AAEC/E575

AAEC/E575

AUSTRALIAN ATOMIC ENERGY COMMISSION

RESEARCH ESTABLISHMENT

LUCAS HEIGHTS RESEARCH LABORATORIES

CLUTCH - AN ORDINARY DIFFERENTIAL EQUATION SOLVER FOR INITIAL VALUE PROBLEMS

by

B.E. CLANCY

August 1983

ISBN 0 642 59780 4

AUSTRALIAN ATOMIC ENERGY COMMISSION
RESEARCH ESTABLISHMENT

LUCAS HEIGHTS RESEARCH LABORATORIES

CLUTCH - AN ORDINARY DIFFERENTIAL EQUATION SOLVER FOR INITIAL VALUE PROBLEMS

by

B. E. CLANCY

ABSTRACT

A description is given of a computer package which provides a simple interface to the LSODE (ordinary differential equation solving) subroutines developed at Lawrence Livermore Laboratory. Standardised input and output procedures help users to obtain numerical solutions to initial value problems with minimal effort. As well as providing data for a problem, the user need only provide the source for a single FORTRAN subroutine which calculates the first derivatives of the problem's dependent variables. The package can operate in batch and interactive modes and the numerical results can be displayed either as tables or graphs by using a set of simple English language commands.

National Library of Australia card number and ISBN 0 642 59780 4

The following descriptors have been selected from the INIS Thesaurus to describe the subject content of this report for information retrieval purposes. For further details please refer to IAEA-INIS-12 (INIS: Manual for Indexing) and IAEA-INIS-13 (INIS: Thesaurus) published in Vienna by the International Atomic Energy Agency.

C CODES; DIFFERENTIAL EQUATIONS; NUMERICAL SOLUTION; REACTOR KINETICS

CONTENTS

1. INTRODUCTION

At the Lucas Heights Research Laboratories, as at other laboratories, a significant number of initial value problems arise for which a solution requires the numerical integration of one or more coupled ordinary differential equations. In the author's experience, these problems are usually solved by writing and running a special purpose computer program which will have its own particular input/output procedures, as well as an individual solution strategy.

These programs take valuable time to develop and to debug; in too many cases the development time is excessive. In an attempt to reduce this wasted time, CLUTCH has been constructed. It has standardised problem definition, input and output routines and, for its integration strategy, employs the robust and well documented LSODE subroutines developed by Hindmarsh [1974, 1981].

Computer programming by or for a prospective user can be limited to the writing of a single FORTRAN subroutine called MYDIFF, which will calculate the first ordinary derivative of the problem's dependent variables. This programming will have been preceded by sufficient analysis to state the problem as a set of first order equations in terms of a single independent variable. The analysis must lead to algorithms by which all of the ordinary first derivatives of the problem's dependent variables can be calculated explicitly. Use of the package is described by reference to two examples.

2. EXAMPLES FOR SOLUTION

2.1 Example 1 - A Simple Rod Pendulum

A point mass is attached to one end of a stiff massless rod which, at its other end, can swing freely about a frictionless pivot. Motion is in a vertical plane.

If U denotes the angle between the rod and the vertical so that U = 0 is the stable equilibrium position, the motion of the rod is well known to be described by the second order equation

$$U'' = -g \sin(U)/L$$

where L is the length of the rod, g is the acceleration due to gravity, and U'' is the second derivative of U with respect to time t.

To solve these equations for various values of the parameter L and for given initial values of U and U' = dU/dt requires calculation of the position U and the angular velocity U' at any time subsequent to the initial time t = 0. We do not restrict U to small values and the usual simple harmonic approximation will not apply.

The problem is easily written as a set of coupled first order equations by introducing a second dependent variable V = U' so that the system of equations is formally

$$dU/dt = V$$

$$dV/dt = -g \sin(U)/L$$

In these equations, g is a fixed constant and L is a parameter which can change from case to case.

2.2 Example 2 - A Reactor Kinetics Problem

We take a model fission reactor and inject into it some excess reactivity. If delayed neutrons are ignored, the reactor power P(t) will start to rise exponentially with some inverse period r. That is, dP/dt is initially equal to rP. The reactor temperature will also rise as energy is deposited in the system, but there is a tendency for coolant flowing through the reactor to remove this energy. After more assumptions have been made we may finish up with an equation for the time dependence of the reactor power in the form

$$P' = P (r - b R)$$

where the prime ' again denotes differentiation with respect to time, and

$$R = \int_{t'=0}^{t'=t} P(t') \exp(-f(t-t')) \, dt'$$

so that R = 0 at t = 0.

The quantity b is proportional to the reactor temperature coefficient of reactivity, and f is proportional to the coolant flow rate.

