User Guide for MEBDF.m and MEBDF_DRIVER.m.

Please direct any comments (good or bad) to Jeff Cash at: j.cash@imperial.ac.uk.

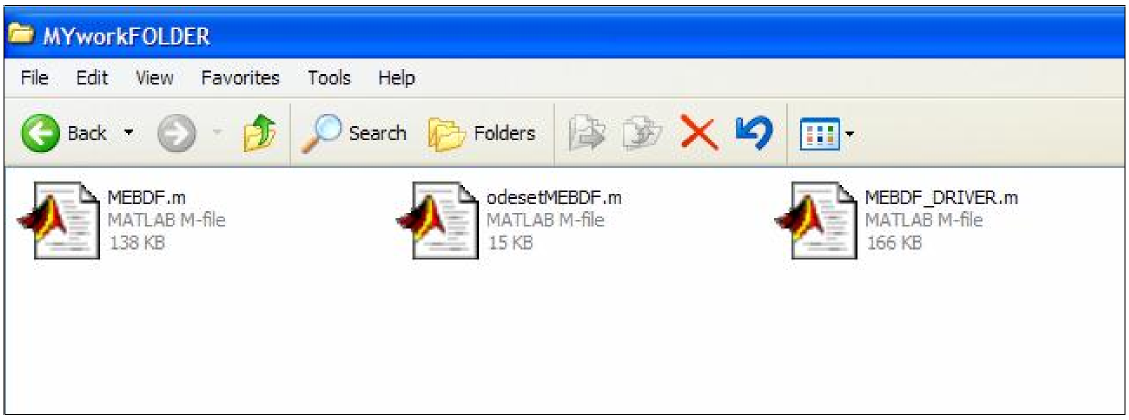
MEBDF.m may be run in two ways. The first way is to use MEBDF_DRIVER.m, an extension to MEBDF.m that allows users to run MEBDF.m similarly to the way other MATLAB solvers are run (like ODE15s for example, indeed it was ODE15s upon which MEBDF_DRIVER.m was based). The second way is to write a DRIVER function and run MEBDF.m directly, recommended for faster run times and more 'serious' experiments. MEBDF.m is slightly more complicated to work than the MEBDF_DRIVER.m. Details of both are found below.

SETUP:

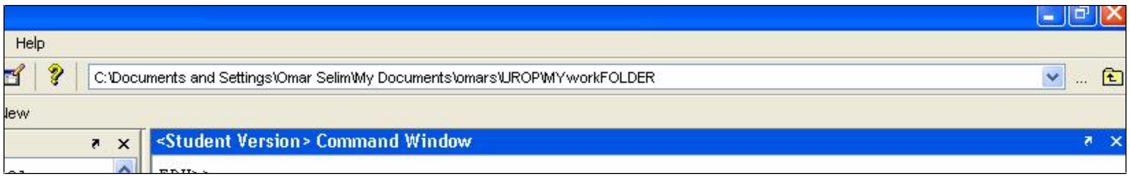
1. To use MEBDF.m or MEBDF_DRIVER.m you must *first* 'copy and paste' the three files,

- MEBDF.m
- MEBDF_DRIVER.m
- odesetMEBDF.m,

into the folder in which you intend to work (see below).



2. Start MATLAB and change the current directory (from the top of the window) to address the work folder containing the above three files.


Help
C:\Documents and Settings\Omar Selim\My Documents\omars\UROP\MYworkFOLDER
<Student Version> Command Window

GETTING STARTED:

To get started with MEBDF_DRIVER.m you must first create three m-files which are to be saved in the same folder as the three files above (see SETUP), they are:

- F.m : Calculates the right hand side of the system to be solved, that is for a system $y' = f(x,y)$, F.m returns the value(s) $f(x,y)$ given the vector y and the point x. This function unlike the next two is 'compulsory' and must be supplied.
- PDERV.m: Calculates the Jacobian matrix of partial derivatives. That is for a system ($y' = f(x,y)$) of *n* differential equations PDERV returns the matrix *J* evaluated at the point x, where,

  $[J]_{ij} = \frac{\partial y_i}{\partial y_j}$ *for* $i, j = 1 \rightarrow n$.

  It is optional but recommended to supply the Jacobian for more reliable results. If PDERV is not to be supplied MF must be set to 22 or 24 (see *OPTIONS and using odesetMEBDF.m.*).
- MAS.m: Calculates the Mass matrix for problems like, $M(x,y)y' = f(x,y)$, an optional function which need be supplied only if the problem requires it.

Skeletons (stencils) of these functions are here provided.
WARNING: The above functions MUST be created even if they are not doing anything (see MAS function under WORKED EXAMPLE 1). Also these stencils must be used strictly with no alterations (including file names) as errors will occur otherwise.

