 MEANS CHORD METHOD WITH JACOBIAN CALCULATED
INTERNAL BY FINITE DIFFERENCES.

ICASE = TYPE OF O.D.E. TO BE SOLVED. ALLOWED VALUES: 1, 2, AND 3.
1 DY/DT = G(Y,T), FULL JACOBIAN MATRIX.
2 DY/CT = G(Y,T), BANDED JACOBIAN MATRIX.
3 A*(DY/DT) = G(Y,T), FULL OR BANDED JACOBIAN MATRIX.

INDEX = INTEGER USED ON INPUT TO INDICATE TYPE OF CALL,
WITH THE FOLLOWING VALUES AND MEANINGS:
1 THIS IS THE FIRST CALL FOR THIS PROBLEM.
0 THIS IS NOT THE FIRST CALL FOR THIS PROBLEM, 
AND INTEGRATION IS TO CONTINUE.
-1 THIS IS NOT THE FIRST CALL FOR THE PROBLEM, 
AND THE USER HAS RESET N AND OR EPS.
2 SAME AS 0 EXCEPT THAT TCUT IS TO BE HIT 
EXACTLY (NO INTERPOLATION IS DONE).
ASSUMES TCUT =GE. THE CURRENT T.
3 SAME AS 0 EXCEPT CTRL RETURN TO CALLING 
PROGRAM AFTER ONE STEP. TCUT IS IGNORED.
SINCE THE NORMAL OUTPUT VALUE OF INDEX IS 0,
IT NEED NOT BE RESET FOR NORMAL CONTINUATION.

ML, MU = THE WIDTHS OF THE LOWER AND UPPER PARTS, RESPECTIVELY, 
OF THE BAND IN THE JACOBIAN MATRIX, NOT COUNTING THE 
MAIN DIAGONAL. THE FULL BANDWIDTH IS ML + MU + 1. 
REQUIRED FOR ICASE = 2 AND FCR ICASE = 3. FCR ICASE 
= 3 AND THE JACOBIAN IS A FULL MATRIX, ML = MU = N-1.
FOR ICASE = 1, ML AND MU DEFAULT TO N-1 IN ODEPAK.
MLA, MUA = THE WIDTHS OF THE LOWER AND UPPER PARTS, RESPECTIVELY, 
OF THE BAND IN THE MATRIX A, NOT COUNTING THE MAIN 
DIAGONAL. THE FULL BANDWIDTH IS MLA + MUA + 1.
MLA AND MUA ARE REQUIRED FOR ICASE = 3. FOR MATRIX A 
FULL, SET MLA = MUA = N-1.
FOR ICASE = 1 OR 2, MLA AND MUA DEFAULT TO 0 IN ODEPAK.

THE USER MUST DIMENSION (AT A MINIMUM) THE ARRAYS YG, IWORK, AND WORK 
IN THE MAIN PROGRAM ACCORDING TO THE FOLLOWING CONDITIONS... 

1) YG(*), WHERE ** = N
2) IWORK(*), WHERE ** = N + 7
3) WORK(*), WHERE **

FOR ICASE = 1; ** = N*(MSIZE+5)+1+IA+IB, WHERE.. 
FOR MITER = 0;
   IA = 1,
   IB = 1.

FOR MITER = 1,2;
   IA = N * N,
   IB = N.

FOR ICASE = 2; ** = N*(MSIZE+5)+1+IA+IB, WHERE.. 
FOR MITER = 0;
   IA = 1,
   IB = 1.

FCR MITER = 1,2;
   IA = N*(2*ML+MU+1),
   IB = N.

FOR ICASE = 3; ** = N*(MSIZE+5)+NPW+NA+NPIV, WHERE..
FOR MITER = G;
   NPW = N*(2*MLA+MUA+1),
   NA = N*(MLA+MUA+1),
   NPIV = 1.

FCR MITER = 1,2,
AND MATRICES PW AND A FULL;
   NPW = N * N,
   NA = N * N,
NIPIV = N.

AND PW FULL AND MATRIX A BANDEC;
NPW = N * N,
NA = N * (MLA+MLA+1),
NIPIV = N.

AND PW BANDED AND MATRIX A FULL;
NPW = MAX(N*N, N*(ML+MLA+1)),
NA = N * N,
NIPIV = N.

AND MATRICES PW AND A BANDED;
NPW = N*(2*MAX(ML,MLA)+MAX(MU,MUA)+1),
NA = N*(MAX(ML,MLA)+MAX(ML,MLA)+1),
NIPIV = N.

AFTER THE INITIAL CALL, IF A NORMAL RETURN OCCURRED AND A NORMAL
CONTINUATION IS DESIRED, SIMPLY RESET TOUT AND CALL AGAIN.
ALL OTHER PARAMETERS WILL BE READY FOR THE NEXT CALL.
A CHANGE OF PARAMETERS WITH INDEX = -1 CAN BE MADE AFTER
EITHER A SUCCESSFUL OR AN UNSUCCESSFULL RETURN.

THE OUTPUT PARAMETERS ARE...

HG  = THE STEP SIZE USED LAST, WHETHER SUCCESSFULLY OR NOT.
Y0  = THE COMPUTED VALUES OF Y AT T = TOUT.
TOUT = THE OUTPUT VALUE OF T. IF INTEGRATION WAS SUCCESSFUL,
AND THE INPUT VALUE OF INDEX WAS NOT 3, TOUT IS
UNCHANGED FROM ITS INPUT VALUE. OTHERWISE, TOUT
IS THE CURRENT VALUE OF T TO WHICH INTEGRATION
HAS BEEN COMPLETED.
INDEX = INTEGER USED ON OUTPUT TO INDICATE RESULTS;
WITH THE FOLLOWING VALUES AND MEANINGS.
0  INTEGRATION WAS COMPLETED TO TOUT OR BEYOND.
-1  THE INTEGRATION WAS HALTED AFTER FAILING TO PASS THE
ERROR TEST EVEN AFTER REDUCING H BY A FACTOR OF
1.E-10 FROM ITS INITIAL VALUE.
-2  AFTER SOME INITIAL SUCCESS, THE INTEGRATION WAS
HALTED EITHER BY REPEATED ERROR TEST FAILURES OR BY
A TEST ON EPS. TOO MUCH ACCURACY HAS BEEN REQUESTED.
-3  THE INTEGRATION WAS HALTED AFTER FAILING TO ACHIEVE
CONVERGENCE EVEN AFTER REDUCING H BY A
FACTOR OF 1.E-10 FROM ITS INITIAL VALUE.
-4  IMMEDIATE HALT BECAUSE OF ILLEGAL VALUES OF INPUT
PARAMETERS. SEE PRINTED MESSAGE.
-5  INDEX WAS -1 ON INPUT, BUT THE DESIRED CHANGES OF
PARAMETERS WERE NOT IMPLEMENTED BECAUSE TOUT
WAS NOT BEYOND T. INTERPOLATION TO T = TOUT WAS
PERFORMED AS ON A NORMAL RETURN. TO TRY AGAIN,
SIMPLY CALL AGAIN WITH INDEX = -1 AND A NEW TOUT.

TO PROVIDE ADDITIONAL STATISTICS ON THE PROBLEM BEING SOLVED,
THE COMMON BLOCK ODE3 MAY BE ACCESSSED EXTERNALLY BY THE USER
IF DESIRED. IT CONTAINS THE STEP SIZE LAST USED (SUCCESSFULLY),
THE CRDER LAST USED (SUCCESSFULLY), THE NUMBER OF STEPS TAKEN
SO FAR, THE NUMBER OF G EVALUATIONS (GFUN CALLS) SO FAR,
IN ADDITION TO CDEPAK, THE FOLLOWING ROUTINES ARE PROVIDED IN THE PACKAGE...

Interp(tcut,y,no,yo) interpolates to get the output values at t = tcut, from the data in the y array.

Stiff(no,psize,npw,na,nipiv,y,ymax,error,save1,save2,save3,pw,a,ipiv) is the core integrator routine. It performs a single step and associated error control.

Coset(meth,nq,el,tc,maxder) sets coefficients for use in the core integrator.

Res(n,t,h,a,y,v,r) computes the residual vector R = H*G(Y,T) - A(Y,T)*V given Y and T, where V is a given vector and H is a given stepsize. Called in all cases.

Pmat(y,a,no,con,iter,ier,ymax,error,save1,save2,pw,ipiv) for icase = 1,2; Fmat computes and processes the matrix J = I - S*CG/DY. For icase = 3; Pmat computes and processes the chord iteration matrix P = A - S*CG/DY, where S is a scalar.

Cec(no,n,ml,mu,b,ip,ierr) performs an LU decomposition on a full matrix or a banded linear system.

Sol(no,n,ml,mu,b,y,ip) solves linear systems A*X = B after decomposition.

Diffun(n,t,y,ydot,a,pw,ipiv) used when icase = 2; Diffun computes ydot = a**(-1)*y, given Y and T, using Adda, Dec, and Sol. Called only on starting up (on the first step), and when restarting from a crash.

Adda1(n,t,y,p,a,no,ml,mu) adds the matrix A(Y,T) to the matrix stored in P. For icase = 1, P is an N by N array and A is the identity matrix, 1.0 is added to the p(i,i) element, yielding a full matrix. For icase = 2, P is an N by (ML+MU+1) array and A is the identity, 1.0 is added to the p(i,j-i+ML+1) element, yielding a banded matrix. For icase = 3, P and A may be either full or banded matrices. If P is full, it is an N by N array; if P is banded, it is an N by (ML+MU+1) array. If A is full, it is an N by N array; if A is banded, it is an N by (MLA+MU+1) array. For both P and A, the result is a banded matrix; for all other configurations of P and A, the result is a full matrix. Called by Pmat and Diffun. Uses Addb and Addbb to assist in summing matrices A and P.

Addb(n,ilc,iup,b,f) adds the full matrix F to the banded matrix E, result is a full matrix. ILC and IUP are the lower and upper bandwidths, respectively, of the banded matrix B.

Addbb(n,nml,mll,maxnp,kl,kll,cl,dl) adds the banded matrices CL and DLL, of possibly different band-
COMMUNICATION...

Each subroutine in the package contains a communication summary as indicated below. The Fortran functions used specifies both the single and double precision names.

PACKAGE ROUTINES CALLED... INTERP, STIFF, URND
USER ROUTINES CALLED... NCNE
CALLED BY... USERS MAIN PROGRAM
FORTRAN FUNCTIONS USED... ABS(CABS), FLOAT(DFLOAT), MAX0, MAX1(DMAX1), SQRT(DSQR)

DIMENSION Y0(1), WCRK(1), IWKCRK(1)
CCMCMN /CDE/ T, H, HMIN, HMAX, EPSC, URCUND, NC, MFC, KFLAG, JSTART
CCMCMN /CDE2/ EPSJ, NSC, MLC, MUC, PM, NML, NOW, MLA, MUAC, MWA
CCMCMN /CDE3/ HLSED, NQLSED, NSTEP, AGE, NJE
CCMCMN /CDE4/ ICASS, ITYPE, JK, NML, NMU, IFULL

IN THE FOLLOWING DATA STATEMENT, SET...
LOLT = THE LOGICAL UNIT NUMBER FOR THE OUTPUT OF MESSAGES
DURING THE INTEGRATION.

DATA LOLT/6/

CHECK FOR INVALID METHOD FLAG.