For a given case r, b and f will be fixed parameters and, at time t = 0, the initial power will be known. A solution will be achieved if the power is calculated for any specified times t > 0. To cast these equations into the required form, we write R as

$$R = \exp(-ft) \int_{t'=0}^{t'=t} P(t') \exp(ft') \, dt'$$

and differentiate with respect to t. It follows that

$$R' = fR + P$$

As well as the reactor power, we are also interested in knowing the time behaviour of the inverse reactor period V, where

$$V = P'/P$$

Recapitulating, the equations are

$$dP/dt = rP - bPR$$

$$dR/dt = P - fR$$

and the initial conditions are

$$P = P0 \text{ at } t = 0$$

where P0 is the given initial power

$$R = 0 \text{ at } t = 0$$

The quantities b,f,r and P0 are fixed parameters, but the quantity V must be calculated at all times of interest.

2.3 Common Features

With a change in notation, the coupled equations in either example can be written in the form

```
dY1/dt = F1(t,Y1,Y2,...,P1,P2,...)
dY2/dt = F2(t,Y1,Y2,...,P1,P2,...)
dY3/dt = F3(t,Y1,Y2,...,P1,P2,...)
etc.
```

where

Y1,Y2,... are the dependent variables,
F1,F2,... are defined functional forms,
P1,P2,... are parameters which may be constant, or which may themselves be given by equations of the form

```
P1  = G1(t,Y1,Y2,...,P1,P2,...)
P2  = G2(t,Y1,Y2,...,P1,P2,...)
etc.
```

3. CODING THE MYDIFF SUBROUTINE

This is the routine which calculates the derivatives from given values of the independent variable, the dependent variables and any of the parameters. The package requires that the routine be coded with a particular argument list and the user may well begin it with the following two statements as standard:

```
SUBROUTINE MYDIFF(N,T,Y,YDOT,NPARAM,PARAM,NFROM,*)
REAL*8 T,Y(N),YDOT(N),PARAM(NPARAM)
```

Here the variables passed to the routine are

N, the number of equations and dependent variables,
T, the current value of the independent variable,
Y, an array of current values of the dependent variables,
NPARAM, the number of parameters,
PARAM , an array with current values of the parameters, and
NFROM , an integer with value 0 or 1 according as the routine is called from the internals of the LSODE package or from the main program.

The remaining array,

YDOT(I),I=1,N

is to be filled by the subroutine with values for the first derivatives. For Example 1, the required coding could consist simply of the following statements:

```
SUBROUTINE MYDIFF(N,T,Y,YDOT,NPARAM,PARAM,NFROM,*)
REAL*8 T,Y(N),YDOT(N),PARAM(NPARAM)
YDOT(1)=Y(2)
YDOT(2)=-9.8D0*DSIN(Y(1))/PARAM(1)
RETURN
END
```

so that in the notation of Section 2.1

Y(1) corresponds to U,
Y(2) corresponds to V,
YDOT(1) corresponds to dU/dt,
YDOT(2) corresponds to dV/dt, and
PARAM(1) corresponds to L.

For Example 2, the variables P,R could be given the FORTRAN names Y(1) and Y(2) while the constant parameters f,b,r and the initial power P0 could be given the FORTRAN names PARAM(1),PARAM(2),PARAM(3),and PARAM(4). The varying parameter V (the reactor period) could be given the FORTRAN name PARAM(5). The coding for MYDIFF could then be as follows:

```
      SUBROUTINE MYDIFF(N,T,Y,YDOT,NPARAM,PARAM,NFROM,*)
      REAL*8 T,Y(N),YDOT(N),PARAM(NPARAM)
C     FIRST THE EQUATION FOR P'
      YDOT(1)=PARAM(3)*Y(1)-PARAM(1)*Y(1)*Y(2)
C     THEN THE EQUATION FOR R'
      YDOT(2)=Y(1)-PARAM(2)*Y(2)
C     FINALLY THE EQUATION TO DETERMINE V, THE INVERSE PERIOD
      PARAM(5)=YDOT(1)/Y(1)
      RETURN
      END
```

3.1 Internally Generated Parameters and the Variable NFROM

In the preceding discussion, most of the parameters have been constant throughout the course of a particular case, only changing from case to case. This within-case constancy is not a necessity, however, and the subroutine

MYDIFF itself may calculate and change values for any of the parameters. During the input phase, the user has only to define initial values for the parameters. The main routine saves these values together with the values available at points T at which the program is asked to calculate a solution. If the user requests them, the values of the parameters may be exhibited by the output routine. For Example 2, the user is interested not only in the reactor power P but also in the behaviour of the inverse period V=(1/P) dP/dt, for which the initial value is equal to r. In the coding of MYDIFF suggested for Example 2, the value of V = PARAM(4) is calculated each time MYDIFF is called. In searching for its solution, the LSODE routines will call MYDIFF with many values of the variables which are not on the final solution curve, and it is a little wasteful to evaluate PARAM(4) at all of these. It is even possible that MYDIFF could be called with the variable Y(1) having a value of zero and, in that event, division by zero would be attempted. It would be more efficient and safer if PARAM(4) were only evaluated in MYDIFF for variable values corresponding to points on the solution curve. To implement this the MYDIFF routine would have to be modified by changing the line before the RETURN statement to

```
IF (NFROM.EQ.1) PARAM(4)=YDOT(1)/Y(1)
```

The conditional IF (NFROM.EQ.1) is inserted because the CLUTCH package sets the variable NFROM equal to zero when searching for a solution point. When it has been found, NFROM is reset to unity and MYDIFF is called again; it is from this call that saveable values of varying parameters are deduced.