Stencil for the F.m function.

```
function [N,X,Y,DF,IPAR,RPAR,ierr] = F(N,X,Y,DF,IPAR,RPAR,ierr)

% N is the dimension of the system to be solved
% X is the point of evaluation.
% Y is the approximated value of the true solution at the point X.
% IPAR and RPAR may be ignored. For more details see the section on Variable description     % below.
% Will be set to zero and should be altered if F, MAS or PDERV encounter an illegal operation % (square root of -1 for example). See the section on Variable description below. If the user is %unsure on how to use this it may be ignore.
% DF contains the returned values of the calculated right hand side. So DF(3) = f(X,Y3), etc.
% For OUTPUT the user need only really concern themselves with 'DF'.

end
```

Stencil for PDERV.m function.

```
function [X,Y,DFY,N,MEBAND,IPAR,RPAR,ierr] = PDERV(X,Y,DFY,N,MEBAND,IPAR,RPAR,ierr)

% X,Y,N,IPAR,RPAR,ierr are the same as above (F.m).
% DFY is the computed Jacobian matrix.
% MEBAND may be ignored.

end
```

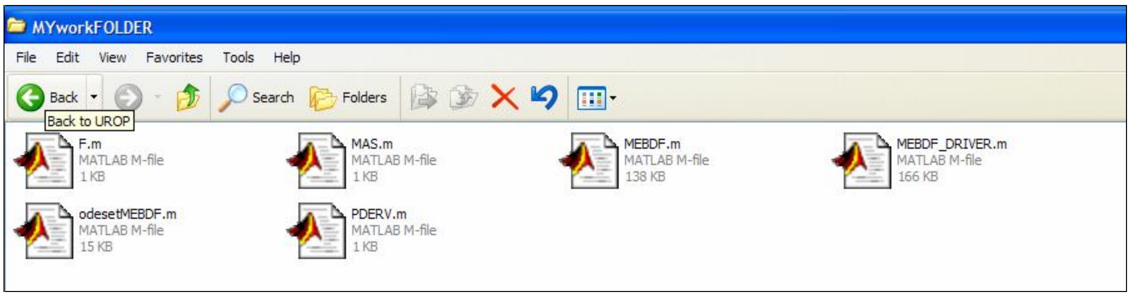
Stencil for MAS.m function.

```
function [N,AM,M,IPAR,RPAR,ierr] = MAS(N,AM,M,IPAR,RPAR,ierr)

% N,IPAR,RPAR,ierr are the same as above (F.m).
% M is the number of rows of the MAS matrix when stored in 'banded form', see entry in Variable
% descriptions.
% AM is the Mass matrix to be returned.

end
```

Again once these functions have been created they must be saved (with the correct names) in the same folder as that of the SETUP section above.



F.m MATLAB M-file 1 KB
MAS.m MATLAB M-file 1 KB
MEBDF.m MATLAB M-file 138 KB
MEBDF_DRIVER.m MATLAB M-file 166 KB
odesetMEBDF.m MATLAB M-file 15 KB
PDERV.m MATLAB M-file 1 KB

<u>MEBDF_DRIVER.m</u>

To run MEBDF_DRIVER.m you must use the following syntax at the command line in the MATLAB window.

Syntax: [X,Y] = MEBDF_DRIVER(SPAN,INITIAL);

or

[X,Y] = MEBDF_DRIVER(SPAN,INITIAL,OPTIONS);

Arguments:

SPAN: A row vector ([0,1,2] for example) that defines the range of integration. If only two points are specified MEBDF_DRIVER.m will return the solution evaluated at every integration step (recommended for plots).

If more than two points are specified then MEBDF_DRIVER.m returns the solution at only those specified points.

INITIAL: A row vector specifying the initial conditions of the problem ([0.4,0,3] for example).

OPTIONS: A structure that allows you to customise the settings of integration. Relative tolerance may be changed for example, from its default of 1E-3. OPTIONS is declared using odesetMEBDF.m (see below).

X: Row vector of mesh points at which the solution was calculated.

Y: Corresponding calculated solution. This is an array and is of the following form,

$$[Y]_{ij} = y_i(x_j)\,.$$

Where $y_i(x_j)$ is the calculated solution of the i^{th} equation at the j^{th} mesh point x_j.

OPTIONS and using odesetMEBDF.m.

For illustrative purposes we first show the syntax for changing the values of RTOL and ATOL and asking the solver to plot the final solutions.

```
OPTIONS = odesetMEBDF(‘RTOL’,1E-7, ‘ATOL’,1E-9, ‘sep_plots’, ‘on’);
```

The following table describes what may be changed using options.

OPTION	**DESCRIPTION**	**DEFAULT VALUE**	**POSSIBLE VALUES**
RTOL	Relative tolerance. See *Variable descriptions*.	1E-3	Any positive Real number.
ATOL	Absolute tolerance. See *Variable descriptions*.	1E-6	Any positive Real number.
ITOL	See *Variable descriptions*.	2	{1,2}
BANDED	To be used if we have a banded system or if MBND or MASBND are to be changed from default values.	‘off’	{on,off}
MF	Method Flag: See *Variable descriptions*.	21	{21,22,23,24}
Hstart	Initial step size.	1E-6	Positive Real number.