IF (MF.NE.10. AND. MF.NE.11. AND. MF.NE.12. AND. MF.NE.20. AND.
*MF.NE.21. AND. MF.NE.22) GO TO 310
ICASS = ICASE
IF (INDEX .EQ. 0) GO TO 20
IF (INDEX .EQ. 2) GO TO 25
IF (INDEX .EQ. -1) GO TO 30
IF (INDEX .EQ. 3) GO TO 40
IF (INDEX .EQ. 1) GO TO 430
IF (EPS .LE. 0.00) GO TO 450
IF (N .LE. 0) GO TO 450
IF ((TO-TOUT)*HO .GE. 0.00) GO TO 420

IF INITIAL VALUES OF YMAX OTHER THAN THOSE SET BELOW ARE DESIRED.
THEY SHOULD BE SET HERE. ALL YMAX(I) MUST BE POSITIVE.
IF VALUES FOR HMIN OR HMAX, THE BOUNDS ON ABS(H), OTHER THAN
THOSE BELOW ARE DESIRED, THEY SHOULD BE SET BELOW.

IN THE FOLLOWING STATEMENT, DETERMINE...
UROUND = THE UNIT ROUNDOFF OF THE MACHINE, I.E. THE SMALLEST
POSITIVE U SUCH THAT 1.0 + U .NE. 1.0 ON THE MACHINE.
CALL URND (URCUND)

C ALLOCATE STORAGE FOR ALL VECTORS REQUIRED IN ODEPAK... Y, YMAX,
C ERROR, SAVE1, SAVE2, SAVE3, PW, A, AND IPIV.
C THIS ALLOCATION IS PERFORMED BETWEEN HERE AND STATEMENT 54.

METH = MF/10
WITER = MF - 10 * METH
MADER = 12
IF (METH .EQ. 2) MADER = 5
MSIZE = MADER + 1
IYSIZ = N * MSIZE
IWORK(1) = IYSIZ + 1
IWORK(2) = IWORK(1) + N
IWORK(3) = IWORK(2) + N
IWORK(4) = IWORK(3) + N
IWORK(5) = IWORK(4) + N

C CC 10 I = 1,N
   WORK(IYSIZ + I) = DABS(YO(I))
   IF (WORK(IYSIZ + I) .EQ. 0.0) WCRK(IYSIZ + I) = 1.0CO
10   WORK(I) = YO(I)

NC = N
T = T0
H = HO
IF (T + H) .EQ. T) WRITE(15,15)
15 FCRMAT(35H WARNING.. T + H = T ON NEXT STEP.)
HMIN = DABS(HO)
HMAX = DABS(TO - TCLT) * 10.0D0
EPS = EPS
MFC = MF
JSTART = 0
NO = N
NW = N
NSQ = NO*NO
NOW = NSQ
EPSJ = D SQRT(URCUND)
NHCT = 0
NML = NO - 1
ITYPE = 0
JK = N
IF (ICASE .EQ. 1) GO TO 42
ML = ML
MUC = MU
NML = ML
NMU = ML
MW = ML + MU + 1
JK = MW

C FCR ICASE = 2 OR 3, CHECK FOR INCONSISTENT VALUES OF ML AND MU.
C FOR ICASE = 2,3 CHECK FOR INCONSISTENT VALUES OF ML AND MU.
C
IF (MW .GT. (N + N - 1)) GO TO 38C
KOML = NO * ML
NOW = NO * NW
IF (ICASE .EQ. 2) CC TC 45

C
C FCR ICASE = 3 CHECK FOR INVALID OR INCONSISTENT VALUES OF MLA

-59-
C AND MUA.
C
IF ((ML + MU) .EQ. (2 * NM1)) JK = N
IF ((ML + MU) .EQ. (2 * NM1)) NCW = NSQ
MLAC = MLA
MUAC = MLA
MWA = MLA + MUA + 1
IF ((MWA - 1) .EQ. 0) GO TO 385
IF (MWA .GT. (N + N - 1)) GO TO 350

C FOR ICASE = 3; IF THE MATRIX A IS FULL, ITYPE = 1; IF THE MATRIX A IS BANDED, ITYPE = 2.
C
IF ((MLA + MUA) .EQ. (2 * NM1)) ITYPE = 1
IF ((MLA + MUA) .LT. (2 * NM1)) ITYPE = 2
NML = MAX0(ML,MLA)
NMU = MAXC(MU,MUA)

C FOR ICASE = 3; IF THE SUM OF THE JACOBIAN MATRIX AND THE MATRIX A IS FULL, IFULL = 1; OTHERWISE, IFULL = 0.
C
IFULL = 1
IF ((NML + NMU) .NE. (2 * NM1)) IFULL = 0
GC TO 47

C TOUTF IS THE PREVIOUS VALUE CF TOUT FOR USE IN HMAX.
C
20  HMAX = DABS(TOUT-TOLTP)*10.0
GC TO 80
C
25  HMAX = DABS(TOUT-TCUTP)*10.0
IF ((T-TOUT)*H .GE. 0.0) GC TO 500
GC TO 85
C
30  IF ((T-TOUT)*H .GE. 0.0) GC TO 440
JSTART = -1
NC = N
EPSC = EPS
C
40  IF ((T+H) .EQ. T) WRITE(LOUT,15)
GC TO 55
C
FOR ICASE = 1, INITIALIZE ML = MU = N - 1.
C
42  ML = NM1
MU = NM1
NML = NM1
NMU = NM1
C
FOR ICASE = 1 OR 2, INITIALIZE MLA = MUA = 0.
C
45  MLA = 0
MUA = 0
MLAC = 0
MUAC = 0
C
47  IF (ICASE .EQ. 3) GC TO 51
NPW = 1
IF (MITER .EQ. 0) GC TO 52
NPW = N * N
-60-
IF (ICASE .EQ. 1) GO TO 52
IF (ICASE .EQ. 2) NPW = N * (2 * ML + MU + 1)
GO TO 52

C

51 NPW = N * (2 * MLA + MUA + 1)
IF (MITER .EQ. 0) GC TC 52
NPW = N * N
IF ((ML + MU) .EQ. (2 * NM1)) GO TO 52
IF ((MLA + MUA) .EQ. (2 * NM1)) NPW = MAXO(NSC, NOW)
IF ((MLA + MUA) .EQ. (2 * NM1)) GC TC 52
NPW = N * (2 * MAXO(ML, MLA) + MAXO(MU, MUA) + 1)

52 IWORK(6) = IWRK(5) + N
NA = 1
IF (ICASE .NE. 3) GC TC 53
NA = N * (MLA + MUA + 1)
IF (MITER .EQ. 0) GC TC 53
IF (((ML + MU) .EQ. (2 * NM1)) .AND. ((MLA + MUA) .NE. (2 * NM1)))
* GO TO 53

NA = N * N
IF ((MLA + MUA) .EQ. (2 * NM1)) GC TC 53
NA = N * (MAXO(ML, MLA) + MAXO(MU, MUA) + 1)

53 IWORK(7) = IWRK(6) + NPW
NIPIV = 1
IF (MITER .NE. 0) NIPIV = N
NWORK = IYSIZ + 5 * N + NPW + NA + NIPIV
NWORK = N + 7

C--------------------------------------------------------
C OUTPUT A MESSAGE TO THE USER INDICATING THE MINIMUM SIZE OF
C YO, WORK, AND IWORK THE MUST DIMENSION IN MAIN ROUTINE.
C--------------------------------------------------------

WRITE(6,541), NWORK, NWORK
54 FORMAT(48H TO AVOID POSSIBLE STORAGE OVERWRITE, USER MUST,
* 15H (AT A MINIMUM), /, 17 DIMENSIONS.. YO(I5, I5, 8H), WCRK(I6,
* 13H), AND IWORK(I5, 2H) )
55 CALL STIFF (NO, MSIZE, NPW, NA, NIPIV, WCRK, WORK(IWORK(1)),
* WORK(IWORK(2)), WCRK(IWORK(3)), WCRK(IWORK(4)),
* WORK(IWORK(5)), WORK(IWORK(6)), WCRK(IWORK(7)),
* IWORK(8))

C

KGO = 1 - KFLAG
GC TC (60, 100, 200, 300), KGC

C KFLAG = 0, -1, -2, -3

C--------------------------------------------------------
C NORMAL RETURN FROM INTEGRATOR.
C--------------------------------------------------------
C THE WEIGHTS YMAX(I) ARE UPDATED. IF DIFFERENT VALUES ARE DESIRED,
C THEY SHOULD BE SET HERE. A TEST IS MADE FOR EPS BEING TOO SMALL
C FOR THE MACHINE PRECISION.
C
C ANY OTHER TESTS OR CALCULATIONS THAT ARE REQUIRED AFTER EVERY
C STEP SHOULD BE INSERTED HERE.
C
C IF INDEX = 3, YO IS SET TO THE CURRENT Y VALUES ON RETURN.
C IF INDEX = 2, H IS CONTROLLED TO HIT TCUT (WITHIN RCUNDFFF
C ERROR), AND THEN THE CURRENT Y VALUES ARE PUT IN YO ON RETURN.
C FOR ANY OTHER VALUE OF INDEX, CONTROL RETURNS TO THE INTEGRATOR
C UNLESS TOUT HAS BEEN REACHED. THEN INTERPOLATED VALUES OF Y ARE
C COMPUTED AND STORED IN YO ON RETURN.
C IF INTERPOLATION IS NOT DESIRED, THE CALL TO INTERP SHOULD BE

-61-
C REMOVED AND CCNTROL TRANSFERRED TO STATEMENT 500 INSTEAD OF 520.
C
C D = 0.0
CC 70 I = 1, N
AYI = CABS(WORK(I))
WORK(IYSIZ + I) = DMAX1(WORK(IYSIZ + I), AYI)
70 D = D + (AYI / WCRK(IYSIZ + I))**2
D = D*(LRCUND/EPS)**2
IF (C * GT. DFLOAT(N)) GO TO 25C
IF (INDEX * EQ. 3) GO TO 500
IF (INDEX * EQ. 2) GO TO 85
80 IF ((T-TOUT)*H *LT. 0.0D0) GO TO 40
CALL INTERP (TOUT, WCRK, NO, YO)
GO TO 520
C
C5 IF (((T+H)-TOUT)*H *LE. 0.0D0) GO TO 40
IF (CABS(T-TOUT) *LE. 100.0D0*URCUND*HMAX) GO TO 500
IF (((T-TOUT)*H *GE. 0.0D0) GO TO 500
H = (TOUT - T)*(1.0D0 - 4.0D0*URCUND)
JSTART = -1
GO TO 40
C
C CN AN ERROR RETURN FROM INTEGRATOR, AN IMMEDIATE RETURN OCCURS IF
C KFLAG = -2, AND RECOVERY ATTEMPTS ARE MADE OTHERWISE.
C TO RECOVER, H AND HMIN ARE REDUCED BY A FACTOR OF .1 UP TO 10
C TIMES BEFORE GIVING UP.
C
100 WRITE (LOUT,105) T
105 FORMAT(/'35H KFLAG = -1 FROM INTEGRATOR AT T = ,E16.8/
* 39H ERROR TEST FAILED WITH CABS(H) = HMIN/
110 IF (NCUT *EQ. 10) GO TO 150
NCUT = NCUT + 1
HMIN = HMIN*.1D0
H = H*.1D0
WRITE (LOUT,115) H
115 F FORMAT('/24H H HAS BEEN REDUCED TO ,E16.8,
* 26H AND STEP WILL BE RETRIEVED/
JSTART = -1
GO TO 40
C
150 WRITE (LOUT,155)
155 F FORMAT('/44H PROBLEM APPEARS UNSOLVABLE WITH GIVEN INPUT/
GO TO 500
C
200 WRITE (LOUT,205) T,H
205 F FORMAT(/'35H KFLAG = -2 FROM INTEGRATOR AT T = ,E16.8,5H H =,
* E16.8/52F THE REQUESTED ERROR IS SMALLER THAN CAN BE HANDLED/
GO TO 500
C
250 WRITE (LOUT,255) T
255 F FORMAT(/'37H INTEGRATION HALTED BY CDEPAK AT T = ,E16.8/
* 56H EPS TOO SMALL TO BE ATTAINED FOR THE MACHINE PRECISION/
KFLAG = -2
GO TO 500
C
300 WRITE (LOUT,305) T
305 F FORMAT(/'35H KFLAG = -3 FROM INTEGRATOR AT T = ,E16.8/
* 45H CORRECTOR CONVERGENCE COULD NOT BE ACHIEVED/
GO TO 110
C
-62-
310 WRITE(LCUT,315)
315 FORMAT(//36H ILLEGAL INPUT.. INVALID VALUE OF MF//)
INDEX = -4
RETURN