3.2 The Error Return

MYDIFF should be coded so that, under normal circumstances, a simple RETURN statement sends control back to the calling routine. For some problems, the user will wish the integration of the equations to stop at some point which cannot be predicted *ab initio* but which is determined by the course of the solutions generated. In the reactor kinetics problem, for example, it may be sensible to stop the integration if the reactor power exceeds a value set by a particular parameter. To implement this, the user should code the MYDIFF subroutine so that if the terminating condition occurs, control is passed back to the calling routine via a RETURN 1 statement. This special return should only be taken if NFROM = 1 . The last item in the MYDIFF argument list is the asterisk *; this simply advises the compiler that an error return may be coded in the source.

4. STANDARD INPUT/OUTPUT PROCEDURES

4.1 Heading and Labelling

Input for a job must begin with a single line of text which the package interprets as a heading with which any plotted output can be marked for identification. Users lacking in imagination can insert a blank line. After this line, the user may provide appropriate labels for the parameters and for the dependent variables. These labels, which can be up to eight characters long with no embedded blanks, are used to label printed and/or plotted output produced by the package. If no labels are supplied at this stage, the package uses the following default labels:

```
Y(1) Y(2) ....
PARM(1) PARM(2) ....
```

which, though unique, are not very informative.

The syntax for the labelling is most easily seen from the following example, which would be useful for Example 1. To label the two variables and the first (and only) parameter, the user could supply in the input stream the line of text

```
LABEL Y1 THETA P1 LENGTH Y2 THETADOT
```

or as many lines as are needed to specify the labels. The syntax for the labelling command is

(i) the keyword LABEL followed by

(ii) pairs of OLDNAME NEWNAME where
OLDNAME is one of Y1 Y2 ... Y20 P1 P2 ... P20 and
NEWNAME is any string of characters which does not include blank or the = character.

The package continues to read these pairs until it finds something which is not a valid OLDNAME. This item is assumed to be the first item of data for a particular case to be solved within the job. This item is processed by the INPUT subroutine.

In the standard form of CLUTCH, data which define a case are read from FORTRAN logical unit 1 by three subroutines, each of which uses, in turn, the SCAN free format input subroutines of Bennett and Pollard [1967]. A complete description of the input procedures is given in that report. Here it is remarked that input quantities can essentially be entered anywhere in columns 1-72 of an input line, that values of integer variables must not have an associated decimal point and that a value for a real variable can be entered in a variety of ways. Thus 123400000. , 1.234E+8 and 1.234E+08 are all valid forms for a real quantity. The routines and the input items they seek are as follows.

4.2 Subroutine INPUT

This subroutine reads the following items from the input stream:

MF a two-digit integer defining the solution method to be used by LSODE. The first digit (METH) must be either 1 or 2, with 2 being necessary for sets of differential equations which exhibit the property of stiffness somewhere in the region of interest. The second digit (MITER) is usually 0 or 2, but other values can be prescribed for special problems. A fuller discussion of these is given in Section 6. For most applications, an MF value of 10 is usually adequate but 22 is recommended for 'stiff' systems. If in doubt use 22.

TSTART a real specifying the initial value of the independent variable, usually 0, for problems where the independent variable is time.

TCRIT a real specifying a critical value for the independent variable, through which the solution must not proceed. A critical value is normally one at which the differential equations become singular; if there is no singularity, a value numerically greater than or equal to 1.0E+60 should be given. The sign is important because the package assumes that the integration is to start at TSTART and proceed towards TCRIT.

N an integer specifying the number of dependent variables, and N must be greater than zero.

Y(I),I=1,N an array of reals specifying the initial values of the dependent variables.

NPARAM an integer specifying the number of parameters in this problem.

PARAM(I),I=1,NPARAM an array of reals specifying the initial values of the parameters. If NPARAM is zero, no values are to be entered as data.

EPS a real specifying the requested local accuracy in the solution. 1.0E-7 is adequate for most problems and, in this event, the package will iterate on its trial solutions in an attempt to reduce the local error in any dependent variable Y(I) to less than ERR(I)=EPS*(DABS(Y(I))+1.0E-13). This error control is then relative for most of the time but for variables starting from zero will be absolute. Values of EPS which are negative or which exceed 1.0 are considered as errors and the default value of 1.0E-7 is used instead. If the error control provided by the formula ERR(I)=EPS*(DABS(Y(I))+1.0E-13) is deemed unsatisfactory, the user may alter this by entering EPS=0.0. Then (and only then) the user must enter 2N reals.

ATOL(I),I=1,N and

RTOL(I),I=1,N from which the code constructs the ERR terms from the formula ERR(I)=ATOL(I)+RTOL(I)*DABS(Y(I)).