MaxDer	Maximum derivative. See *Variable descriptions*.	7	{1,2,…,7}
Stats	Print computational statistics of experiment. Will appear in a file named 'REPORT.txt' in the current folder. Details such as CPU time, Number of function evaluations… etc. can be found here.	'off'	{on,off}
MBND	See *Variable descriptions*.	[0, 0, 0, 0]	Entries can be any non-negative integers.
MASBND	See *Variable descriptions*.	[0, 0, 0, 0]	Entries can be any non-negative integers.
I_WORK_14	Maximum number of integration steps. See *Variable descriptions*, under IWORK(14)	200000	Positive Integer.
IWORK_1	See *Variable descriptions*, under IWORK(1)	0	Non-negative integer.
IWORK_2	See *Variable descriptions*, under IWORK(2)	0	Non-negative integer.
IWORK_3	See *Variable descriptions*, under IWORK(3)	0	Non-negative integer.
ONE_PLOT	Creates a plot of the solutions on ONE figure. The plot will be of Y1 v X, Y2 vX…etc.	'off'	{'off', dimension of grid for the plots, [2,3] for example}
SEP_PLOTS	Creates plots of solution on separate figures. As above.	'off'	{on,off}
Y_PLOTS	In the case of a dimension 2 system, this option will plot Y1 v Y2, but will return an error if dimension of the system ~= 2.	'off'	{on,off}

WORKED EXAMPLE: The Van Der Pol equation.

For this problem we have:

$$
\begin{aligned}
y_1' &= y_2 & y_1(0) &= 2 \\
y_2' &= \mu^2((1-y_1^2)y_2 - y_1) & y_2(0) &= 0
\end{aligned}
$$

For this example, we will be using $\mu = 10^3$.

First we set up the functions F.m, MAS.m and PDERV.m according to the stencils above. So,

F.m.

```
function [N,X,Y,DF,IPAR,RPAR,ierr] = F(N,X,Y,DF,IPAR,RPAR,ierr)


EPS=1E-6;
DF(1)=Y(2);
PROD=1-Y(1)*Y(1);
DF(2)=(PROD*Y(2)-Y(1))/EPS;

end
```

MAS.m

```
function [N,AM,M,IPAR,RPAR,ierr] = MAS(N,AM,M,IPAR,RPAR,ierr)

 %DUMMY ROUTINE
end
```

PDERV.m

```
function [X,Y,DFY,N,MEBAND,IPAR,RPAR,ierr] =  PDERV(X,Y,DFY,N,MEBAND,IPAR,RPAR,ierr)

    EPS=1E-6;
    DFY(1,1)=0;
    DFY(1,2)=1;
    DFY(2,1)=(-2*Y(1)*Y(2)-1)/EPS;
    DFY(2,2)=(1-Y(1)^2)/EPS;

end
```

It is possible to solve this problem with just the default settings, so we may type at the command prompt:

```
[X,Y] = MEBDF_DRIVER([0,1,2,3,4,5],[2,0]);
```

Which will return the following OUTPUTS:

X = 0 1 2 3 4 5

```
Y =


   2.0000        0
  -1.8649   0.7526
   1.7072  -0.8909
  -1.5116   1.1765
   1.1965  -2.7723
   1.8903  -0.7346
```

Alternatively we may wish to change RTOL and ATOL for more accurate results, find out the computational statistics of the run and obtain a plot of the final solution.

We first set options,

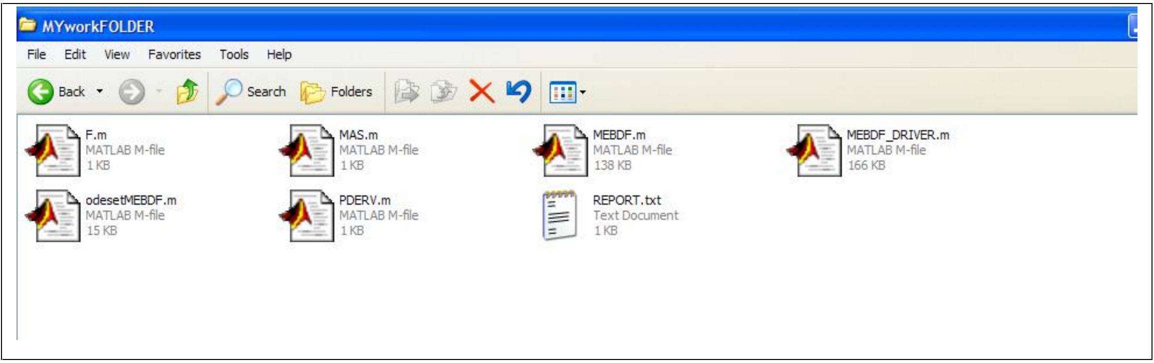
```
options = odesetMEBDF(‘RTOL’,1E-7, ‘ATOL’, 1E-7, ‘stats’, ‘on’, ‘y_plots’, ‘on’);
```

Now we call MEBDF_DRIVER.m with,

```
[X,Y] = MEBDF_DRIVER([0,5],[2,0],options);
```

Notice here we only specify two points in the SPAN, this tells the solver to return the values at all the integration steps, giving a smoother plot.