C
380 WRITE(LCUT,381)
381 FORMAT(//48H ILLEGAL INPUT.. ML AND WU DC NCT CCNFORM WITH N//)
INDEX = -4
RETURN

C
385 WRITE(LOUT,386)
386 FORMAT(//52H ILLEGAL FCRM FOR MATRIX A.. MATRIX A IS A DIAGONAL ,
*6H:MATRX//)
INDEX = -4
RETURN

C
390 WRITE(LOUT,395)
395 FORMAT(//50H ILLEGAL INPUT.. MLA AND WLU DC NCT CCNFORM WITH N//)
INDEX = -4
RETURN

C
400 WRITE (LOUT,405)
405 FORMAT(//29H ILLEGAL INPUT.. EPS .LE. 0.//)
INDEX = -4
RETURN

C
410 WRITE (LOUT,415)
415 FORMAT(//25H ILLEGAL INPUT.. N .LE. C//)
INDEX = -4
RETURN

C
420 WRITE (LOUT,425)
425 FORMAT(//36H ILLEGAL INPUT.. (TC-TOUT)*H .GE. 0.//)
INDEX = -4
RETURN

C
430 WRITE (LOUT,435) INDEX
435 FORMAT(//24H ILLEGAL INPUT.. INDEX = ,I5//)
INDEX = -4
RETURN

C
440 WRITE(LOUT,445) T,TCUT,H
445 FORMAT(//44H INDEX = -1 CN INPUT WITH (T-TOUT)*H .GE. 0./
* 4H T =,E16.8,SH  TCUT =,E16.8,6T  H =,E16.8/
* 44H INTERPOLATION WAS DONE AS CN NORMAL RETURN./
* 41H DESIRED PARAMETER CHANGES WERE NCT MADE.)
CALL INTERP (TOUT, WORK, H0, Y0)
INDEX = -5
RETURN

C
500 TCUT = T
DC 510 I = 1,
510 Y0(I) = WORK(I)
520 INDEX = KFLAG
TCUTP = TCUT
H0 = HUSEC
IF (KFLAG .NE. 0) H0 = H
RETURN

C------------------------ END OF SUBROUTINE ODEPAK ------------------------
END

-63-
SUBROUTINE interp (t0t, y, nc, yc)
IMPLICIT REAL*8 (A-H,O-Z)
C
C SUBROUTINE interp completes interpolated values of the dependent
C variable y and stores them in yc. the interpolation is to the
C point t = t0t, and uses the Nordsieck History array y, as follows...
C
C yc(i) = sum y(i,j+1)*s**j ,
C                j=0
C
C WHERE s = -(t-t0t)/h.
C
C PACKAGE ROUTINES CALLED.. ncne
C USER ROUTINES CALLED.. ncne
C CALLED BY.. odepak
C FORTRAN FUNCTIONS USED.. none
C
C DIMENSION yc(nc,0),y(nc,1)
C COMMON /ODE1/ t,h,cummy(4),n,idummy(2),jstart
C
DC 10 i = 1,n
10     yc(i) = y(i,1)
DC 30 j = 2,l
S1 = 1.00
DC 20 i = 1,n
      yc(i) = yc(i) + s1*y(i,j)
      CONTINUE
C
RETURN
C------ END OF SUBROUTINE interp ----------------------
END
SUBROUTINE STIFF (NC, MSIZE, NPW, NA, NPIV, Y, YMAX, ERRCR, 
  SAVE1, SAVE2, SAVE3, PW, A, IPIV)
IMPLICIT REAL*8(A-H,0-Z)
C
C STIFF PERFORMS ONE STEP OF THE INTEGRATION OF AN INITIAL VALUE 
C PROBLEM FOR A SYSTEM OF ORDINARY DIFFERENTIAL EQUATIONS.
C STIFF IS A VERSION FOR THE FULL AND BANDED FORMALISM OF THE JACOBIAN 
C MATRIX, AND FOR THE IMPLICIT ODE WITH FULL OR BANDED JACOBIAN 
C MATRIX.
C
C COMMUNICATION WITH STIFF IS DONE WITH THE FOLLOWING VARIABLES...
C
C ICASE = TYPE OF ODE TO BE SOLVED.
C 1 DY/DT = G(Y,T), FULL JACOBIAN MATRIX
C 2 DY/DT = G(Y,T), BANDED JACOBIAN MATRIX
C 3 A*(DY/DT) = G(Y,T), FULL OR BANDED JACOBIAN MATRIX
C
C ITYPE = FORM OF MATRIX A USED WHEN ICASE = 3.
C 0 DEFAULT VALUE FOR ICASE = 1, 2.
C 1 A IS A FULL N BY N MATRIX.
C 2 A IS A BANDED N BY (ML+MU+1) MATRIX.
C
C Y AN N BY LMAX ARRAY CONTAINING THE DEPENDENT VARIABLES 
C AND THEIR SCALED DERIVATIVES. LMAX IS 13 FOR THE ADAMS 
C METHODS AND 6 FOR THE GEAR METHODS. LMAX - 1 = MAXDER 
C IS THE MAXIMUM ORDER AVAILABLE. SEE SUBROUTINE CSGET.
C Y(I,J+1) CONTAINS THE J-TH DERIVATIVE OF Y(I), SCALED BY 
C H**J/FACCTORIAL(J) (J = 1, ..., NC).
C
C NO A CONSTANT INTEGER GE N, USED FOR DIMENSIONING PURPOSES.
C T THE INDEPENDENT VARIABLE. T IS UPDATED ON EACH STEP TAKEN.
C
C H THE STEP SIZE TO BE ATTEMPTED ON THE NEXT STEP.
C H IS ALTERED BY THE ERROR CONTROL ALGORITHM DURING THE 
C PROBLEM. H CAN BE EITHER POSITIVE OR NEGATIVE, BUT ITS 
C SIGN MUST REMAIN CONSTANT THROUGHOUT THE PROBLEM.
C
C HMIN, HMAX THE MINIMUM AND MAXIMUM ABSOLUTE VALUE OF THE STEP SIZE 
C TO BE USED FOR THE STEP. THESE MAY BE CHANGED AT ANY 
C TIME, BUT WILL NOT TAKE EFFECT UNTIL THE NEXT H CHANGE.
C
C EPS THE RELATIVE ERROR BOUND. SEE DESCRIPTION IN ODEPAK.
C
C UROUND THE UNIT ROUND OFF OF THE MACHINE.
C N THE NUMBER OF FIRST-ORDER DIFFERENTIAL EQUATIONS.
C MF THE METHOD FLAG. SEE DESCRIPTION IN ODEPAK.
C KFLAG A COMPLETION CODE WITH THE FOLLOWING MEANINGS...
C 0 THE STEP WAS SUCCESSFUL.
C -1 THE REQUESTED ERRCR COULD NOT BE ACHIEVED 
C WITH ABS(H) = HMIN.
C -2 THE REQUESTED ERRCR IS SMALLER THAN CAN 
C BE HANDLED FOR THIS PROBLEM.
C -3 CORRECTOR CONVERGENCE COULD NOT BE 
C ACHIEVED FOR ABS(H) = HMIN.
C
C ON A RETURN WITH KFLAG NEGATIVE, THE VALUES OF T AND 
C THE Y ARRAY ARE AS OF THE BEGINNING OF THE LAST 
C STEP, AND H IS THE LAST STEP SIZE ATTEMPTED.
C
C JSTART AN INTEGER USED ON INPUT AND OUTPUT.
C ON INPUT, IT HAS THE FOLLOWING VALUES AND MEANINGS...
C 0 PERFORM THE FIRST STEP.
C +1 TAKE A NEW STEP CONTINUING FROM THE LAST.
C +2 TAKE THE NEXT STEP WITH A NEW VALUE OF 
C H, EPS, N, AND/OR MF.
C
C ON EXIT, JSTART IS NC, THE CURRENT ORDER OF THE METHOD.
C
C YMAX AN ARRAY OF N ELEMENTS WITH WHICH THE ESTIMATED LOCAL 
C ERRORS IN Y ARE COMPARED.
C
ERROR
AN ARRAY OF N ELEMENTS. ERRCR(I)/TQ(2) IS THE ESTIMATED
ONE-STEP ERROR IN Y(I).
SAVE1, SAVE2, SAVE3 THREE WORKING STORAGE ARRAYS, EACH OF LENGTH N.
PW A BLOCK OF LOCATIONS USED FOR PARTIAL DERIVATIVES IF
MTER IS NOT 0. SEE DESCRIPTION IN ODEPAK.
IPIV AN INTEGER ARRAY OF LENGTH N USED FOR PIVOTING
INFORMATION IF MTER = 1 OR 2.
ML, NU THE LOWER AND UPPER HALves BANDWIDTHS, RESPECTIVELY, OF THE
CHORD ITERATION MATRIX. SEE DESCRIPTION IN ODEPAK.
MLA, MLA THE LOWER AND UPPER HALVES BANDWIDTHS, RESPECTIVELY, OF THE A MATRIX; USED WHEN ICASE = 3. SEE DESCRIPTION IN
ODEPAK.

PACKAGE ROUTINES CALLED.. COSET, DIFFUN, FMAT, RES, SCL
USER ROUTINES CALLED.. GFUN
CALLED BY.. CDEPAK
FORTRAN FUNCTIONS USED.. ABS, DABS, FLCAT, DFCAT, MAX1, DMAX1,
MIN1, DMIN1

DIMENSION Y(N, MSIZE), YMAX(N0), ERROR(N0), SAVE1(NC), SAVE2(N0)
DIMENSION SAVE3(N0), PW(NPW), A(NA), IFIV(NIPIV)
DIMENSION EL(13), TQ(19)
COMMON /ODE1/ T, H, HMIN, HMAX, EPS, URFUNC, N, M, KFLAG, JSTART
COMMON /ODE2/ EPSJ, NSC, MLC, MUC, PW, NM1, NOML, NCK, MLA, MLA, MLA
COMMON /ODE3/ HUSE, NCUSE, NSTEP, NGE, NJE
COMMON /ODE4/ ICASE, ITYPE, JK, NML, NML, IFULL
DATA EL(2)/1.0D0/, OLDLO/1.0D0/

KFLAG = 0
TDLO = T
IF (JSTART.GT.0) GO TO 200
IF (JSTART.NE.0) GO TO 120

CN THE FIRST CALL, THE ORDER IS SET TO 1 AND THE INITIAL YDOT IS
CALCULATED. RMAX IS THE MAXIMUM RATIO BY WHICH H CAN BE INCREASED
IN A SINGLE STEP. IT IS INITIALLY 1.E4 TO COMPENSATE FOR THE SMALL
INITIAL H, BUT THEN IS NORMALLY EQUAL TO 10. IF A FAILURE
OCcURS (IN CORRECTOR CONVERGENCE OR ERRCR TEST), RMAX IS SET AT 2
FOR THE NEXT INCREASE.

