1 SUMMARY

To solve a system of ordinary differential equations or differential algebraic equations of index less than or equal to one. There are N variables, Y_i, $i=1,\ldots,N$, which satisfy the equations

$$F_j\,(t, Y_i, Y_i') = 0\,, \qquad j=1,\ldots,N, \tag{1}$$

where

$$Y_i' = \frac{dY_i}{dt} \text{ is the } t \text{ derivative of the variable } Y_i.$$

`DC06A/AD` solves the **initial value problem** for these equations; that is, given the initial values of the variables, $Y_i(t_0)$, at some point, t_0, it advances the solution to equations (1) forward in t.

ATTRIBUTES — Version: 1.0.0. **Types:** `DC06A`, `DC06AD`. **Calls:** `_GEFA`, `_GESL`, `_GBFA`, `_GBSL`. **Original date:** July 1991. **Origin:** A.R.Curtis, Harwell.

2 HOW TO USE THE PACKAGE

2.1 General remarks

We write equations (1) in vector notation as

$$\mathbf{F}(t, \mathbf{Y}, \mathbf{Y}') = \mathbf{0}. \tag{2}$$

Some of the components of **Y** are purely algebraic and thus their derivatives are absent from equation (2). If there are N_a such algebraic variables we can split **Y** into two vectors: $\mathbf{Y}_1$ of length N_a, these are known as **algebraics**, and $\mathbf{Y}_2$ of length $N-N_a$ which we shall refer to as **variables**.

In turn (2) may be rewritten as

$$\begin{aligned} \mathbf{F}_1(t, \mathbf{Y}_1, \mathbf{Y}_2) &= \mathbf{0}\,, \\ \mathbf{F}_2(t, \mathbf{Y}_1, \mathbf{Y}_2, \mathbf{Y}_2') &= \mathbf{0}\,, \end{aligned} \tag{3}$$

where $\mathbf{F}_1$ and $\mathbf{F}_2$ are vectors of length N_a and $N-N_a$ respectively.

A special case, which often occurs in practice, is when the second equation in (3) can be inverted to find a solution for the derivatives. When this happens the equations are said to take an **explicit form**

$$\begin{aligned} \mathbf{0} &= \mathbf{G}_1(t, \mathbf{Y}_1, \mathbf{Y}_2)\,, \\ \mathbf{Y}_2' &= \mathbf{G}_2(t, \mathbf{Y}_1, \mathbf{Y}_2)\,. \end{aligned} \tag{4}$$

Another special (and common) case occurs when there are no algebraic variables, $N_a=0$. In this case we have the pure ordinary differential equation system (of index 0)

$$\mathbf{F}_2(t, \mathbf{Y}_2, \mathbf{Y}_2') = \mathbf{0}\,,$$

or

$$\mathbf{Y}_2' = \mathbf{G}_2(t, \mathbf{Y}_2)\,, \tag{5}$$

if the equations are also explicit.

2.1.1 Information needed to integrate the equations

When integrating a (stiff) problem the integrator needs to be able to form a **Newton iteration matrix** which is of the form

$$\begin{pmatrix} W_{11} & W_{12} \\ W_{21} & W_{22} \end{pmatrix} = \begin{pmatrix} -J_{11} & \alpha J_{12} \\ -J_{21} & \alpha J_{22} + \frac{\beta}{h} K_{22} \end{pmatrix} ,$$

where J is the **Jacobian**

$$J_{ij} = \frac{\partial F_i}{\partial Y_j} ,$$

and K is the **K-Jacobian**

$$K_{ij} = \frac{\partial F_i}{\partial Y'_j} .$$

DC06A/AD therefore needs to be able to evaluate the Jacobian. In addition it will need to be able to form the K-Jacobian if the equations are implicit. This can be done either automatically using finite difference techniques or analytically if the user is able to supply FORTRAN subroutines to evaluate them.

The linear algebra method used to invert W can be chosen to reflect the special form of the iteration matrix. There are two linear algebra options available; full and banded.

2.1.2 Modes of operation of the subroutine

The subroutine has been written so that it may be called in one of four modes.