Note that neither N nor NPARAM can exceed 20 in the standard form of the package. All values supplied as data are checked by the program which may abort the case if certain input errors are detected.

4.3 Subroutine GETOUT

This subroutine carries on from where INPUT leaves off. It reads the successive values of T (the independent variable) at which the other variables and parameters are to be calculated. These T values should be monotonic but the package can integrate in either direction, increasing or decreasing T. To signal the end of a case, the user simply enters a value of T back before the starting value TSTART. When the program senses this, it proceeds to the output stage controlled by the next subroutine.

4.4 Subroutine OUTPUT

The type of output produced for each case is determined solely by the presence of keywords detected in the input stream. These keywords are PUNCH,

PLOT and PRINT. To produce results, each keyword must be followed by a list of items to be punched, plotted or printed; this list is a sequence of names, either OLDNAME or NEWNAME, as described in Section 4.1. To signal the end of a list, the user must enter the digit 0. To print a table of values of the angular position and angular velocity for Example 1, the input items could be

```
PRINT Y1 Y2 0
```

or, if the naming suggested in Section 4.1 had been carried out,

```
PRINT THETA THETADOT 0
```

The printed output is a labelled columnar table of values which includes the corresponding values of the independent variable T. This table is supplied by default for all values of T at which solutions were obtained. The user may only be interested in the values over a small subrange of the the calculated range; in this event the user should precede the output command keyword PRINT with the entry

```
TRANGE t1 t2
```

where the reals t1 and t2 mark the lower and upper limits of T values of interest. The T range thus defined remains operative for all subsequent output commands, unless overridden by a further TRANGE command.

Syntax for the PUNCH command is identical to that for PRINT. The output is sent to FORTRAN logical unit 2 in card image format.

Graphs are requested in a similar fashion. The command

```
PLOT THETA THETADOT 0
```

will result in a page on which THETA and THETADOT are each plotted against the independent variable. Different linetypes are used for each curve and a separately annotated axis is drawn for each quantity. The default plot is of the linear-linear type but can be overridden by using modifiers before the command keyword, e.g.

```
LOG LIN PLOT list 0

LIN LOG PLOT list 0
```

or

```
LOG LOG PLOT list 0
```

Should a variable be non-positive when a logarithmic plot is requested, the package will revert to linear scaling for this variable.

It is possible to produce plots in which the x-axis is used for other than the independent variable. For instance the command

```
PLOT THETADOT AGAINST THETA
```

or

```
PLOT Y2 AGAINST Y1
```

would produce a curve for which the x- and y-coordinates represented THETA and THETADOT respectively.

When all required output commands have been given, the user may then enter the command

```
MORE
```

which causes the package to return to subroutine INPUT to seek data for another case, or the command

```
GOON
```

which causes the package to return to subroutine GETOUT and there obtain further values for the independent variable. The integration continues from the point last reached. If no further cases are to be attempted, the keyword

```
STOP
```

terminates the job.

Printed output produced as a result of the PRINT command appears on FORTRAN logical unit 3, which is normally directed to the system line printer. If the user produces output with the PUNCH command, the resulting information is identical to that produced by PRINT but the results are written on logical unit 2, which can be directed to any appropriate dataset for processing outside the CLUTCH package. Additional printed output is produced via logical unit 6. This will contain the input items read by the SCAN routines together with any error messages produced by the package.

5. NON-STANDARD INPUT/OUTPUT PROCEDURES

It is believed that use of CLUTCH with the standard input/output procedures should suffice for most applications. For special problems, it may be appropriate to develop a special program. There is yet another class of problems which can use the CLUTCH skeleton but which would be more powerful if

different versions of the routines INPUT, GETOUT and/or OUTPUT were available. Programs for this last class can be developed quite simply if the user includes the appropriate FORTRAN source cards for any of these subroutines together with the source for the MYDIFF subroutine when submitting the job. For these special subroutines, the calling sequence conventions followed in the skeletal source programs listed in Section 9 must be adhered to.

There is an intermediate class of problems which, although adequately served by the basic pattern of CLUTCH, requires additional processing of the calculated solution values. This processing might present part of the results as a special table or graph. Alternatively, it might require more complicated computation such as an integral of a combination of the results. To achieve this, the user should provide as part of the FORTRAN input to the package a subroutine EXPERT which performs the required processing. This subroutine will be executed if, in the OUTPUT stage of the job, the user enters the output command EXPERT. The supplied routine should conform to the calling sequence used by the default routine listed in Section 9.2.

6. THE INPUT ITEM MF AND THE MYPEDE SUBROUTINE

The input item MF was mentioned briefly in Section 4.2 where two suggested values were given. A full discussion of this quantity is given by Hindmarsh [1981] and should be consulted by any user who wishes to make use of the package for any major task. MF is a two decimal digit number of which

the first	METH must be 1 or 2,
and the second	MITER must be 0,1,2,3,4 or 5

If METH is 1, the solution strategy is based on Adams-Moulton predictor-corrector methods, whereas if METH is 2, the backward difference (implicit) methods of Gear [1971], developed for stiff systems, are implemented.