CALL GFUN (N, T, Y, SAVE1)
ANN = NSQ
IF (ICASE .EQ. 2) ANN = KNOW
IF (ICASE .EQ. 3) GC TO 105
DO 100 I = 1,ANN
100 PW(I) = 0.0D0
105 IF (ICASE .EQ. 3) CALL DIFFUN (N, T, Y, SAVE1, A, PW, IPIV)
DO 110 I = 1, N
110 Y(I,2) = H*SAVE1(I)
METH = MF/10
MTER = MF - 10*METH
NC = 1
L = 2
IDQUB = 3
RMAX = 1.0D04
RC = 0.0D0
CRATE = 1.0D0
EPSCLD = EPS
HOLD = T
C IF THE CALLER HAS CHANGED METH, CCSET IS CALLED TO SET
C THE COEFFICIENTS OF THE METHOD. IF THE CALLER HAS CHANGED
C N, EPS, OR METH, THE CONSTANTS E, EDA, EUP, AND BND MUST BE RESET.
C E IS A COMPARISON FOR ERRORS OF THE CURRENT ORDER NG. EUP IS
C TO TEST FOR INCREASING THE ORDER, EDA FOR DECREASING THE ORDER.
C BND IS USED TO TEST FOR CONVERGENCE OF THE CORRECTOR ITERATES.
C IF THE CALLER HAS CHANGED H, Y MUST BE RESCALED.
C IF H OR METH HAS BEEN CHANGED, ICCUB IS RESET TO L + 1 TO PREVENT
C FURTHER CHANGES IN H FOR THAT MANY STEPS.
C
120 IF (MF .EQ. MFAIL) GO TO 150
     MEO = METH
     MIO = MITER
     METH = MF/10
     MITER = MF - 10*METH
     MFAIL = MF
     IF (MITER .NE. MIO) IMOVAL = MITER
     IF (METH .EQ. MEC) GO TO 150
     IDOUB = L + 1
     IRET = 1
130 CALL CCSET (METH, NG, EL, TG, MAXCER)
     LMAX = MAXDER + 1
     RC = RC*EL(1)/OLDLO
     OLDLO = EL(1)
140 FN = DFNCAT(N)
     EDN = FAA*(TQ(1)*EPS)**2
     E = FAA*(TQ(2)*EPS)**2
     EUP = FAA*(TQ(3)*EPS)**2
     BND = FAA*(TQ(4)*EPS)**2
     GO TO (160, 170, 200), IRET
C
C 150 IF ((EPS .EQ. EPSCLC) .AND. (NG .EQ. NCLD)) GO TO 160
     EPSOLD = EPS
     NCLD = NG
     IRET = 1
     GC TO 140
C
C 160 IF (H .EQ. HCLD) GC TO 200
     RF = H/FCLD
     H = HCLD
     IREDD = 3
     GC TO 175
C
C 170 RH = DMAX1(RH, HMIN/CABS(H))
C 175 RH = DMIN1(RH, HMAX/CABS(H), RMAX)
     R1 = 1.0
     DO 180 J = 2, L
         R1 = R1*RH
         DO 180 I = 1, N
180     Y(I, J) = Y(I, J)*R1
     -67-
\[ H = H^*RH \]
\[ RC = RC^*RH \]
\[ IDOUB = L + 1 \]

IF (IRECO .EQ. 0) GC TC 690

C-----------------------------------------------
C THIS SECTION COMPUTES THE PREDICTED VALUES BY EFFECTIVELY
C MULTIPLYING THE \( Y \) ARRAY BY THE PASCAL TRIANGLE MATRIX.
C RC IS THE RATIO OF NEW TO OLD VALUES OF THE COEFFICIENT \( H^*EL(1) \).
C WHEN RC DIFFERS FROM 1 BY MORE THAN 30 PERCENT, OR THE CALLER HAS
C CHANGED MITER, IWEVAL IS SET TO MITER TO FORCE THE PARTIALS, \( P \), TO
C BE UPDATED, IF PARTIALS ARE USED. IN ANY CASE, THE PARTIALS
C ARE UPDATED AT LEAST EVERY 20-TH STEP. FOR ICASE = 1, 2;
C \( P = 1 - H^*EL(1)^*(DG/DY) \). FOR ICASE = 3, \( P \) IS THE \( C^*CRC \) ITERATION MATRIX
C \( A = -H^*EL(1)^*(DG/DY) \).

\[ 200 \]
IF (CABS(RC - 1.0D0) .GT. 0.3D0) IWEVAL = MITER
IF (NSTEP .GE. NSTEFJ+20) IWEVAL = MITER
T = T + H
CC 210 J1 = 1, NQ
DO 210 J2 = J1, NQ
   J = (NQ + J1) - J2
   DO 210 I = 1, N
      Y(I, J) = Y(I, J) + Y(I, J+1)

C-----------------------------------------------
C UP TO 3 CORRECTOR ITERATIONS ARE TAKEN. A CONVERGENCE TEST IS
C MADE ON THE R.M.S. NORM \( \| \) OF EACH CORRECTION, USING BND, WHICH
C IS DEPENDENT ON EPS. THE SUM OF THE CORRECTIONS IS ACCUMULATED
C IN THE VECTOR ERROR(I). THE \( Y \) ARRAY IS NOT ALTERED IN THE CORRECTOR
C LOOP. THE UPDATED \( Y \) VECTOR IS STORED TEMPORARILY IN SAVE1.
C THE UPDATED \( H^*Y \) VECTOR IS STORED IN SAVE2.

\[ 220 \]
CC 230 I = 1, N
   SAVE2(I) = Y(I, 2)

\[ 230 \]
ERROR(I) = 0.0D0
M = 0
CALL RES(N, T, H, A, Y, SAVE2, SAVE3)
NGE = NGE + 1
IF (IWEVAL .LE. 0) GO TO 290

C-----------------------------------------------
C IF INDICATED, THE MATRIX \( P \) IS REEVALELATED BEFORE STARTING THE
C CORRECTOR ITERATION. IWEVAL IS SET TO 0 AS AN INDICATOR THAT THIS
C HAS BEEN DONE. \( P \) IS COMPUTED AND PROCESSED IN PSET.
C
C IWEVAL = 0
RC = 1.00
NGE = NGE + 1
NSTEFJ = NSTEP
IF (ICASE .EQ. 3) GO TO 250
GO TO (250, 240), MITER

C-----------------------------------------------
C IF ICASE = 1, JK = L
C IF ICASE = 2, JK = Mh
C
\[ 240 \]
NGE = NGE + JK
\[ 250 \]
CON = -1.00D0 * H * EL(1)
CALL FMAT(Y, A, N0, CCA, MITER, IER, YMAX, ERROR, SAVE1, SAVE2,
* PH, IPIV)
IF (IER .NE. 0) GO TO 42C
GO TO 350

-68-
290 IF ((MITER .EQ. 0) .AND. (ICASE .NE. 3)) GO TO 370

C COMPUTE THE CORRECTOR ERROR, R SLB (M), AND SOLVE THE LINEAR
C SYSTEM WITH THAT AS THE RIGHT-HAND SIDE AND P AS THE COEFFICIENT
C MATRIX, USING THE LU DECOMPOSITION IF MITER = 1 OR 2.
C
350 CALL SCL (NO, N, NPL, NPU, PW, SAVE3, IPIV)
370 D = 0.0
DC 380 I = 1,N
   ERROR(I) = ERRCR(I) + SAVE3(I)
   D = D + (SAVE3(I) / YMAX(I))**2
   SAVE1(I) = Y(I,1) + EL(I) * ERROR(I)
380 SAVE2(I) = Y(I,2) + ERROR(I)
C
C TEST FOR CONVERGENCE. IF K .GT. 0, AN ESTIMATE OF THE CONVERGENCE
C RATE CONSTANT IS STORED IN CRATE, AND THIS IS USED IN THE TEST.
C
460 IF (M .NE. 0) CRATE = DMAX1(1.,9DO*CRATE,D/D1)
   IF ((D*DMINI(1.,DO,2.,DC*CRATE)) .LE. BND) GC TC 450
   D1 = D
   M = M + 1
IF (M .EQ. 3) GO TO 410
CALL RES (N, T, H, A, SAVE1, SAVE2, SAVE3)
GO TO 250
C
C THE CORRECTOR ITERATIONS FAILED TO CONVERGE IN 3 TRIES. IF PARTIALS
C ARE INVOLVED BUT ARE NOT UP TO DATE, THEY ARE REEVALUATED FOR THE
C NEXT TRY. OTHERWISE THE Y ARRAY IS RETRACTED TO ITS VALUES
C BEFORE PREDICTION, AND H IS REDUCED, IF POSSIBLE. IF NOT, A
C NO-CONVERGENCE EXIT IS TAKEN.
C
410 NGE = NGE + 2
   IF (IWEVAL .EQ. -1) GC TO 440
420 T = TOLD
   RMAX = 2.0
DC 430 J1 = 1,N
DC 430 J2 = J1,NQ
   J = (NQ + J1) - J2
DC 430 I = 1,N
430 Y(I,J) = Y(I,J) - Y(I,J+1)
   IF (CABS(H) .LE. -PROC*1.0000001DC) GO TO 680
   RH = .25CC
   IREDO = I
GO TO 170
C
440 IWEVAL = MITER
GC TO 220
C
C THE CORRECTOR HAS CONVERGED. IWEVAL IS SET TO -1 IF PARTIAL
C DERIVATIVES WERE USED, TO SIGNAL THAT THEY MAY NEED UPDATING ON
C SUBSEQUENT STEPS. THE ERROR TEST IS MADE AND CONTROL PASSES TO
C STATEMENT 500 IF IT FAILS.
C
450 IF (MITER .NE. 0) IWEVAL = -1
   NGE = NGE + M
   D = 0.0
DC 460 I = 1,N
460 D = D + (ERROR(I)/YMAX(I))**2
   IF (D .GT. E) GC TC 500

-69-
C --- A SUCCESSFUL STEP, UPDATE THE Y ARRAY.
C CONSIDER CHANGING H IF IDCUB = 1. OTHERWISE DECREASE IDCUB BY 1.
C IF IDCUB IS THEN 1 AND NC .LT. MAXCR, THEN ERROR IS SAVED FOR
C USE IN A POSSIBLE ORDER INCREASE ON THE NEXT STEP.
C IF A CHANGE IN H IS CONSIDERED, AN INCREASE OR DECREASE IN ORDER
C BY ONE IS CONSIDERED ALSO. A CHANGE IN H IS MADE ONLY IF IT IS BY A
C FACTOR OF AT LEAST 1.1. IF NOT, IDCUB IS SET TO 10 TO PREVENT
C TESTING FOR THAT MANY STEPS.

KFLAG = 0
IREDC = 0
NSTEP = NSTEP + 1
HUSED = H
NQUSED = NQ
DO 470 J = 1,L
    DO 470 I = 1,N
        470 Y(I,J) = Y(I,J) + EL(J)*ERRCR(I)
    IF (IDCUB .EQ. 1) GO TO 520
    IDCUB = IDCUB - 1
    IF (IDCUB .GT. 1) GO TO 700
    IF (L .EQ. LMAX) GO TO 700
    DC 490 I = 1,N
    490 Y(I,LMAX) = ERRCR(I)
    GO TO 700

C --- THE ERROR TEST FAILED. KFLAG KEEPS TRACK OF MULTIPLE FAILURES.
C RESTORE T AND THE Y ARRAY TO THEIR PREVIOUS VALUES, AND PREPARE
C TO TRY THE STEP AGAIN. COMPUTE THE CURRENT STEP SIZE FOR THIS OR
C ONE LOWER ORDER.