As we asked for ‘stats’ a file ‘REPORT.txt’ is created with the statistics.



F.m MATLAB M-file 1 KB
MAS.m MATLAB M-file 1 KB
MEBDF.m MATLAB M-file 138 KB
MEBDF_DRIVER.m MATLAB M-file 166 KB
odesetMEBDF.m MATLAB M-file 15 KB
PDERV.m MATLAB M-file 1 KB
REPORT.txt Text Document 1 KB

This is what the report should say:

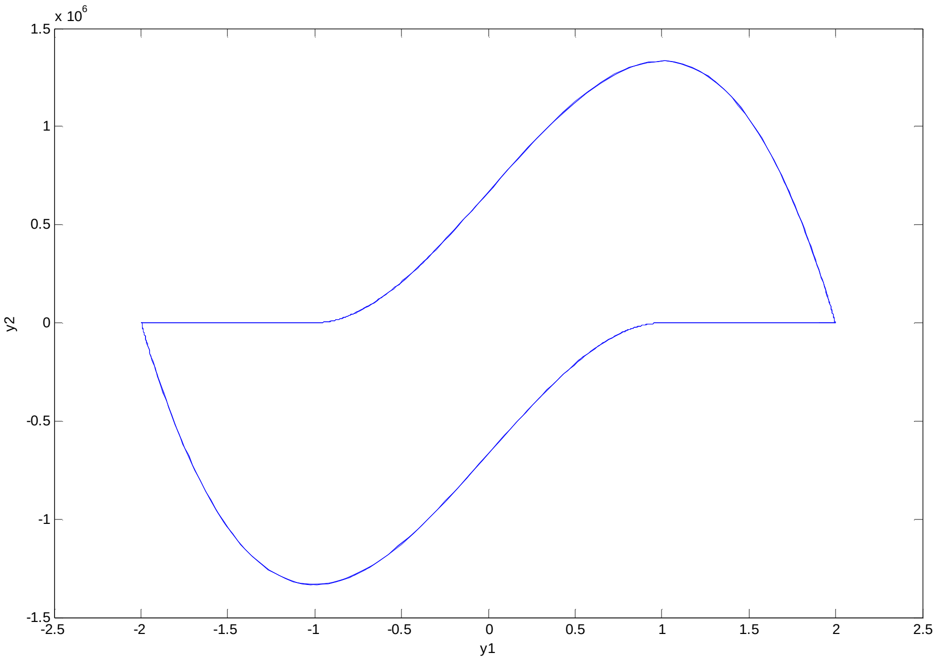
```
COMPUTATIONAL STATISTICS OF RUN

 NUMBER OF EQUATIONS: 2

 RESULTS WITH THE FOLLOWING TOL :
 RTOL = 1.000000E-007
 ATOL = 1.000000E-007


 LAST STEP SIZE: 2.507530E-002
 LAST ORDER OF THE METHOD: 5
 MAXIMUM ORDER USED SO FAR: 7
 TOTAL NUMBER OF STEPS TAKEN: 3588
 TOTAL NUMBER OF FAILED STEPS: 163
 NUMBER OF FUNCTION EVALUATIONS: 6464
 NUMBER OF JACOBIAN EVALUATIONS: 344
 NUMBER OF FACTORIZATION: 344
 NUMBER OF BACKSOLVES: 13455
 CPU TIME: 7.66102
```

The plot obtained should look like,



MEBDF.m

To run MEBDF.m you need to follow the same instructions under SETUP and GETTING STARTED. Then a DRIVER routine is needed, these provide MEBDF.m with the necessary initial parameters. The DRIVER routine is to be saved in the same file as that mentioned in SETUP, and it is this function that is called from the MATLAB command line. The user may find a large bank of working example of DRIVERS at www.imperial.ac.uk/~jcash. It is recommend that these be referred to in conjunction with the notes of the original MEBDF.f (See below). Remarks: The general structure will be found in the above mentioned examples, but it should be noted that there is a subtle difference between DRIVERs which declare non-zero MEBND and those which declare MBND = [0,0,0,0], These differences can be seen in the decleration of WORK_I9 for example.

Variable Descriptions.