One step integration mode: In this mode the integrator will integrate the equations for one step over a step size that it deems appropriate. If the program is called in this mode then it will return with the values of t and the variables and derivatives at the end of the next step.

Interval integration mode: In this mode the program will take as many steps as are necessary to integrate the equations up to a given value of t. It will return with the variable and derivative values at the given t point. Any information obtain on intermediate steps will not be available to the user.

Interpolation mode: In this mode the program can be used to evaluate the values of any of the variables or their derivatives at any value of t within the last completed step.

Extrapolation mode: In this mode the program can be used to evaluate estimates for the values of any of the variables or their derivatives at values of t just outside the last completed step.

2.2 Argument lists and calling sequences

2.2.1 Arguments for the first call

The first call to `DC06A/AD` initialises the problem. It is this call that is used to set up the problem in hand and the linear algebra solvers that are going to be used in its solution. In this section we detail the parameters that must be passed to `DC06A/AD` on such a call and the values it returns.

The single precision version

```
      CALL  DC06A(MODE,NEW,JAC,KAC,JPT,KPT,FCN,RPAR,IPAR,NEQ,
     *            NALGB,T,TEND,TSTOP,LSTOP,LDMAIN,YVAL,YDER,TOL,
     *            HMAX,INFORM,RVALS,IVALS,RWORK,LRW,IWORK,LIW,
     *            LPS1,LPO1,IDEBUG,IDID)
```

The double precision version

```
   CALL  DC06AD(MODE,NEW,JAC,KAC,JPT,KPT,FCN,RPAR,IPAR,NEQ,
  *           NALGB,T,TEND,TSTOP,LSTOP,LDMAIN,YVAL,YDER,TOL,
  *           HMAX,INFORM,RVALS,IVALS,RWORK,LRW,IWORK,LIW,
  *           LPS1,LPO1,IDEBUG,IDID)
```

`MODE` is an `INTEGER` variable. This must be set to 1 if the subroutine is to return after integrating one step or 2 if the subroutine is to return after reaching a given *t* value `TEND`.

`NEW` is an `INTEGER` variable. This must be set to 0 on the first call to `DC06A/AD`.

`JAC` is a user supplied `LOGICAL` function for evaluating the Jacobian. This is only required if the user sets `INFORM(4)=2`. In all other cases the name of a dummy routine may be supplied to `DC06A/AD` and it will evaluate the Jacobian numerically. Details of this function are given in section 2.3.2

`KAC` is a user supplied `LOGICAL` function for evaluating the K-Jacobian. This is only required if the equations are implicit, `INFORM(1)=2`, and the user sets `INFORM(5)=2`. In all other cases the name of a dummy routine may be supplied to `DC06A/AD` and it will evaluate the K-Jacobian numerically. Details of this function are given in section 2.3.2

`JPT` is a user supplied `LOGICAL` function for

`JPT` is a user supplied `LOGICAL` function. This option is not implemented so name of a dummy routine should be supplied to `DC06A/AD`.

`KPT` is a user supplied `LOGICAL` function. This option is not implemented so name of a dummy routine should be supplied to `DC06A/AD`.

`FCN` is a user supplied `LOGICAL` function for evaluating the derivative function. This function specifies the problem and it must always be present. The form of this function is described in section 2.3.1

`RPAR` is a `REAL` (`DOUBLE PRECISION` in the D version) array of length chosen by the user that is used to pass parameter values to the derivative function `FCN` and the Jacobian evaluation subroutines `JAC`, `KAC`. This is described in section 2.3.2 Its contents are unaltered by `DC06A/AD`.

`IPAR` is an `INTEGER` array of length chosen by the user that is used to pass parameter values to the derivative function `FCN` and the Jacobian evaluation subroutines `JAC`, `KAC`. This is described in section 2.3.2 Its contents are unaltered by `DC06A/AD`.

`NEQ` is an `INTEGER` variable set to the total number of equations. It must be greater than 0.

`NALGB` is an `INTEGER` variable set to the total number of purely algebraic equations. It can be set equal to 0.

`T` is a `REAL` (`DOUBLE PRECISION` in the D version) variable set to the initial value of the independent variable, *t*. On return it contains the value of the dependent variable at the point at which the integration ceased. Assuming that it is not an error return then this is either the end of the first step if `MODE=1` or `TEND` if `MODE=2`.