500 KFLAG = KFLAG - 1
    T = TOLD
    DC 510 J1 = 1,NQ
    DC 510 J2 = J1,NQ
    J = (NQ + J1) - J2
    DO 510 I = 1,N
        510 Y(I,J) = Y(I,J) - Y(I,J+1)
    RMAX = 2.00
    IF (DABS(H) .LE. HKMIN*1.00001CC) GO TO 660
    IF (KFLAG .LE. -3) GO TO 640
    IREDC = 2
    PR3 = 1.020
    GO TO 540

C --- REGARDLESS OF THE SUCCESS OR FAILURE OF THE STEPS, FACTORS
C PR1, PR2, AND PR3 ARE COMPUTED, BY WHICH H COULD BE DIVIDED
C AT ORDER NC - 1, ORDER NC, OR ORDER NC + 1, RESPECTIVELY.
C IN THE CASE OF FAILURE, PR3 = 1.E20 TO AVOID AN ORDER INCREASE.
C THE SMALLEST OF THESE IS DETERMINED AND THE NEW ORDER CHOSEN
C ACCORDINGLY. IF THE ORDER IS TO BE INCREASED, WE COMPUTE ONE
C ADDITIONAL SCALED DERIVATIVE.

520 PR3 = 1.E20
    IF (L .EQ. LMAX) GO TO 540
    D1 = 0.00
    DC 530 I = 1,N
      530 D1 = C1 + ((ERRCR(I) - Y(I,LMAX))/YMAX(I))**2
          ENQ3 = .5C0/DFLOAT(L+1)
          PR3 = ((D1/EUP)**ENQ3)*1.40D0 + 1.40-06

---
ENQ2 = .5CC/DFLOAT(L)
PR2 = ((D/E)**ENQ2)*1.2D0 + 1.2C-06
PR1 = 1.0D2C
IF (NG .EQ. 1) GO TO 560
C = C*DC
DO 550 I = 1,N
D = D + (Y(I,L)/YMAX(I))**2
ENQ1 = .5DO/DFLCAT(NQ)
PR1 = ((D/EDN)**ENQ1)*1.3D0 + 1.3C-06
560 IF (PR2 .LE. PR3) GC TO 570
IF (PR3 .LT. PR1) GC TO 550
GC TO 580
C
570 IF (PR2 .GT. PR1) GC TO 560
NEWQ = NQ
RH = 1.0C/PR2
GC TO 620
C
580 NEWQ = NQ - 1
RH = 1.0C/PR1
GC TO 620
C
590 NEWQ = L
RH = 1.0D/PR3
IF (RH .LT. 1.0D0) GO TO 610
CC 600 I = 1,N
600 Y(I,NEWQ+1) = ERRCR(I)*EL(L)/DFLOAT(L)
GO TO 630
C
610 IDOUB = 10
GO TO 700
C
620 IF (((KFLAG .EQ. 0) .AND. (RH .LT. 1.1C0)) GC TO 610
C-----------------------------------------
C IF THERE IS A CHANGE OF CRDER, RESET NQ, L, AND THE COEFFICIENTS.
C IN ANY CASE RH IS RESET ACCORDING TO RH AND THE Y ARRAY IS RESCALED.
C THEN EXIT FROM 690 IF THE STEP WAS OK, OR REDO THE STEP OTHERWISE.
C-----------------------------------------
C IF (NEWQ .EQ. NQ) GC TO 170
630 NQ = NEWQ
L = NQ + 1
IRET = 2
GC TO 130
C-----------------------------------------
C CONTROL REACHES THIS SECTION IF 3 OR MORE FAILURES HAVE OCCURRED.
C IT IS ASSUMED THAT THE DERIVATIVES THAT HAVE ACCUMULATED IN THE
C Y ARRAY HAVE ERRORS OF THE WRONG ORDER. HENCE THE FIRST
C DERIVATIVE IS RECOMPUTED, AND THE CRDER IS SET TO 1. THEN
C RH IS REDUCED BY A FACTOR OF 10, AND THE STEP IS RETRIED.
C AFTER A TOTAL OF 7 FAILURES, AN EXIT IS TAKEN WITH KFLAG = -2.
C-----------------------------------------
640 IF (KFLAG .EQ. -7) GO TO 67C
RH = .1C0
RH = DMAX1(HMIN/DABS(H),RH)
H = H*RH
IF (ICASE .NE. 3) NGE = NGE + 1
CALL GFUN (N, T, Y, SAVE1)
IF (ICASE .EQ. 3) CALL DIFFUN (N, T, Y, SAVE1, A, PW, IPIV)
IF (ICASE .EQ. 3) NJE = NJE + 1
DO 650 I = 1,N
650
Y(1,2) = H*SAVE1(I)
IWEVAL = MITER
I0CUB = 10
IF (NQ .EQ. 1) GO TO 200
NC = 1
L = 2
IRET = 3
GO TO 130

C ALL RETURNS ARE MADE THROUGH THIS SECTION. H IS SAVED IN HCLD
C TO ALLOW THE CALLER TO CHANGE H ON THE NEXT STEP.

660 KFLAG = -1
GO TO 700

C
670 KFLAG = -2
GO TO 700

C
680 KFLAG = -3
GO TO 700

C
690 RMAX = 10.00
700 HCLD = H
JSTART = NQ
RETURN

C------------------------ END OF SUBROUTINE STIFF ------------------------
END
SUBROUTINE CCSET (METH, NQ, EL, TC, MAXDER)
IMPLICIT REAL*8(A-H,O-Z)
REAL PERST

C
C CCSET IS CALLED BY STIFF AND SETS COEFFICIENTS FOR USE THERE.
C THE VECTOR EL, OF LENGTH NQ + 1, DETERMINES THE BASIC METHOD.
C THE VECTOR TQ, OF LENGTH 4, IS INVOLVED IN ADJUSTING THE STEP SIZE
C IN RELATION TO TRUNCATION ERROR. ITS VALUES ARE GIVEN BY THE
C PERST ARRAY.
C THE VECTORS EL AND TQ DEPEND ON METH AND NQ.
C CCSET ALSO SETS MAXDER, THE MAXIMUM ORDER OF THE METHOD AVAILABLE.
C CURRENTLY THIS IS 12 FOR THE ADAMS AND 5 FOR THE BDF METHODS.
C LMAX = MAXDER + 1 IS THE NUMBER OF COLUMNS IN THE Y ARRAY.
C THE MAXIMUM ORDER USED MAY BE REDUCED SIMPLY BY CHANGING THE
C THE NUMBERS IN STATEMENTS 1 AND 2 BELOW.
C
C THE COEFFICIENTS IN PERST NEED BE GIVEN TO ONLY ABOUT
C ONE PERCENT ACCURACY. THE ORDER IN WHICH THE GROUPS APPEAR BELOW
C IS: COEFFICIENTS FOR ORDER NQ - 1, COEFFICIENTS FOR ORDER NQ,
C COEFFICIENTS FOR ORDER NQ + 1. WITHIN EACH GROUP ARE THE
C COEFFICIENTS FOR THE ADAMS METHODS, FOLLOWED BY THOSE FOR THE
C GEAR METHODS. SEE REFERENCE 3 FOR DETAILS.
C
C PACKAGE ROUTINES CALLED... NONE
C USER ROUTINES CALLED... NONE
C CALLED BY... STIFF
C FORTRAN FUNCTIONS USED... DBLE,FLOAT,CFLOT

C
C DIMENSION PERST(12,2,3),EL(13),TC(4)
C DATA PERST / 1.0EC,1.0EC,2.0EC,1.0EC,3.0EC,4.0EC,5.0EC,6.0EC,7.0EC,8.0EC,9.0EC,10.0EC/

C GO TO (1,2), METH
C
1 MAXDER = 12
C GO TO (101,102,103,104,105,106,107,108,109,110,111,112), NQ
C
2 MAXDER = 5
C GO TO (201,202,203,204,205), NQ

C THE FOLLOWING COEFFICIENTS SHOULD BE DEFINED TO MACHINE ACCURACY.
C FOR A GIVEN ORDER NQ, THEY CAN BE CALCULATED BY USE OF THE
C GENERATING POLYNOMIAL L(T), WHOSE COEFFICIENTS ARE EL(I).
C L(T) = EL(1) + EL(2)*T + ... + EL(NQ+1)*T**NQ.
C FOR THE IMPLICIT ADAMS METHODS, L(T) IS GIVEN BY
C DL/DT = (T+1)*(T+2)*...*(T+NQ-1)/K, L(-1) = C,
C WHERE
C K = FACTCRL(NQ-1).
C FOR THE GEAR METHODS,
L(T) = (T+1)*(T+2)* ... *(T+NQ)/K,
WHERE K = FACTORIAL(NQ)*(1 + 1/2 + ... + 1/NQ).

THE ORDER IN WHICH THE GROUPS APPEAR BELOW IS:
IMPLICIT ADAMS METHODS OF ORDERS 1 TO 12,
BACKWARD DIFFERENTIATION METHODS OF ORDERS 1 TO 5.

101 EL(1) = 1.00D0
GO TO 900

102 EL(1) = 0.50D0
EL(3) = 0.50D0
GO TO 900

103 EL(1) = 4.1666666666666667D-01
EL(3) = 0.75D0
EL(4) = 1.6666666666666667D-01
GO TO 900

104 EL(1) = 0.375D0
EL(3) = 9.1666666666666667D-01
EL(4) = 3.333333333333333D-01
EL(5) = 4.1666666666666667D-02
GO TO 900

105 EL(1) = 3.4861111111111111D-01
EL(3) = 1.0416666666666667D0
EL(4) = 4.611111111111111D-01
EL(5) = 1.0416666666666667D-01
EL(6) = 8.333333333333333D-03
GO TO 900

106 EL(1) = 3.2986111111111111D-01
EL(3) = 1.14166666666666667D0
EL(4) = 0.625D+00
EL(5) = 1.770833333333333D-01
EL(6) = 0.025D+00
EL(7) = 1.388888888888888D-03
GO TO 900

107 EL(1) = 3.1559193121693122D-01
EL(3) = 1.225D+00
EL(4) = 7.5185185185185185D-01
EL(5) = 2.552083333333333D-01
EL(6) = 4.861111111111111D-02
EL(7) = 4.861111111111111D-03
EL(8) = 1.9841269841269841D-04
GO TO 900

108 EL(1) = 3.0422453703703704D-01
EL(3) = 1.2964257142571425D+00
EL(4) = 8.6851651851851851D-02
EL(5) = 3.35783888888888889D-01
EL(6) = 7.77777777777777778D-02
EL(7) = 1.0648148148148148D-02
EL(8) = 7.9365079365079365D-04
EL(9) = 2.48015873C15E73C2D-05
GO TO 900

109 EL(1) = 2.5486860G04400171D-01
EL(3) = 1.3589285714285714D+00
EL(4) = 9.76554238423280D+01
EL(5) = 0.4171875D+00
EL(6) = 1.1125416666666667D-01
EL(7) = 0.01875D+00
EL(8) = 1.9345238055238055D-03
EL(9) = 1.1160714257142571D-04
EL(10) = 2.7557319223985891D-06
GO TO 900