As the solution proceeds, the LSODE routines build up a knowledge of the behaviour of the variables and calculate the numerical values of the derivatives through the MYDIFF routine only when necessary. These values may be used to construct within the package approximations to the Jacobian matrix for the problem, i.e. the matrix [PD(i,j)] , i and j ranging from 1 to N the number of equations. A particular element PD(i,j) is equal to the partial derivative with respect to Y(j) of the derivative dY(i)/dt, in the notation of

Section 2.3 . The values selected for MITER tell the package how to approximate the PD matrix.

If MITER has the value 0 , no attempt is made to build up an estimate of the Jacobian. For non-stiff equations, this may not result in any computational inefficiency. For stiff equations a very efficient method corresponds to MITER = 1 but to implement this, the user must supply the source for an additional subroutine named MYPEDE which calculates the full Jacobian. A skeleton giving the required argument list is included in Section 9. When MITER has the value 1, the variables ML and MU in the argument list are dummies and can be ignored.

Use of the value MITER = 2 forces the package to build up estimates for all the elements of the Jacobian. This is done by examining differences in the derivatives found when the dependent variables are slightly changed.

If MITER is 3, the package builds up and uses a purely diagonal matrix estimate of the Jacobian. If the equations are only loosely coupled, this will be quite satisfactory, and it even works surprisingly well for some tightly coupled systems.

For some problems, the Jacobian matrix has a banded structure, so its only non-zero elements are along the main diagonal or along a few subdiagonals above or below the main one. The existence of this structure leads to computational efficiency if the CLUTCH package and LSODE are made aware of the fact. This can be done by entering as data a value for MF which implies that MITER is 4 or 5. In this event, the input routine expects that integer values for ML and MU will then be supplied in the input stream. ML is the number of subdiagonals which are non-zero, whereas MU is the number of non-zero superdiagonals. Negative values are illegal, but zero values for either ML or MU are acceptable. Of course, neither value can reach or exceed N, the order of the full Jacobian matrix.

If MITER is 4, the package expects that the user will have coded a version of the MYPEDE subroutine which will calculate the non-zero diagonal terms. As distinct from the situation with MITER = 1 the variables ML,MU in the subroutine argument list then correspond exactly to the numbers of non-zero diagonals and the subroutine coding should use their values.

The subroutine must load the two-dimensional array PD with values for the diagonals in a special banded form most easily understood from an example.

Suppose that we have a system of eight equations and the 8 by 8 Jacobian matrix really in the form

1,1	1,2	1,3	1,4	0	0	0	0
2,1	2,2	2,3	2,4	2,5	0	0	0
3,1	3,2	3,3	3,4	3,5	3,6	0	0
0	4,2	4,3	4,4	4,5	4,6	4,7	0
0	0	5,3	5,4	5,5	5,6	5,7	5,8
0	0	0	6,4	6,5	6,6	6,7	6,8
0	0	0	0	7,5	7,6	7,7	7,8
0	0	0	0	0	8,6	8,7	8,8

where the zero terms in the matrix are marked as 0. In this case MU = 3 (because there are three non-zero superdiagonals) and ML = 2 (because there are two non-zero subdiagonals.)

Subroutine MYPEDE will be passed the values of ML and MU, and must load only six rows of the array PD as follows:

0	0	0	1,4	2,5	3,6	4,7	5,8
0	0	1,3	2,4	3,5	4,6	5,7	6,8
0	1,2	2,3	3,4	4,5	5,6	6,7	7,8
1,1	2,2	3,3	4,4	5,5	6,6	7,7	8,8
2,1	3,2	4,3	5,4	6,5	7,6	8,7	0
3,1	4,2	5,3	6,4	7,5	8,6	0	0

This odd layout is a requirement of the LSODE subroutines.

If MITER is 5, the package constructs internally an approximation to the banded Jacobian by repeated calls to the MYDIFF subroutine with various values of the dependent variables. Values for ML and MU are still required.

7. ACCESSING THE PACKAGE

CLUTCH can be invoked through a catalogued procedure, and an input deck for any job will follow the following pattern.

```
//BECCRUNA JOB ('account/project',N1),B.E.CLANCY,
// CLASS=,TIME=
/*ROUTE  PUNCH VIEW
// EXEC CLUTCH
//FORT.SYSIN DD *
      FORTRAN source for MYDIFF
      and if desired for MYPEDE,INPUT,GETOUT,OUTPUT,EXPERT,EWSET,VNORM
//GO.SYSIN DD *
     Data cards as described in Section 4.
/*
```

Compilation of the source cards is carried out by the FORTRAN H compiler using level 2 optimisation as default. This is the first step of a three-step procedure. The second step is the use of the linkage editor to link the compiled modules with the rest of the CLUTCH package which contains default versions of the subroutines which the user may provide. The user's source code may include references to other subprograms for which source code is supplied or references to subprograms on existing system libraries. The linkage editor may be directed to these system libraries by use of the PLIB symbols defined in the catalogued procedure listed in Section 9.