`TEND` is a `REAL` (`DOUBLE PRECISION` in the D version) variable set to the value of the independent variable, *t*, at which the integration should cease. This will only take effect if the program is called with `MODE=2`. In this case it may also be used to estimate an initial stepsize if no value is given for `HMAX`.

`TSTOP` is a `REAL` (`DOUBLE PRECISION` in the D version) variable set to the upper limit of the independent variable. This should be set when the equations are not defined beyond a given value of *t* (because of a singularity in the derivative function, for example). In such a case set `TSTOP` equal to this value and `LSTOP` to `.TRUE.` If `LSTOP` is set to `.FALSE.` the value of `TSTOP` is ignored.

`LSTOP` is a `LOGICAL` variable set to `.TRUE.` if a `TSTOP` has been set and `.FALSE.` otherwise.

`LDMAIN` is a `LOGICAL` variable set to `.TRUE.` to turn on checking of the domain bounds and `.FALSE.` otherwise. This option is described in section 2.2.5

YVAL is a REAL (DOUBLE PRECISION in the D version) array of length NEQ set to the initial values of the dependent variables $\mathbf{Y}$. The values of the algebraic variables, $\mathbf{Y}_1$, need only be 'good' initial guesses. On return this array holds the values of the dependent variables, $\mathbf{Y}$, at the value of the independent variable given in T.

YDER is a REAL (DOUBLE PRECISION in the D version) array of length NEQ. If the equations are implicit then set this array to initial guesses of $\mathbf{Y}' = d\mathbf{Y}/dt$; zero is a good enough initial guess for any component which occurs linearly in equation (2). Initial guesses for derivatives of algebraic components, $\mathbf{Y}'_1$, are not required. On return this array holds the values of the derivatives of the dependent variables, $\mathbf{Y}'$, at the value of the independent variable given in T.

TOL is A REAL (DOUBLE PRECISION in the D version) variable used to specify the relative accuracy required in the solution components (actually, relative to comparison values YTST(I) whose default calculation is described in the writeup for DC05). TOL must be given a positive value. If the value specified is larger than 10^{-2} or smaller than a number TOLMIN (which is equal to 10^{-10} on most computers), it will be reset to the corresponding limit. The integration will then be performed with this modified tolerance.

HMAX is a REAL (DOUBLE PRECISION in the D version) variable set to a trial initial step size that DC06A/AD will not exceed. It is used as a guide to the scale of the independent variable t. If a non-positive value is given and DC06A/AD has been called with MODE=2 then it will use the default value of TOL*(TEND-T). If DC06A/AD is unable to form a trial initial step size then it will return with an error.

INFORM is an INTEGER array of length 5. This array is used on the first call to DC06A/AD to communicate details of the equations and the linear algebra methods that will be used to solve them. To set the entries answer the following five questions:

(i) Are the equations implicit?

No –set INFORM(1)=1
Yes –set INFORM(1)=2

(ii) Is there a KMAX, maximum order of integration formulae, set?

No –set INFORM(2)=0
Yes –set INFORM(2)=KMAX

(iii) Which linear algebra option do you require?

Full –set INFORM(3)=1
Banded –set INFORM(3)=2 and set IVALS(1)=ML and IVALS(2)=MU.

where ML is the number of non-zero diagonals below the main diagonal and MU is the number of non-zero diagonals above the main diagonal.

(iv) How is the Jacobian to be evaluated?

Numerically –set INFORM(4)=1
Analytically –set INFORM(4)=2 and supply the user subroutine given as JAC.

(v) How is the K-Jacobian to be evaluated?

Numerically –set INFORM(5)=1
Analytically –set INFORM(5)=2 and supply the user subroutine given as KAC.

No check is made to see whether the elements of INFORM are changed between calls but its values are only used when NEW is set equal to 0 or 1.

RWORK is a REAL (DOUBLE PRECISION in the D version) array of length LRW which is used to provide work space.