C
110  EL(1) = 2.8697544642857143D-01
EL(3) = 1.4144841268584127D+00
EL(4) = 1.0772156044656685D+00
EL(5) = 4.9856701940035273D-01
EL(6) = 0.1484375D+00
EL(7) = 2.506357037654321D-02
EL(8) = 3.7202380952380952D-03
EL(9) = 2.996596566604656D-04
EL(10) = 1.3778659611992945D-05
EL(11) = 2.7557319223985891D-07
GO TO 900

C
111  EL(1) = 2.801895644393672D-01
EL(3) = 1.4644841265564127D+00
EL(4) = 1.1715145502645503D+00
EL(5) = 5.79358190C3527337D-01
EL(6) = 1.883226155202822D-01
EL(7) = 4.1430362654320988D-02
EL(8) = 6.211441798541795D-03
EL(9) = 6.2520667985417989D-04
EL(10) = 4.0417401528512640D-05
EL(11) = 1.5156525573192240D-06
EL(12) = 2.5052108385441719D-08
GO TO 900

C
112  EL(1) = 2.74265540C3155966D-01
EL(3) = 1.509536724386724D+00
EL(4) = 1.2602711664021164D+00
EL(5) = 6.59234182C987543D-01
EL(6) = 2.30458002645502650D-01
EL(7) = 5.5697261C522216D-02
EL(8) = 9.4394841265841270D-03
EL(9) = 1.1192749665312169D-03
EL(10) = 5.09391534353439D-05
EL(11) = 4.8225308641575309D-06
EL(12) = 1.5031265021265031D-07
EL(13) = 2.087675667668695D-08
GO TO 900

C
201  EL(1) = 1.0D+00
GO TO 900

C
202  EL(1) = 6.666666666666666D-01
EL(3) = 3.333333333333333D-01
GO TO 900

C
203  EL(1) = 5.454545454545454D-01
EL(3) = EL(1)
EL(4) = 5.09090909C9C9D9D-02
-75-
GC TO 900

C
204  EL(1) = 0.48D+00
     EL(3) = 0.7D+00
     EL(4) = 0.2D+00
     EL(5) = 0.02D+00
     GC TO 900

C
205  EL(1) = 4.375562043756204D-01
     EL(3) = 8.2116788321167883D-01
     EL(4) = 3.1021857810218578D-01
     EL(5) = 5.4744525547445255D-02
     EL(6) = 3.6496350364963504D-03

C
900  DC 910 K = 1,3
910   TG(K) = DBLE(PERTST(NQ,METH,K))
     TG(4) = 0.5D0 * TG(2)/DFLOAT(NQ + 2)
     RETURN

C---------------------------------- END OF SUBROUTINE COSET ----------------------------------
END
SUBROUTINE RES (N, T, H, A, Y, V, R)
IMPLICIT REAL*8(A-H, O-Z)

C-----------------------------
C RES COMPUTES THE RESIDUAL VECTOR R = H * G(Y,T) - A(Y,T) * V
C GIVEN Y AND T, WHERE V IS A GIVEN VECTOR AND H IS A GIVEN STEP SIZE.
C CALLED FOR ALL VALUES CF MF.
C
C FCRT ITYPE = 0, A IS THE IDENTITY MATRIX.
C FCRT ITYPE = 1, A IS A FULL MATRIX.
C FOR ITYPE = 2, A IS A BANDED MATRIX WITH UPPER AND LOWER BAND-
C WIDTHS, MLA AND MUA, RESPECTIVELY. THE TOTAL
C BANDWITH IS MWA = MLA+MUA+1.
C
C PACKAGE ROUTINES CALLED. NONE
C USER ROUTINES CALLED. GFUN
C CALLED BY. STIFF
C FORTRAN FUNCTIONS USED. NONE
C-----------------------------

DIMENSION A(1), Y(1), V(1), R(1)
COMMON /ODE2/ EPSJ, NSQ, MLC, MUC, MW, NM1, NOML, NOW, MLA, MUA, MWA
COMMON /ODE4/ ICASE, ITYPE, JK, NML, NMU, IFULL

CALL GFUN (N, T, Y, R)
IT1 = ITYPE + 1
GO TO (10, 30, 50), IT1

C-----------------------------
C THE MATRIX A IS THE IDENTITY MATRIX; SC THE DESIRED RESULT IS
C R = H * G(Y,T) - V.
C-----------------------------
10 CO 20 I = 1, N
20 R(I) = H * R(I) - V(I)
RETURN

C-----------------------------
C THE MATRIX A IS A FULL MATRIX; SC THE DESIRED RESULT IS
C R = H * G(Y,T) - A(Y,T) * V.
C-----------------------------
30 DO 40 I = 1, N
40 R(I) = H * R(I)
30 DO 40 J = 1, N
40 R(I) = R(I) - A(I + N * (J - 1)) * V(J)
RETURN

C-----------------------------
C THE MATRIX A IS A BANDED MATRIX; SC THE DESIRED RESULT IS
C R = H * G(Y,T) - A(Y,T) * V.
C-----------------------------
50 IF = N * MWA
   IJ = MLA * N + 1
50 DO 70 I = 1, N
60 R(I) = H * R(I)
50 DO 70 J = 1, N
   IK = IJ + N * (J - 1)
   IF (IK .LT. 1) GO TO 60
   IF (IK .GT. IP) GC TO 60
   R(I) = R(I) - A(IK) * V(J)
60 CONTINUE
7C  IJ = IJ - N + 1
C
    RETURN
C------------------ END OF SUBROLTINE RES ------------------
END
SUBROUTINE PMAT (Y, A, NO, CCN, MITER, IER, YMAX, ERROR, SAVEL1,
     * SAVE2, PW, IPIV)
IMPLICIT REAL*8(A-H,O-Z)

PMAT IS CALLED TO COMPLETE AND PROCESS THE MATRIX
P = A - l*EL(1)*J, WHERE J IS AN APPROXIMATION TO THE JACOBIAN
(NCTE.. FOR ICASE = 1; A = I). MATRICES A AND J = DG/DY MAY
EACH BE TREATED IN EITHER FULL OR BANDED FORM. J IS COMPUTED,
EITHER BY THE USER-SUPPLIED ROUTINE JMAT IF MITER = 1, OR BY FINITE
DIFFERING IF MITER = 2. J IS STORED IN PW AND IS REPLACED BY P,
USING CON = -H*EL(1). THEN P IS SUBJECTED TO LU DECOMPOSITION
IN PREPARATION FOR LATER SOLUTION OF LINEAR SYSTEMS WITH P AS THE
COEFFICIENT MATRIX.

IN ADDITION TO VARIABLES DESCRIBED PREVIOUSLY, COMMUNICATION
WITH PMAT USES THE FOLLOWING:
EPSJ = SQRT(UROUND), USED IN THE NUMERICAL JACOBIAN INCREMENTS.
MW = ML + MU + 1.
NM1 = NO - 1.
NOML = NO * ML.
NCW = NO * MW.

ICASE = TYPE OF O.C.E. TO BE SOLVED.
1 DY/DT = G(Y,T), FULL JACOBIAN MATRIX
2 DY/DT = G(Y,T), BANDED JACOBIAN MATRIX
3 A*(DY/DT) = G(Y,T), FULL OR BANDED JACOBIAN MATRIX

PACKAGE ROUTINES CALLED. . ADDA,CEC
USER ROUTINES CALLED. . GFUN,JMAT
CALLED BY. . . . . . . . STIFF
FORTRAN FUNCTIONS USED. . . ABS(CABS),MAXO,MAX1(DMAX1),MINO,
SQT(DSCRT)

DIMENSION Y(NO,1),A(1),YMAX(1),ERROR(1),SAVEM1(1),SAVE2(1)
DIMENSION PW(1),IPIV(1)
COMM CN /CDE1/ T,H,DUMM(3),URCUND,N,ICUMM(3)
COMM CN /CDE2/ EPSJ,NSQ,ML,MU,MW,NM1,NOML,NOW,MLA,MUA,MWA
COMM CN /CDE4/ ICASE,ITYPE,JK,NML,NMU,IFULL

IF (MITER .EQ. 2) GO TO 20

IF MITER = 1, CALL JMAT AND MULTIPLY BY SCALAR.

CALL JMAT (N, ML, MU, T, Y, PW)

FOR ICASE = 1 AND ICASE = 3 WITH FULL JACOBIAN, NOW = NSQ.

DO 10 I = 1,NOW
10 PW(I) = PW(I) * CCN
GO TO 140

IF ICASE = 1 AND MITER = 2, MAKE N+1 CALLS TO GFUN TO APPROX. J.
IF ICASE = 2 AND MITER = 2, MAKE MW+1 CALLS TO GFUN TO APPROX. J.
IF ICASE = 3 AND MITER = 2, MAKE MW+1 CALLS TO GFUN TO APPROX. J.

20 CALL GFUN (N, T, Y, SAVE2)
   D = 0.000
   DO 30 I = 1,N
   30 CONTINUE

-79-
30     D = D + SAVE2(I)**2
RO=DABS(H)*DSQRT(D)**1.0D0*URCUND
C---------------------------------------
C THE ORIGINAL VALUES OF Y(I,1) ARE SAVED TEMPORARILY IN ERRCR( ).
C---------------------------------------
C CC 40 I = 1,N
40     ERROR(I) = Y(I,1)
50     J1 = 0
C---------------------------------------
C FOR ICASE = 1; JK = N
C FOR ICASE = 2; JK = MW
C FOR ICASE = 3 AND FULL JACOBIAN; JK = N
C FOR ICASE = 3 AND BANDED JACOBIAN; JK = MW
C---------------------------------------
JIK = MIN0(JK,N)
CG 130  J = 1,JIK
       KMAX = (N-J)/MW + 1
DO 70 K = 1,KMAX
I = J + (K-1) * MW
R = DMAX1(EPSJ*YMAX(I),RO)
70     Y(I,1) = Y(I,1) + R
CALL GFUN (N, T, Y, SAVE1)
IF (ICASE .EQ. 1) GO TO 110
IF ((ICASE .EQ. 3) .AND. ((ML + ML) .EQ. (2 * NM1))) GO TO 110
J1 = J * NO + NCML
DO 100 K = 1,KMAX
JJ = J + (K-1) * MW
R = DMAX1(EPSJ*YMAX(JJ),RO)
O = CON/R
110     II = MAX0(JJ-MU,1)
I2 = MIN0(JJ+ML,N)
II = J1 - NM1 + I1
DO 90 I = II-I2
       PW(II) = (SAVE1(I) - SAVE2(I)) * D
90     II = II - NM1
     Y(JJ,1) = ERRCR(JJ)
ERRCR(JJ) = 0.0D0
100     J1 = J1 + NO
GO TO 130
C---------------------------------------
C 110     D = CCN/R
DO 120 I = 1,N
120     PW(I+J1) = (SAVE1(I) - SAVE2(I)) * D
     Y(I,1) = ERROR(J)
ERROR(J) = COO
J1 = J1 + NO
130     CONTINUE
C---------------------------------------
C FOR ICASE=1,2; ADD THE IDENTITY MATRIX TO PW.
C FOR ICASE=3; ADD THE MATRIX A(Y,T) TO PW.
C---------------------------------------
C PERFORM LU DECOMPOSITION CN PW.
C---------------------------------------
140     CALL ADCA (N, T, Y, PW, A, NO, ML, MU)
     CALL DEC (NO, N, ML, MU, PW, IPIV, IER)
RETURN
C--------------------------------------- END OF SUBROUTINE PMAT ---------------------------------------
SUBROUTINE DEC (NDIM, N, ML, MU, E, IF, IER)
IMPLICIT REAL*8(A-H, I-Z)