7.1 Interactive Version

The package may be run in an interactive mode by beginning the data for any case with the single word TERMINAL . For a fully interactive job, the title card and any labelling items must still be given before the TERMINAL keyword. After interaction starts, all further input is taken from the user's terminal and, when the time comes to insert appropriate input items, the user is prompted by the package. The prompting procedure allows the user to retain previous values of input items without having to re-enter them from the keyboard. In preparing the package, care has been taken to make it react in a friendly way to the user in the interactive mode, so many errors which would cause the job to abend in a batch mode are trapped for correction by the user.

In the interactive mode, an additional output command keyword LIST can be used together with those described in Section 4.4 . The syntax and effect of this command is just like that of the command PRINT except that the resulting output is sent directly to the user's terminal. Such output will be very expansive when the case input requests solution at a large number of T-values, and the terminal listing may be interrupted at any time so that the user can

browse through the results or terminate the listing if that is desirable. Touching any key on the terminal will cause the interrupt.

When using the package in an interactive mode, the user will often wish to vary initial values of parameters or variables and to resolve the equations over the same range of T-values. To avoid having to enter the T range over and over again the user may, when asked to enter the T-values, simply enter the keyword SAME at the terminal. The previously used values are then inserted by the package.

In interactive mode the package also remembers the last output-producing command entered together with any modifiers. This last command can be re-invoked by entering the pseudo-command REPEAT at the terminal.

Help is available to the interactive user at the output stage. The appropriate command is HELP. This produces a list of available commands together with a brief description of their effects.

8. ERROR CONTROL SUBROUTINES

For some problems the user may wish to specify the local error control in a fashion different from that implemented in the standard form of the package. Before implementing such a procedure the user should read the section in the LSODE User's Manual dealing with optionally replaceable solver routines [Hindmarsh, 1981].

The two routines which can be replaced are

(i) SUBROUTINE EWSET (N, ITOL, RTOL, ATOL, YCUR, EWT) in which YCUR contains, on entry, the current dependent variable vector and EWT is the array of positive weights to be set by EWSET.

(ii) DOUBLE PRECISION FUNCTION VNORM (N, V, EWT) in which EWT contains, on entry, the array of positive weights as set by EWSET and VNORM is to be calculated as a norm of the vector V of length N.

The default version of EWSET returns in (EWT(I),I=1,N) values which are those quantities ERR(I) described in Section 4.2. If the user supplies EWSET, it must also return in (EWT(I),I=1,N) positive quantities suitable for

comparing with errors in Y(I). The EWT array returned by EWSET is passed to the VNORM routine and also used by LSODE in the computation of related quantities.

The default version of VNORM returns a root mean square norm so that N times the square of VNORM is the sum of the squares of elements V(I)/EWT(I). If the user supplies this function, it should return a non-negative value of VNORM suitable for use in the error control in LSODE. The weight array EWT may be used as needed, but should not be altered. for example, a user-supplied VNORM routine might:

- substitute a max-norm of (V(I)/EWT(I)) for the r.m.s-norm, or
- ignore some components of V in the norm, with the effect of suppressing the error control on those components of Y.