LRW is an INTEGER constant set to the length of RWORK. This will vary depending on which options are chosen. An estimate of the minimum size of RWORK for each of the two linear algebra options is given below:

Full LRW = 65+NEQ*(12+2*NEQ)+(NEQ-NALGB)*(NEQ-NALGB).

Banded	`LRW = 65+NEQ*(14+3*ML+2*MU)+(NEQ-NALGB)*(ML+MU-1).`

where `ML` is the number of non-zero diagonals below the main diagonal and `MU` is the number of non-zero diagonals above the main diagonal.

`IWORK` is an `INTEGER` array of length `LIW` which is used to provide work space.

`LIW` is an `INTEGER` constant set to the length of `IWORK`. This will vary depending on which options are chosen. An estimate of the minimum size of `IWORK` for each of the two linear algebra options is given below:

Full	`LIW = 40+NEQ.`
Banded	`LIW = 40+NEQ`

`LPS1` is an `INTEGER` constant set to the unit number on which the user wishes any standard output, including fatal error messages, to appear.

`LPO1` is an `INTEGER` constant set to the unit number on which the user wishes any optional output, including warning error messages, to appear.

`IDEBUG` is an `INTEGER` constant that controls the level of debug output.

`0`	No debug output.
`1`	Monitor output only.
`2`	Monitor output plus the values of *t*, variables and derivatives at any point where `DC05A/AD` tries to evaluate the derivative function outside its domain.

The output appears on unit `LP01`.

`IDID` is an `INTEGER` variable set by the code. A value of `IDID=1` signifies that `DC06A/AD` has successfully performed the required integration on the new problem. A negative value denotes an error return. These are described fully in section 2.6.

2.2.2 Subsequent calls of DC06A/AD

On subsequent calls to the program the user may either continue the integration, before or after modifying the defining equations, or interpolate values of the variables and derivatives at a point in the last completed step. In exceptional circumstances they might also wish to use the interpolating polynomials to extrapolate values outside this step. These four options are discussed in turn.

2.2.2.1 Continue integration

The user may continue the integration in one of two modes. These modes may be changed on subsequent calls to the subroutine. In this section we only discuss parameter values different from those of section 2.2.1 All other arguments **must not be altered** since the last call to `DC06A/AD` with `NEW` equal to 0 or 1.

The single precision version

```
      CALL  DC06A(MODE,NEW,JAC,KAC,JPT,KPT,FCN,RPAR,IPAR,NEQ,
     *            NALGB,T,TEND,TSTOP,LSTOP,LDMAIN,YVAL,YDER,TOL,
     *            HMAX,INFORM,RVALS,IVALS,RWORK,LRW,IWORK,LIW,
     *            LPS1,LPO1,IDEBUG,IDID)
```

The double precision version

```
      CALL  DC06AD(MODE,NEW,JAC,KAC,JPT,KPT,FCN,RPAR,IPAR,NEQ,
     *            NALGB,T,TEND,TSTOP,LSTOP,LDMAIN,YVAL,YDER,TOL,
     *            HMAX,INFORM,RVALS,IVALS,RWORK,LRW,IWORK,LIW,
     *            LPS1,LPO1,IDEBUG,IDID)
```

`MODE` is an `INTEGER` variable set to 1 for one more step or 2 for integration up to `TEND`.

`NEW` is an `INTEGER` variable set to equal to 2.

`T` is a `REAL` (`DOUBLE PRECISION` in the D version) variable. On return this is set to the value of the independent variable at the point at which the integration ceased. Assuming that it is not an error return then this is either the end of the first step if `MODE=1` or `TEND` if `MODE=2`.

`TEND` is a `REAL` (`DOUBLE PRECISION` in the D version) variable set to the value of the independent variable, *t*, at which the integration should cease. This will only take effect if the program with `MODE=2`.

`TSTOP` is a `REAL` (`DOUBLE PRECISION` in the D version) variable set to the upper limit of the independent variable. This should be set when the equations are not defined beyond a given value of *t* (because of a singularity in the derivative function, for example). In such a case set `TSTOP` equal to this value and `LSTOP` to `.TRUE.` If `LSTOP` is set to `.FALSE.` the value of `TSTOP` is ignored.