C-----------------------------------------
C SUBROUTINE DEC IS DIVIDED INTO TWO PORTIONS. ONE PORTION
C PERFORMS THE MATRIX TRIANGULARIZATION BY GAUSSIAN ELIMINATION
C TO A MATRIX IN FULL FORM; THE SECOND PORTION PERFORMS THE LU
C DECOMPOSITION ON A MATRIX IN BANDED FORM.
C
C PACKAGE ROUTINES CALLED.. NONE
C USER ROUTINES CALLED.. NONE
C CALLED BY.. DIFFUN, PMAT
C FORTRAN FUNCTIONS USED.. ABS(CABS), MIN0
C
C-----------------------------------------
DIMENSION B(NDIM, 1), IP(N)
CCMACCN /CCE4/ ICASE, ITYPE, JK, NML, NMU, IFULL
C
C---------------
C DETERMINE WHETHER THE MATRIX B IS FULL OR BANDED.
C
IF (ITYPE .EQ. 0) G0 TO (1, 100), ICASE
IF ((ICASE .EQ. 3) .AND. (IFULL .EQ. 0)) GO TO 100
C
C-------------------
C MATRIX TRIANGULARIZATION BY GAUSSIAN ELIMINATION.
C INPLT..
C N = CRDER OF MATRIX.
C NDIM = DECLARED DIMENSION OF ARRAY B.
C B = MATRIX TO BE TRIANGULARIZED.
C CUTFT..
C B(I, J), I .LE. J = UPPER TRIANGULAR FACTOR, U.
C B(I, J), I .GT. J = MULTIPLIERS = LOWER TRIANGULAR FACTOR, L.
C IP(K), K .LT. N = INDEX OF K-TH Pivot Row.
C IP(N) = (-1)**(NUMBER OF INTERCHANGES) OR C.
C IER = 0 IF MATRIX B IS NONSINGULAR, OR K IF FACTOR TO BE
C SINGULAR AT STAGE K.
C USE SOL TO OBTAIN Solution OF LINEAR SYSTEM.
C DETERM(B) = IP(N)*B(1, 1)*B(2, 2)*...*B(N, N).
C IF IP(N) = 0, B IS SINGULAR, SOL WILL CIVICE BY ZERO.
C
C REFERENCE..
C C. B. MCCLER, ALGORITHM 423, LINEAR EQUATION SOLVER,
C C. A. C. M. 15 (1972), P. 274.
C
1  IER = 0
2  IF(N) = 1
3  IF (N .EQ. 1) G0 TO 70
4  NML = N - 1
5  CC 60 K = 1, NML
6  KPI = K + 1
7  M = K
8  DO 10 I = KPI, N
9     IF (DABS(B(I, K)) .GT. DABS(B(K, K))) M = I
10    IP(K) = M
11    T = B(M, K)
12    IF (M .EQ. K) GO TO 20
13    IP(N) = -IP(N)
14    B(M, K) = B(K, K)
15    B(K, K) = T

-81-
20 IF (T .EQ. 0.0D0) GO TO 80
   T = 1.0D0/T
   DO 30 I = KPI1,N
   30   B(I,K) = -B(I,K)*T
   DO 50 J = KPI1,N
   50   T = B(K,J)
   60   B(K,J) = T
   IF (T .EQ. 0.0D0) GO TO 50
   DO 40 I = KPI1,N
   40   B(I,J) = B(I,J) + B(I,K)*T
   70   CONTINUE
   80   CONTINUE
   90   IF (B(N,N) .EQ. 0.0D0) GO TO 8C
   RETURN
C
 8C   IER = K
   IP(N) = 0
   RETURN

C

LU DECOMPOSITION OF BAND MATRIX A. L*U = P*A, WHERE P IS A
PERMUTATION MATRIX, L IS A UNIT LOWER TRiANGLE MATRIX,
AND U IS AN UPPER TRIANGLE MATRIX.
N = ORDER OF MATRIX.
B = N BY (2*ML+MU+1) ARRAY CONTAINING THE MATRIX A ON INPUT
AND ITS FACTORED FORM ON OUTPUT.
ON INPUT, B(I,K) (1.LE.I.LE.N) CONTAINS THE K-TH
DIAGONAL OF A, OR A(I,J) IS STORED IN B(I,J-I+ML+1).
ON OUTPUT, B CONTAINS THE L AND U FACTORS, WITH
U IN COLUMNS 1 TO ML+MU+1, AND L IN COLUMNS
ML+MU+2 TO 2*ML+MU+1.
ML,MU = WIDTHS OF THE LOWER AND UPPER PARTS OF THE BAND, NOT
COUNTING THE MAIN DIAGONAL. TOTAL BANDWIDTH IS ML+MU+1.
NDIM = THE FIRST DIMENSION (COLUMN LENGTH) OF THE ARRAY E,
NDIM MUST BE .GE. N.
IP = ARRAY OF LENGTH N CONTAINING PIVOT INFORMATION.
IER = ERROR INDICATOR.
   = 0 IF NO ERRORS.
   = K IF THE K-TH PIvOT CHOSEN WAS ZERO (A IS SINGULAR).
THE INPUT ARGUMENTS ARE NDIM, N, ML, MU, B.
THE OUTPUT ARGUMENTS ARE E, IP, IER.

100   IER = 0
   IF (N .EQ. 1) GO TO 192
   LL = ML + MU + 1
   NI = N - 1
   IF (ML .EQ. 0) GO TO 132
   DO 130 I = 1,ML
      II = MU + I
      K = ML + 1 - I
      DO 110 J = 1,II
     110   B(I,J) = B(I,J+K)
      K = II + 1
      DO 120 J = K,LL
     120   B(I,J) = 0.0
   DO 130 CONTINUE
   132   LR = ML
DC 190 NR = 1,N1
NP = NR + 1
IF (LR .NE. N) LR = LR + 1
MX = NR
XM = CABS(B(NR,1))
IF (ML .EQ. 0) GC TC 142
DO 140 I = NP,LR
    IF (CABS(B(I,1)) .LE. XM) GC TO 140
    MX = I
    XM = CABS(B(I,1))
140    CCNTINUE
142    IP(NR) = MX
    IF (MX .EQ. NR) GC TC 160
DO 150 I = 1,LL
    XX = B(NR,I)
    B(NR,I) = B(MX,I)
150    E(MX,I) = XX
160    XM = B(NR,1)
    IF (XM .EQ. 0.CC) GO TO 200
    B(NR,1) = 1.DO/XM
    IF (ML .EQ. 0) GC TC 190
    XM = -E(NR,1)
    KK = MIN0(N-NR,LL-1)
DO 180 I = NP,LR
    J = LL + I - NR
    XX = B(I,1)*XM
    B(NR,J) = XX
170    B(I,II) = B(I,II+1) + XX*B(NR,II+1)
180    B(I,LL) = 0.DO
190    CCNTINUE
192    NR = N
    IF (B(N,1) .EQ. 0.CC) GO TO 200
    B(N,1) = 1.DO/B(N,1)
RETURN
C
200    IER = NR
RETURN
C------------------------ END OF SUBROUTINE CEC ------------------------
END
SUBROUTINE SOL (NDIM, N, ML, MU, B, Y, IP)
IMPLICIT REAL*8(A-H,O-Z)

C-----

C SUBROUTINE SOL IS DIVIDED INTO TWO PORTIONS. ONE PORTION
C SOLVES THE LINEAR SYSTEM B*X = Y, WHERE MATRIX B IS A TRIANGULAR-
C UPPER MATRIX OF FULL FORM; THE SECOND PORTION SOLVES THE LINEAR
C SYSTEM A*X = C, WHERE MATRIX A IS THE LU DECOMPOSITION MATRIX IN
C BANDED FORM.
C
C PACKAGE ROUTINES CALLED... NONE
C USER ROUTINES CALLED... NCNE
C CALLED BY... DIFFUN,STIFF
C FORTRAN FUNCTIONS USED... MIN

C

DIMENSION B(NCIM,1),Y(N),IP(N)
COMMON /ODE4/ ICASE,ITYPEJK,AML,AML,IFULL

C

C DETERMINE WHETHER THE MATRIX B IS FULL OR BANDED.
C
IF (ITYPE .EQ. 0) GO TO (1,100), ICASE
IF ((ICASE .EQ. 3) .AND. (IFULL .EQ. 0)) GO TO 1CC

C

C SOLUTION OF LINEAR SYSTEM, B*X = Y.
C INPUT...
C N = ORDER OF MATRIX.
C NDIM = DECLARED DIMENSION OF ARRAY B.
C B = TRIANGULARIZED MATRIX OBTAINED FROM DEC.
C Y = RIGHT HAND SIDE VECTOR.
C IP = PIVOT VECTOR OBTAINED FROM DEC.
C DO NOT USE IF DEC HAS SET IER .NE. 0.
C OUTPUT...
C Y = SOLUTION VECTOR, X.

1    IF (N .EQ. 1) GO TO 50
    KM1 = N - 1
    DO 20 K = 1,NM1
        KPI = K + 1
        M = IP(K)
        T = Y(M)
        Y(K) = T
        DO 10 I = KPI,N
            Y(I) = Y(I) + B(I,K)*T
10     CONTINUE
    CONTINUE
20 CC 40 KE = 1,NM1
    KM1 = N - KE
    K = KM1 + 1
    Y(K) = Y(K)/B(K,K)
    T = -1.0DOO * Y(K)
    DO 30 I = 1,KM1
        Y(I) = Y(I) + B(I,K)*T
30     CONTINUE
40 CONTINUE
50 Y(1) = Y(1)/B(1,1)
RETURN
C
C
C SOLUTION OF A*X = C GIVEN LU DECOMPOSITION OF A FROM DEC.
C Y = RIGHT-HAND VECTOR C, OF LENGTH N, CN INPUT,
C X = SOLUTION VECTOR X CN OUTPUT.
C ALL THE ARGUMENTS ARE INPUT ARGUMENTS.
C THE OUTPUT ARGUMENT IS Y.
C
100 IF (N .EQ. 1) GO TO 160
   N1 = N - 1
   LL = ML + MU + 1
   IF (ML .EQ. 0) GO TO 132
   DO 130 NR = 1,N1
      IF (IP(NR) .EQ. NR) GO TO 110
      J = IP(NR)
      XX = Y(NR)
      Y(NR) = Y(J)
      Y(J) = XX
   110   KK = MIN0(N-NR,ML)
   DC 120 I = 1, KK
   120   Y(NR+I) = Y(NR+I) + Y(NR)*B(NR,LL+I)
130   CONTINUE
132   LL = LL - 1
   Y(N) = Y(N)*B(N,1)
   KK = 0
   DC 150 NB = 1,N1
   NR = N - NB
   IF (KK .NE. LL) KK = KK + 1
   DO = C.DO
   IF (LL .EQ. 0) GO TO 150
   DO 140 I = 1, KK
140   DP = DP + B(NR,I+1)*Y(NR+I)
150   Y(NR) = (Y(NR) - DP)*B(NR,1)
   RETURN
C
160   Y(1) = Y(1)*B(1,1)
   RETURN
C----------------------------- END OF SUBROUTINE SOL -----------------------------
SUBROUTINE DDIFFUN (N, T, Y, YDOT, A, PW, IPIV)
IMPLICIT REAL*8 (A-H, O-Z)