9. CODE LISTINGS

9.1 Catalogued Procedure

```
    //* CLUTCH - AN INTERFACE TO THE LSODE SUBROUTINES - B.E.CLANCY.
//*********************************************************************
//* USER PROVIDES SOURCE FOR SUBROUTINE MYDIFF STARTING LIKE THIS
//*       SUBROUTINE MYDIFF(N,T,Y,YDOT,NPARAM,PARAM,NFROM,*)
//*       REAL*8 T,Y(N),YDOT(N),PARAM(NPARAM)
//*C      T=INDEPENDENT VARIABLE
//*C      Y IS ARRAY OF DEPENDENT VARIABLES
//*C      YDOT IS ARRAY OF DERIVATIVES TO BE CALCULATED IN MYDIFF
//*C      PARAM IS ARRAY OF PARAMETERS
//*C      NFROM=0 WHEN CALLED FROM INTERNALS OF THE PACKAGE,
//*C           =1 WHEN CALLED FROM THE MAIN ROUTINE AT THE END OF A STEP
//*C ERROR RETURN TO FORCE TERMINATION OF CASE.
//*********************************************************************
//CLUTCH        PROC OPT=2,PLIB='AAE.FORTLIB',PLIB1='AAE.FORTLIB',
//              PLIB2='AAE.FORTLIB',PLIB3='AAE.FORTLIB',
//              GEAR=LOWGEAR,PLKED=LIST,REGGO=800K,FORTLIB=R201
//FORT    EXEC  PGM=IEKAA00,REGION=&REGGO,PARM='OPT=&OPT'
//STEPLIB  DD   DSN=&FORTLIB..PLINKLIB,DISP=SHR
//SYSPUNCH DD   SYSOUT=B
```

```
//SYSLIN    DD  DSN=&&OBJ,UNIT=VIO,DISP=(,PASS),SPACE=(3200,(20,40)),
//              DCB=(RECFM=FB,LRECL=80,BLKSIZE=3200)
//SYSPRINT DD   SYSOUT=A
//SYSUT2    DD  DSN=&&UT2,UNIT=VIO,SPACE=(1700,(50,100))
//LKED    EXEC  PGM=IEWL,PARM='&PLKED',COND=(5,LT)
//SYSLIN    DD  DSN=&&OBJ,DISP=(OLD,DELETE)
//          DD  DSN=BEC.CLUTCH.SYSLIN(&GEAR),DISP=SHR
//          DD  DDNAME=SYSIN
//SYSPRINT DD   SYSOUT=A
//SYSUT1    DD  DSN=&&UT1,UNIT=VIO,SPACE=(1700,(200,100))
//SYSLIB    DD  DSN=&FORTLIB..FORTLIB,DISP=SHR
//          DD  DSN=BEC.CLUTCH.PROGRAM,DISP=SHR
//          DD  DSN=PHYS.FORTLIB,DISP=SHR
//          DD  DSN=&PLIB,DISP=SHR
//          DD  DSN=&PLIB1,DISP=SHR
//          DD  DSN=&PLIB2,DISP=SHR
//          DD  DSN=&PLIB3,DISP=SHR
//SYSLMOD   DD  DSN=&&CLUTCH(USERSPGM),
//              UNIT=VIO,SPACE=(TRK,(40,20,01)),DISP=(NEW,PASS)
//SYSIN     DD  DUMMY
//GO      EXEC  PGM=*.LKED.SYSLMOD,COND=(5,LT),REGION=&REGGO
//FT01F001 DD   DDNAME=SYSIN
//FT02F001 DD   SYSOUT=B
//FT03F001 DD   SYSOUT=A
//FT06F001 DD   SYSOUT=A
//FT19F001 DD   DSN=&&UNIT19,UNIT=VIO,DISP=(,PASS),
//              DCB=(RECFM=FB,LRECL=80,BLKSIZE=800),SPACE=(80,(2,4))
//FT20F001 DD   DSN=&&UNIT20,UNIT=VIO,DISP=(,PASS),
//              DCB=(RECFM=FB,LRECL=80,BLKSIZE=800),SPACE=(80,(2,4))
//AEPLOT    DD  SYSOUT=C
//* END OF PROCEDURE CLUTCH
```