`LSTOP` is a `LOGICAL` variable set to `.TRUE.` if a `TSTOP` has been set and `.FALSE.` otherwise.

`LDMAIN` is a `LOGICAL` variable set to `.TRUE.` to turn on checking of the domain bounds and `.FALSE.` otherwise. This option is described in section 2.2.5

`YVAL` is a `REAL` (`DOUBLE PRECISION` in the D version) array of length `NEQ`. On return this array holds the values of the independent variables, **Y**, at the value of the independent variable given in `T`.

`YDER` is a `REAL` (`DOUBLE PRECISION` in the D version) array of length `NEQ`. On return this array holds the values of the derivatives of the dependent variables, **Y'**, at the value of the independent variable given in `T`.

`TOL` is A `REAL` (`DOUBLE PRECISION` in the D version) variable used to specify the relative accuracy required in the solution components (actually, relative to comparison values `YTST(I)` whose default calculation is described in the writeup for `DC05`). `TOL` must be given a positive value. The tolerance may be changed at any time. Increasing it may make the equations easier to integrate. Conversely decreasing it makes them harder to integrate.

`HMAX` is a `REAL` (`DOUBLE PRECISION` in the D version) variable set to the maximum step size that `DC06A/AD` is allowed to take on the next step. The maximum step size may be altered on each call to `DC06A/AD` if required.

`IDEBUG` is an `INTEGER` constant that controls the level of debug output.

`0`	No debug output.
`1`	Monitor output only.
`2`	Monitor output plus the values of *t*, variables and derivatives at any point where `DC05A/AD` tries to evaluate the derivative function outside its domain.

The output appears on unit `LP01`.

`IDID` is an `INTEGER` variable set by the code. A value of `IDID=1` signifies that `DC06AD` has successfully performed the required integration. A negative value denotes an error return. These are described fully in section 2.6

2.2.2.2 Change equations and continue integration

There are times when the user may wish to change the problem at a particular value of *t* and then carry on the integration. Some examples of this are given below:

• A discontinuity in the derivative function.
• Changes to `RPAR` and/or `IPAR`.
• A discontinuous change in a value of one or more of the variables.
• A change in the number of equations.

The integration may be continued after such alterations have been made by calling `DC06A/AD` with `NEW` set to 0 or 1. A value of 0 resets `DC06A/AD`'s statistical counts whilst a value of 1 does not; this is the only difference between them. If this is done the problem is then treated as new one and the user should refer to section 2.2.1 It may be that most of the arguments may be left unaltered from the previous call, especially if the last returned values of *t*, the variables and the derivatives are the ones from which the integration is to continue.

2.2.2.3 Interpolation

As well as returning the end of step values `DC06A/AD` has polynomial approximations of the variables and derivatives that may be used to find interpolated values over the last complete step. In this section we only discuss parameter values different from those of section 2.2.1 All other arguments **must not be altered** since the last call to `DC06A/AD` with `NEW` equal to 0 or 1.

The single precision version

```
      CALL  DC06A(MODE,NEW,JAC,KAC,JPT,KPT,FCN,RPAR,IPAR,NEQ,
     *           NALGB,T,TEND,TSTOP,LSTOP,LDMAIN,YVAL,YDER,TOL,
     *           HMAX,INFORM,RVALS,IVALS,RWORK,LRW,IWORK,LIW,
     *           LPS1,LPO1,IDEBUG,IDID)
```

The double precision version

```
      CALL  DC06AD(MODE,NEW,JAC,KAC,JPT,KPT,FCN,RPAR,IPAR,NEQ,
     *           NALGB,T,TEND,TSTOP,LSTOP,LDMAIN,YVAL,YDER,TOL,
     *           HMAX,INFORM,RVALS,IVALS,RWORK,LRW,IWORK,LIW,
     *           LPS1,LPO1,IDEBUG,IDID)
```

`MODE` is an `INTEGER` constant set to 3.

`NEW` is an `INTEGER` constant set to 2.

`T` is a `REAL` (`DOUBLE PRECISION` in the D version) variable set to the value of the independent variable at which the interpolated values are required.