C-----------------------------------------------
C SUBROUTINE DDIFFUN IS CALLED ONLY FOR ICASE = 3. THIS ROUTINE
C COMPUTES F(Y,T) = A(Y,T)**-1 * G(Y,T).
C
C PACKAGE Routines called:  ADDA,DEC,SOL
C USER Routines called:  NONE
C CALLED BY:  STIFF
C FORTRAN Functions used:  MAX0
C-----------------------------------------------

DIMENSION Y(N),YDOT(N),A(I),PW(N),IPIV(I)
COMMON /ODE2/ EPSJ,ASQ,ML,AL,MU,MW,NM1,NOML,NOW,MLA,NUA,MWA
COMMON /CDE4/ ICASE,ITYPE,JK,NML,AML,IFULL

C
NN = NSC
IF ((ML + MU) .EQ. (2 * NM1)) GO TO 20
IF ((MLA + MUA) .NE. (2 * NM1)) GO TO 10
NN = MAX0(NOW,NSQ)
GO TO 20

10 NN = N * (NML + NMU + 1)
20 CO 30 I = 1,NN
30 PW(I) = 0.0000
NO = NM1 + 1
CALL ADDA (N, T, Y, PW, A, NO, ML, ML)
CALL DEC (NO, N, NML, NMU, PW, IPIV, IER)
CALL SOL (NO, N, NML, NMU, PW, YDCT, IPIV)
RETURN
C----------------------------------------------- END OF SUBROUTINE DDIFFUN -----------------------------------------------
END
SUBROUTINE ADCA (N, T, Y, PW, A, NO, ML, MU)
IMPLICIT REAL*8(A-H,O-Z)

C

C ADDS THE MATRIX A(Y,T) TO THE MATRIX P. BOTH MATRICES, A
C AND P, MAY BE IN EITHER FULL OR BANDED FORM.
C
C ICASE = TYPE OF O.D.E. TO BE SOLVED.
C 1 DY/DT = G(Y,T), FULL JACOBIAN MATRIX
C 2 DY/DT = G(Y,T), BANDED JACOBIAN MATRIX
C 3 A*(DY/DT) = G(Y,T), FULL OR BANDED JACOBIAN MATRIX
C
C ITYPE = FORM OF MATRIX A USED WHEN ICASE = 3.
C 0 DEFAULT VALUE FOR ICASE = 1,2.
C 1 A IS A FULL N BY N MATRIX.
C 2 A IS A BANDED N BY (MLA+MU+1) MATRIX.
C
C FOR ICASE = 3, A CALL TO USER-SUPPLIED ROUTINE AMAT DEFINES THE
C MATRIX A.
C
C PACKAGE ROUTINES CALLED.. ADDB,ADDBB
C USER ROUTINES CALLED.. AMAT
C CALLED BY.. DIFFUN,FMAT
C FORTRAN FUNCTIONS USED.. NONE
C

C DIMENSION PW(1),Y(1),A(1)
C COMMON /ODE2/ EPSJ,ASC,MLC,MUC,MN,M1,M1,M2,M2,M2,M2
C COMMON /ODE4/ ICASE,ITYPE,JK,NML,NML,IFUL
C
C C.C.E. IS CF FORM ICASE = 1; ADD 1.C TO DIAGONAL ELEMENTS CF PW.
C
C 10 J = 1
C DC 20 I = 1,N
C PW(J) = PW(J) + 1.0CC
C 20 J = J + (NO + 1)
C RETURN
C
C C.C.E. IS CF FORM ICASE = 2; ADD 1.C TO APPROPRIATE DIAGONAL
C ELEMENTS OF BANDED MATRIX PW.
C
C 30 DC 40 I = 1,N
C 40 PW(NOML + I) = PW(NOML + I) + 1.0CC
C RETURN
C
C C.C.E. IS CF FORM ICASE = 3.
C
C 50 CALL AMAT (N, MLA, MLA, T, Y, A)
C
C -87-
C CHECK TO SEE IF MATRIX A AND MATRIX PW IS INPUT AS FULL OR BANDED,
C TREAT THE COMBINATION APPROPRIATELY.
C
IF (ITYPE .NE. 1) GC TO 70
IF ((ML + MU) .NE. (2 * NM1)) GC TO 80
C
C C.O.E. IS CF FORM ICASE = 3, BOTH A AND PW ARE FULL. THE
C RESULTING SUM OF A AND PW IS A FULL MATRIX, PW.
C
DO 60 I = 1, NSQ
  60     PW(I) = PW(I) + A(I)
RETURN
C
C C.O.E. IS CF FORM ICASE = 3; A IS BANDED WHILE PW IS BANDED OR
C FULL; THE RESULTING SUM WILL BE A FULL OR BANDED MATRIX PW.
C
70 IF ((ML + MU) .NE. (2 * NM1)) GC TO 100
C
AT THIS POINT, A IS CF BANDED FORM, WHILE PW IS FULL. THE
C RESULTING SUM OF A AND PW IS A FULL MATRIX, PW. CALL ADDB TO ADD
C THE FULL MATRIX PW TO THE BANDED MATRIX A.
C
CALL ADDB (N, MLA, MUA, A, PW)
RETURN
C
C AT THIS POINT, A IS FULL, WHILE PW IS BANDED. THE RESULTING
C FULL OF A AND PW IS A FULL MATRIX, PW. CALL ADDB TO ADD THE FULL
C MATRIX A TO THE BANDED MATRIX PW.
C
80 CALL ADDB (N, ML, ML, PW, A)
DC 90 I = 1, NSQ
  90     PW(I) = A(I)
    CALL AMAT (N, MLA, MUA, T, Y, A)
RETURN
C
C C.O.E. IS CF FORM ICASE = 3; A IS A BANDED N BY (MLA + MUA + 1)
C MATRIX AND PW IS A BANDED N BY (ML + MU + 1) MATRIX. THE
C RESULTING MATRIX IS CF BANDED FORM WITH BANDWIDTHS OF NML AND
C NMU, WHERE NML = MAXIMUM(ML, MLA) AND NMU = MAXIMUM(MU, MUA).
C
100 MAXNM = NO * (NML + NMU + 1)
  KKPPL = NO * (NML + MU + 1)
  KKPAU = NO * (NML + MUA + 1)
IF (ML .LT. MLA) GC TO 110
C
THE LOWER BANDWIDTH OF MATRIX P IS GREATER THAN OR EQUAL TO THE
C LOWER BANDWIDTH OF MATRIX A.
C
    CALL ADDBB (N, NML, MLA, MAXNM, KKPAU, KKPPL, A, PW)
RETURN
C
THE LOWER BANDWIDTH OF MATRIX A IS GREATER THAN THE LOWER BANDWIDTH
C OF MATRIX P.
C
110 CALL ADDBB (N, NML, ML, MAXNM, KKPPL, KKPAU, PW, A)
CO 120 I = 1, MAXNM
120    PW(I) = A(I)
      CALL AMAT (N, MLA, MUA, T, Y, A)
      RETURN
C------------------------ END OF SUBROUTINE ADDA ------------------------
END
SUBROUTINE ADDB (N, ILC, IUP, B, F)
IMPLICIT REAL*8(A-H,O-Z)

C---------------------------------------------
C
C SUBROUTINE ADDB ADDS THE FULL MATRIX F TO THE BANDED MATRIX B.
C ILC AND IUP ARE THE LOWER AND UPPER BANDWIDTHS, RESPECTIVELY,
C OF THE BANDED MATRIX B. THE DIMENSION OF THE FULL MATRIX F IS
C N BY N, WHILE THE DIMENSION OF THE BANDED MATRIX B ON INPUT IS
C N BY (ILC+IUP+1). THE RESULTING SUM OF MATRICES B AND F IS PLACED
C IN MATRIX F IN FULL FORM.
C
C PACKAGE ROUTINES CALLED.. NCNE
C USER ROUTINES CALLED.. NCNE
C CALLED BY.. ADDA
C FORTRAN FUNCTIONS USED.. NCNE

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

C DIMENSION B(1),F(1)

C
MN = ILC + IUP + 1
DO 10 J = 1,N
DO 10 I = 1,N
K = J - I + ILC + 1
IF (K .LE. 0) GC TO 10
IF (K .GT. MN) GC TO 10
IJ = N * (J - 1) + I
IK = N * (K - 1) + I
F(IJ) = F(IJ) + E(IK)
10 CONTINUE
RETURN

C--------------------------------------------- END OF SUBROUTINE ADDB ---------------------------------------------
END
SUBROUTINE ADDBB (N, NML, MLL, MAXNM, KL, KLL, CL, DLL)
IMPLICIT REAL*8(A-H, O-Z)

C SUERCUTINE ADDBB ADDS THE BANDED MATRICES CL AND DLL. THE LOWER
C BANDWIDTH OF MATRIX DLL IS GREATER THAN OR EQUAL TO THE LOWER
C BANDWIDTH OF MATRIX CL. THE RESULTING SUM OF CL AND DLL IS PLACED
C IN MATRIX DLL IN BANDED FORM.
C
C PACKAGE ROUTINES CALLED.. NCNE
C USER ROUTINES CALLED.. NCNE
C CALLED BY.. ADDA
C FORTRAN FUNCTIONS USED.. NCNE
C
C DIMENSION CL(I),DLL(I)
C
KKL = N * (NML - MLL)
DC 30 I = 1,MAXNM
   XDLL = 0.000
   XCL = 0.000
   IF (I GT KKL) GO TO 10
   GO TO 20
C
10   JL = I - KKL
    IF (I LE KL) XCL = CL(JL)
20   IF (I LE KLL) XDLL = DLL(I)
30   DLL(I) = XDLL + XCL
RETURN
C------------------- END OF SUBROUTINE ADDBB -------------------
END
SUBROUTINE URND (U)
IMPLICIT REAL*8 (A-H, O-Z)
-------------------

C U IS THE SMALLEST POSITIVE NUMBER SUCH THAT (1.0+U) .GT. 1.0
C U IS COMPUTED APPROXIMATELY AS A POWER OF 1./2.
C
C THIS CODE IS COMPLETELY EXPLAINED AND DOCUMENTED IN THE TEXT,
C COMPUTER SOLUTION OF ORDINARY DIFFERENTIAL EQUATIONS: THE INITIAL
C VALUE PROBLEM BY L. F. SHAMPINE AND M. K. GORDON.
C
C PACKAGE ROUTINES CALLED.. NCNE
C USER ROUTINES CALLED.. NCNE
C CALLED BY.. ODEPAK
C FORTRAN FUNCTIONS USED.. NCNE

-------------------

HALFU = 0.5D0
50  TEMPI = 1.0D0 + HALFU
  IF(TEMPI .LE. 1.0D0) GO TO 100
  HALFU = 0.5D0 * HALFU
  GO TO 50

C 100  U = 2.0D0 * HALFU
RETURN

-------------------- END OF SUBROUTINE URND --------------------
ODEPAKK: AN ORDINARY DIFFERENTIAL EQUATIONS PACKAGE

by

JOHN W. SHELLENBERGER

B. S., Washburn University, 1973
B. S., Kansas State University, 1975

AN ABSTRACT OF A MASTERS REPORT

Submitted in partial fulfillment of

the requirements for the degree

MASTER OF SCIENCE

Department of Computer Science

KANSAS STATE UNIVERSITY
Manhattan, Kansas
1975
ABSTRACT

ODEPAKK is a FORTRAN software package for the numerical solution of the initial value problem for systems of ordinary differential equations. The package is a combination and modification of three previously developed ordinary differential equation solvers.

ODEPAKK allows the user to employ either a variable-stepsize, variable-order implicit Adams method or a variable-stepsize, variable-order backward differentiation method in solving a given system of ordinary differential equations.