We refer to the original code MEBDF.f for this part.

```
% C    -------------------------------------------------------------
% C    OVDRIV IS TO BE CALLED ONCE FOR EACH OUTPUT VALUE OF T, AND
% C    IN TURN MAKES REPEATED CALLS TO THE CORE INTEGRATOR STIFF.
% C
% C    THE INPUT PARAMETERS ARE ..
% C    N     =  THE NUMBER OF FIRST ORDER DIFFERENTIAL EQUATIONS.
% C    T0    =  THE INITIAL VALUE OF T, THE INDEPENDENT VARIABLE
% C           (USED ONLY ON THE FIRST CALL)
% C    HO    =  THE NEXT STEP SIZE IN T (USED FOR INPUT ONLY ON THE
% C           FIRST CALL)
% C    Y0    =  A VECTOR OF LENGTH N CONTAINING THE INITIAL VALUES OF Y
% C           (USED FOR INPUT ONLY ON FIRST CALL)
% C    TOUT  =  THE VALUE OF T AT WHICH OUTPUT IS DESIRED NEXT.
% C           INTEGRATION WILL NORMALLY GO SLIGHTLY BEYOND TOUT
% C           AND THE PACKAGE WILL INTERPOLATE TO T = TOUT
% C    TEND  =  END OF THE RANGE OF INTEGRATION.
% C    MF    =  THE METHOD FLAG.  AT PRESENT MF=21,22,23 OR 24 IS
% C           ALLOWED. THESE ARE EXTENDED BACKWARD DIFFERENTIATION
% C           FORMULAE USING THE CHORD METHOD WITH ANALYTIC OR NUMERICAL
% C           JACOBIAN FOR MF=21,22 RESPECTIVELY. MF=23/24 ARE  THE SAME
% C           AS FOR 21/22 BUT THE JACOBIAN IS NOW BANDED.   THE USER
% C           NEEDS TO SPECIFY SUBROUTINE PDERVIF MF=21 OR 23.
% C    IDID   = THE INTEGER USED ON INPUT TO INDICATE THE TYPE OF CALL.
% C           1   THIS IS THE FIRST CALL FOR THE PROBLEM.
% C           0   THIS IS NOT THE FIRST CALL FOR THIS PROBLEM
% C              AND INTEGRATION IS TO CONTINUE.
% C          -1   THIS IS NOT THE FIRST CALL FOR THE PROBLEM,
% C              AND THE USER HAS RESET N, RTOL, ATOL,H  AND/OR MF.
% C           2   SAME AS 0 EXCEPT THAT TOUT HAS TO BE HIT
% C               EXACTLY (NO INTERPOLATION IS DONE).
% C               ASSUMES TOUT .GE. THE CURRENT T.
% C             3   SAME AS 0 EXCEPT CONTROL RETURNS TO CALLING
% C               PROGRAM AFTER ONE STEP. TOUT IS IGNORED, UNTIL THE
% C               INTEGRATION REACHES TOUT OR BEYOND. IF IT PASSES TOUT
% C               THE PROGRAM INTERPOLATES THE SOLUTION VALUES AND
% C               RETURNS THE SOLUTION VALUE AT TOUT.
% C             SINCE THE NORMAL OUTPUT VALUE OF IDID IS 0,
% C             IT NEED NOT BE RESET FOR NORMAL CONTINUATION.
% C             THE FIRST CALL TO THE DRIVER IS WITH IDID=1 AND FOR
% C             A SUCCESSFUL STEP THE DRIVER RETURNS WITH IDID=1.THUS
% C             THE CALL WITH IDID = 1 IS SIMPLY THE FIRST
% C             INITIALISING STEP FOR THE CODE.  THE USER
% C             THEN NEEDS TO CONTINUE WITH IDID=0,-1,2 OR 3 AS ABOVE.
% C     LOUT   = THE LOGICAL OUTPUT CHANNEL FOR MESSAGE PASSING.
% C     MBND   = AN ARRAY OF DIMENSION 4 FOR USE WHEN THE JACOBIAN
% C             MATRIX IS BANDED.  IF THE JACOBIAN HAS ML DIAGONALS
% C             BELOW THE MAIN DIAGONAL AND MU DIAGONALS ABOVE THE
% C             MAIN DIAGONAL THEN:
% C             MBND(1) = ML
% C             MBND(2) = MU
% C             MBND(3) = MU + ML + 1
% C             MBND(4) = 2*ML + MU + 1
% C     MASBND = AN ARRAY OF DIMENSION 4 DESCRIBING THE MASS MATRIX.
% C             MASBND(1) = IMAS IS ZERO IF THE MASS MATRIX IS THE
% C             IDENTITY. OTHERWISE IMAS = 1.
% C             MASBND(2) = N IF THE MASS MATRIX IS FULL. OTHERWISE
% C            = MLMAS THE LOWER BANDWIDTH.