`LDMAIN` is a `LOGICAL` variable set to `.TRUE.` to turn on checking of the domain bounds and `.FALSE.` otherwise. This option is described in section 2.2.5

`YVAL` is a `REAL` (`DOUBLE PRECISION` in the D version) array of length `NEQ`. On return this array holds the values of the dependent variables, **Y**, at the point of interpolation.

`YDER` is a `REAL` (`DOUBLE PRECISION` in the D version) array of length `NEQ`. On return this array holds the values of the derivatives of the dependent variables, **Y'**, at the point of interpolation.

`IDEBUG` is an `INTEGER` constant that controls the level of debug output.

`0`	No debug output.
`1`	Monitor output only.
`2`	Monitor output plus the values of *t*, variables and derivatives at any point where `DC05A/AD` tries to evaluate the derivative function outside its domain.

The output appears on unit `LP01`.

`IDID` is an `INTEGER` variable set by the code. This will return with a value 1 if the interpolation was successful. A negative value denotes an error return. These are described fully in section 2.6 The most important errors specific to the interpolation mode are:

`IDID=-104`	This was the first call to `DC06A/AD` so no interpolated values could be returned.
`IDID=-204`	Supplied *t* point was outside the last completed step so that the returned values are extrapolated rather than interpolated values. As a consequence they may be grossly inaccurate.

Both these errors are fatal errors. The user may, however, continue calling DC06A/AD after the latter one if they are really sure that they either want to use the returned values or just to ignore them. To do this they must reset IDID to 1.

2.2.2.4 Extrapolation

The user may occasionally wish to look at extrapolated values of the variables and derivatives, that is their predicted values for *t* just outside the last completed step. As this is an extrapolation rather than interpolation the accuracy of the values may be very poor. However, it may be useful in predicting when a condition might occur so as to allow the maximum stepsize to be modified accordingly. In this section we only discuss parameter values different from those of section 2.2.1 All other arguments **must not be altered** since the last call to DC06A/AD with NEW equal to 0 or 1.

The single precision version

```
      CALL  DC06A(MODE,NEW,JAC,KAC,JPT,KPT,FCN,RPAR,IPAR,NEQ,
     *            NALGB,T,TEND,TSTOP,LSTOP,LDMAIN,YVAL,YDER,TOL,
     *            HMAX,INFORM,RVALS,IVALS,RWORK,LRW,IWORK,LIW,
     *            LPS1,LPO1,IDEBUG,IDID)
```

The double precision version

```
      CALL  DC06AD(MODE,NEW,JAC,KAC,JPT,KPT,FCN,RPAR,IPAR,NEQ,
     *            NALGB,T,TEND,TSTOP,LSTOP,LDMAIN,YVAL,YDER,TOL,
     *            HMAX,INFORM,RVALS,IVALS,RWORK,LRW,IWORK,LIW,
     *            LPS1,LPO1,IDEBUG,IDID)
```

MODE is an INTEGER constant set to 4.

NEW is an INTEGER constant set to 2.

T is a REAL (DOUBLE PRECISION in the D version) variable set to the value of the independent variable at which the interpolated values are required.

LDMAIN is a LOGICAL variable set to .TRUE. to turn on checking of the domain bounds and .FALSE. otherwise. This option is described in section 2.2.5

YVAL is a REAL (DOUBLE PRECISION in the D version) array of length NEQ. On return this array holds the values of the dependent variables, **Y**, at the point of extrapolation.

YDER is a REAL (DOUBLE PRECISION in the D version) array of length NEQ. On return this array holds the values of the derivatives of the dependent variables, **Y'**, at the point of extrapolation.

IDEBUG is an INTEGER constant that controls the level of debug output.

0	No debug output.
1	Monitor output only.
2	Monitor output plus the values of the *t*, variables and derivatives at any point where DC05A/AD tries to evaluate the derivative function outside its domain.

The output appears on unit LP01.

IDID is an INTEGER constant set by the code. This will return with a value 1 if the extrapolation was successful. A negative value denotes an error return. These are described fully in section 3. In particular the main ones that are specific to the extrapolation mode are:

IDID=-105	This was the first call to DC06A/AD so no extrapolated values could be returned.
IDID=-24	Supplied *t* point was inside the last completed step so that the returned

values are interpolated rather than extrapolated values.