9.2 Basic Coding for Replaceable Subroutines

```
      SUBROUTINE INPUT(MF,EPS,TSTART,N,Y,NPARAM,PARAM,IC,*)
     REAL*8 Y(20),PARAM(20),EPS,TSTART
C ... INPUT MUST RETURN VALUES FOR MF,EPS,TSTART,N,Y,NPARAM,PARAM
C ... IC MUST ONLY BE USED AS COLUMN POINTER IN SCAN CALLS
C ... ERROR RETURN CAN BE TAKEN IF INPUT ERRORS ARE DETECTED
C -------------------------------------------------------------
```

```
      etc.
      RETURN
      END
      SUBROUTINE GETOUT(TOUT,IC,*)
      REAL *8 TOUT
C RETURNS VALUE FOR TOUT, THE NEXT OUTPUT POINT
C IC MUST ONLY BE USED AS COLUMN POINTER IN CALLS TO SCAN
C INTERACTIVE VERSION MAY BE CALLED, IT CAN
C USE ERROR RETURN 1 IF INPUT ERROR OR TO STOP INTEGRATION AT LAST TOUT
C -------------------------------------------------------------------
      etc.
      RETURN
      END
      SUBROUTINE OUTPUT(TVALUE,YVALUE,N,PVALUE,NPARAM,VALUE,NVALUE,IC,
     &*)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION TVALUE(2001)
      DIMENSION YVALUE(2001,20),PVALUE(2001,20),VALUE(2001,40)
C ...TVALUE IS ARRAY OF INDEPENDENT VARIABLE VALUES (NVALUE IN LENGTH)
C ...YVALUE "  "     "    DEPENDENT "        "
C ...PVALUE "  "     "    PARAMETER VALUES
C ...THE CALLING PROGRAM CONTAINS THE FOLLOWING EQUIVALENCE STATEMENT
C ...EQUIVALENCE (VALUE(1,1),YVALUE(1,1)),(VALUE(1,21),PVALUE(1,1))
C ...IC MUST ONLY BE USED AS COLUMN POINTER IN CALLS TO SCAN
C -----------------------------------------------------------
      etc.
      RETURN
      END
      SUBROUTINE MYPEDE(N,T,Y,ML,MU,PD,NO,NPARAM,PARAM,NFROM,*)
      REAL*8 PD(NO,NO),Y(N),PARAM(NPARAM),T
C     T IS CURRENT VALUE OF INDEPENDENT VARIABLE
C     Y IS ARRAY OF CURRENT VALUES OF DEPENDENT VARIABLES
C IF(MITER.EQ.1)  PD IS FULL JACOBIAN ARRAY TO BE CALCULATED
C                 PD(I,J)=DERIVATIVE OF YDOT(I) WITH RESPECT TO Y(J)
C
C IF(MITER.EQ.4)  PD IS NON-ZERO ELEMENTS OF BANDED JACOBIAN
C                 ROWS (1    TO MU     ) ARE SUPERDIAGONALS OF JACOBIAN
C                 ROW  (MU+1           )  IS  MAIN  DIAGONAL
C                 ROWS (MU+2 TO MU+ML+1) ARE   SUBDIAGONALS
C-------------------------------------------------------------------
```

```
      etc.
      RETURN
      END
      SUBROUTINE EXPERT(TVALUE,YVALUE,N,PVALUE,NPARAM,VALUE,NVALUE,IC)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION TVALUE(2001)
      DIMENSION YVALUE(2001,20),PVALUE(2001,20),VALUE(2001,40)
C -------------------------------------------------------
C ...TVALUE IS ARRAY OF INDEPENDENT VARIABLE VALUES (NVALUE IN LENGTH)
C ...YVALUE "  "     "    DEPENDENT "        "
C ...PVALUE "  "     "    PARAMETER VALUES
C ...THE CALLING PROGRAM CONTAINS THE FOLLOWING EQUIVALENCE STATEMENT
C ...EQUIVALENCE (VALUE(1,1),YVALUE(1,1)),(VALUE(1,21),PVALUE(1,1))
C ...IC MUST ONLY BE USED AS COLUMN POINTER IN CALLS TO SCAN
C -------------------------------------------------------
CONTRL-----------------------------------------------------------------
      COMMON/CONTRL/
     & INTACT,IWHERE,LABEL0,LABEL8(40),LABEL4(80),NOCHAN(40)
     &,HEADER(10),IINPUT,IOUTPT,IPEDER,ISYSIN,ISYSOU,ISYSPU,INTIN,INTOUT
     &,RWORK(1422),ATOL(20),RTOL(20),IWORK(40),LRW,LIW
      REAL*8 LABEL0,LABEL8
C ISYSIN        ISYSOU        ISYSPU        INTIN          INTOUT
C SYSIN         PRINTER       PUNCH         KEYBOARD       TERMINAL
C                                            INPUT          OUTPUT
C                                          ( WHEN JOB IS INTERACTIVE )
C DEFINED IN MAIN
      REAL*8 TCRIT,HU,HCUR,TCUR,TOLSF,H0,HMAX,HMIN
      EQUIVALENCE
     & (RWORK( 1),TCRIT ),
     & (RWORK( 5),H0    ),(RWORK( 6),HMAX  ),(RWORK( 7),HMIN  ),
     & (RWORK(11),HU    ),(RWORK(12),HCUR  ),(RWORK(13),TCUR  ),
     & (RWORK(14),TOLSF )
      EQUIVALENCE
     & (IWORK( 1),ML    ),(IWORK( 2),MU    ),(IWORK( 5),MAXORD),
     & (IWORK( 6),MXSTEP),(IWORK( 7),MXHNIL),
     & (IWORK(11),NST   ),(IWORK(12),NFE   ),(IWORK(13),NJE   ),
     & (IWORK(14),NQU   )
CONTRL-----------------------------------------------------------------
C USELESS EXPERT
      WRITE(IOUTPT,1)
```

```
1 FORMAT(' NO EXPERT HELP AVAILABLE')
  RETURN
  END
```

10. CAVEAT

CLUTCH is not a panacea for all problems. Some thought and problem analysis is necessary before it can be used. Mistakes made in casting a problem into a form suitable for solution with the package will be reflected in results which are just as wrong as the input.

For some problems a high degree of accuracy in the numerical integration procedure may be necessary if sensible answers are to be produced; and the user must always make a technical judgement of the correctness of any numerical results produced by the package. For such problems, the user is recommended to experiment with different values of the input items MF and EPS.

Some problems are by their very nature, unsuited to solution by direct means because small numerical inaccuracies can accumulate as 'solution' proceeds and may produce unexpected results. The simple Example 1 given in Section 2 provides a case in point and the curious user should attempt to execute this problem with the pendulum initially at rest in an unstable equilibrium position at time zero. Even so, the package may still be useful for a number of applications.

11. REFERENCES

Bennett N. W. and Pollard J. P. [1967] SCAN - A free input subroutine for the IBM360. AAEC/TM399

Gear C. W. [1971] Numerical initial value problems in ordinary differential equations. Prentice-Hall, Englewood Cliffs, N.J.

Hindmarsh A. C. [1974] GEAR ordinary differential equation solver. UCID-30001, Rev 3.

Hindmarsh A. C. [1981] LSODE. National Energy Software Center Note 81-70, Argonne National Laboratory.

Hindmarsh A. C. [1981] LSODE. National Energy Software Center Note 81-70, Argonne National Laboratory.