% C             MASBND(3) = MUMAS THE UPPER BANDWIDTH.  IF THE MASS MATRIX
% C             IS FULL THIS NEED NOT BE SPECIFIED.
% C             MASBND(4) = LMAS = N IF THE MASS MATRIX IS FULL.
% C                         = 0 IF IMAS = 0
% C                         = MLMAS + MUMAS + 1 OTHERWISE
% C             THE RESTRICTION MLMAS .LE. ML AND MUMAS .LE. MU IS
% C             ALSO IMPOSED.
% C     MAXDER=  THE MAXIMUM ORDER IS MAXDER + 1.
% C             THE VALUE OF MAXDER CANNOT EXCEED 7.  THIS IS THE
% C             VALUE RECOMMENDED UNLESS IT IS BELIEVED THAT THERE
% C             ARE SEVERE STABILITY PROBLEMS IN WHICH CASE MAXDER=3
% C             OR 4 SHOULD BE TRIED INSTEAD.
% C     ITOL  =  AN INDICATOR OF THE TYPE OF ERROR CONTROL. SEE
% C             DESCRIPTION BELOW UNDER ATOL.
% C     RTOL  =  A RELATIVE ERROR TOLERANCE PARAMETER. CAN BE EITHER A
% C             SCALAR OR AN ARRAY OF LENGTH N.  SEE DESCRIPTION
% C             BELOW UNDER ATOL.
% C     ATOL  =  THE ABSOLUTE ERROR BOUND.
% C             THE INPUT PARAMETERS ITOL, RTOL AND ATOL DETERMINE
% C             THE ERROR CONTROL PERFORMED BY THE SOLVER.  THE
% C             SOLVER WILL CONTROL THE VECTOR e = (e(i)) OF ESTIMATED
% C             LOCAL ERRORS IN y ACCORDING TO AN INEQUALITY OF THE FORM
% C               RMS-NORM OF (e(i)/ewt(i)) .LE. 1
% C             THE ROOT MEAN SQUARE NORM IS
% C                RMS-NORM(V) = SQRT((SUM v(i)**2)/N).  HERE
% C              ewt = (ewt(i)) IS A VECTOR OF WEIGHTS WHICH MUST
% C             ALWAYS BE POSITIVE, AND THE VALUES OF RTOL AND ATOL
% C             SHOULD BE NON-NEGATIVE. IF ITOL = 1 THEN SINGLE STEP ERROR
% C           ESTIMATES DIVIDED BY YMAX(I) WILL BE KEPT LESS THAN 1
% C           IN ROOT-MEAN-SQUARE NORM.  THE VECTOR YMAX OF WEIGHTS IS
% C           COMPUTED IN OVDRIV. INITIALLY YMAX(I) IS SET AS
% C           THE MAXIMUM OF 1 AND ABS(Y(I)).  THEREAFTER YMAX(I) IS
% C           THE LARGEST VALUE OF ABS(Y(I)) SEEN SO FAR, OR THE
% C           INITIAL VALUE YMAX(I) IF THAT IS LARGER.
% C           IF ITOL = 1 THE USER NEEDS TO SET ATOL = RTOL =
% C           THE PRECISION REQUIRED.  THEN
% C                   ewt(i) = RTOL(1)*YMAX(i)
% C           IF ITOL IS GREATER THAN 1 THEN
% C             ewt(i) = rtol(i)*abs(y(i)) + atol(i)
% C           THE FOLLOWING TABLE GIVES THE TYPES (SCALAR/ARRAY)
% C           OF RTOL AND ATOL, AND THE CORRESPONDING FORM OF ewt(i)
% C              ITOL   RTOL      ATOL       ewt(i)
% C               2    SCALAR    SCALAR   rtol*abs(y(i))   + atol
% C               3    SCALAR    ARRAY    rtol*abs(y(i))   + atol(i)
% C               4    ARRAY     SCALAR   rtol(i)*abs(y(i))+ atol
% C               5    ARRAY     ARRAY    rtol(i)*abs(y(i))+ atol(i)
% C           IF EITHER OF THESE PARAMETERS IS A SCALAR, IT NEED
% C           NOT BE DIMENSIONED IN THE USER'S CALLING PROGRAM.
% C    NIND1    = THE NUMBER OF VARIABLES OF INDEX 1,2,3 RESPECTIVELY.
% C    NIND2,NIND3  THESE ARE SET IN IWORK(1),(2),(3).
% C           THE EQUATIONS MUST BE DEFINED SO THAT THE INDEX 1
% C           VARIABLES PRECEDE THE INDEX 2 VARIABLES WHICH IN
% C           TURN PRECEDE THE INDEX 3 VARIABLES.
% C    RPAR, IPAR  REAL AND INTEGER PARAMETERS (OR PARAMETER ARRAYS)
% C           WHICH CAN BE USED FOR COMMUNICATION BETWEEN  THE USER'S
% C           CALLING PROGRAM AND THE F, PDERV AND MAS SUBROUTINES.