The first of these errors is fatal.

2.2.3 Domain checking options

If the domain checking option is turned on then DC06A/AD passes information to the integrator DC05A/AD as to whether the variables it supplied where in the domain of the derivative function each time such a function evaluation is requested. The modified DC05A/AD then uses this to ensure that the calculated solution remains in this domain. If it is turned off then DC06A/AD always tells DC05A/AD that the variables were in the domain. In this case exactly the original algorithm that was used in DC05A/AD will be used except that an additional function evaluation will be performed each step along with an extra one per start/restart. The effects of this will be minimised if the user writes their derivative function simply to do domain bounds checking if DC06A/AD requests it. The functions required are described in the next sections.

WARNING: DC06A/AD does not worry about whether the variables it uses to form a numerical approximation to the Jacobian lie inside the domain of the derivative function. This assumption will be valid as long as the values supplied to the Jacobian routine and ‘small’ perturbations about them are in the domain.

2.2.3.1 Domain checking and interpolation In general DC05A/AD has a first to fifth order polynomial that it can use to interpolate, to the required accuracy, within a completed step. However such a high order polynomial is not guaranteed to give results that lie in the domain of definition, even if the end of step values always do. Linear interpolation ensures that the interpolated values lie in the domain of definition of the derivative function if:

(1) The values at the beginning and the end of the step also lie in the domain.

(2) The domain is **convex**; that is, any two sets of values of the variables lying within the domain can be joined by a straight line that remains completedly within the domain. In particular this means that the domain must be simply connected.

Linear interpolation will not however be as accurate as the full interpolation. As a compromise between these two desired properties DC06A/AD does the following if the domain checking option is turned on. Firstly it does a full interpolation in the last completed step. Then it checks to see that the interpolated values lie in the domain. This involves an extra call to the derivative function in domain checking mode. If they do lie in the domain then it returns these values. If they do not DC06A/AD is forced to perform a linear interpolation. In this case it will return with the warning error code IDID=-1 to denote the fact that the results are not as accurate. The user can always force full interpolation at each point by turning the domain checking option off.

2.2.3.2 Domain checking and extrapolation Extrapolation is **not allowed** when the domain checking option is turned on. An attempt to invoke it will cause a fatal error.

2.3 Definition of the subroutines to be supplied by the user

2.3.1 FCN: Derivative function

The user must supply a FORTRAN logical function to evaluate the derivative function. This function should return the value .TRUE. if the variables passed to it lie in the domain of the derivative function and .FALSE. otherwise. The function must be of the form

Single precision version

```
LOGICAL FUNCTION FCN(IDMN,T,YVAL,YDER,RPAR,IPAR,FVAL)
IMPLICIT REAL (A-H,O-Z)
IMPLICIT INTEGER (I-N)
DIMENSION RPAR(*), IPAR(*), YVAL(*), YDER(*) ,FVAL(*)
```

Double precision version

```
        LOGICAL FUNCTION FCN(IDMN,T,YVAL,YDER,RPAR,IPAR,FVAL)
        IMPLICIT DOUBLE PRECISION (A-H,O-Z)
        IMPLICIT INTEGER (I-N)
        DIMENSION RPAR(*), IPAR(*), YVAL(*), YDER(*) ,FVAL(*)
```

IDMN is an INTEGER constant set to 0 if the function is required to perform a full function evaluation as well as a domain check. If it has a value of 1 then only a domain check is required.

T is a REAL (DOUBLE PRECISION in the D version) variable containing the value of the independent variable.

YVAL is a REAL (DOUBLE PRECISION in the D version) array containing the current values of the dependent variables.

YDER is a REAL (DOUBLE PRECISION in the D version) array containing the current values of the derivatives.

RPAR is a REAL (DOUBLE PRECISION in the D version) array that enables the user to pass parameters to FCN.

IPAR is an INTEGER array that enables the user to pass parameters to FCN.

FVAL is a REAL (DOUBLE PRECISION in the D version) array in which the values of the derivative function should be returned. If the equations are implicit (INFORM(1)=1) then the derivative function should consist of the left hand side of equation (3). If the equations are explicit (INFORM(1)=2) then the derivative function should consist of the right hand side of equation (4).