% C    IERR     IERR IS AN INTEGER FLAG WHICH IS ALWAYS EQUAL TO ZERO
% C           ON INPUT.  SUBROUTINES F, PDERV AND MAS SHOULD ALTER
% C           IERR ONLY IF ONE OF THEM ENCOUNTERS AN ILLEGAL OPERATION SUCH
% C           AS THE SQUARE ROOT OF A NEGATIVE NUMBER OR EXPONENT
% C           OVERFLOW. THE USER CAN THEN ALTER H AND CALL THE
% C           SUBROUTINE AGAIN WITH IDID=-1 IF HE WISHES.
% C
% C    AFTER THE INITIAL CALL, IF A NORMAL RETURN OCCURED AND A NORMAL
% C    CONTINUATION IS DESIRED, SIMPLY RESET TOUT AND CALL AGAIN.
% C    ALL OTHER PARAMETERS WILL BE READY FOR THE NEXT CALL.
% C    A CHANGE OF PARAMETERS WITH IDID = -1 CAN BE MADE AFTER
% C    EITHER A SUCCESSFUL OR AN UNSUCCESSFUL RETURN.
% C
% C    THE OUTPUT PARAMETERS ARE..
% C    T0    =  THE VALUE OF T WHICH RELATES TO THE CURRENT SOLUTION
% C           POINT Y0()
% C    HO    =  THE STEPSIZE H USED LAST, WHETHER SUCCESSFULLY OR NOT.
% C    Y0    =  THE COMPUTED VALUES OF Y AT T = TOUT
% C    TOUT  =  UNCHANGED FROM ITS INPUT VALUE.
% C    IDID  =  INTEGER USED ON OUTPUT TO INDICATE RESULTS, WITH
% C           THE FOLLOWING VALUES AND MEANINGS..
% C
% C     0   INTEGRATION WAS COMPLETED TO TOUT OR BEYOND.
% C
% C    -1   THE INTEGRATION WAS HALTED AFTER FAILING TO PASS THE
% C        ERROR TEST EVEN AFTER REDUCING H BY A FACTOR OF
% C        1.E10 FROM ITS INITIAL VALUE.
% C
% C    -2   AFTER SOME INITIAL SUCCESS, THE INTEGRATION WAS
% C         HALTED EITHER BY REPEATED ERROR TEST FAILURES OR BY
% C         A TEST ON RTOL/ATOL.  TOO MUCH ACCURACY HAS BEEN REQUESTED.
% C
% C    -3   THE INTEGRATION WAS HALTED AFTER FAILING TO ACHIEVE
% C         CORRECTOR CONVERGENCE EVEN AFTER REDUCING H BY A
% C         FACTOR OF 1.E10 FROM ITS INITIAL VALUE.
% C
% C    -4   IMMEDIATE HALT BECAUSE OF ILLEGAL VALUES OF INPUT
% C         PARAMETERS.  SEE PRINTED MESSAGE.
% C
% C    -5   IDID WAS -1 ON INPUT, BUT THE DESIRED CHANGES OF
% C         PARAMETERS WERE NOT IMPLEMENTED BECAUSE TOUT
% C         WAS NOT BEYOND T.  INTERPOLATION AT T = TOUT WAS
% C         PERFORMED AS ON A NORMAL RETURN.  TO TRY AGAIN,
% C         SIMPLY CALL AGAIN WITH IDID = -1 AND A NEW TOUT.
% C
% C    -6   MAXIMUM ALLOWABLE NUMBER OF INTEGRATION STEPS EXCEEDED.
% C         TO CONTINUE THE USER SHOULD RESET IWORK(14).
% C
% C
% C    -7   STEPSIE IS TOO SMALL (LESS THAN SQRT(UROUND)/100)
% C
% C
% C    -11   INSUFFICIENT REAL WORKSPACE FOR THE INTEGRATION
% C
% C    -12   INSUFFICIENT INTEGER WORKSPACE FOR THE INTEGRATION
% C
% C
% C    IN ADDITION TO OVDRIVE, THE FOLLOWING ROUTINES ARE PROVIDED
% C    IN THE PACKAGE..
% C
% C    INTERP( - )   INTERPOLATES TO GET THE OUTPUT VALUES
% C               AT T=TOUT FROM THE DATA IN THE Y ARRAY.
% C    STIFF( - )    IS THE CORE INTEGRATOR ROUTINE.  IT PERFORMS A
% C               SINGLE STEP AND ASSOCIATED ERROR CONTROL.
% C    COSET( - )    SETS COEFFICIENTS FOR BACKWARD DIFFERENTIATION
% C               SCHEMES FOR USE IN THE CORE INTEGRATOR.
% C    PSET( - )     COMPUTES AND PROCESSES THE JACOBIAN
% C               MATRIX J = DF/DY
% C    DEC( - )      PERFORMS AN LU DECOMPOSITION ON A MATRIX.
% C    SOL( - )      SOLVES LINEAR SYSTEMS A*X = B AFTER DEC
% C               HAS BEEN CALLED FOR THE MATRIX A
% C    DGBFA ( - )   FACTORS A DOUBLE PRECISION BAND MATRIX BY
% C               ELIMINATION.
% C    DGBSL ( - )   SOLVES A BANDED LINEAR SYSTEM A*x=b
% C
% C               ALSO SUPPLIED ARE THE BLAS ROUTINES
% C
% C               daxpy, dscal, idamax, ddot.
% C
% C
% C    THE FOLLOWING ROUTINES ARE TO BE SUPPLIED BY THE USER AND
% C               SHOULD BE DECLARED AS EXTERNAL.
% C
% C    F(N,T,Y,YDOT,IPAR,RPAR)   COMPUTES THE FUNCTION YDOT = F(Y,T),
% C                   THE RIGHT HAND SIDE OF THE O.D.E.
% C                   HERE Y AND YDOT ARE VECTORS OF LENGTH N
% C    PDERV(T,Y,PD,N,MEBAND,IPAR,RPAR)  COMPUTES THE N*N JACOBIAN MATRIX
% C                   OF PARTIAL DERIVATIVES AND STORES IT IN PD
% C                   AS AN N BY N ARRAY IF THE JACOBIAN IS FULL.
% C                   IF THE JACOBIAN IS BANDED THE ARRAY PD IS OF
% C                   SIZE MBND(4)*N.
% C                   IF THE JACOBIAN IS FULL PD(I,J) IS TO BE SET
% C                   TO THE PARTIAL DERIVATIVE OF YDOT(I) WITH
% C                   RESPECT TO Y(J).  IF THE JACOBIAN IS BANDED
% C                   WITH mu DIAGONALS ABOVE THE MAIN DIAGONAL
% C                   THE PARTIAL DERIVATIVE DF(I)/DY(J) SHOULD BE
% C                   PUT IN PD(i-j+mu+1,j). PDERV IS CALLED ONLY IF
% C                   MITER = 1 OR 3.  OTHERWISE A DUMMY ROUTINE CAN
% C                   BE SUBSTITUTED.
% C
% C    MAS(N,AM,MASBND(4),IPAR,RPAR) COMPUTES THE MASS MATRIX M.  IF IMAS=0
% C                 THIS MATRIX IS THE IDENTITY MATRIX AND THE
% C                 SUBROUTINE MAS CAN BE DUMMY.  IF IMAS=1
% C                 THE SUBROUTINE MAS IS OF THE FORM
% C                 MAS(N,AM,MASBND(4))
% C                 IF MASBND(4) = N THE MASS MATRIX IS STORED AS
% C                 A FULL MATRIX WITH AM(I,J) = M(I,J). OTHERWISE
% C                 THE MATRIX IS BANDED AND IS STORED
% C                 COMPONENTWISE AS AM(I-J+MUMAS+1,J)=M(I,J).
% C
% C    THE DIMENSION OF PD/ MAS  MUST BE AT LEAST N**2 FOR A FULL JACOBIAN
% C    /MASS MATRIX AND MBND(4)*N/ MASBND(4)*N FOR A BANDED JACOBIAN
% C    / MASS MATRIX(MF=23 OR 24) .  THE DIMENSIONS
% C    OF YMAX,ERROR,SAVE1,SAVE2,IPIV AND THE FIRST DIMENSION
% C    OF Y SHOULD ALL BE AT LEAST N.
% C
% C
% C    UROUND   THIS IS THE UNIT ROUNDOFF AND HAS TO BE SET AS
% C                  UROUND = DLAMCH('Epsilon')
% C    EPSJAC   = sqrt(UROUND).
% C
% C    HUSED  (=WORK(2))    LAST STEPSIZE SUCCESSFULLY USED BY THE INTEGRATOR
% C    NQUSED (=IWORK(4))   LAST ORDER SUCCESSFULLY USED
% C    NSTEP  (=IWORK(5))   NUMBER OF SUCCESSFUL STEPS TAKEN SO FAR
% C    NFAIL  (=IWORK(6))   NUMBER OF FAILED STEPS
% C    NFE    (=IWORK(7))   NUMBER OF FUNCTION EVALUATIONS  SO FAR
% C    NJE    (=IWORK(8))   NUMBER OF JACOBIAN EVALUATIONS  SO FAR
% C    NDEC   (=IWORK(9))   NUMBER OF LU DECOMPOSITIONS  SO FAR
% C    NBSOL  (=IWORK(10))   NUMBER OF 'BACKSOLVES'  SO FAR
% C    NPSET  (=IWORK(11))   NUMBER OF TIMES A NEW COEFFICIENT MATRIX HAS BEEN
% C           FORMED SO FAR
% C    NCOSET (=IWORK(12))   NUMBER OF TIMES THE ORDER OF THE METHOD USED HAS
% C           BEEN CHANGED SO FAR
% C    MAXORD (=IWORK(13))   THE MAXIMUM ORDER USED SO FAR IN THE INTEGRATION
% c    MAXSTP (=IWORK(14))   THE MAXIMUM ALLOWED NUMBER OF STEPS SET BY THE
USER
% C
% C    IF IT IS ONLY REQUIRED TO CONTROL THE ACCURACY IN THE
% C    DIFFERENTIAL VARIABLES THEN THE USER SHOULD FIND THE
% C    STRING 'AMMEND' AND MAKE CHANGES THERE
% C
% c
```