An example program to obtain the derivative function is given in section 5.

2.3.2 JAC: Jacobian (and KAC: K-Jacobian)

The user may supply a FORTRAN logical function to evaluate the Jacobian. This function should return the value .TRUE. if the variables passed to it lie in the domain of the derivative function and .FALSE. otherwise. The function must be of the form

Single precision version

```
      LOGICAL FUNCTION JAC(T,YVAL,JACARR,MY,IWORK,LIW,RPAR,IPAR)
      IMPLICIT REAL (A-H,O-Z)
      IMPLICIT INTEGER (I-N)
      DIMENSION RPAR(*), IPAR(*), IWORK(LIW)
```

Double precision version

```
      LOGICAL FUNCTION JAC(T,YVAL,JACARR,MY,IWORK,LIW,RPAR,IPAR)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      IMPLICIT INTEGER (I-N)
      DIMENSION RPAR(*), IPAR(*), IWORK(LIW)
```

T is a REAL (DOUBLE PRECISION in the D version) variable containing the value of the independent variable.

YVAL is a REAL (DOUBLE PRECISION in the D version) array containing the current values of the variables.

JACARR

RPAR is a REAL (DOUBLE PRECISION in the D version) array that enables the user to pass parameters to FCN.

IPAR is an INTEGER array that enables the user to pass parameters to FCN.

MY

IWORK is an INTEGER array of length LIW which is used to provide work space.

LIW is an INTEGER constant set to the length of IWORK. This will vary depending on which options are chosen. An estimate of the minimum size of IWORK for each of the two linear options is given below:

Full LIW = 40+NEQ.

RPAR is a REAL (DOUBLE PRECISION in the D version) array that enables the user to pass parameters to FCN.

IPAR is an INTEGER array that enables the user to pass parameters to FCN.

The logical function for the K-Jacobian has the same form as that for the Jacobian except that the function name is KAC rather than JAC.

An example program to evaluate the Jacobian function is given in section 5.

2.4 Summary of calling parameters.

	Start integration		Continue integration		Interpolate/Extrapolate
	NEW=0,1		NEW=2		NEW=2
VARIABLE	MODE=1	MODE=2	MODE=1	MODE=2	MODE=3,4
MODE	I	I	I	I	I
NEW	I	I	I	I	I
JAC	I	I	–	–	–
KAC	I	I	–	–	–
JPT	I	I	–	–	–
KPT	I	I	–	–	–
FCN	I	I	–	–	–
RPAR	I	I	–	–	–
IPAR	I	I	–	–	–
NEQ	I	I	–	–	–
NALGB	I	I	–	–	–
T	I/O	I/O	O	O	I/O
TEND	?	I	?	I	?
TSTOP	I	I	(I)	(I)	?
LSTOP	I	I	(I)	(I)	?
LDMAIN	I	I	I	I	I
YVAL	I/O	I/O	O	O	O
YDER	I/O	I/O	O	O	O
TOL	I †	I †	I †	I †	?
HMAX	(I)	(I)	(I)	(I)	?
INFORM	I	I	–	–	–
RVALS	I	I	–	–	–
IVALS	I	I	–	–	–
RWORK	I	I	–	–	–
LRW	I	I	–	–	–
IWORK	I	I	–	–	–
LIW	I	I	–	–	–
LPS1	I	I	–	–	–
LPO1	I	I	–	–	–
IDEBUG	I	I	I	I	I
IDID	O	O	O	O	O

I	= Input, value must be specified.
(I)	= Optional input value.
O	= Output.
–	= Do not alter (after first call)
?	= Value unimportant.

All input values, I and (I), may be changed from call to call.

Notes:

^{A}	denotes the fact that either `TEND` or `HMAX` must be given a non-zero value.
†	= Input, may be modified by `DC06A/AD` if out of range.

2.5 Error returns

There are five stages of the integration process at which the program may return with an error: (1) failure to initialise or re-initialise the problem; (2) error return from the integration subroutine `DC05AD`; (3) error return from the Jacobian/K-Jacobian calculation; (4) error return from the formation of the Newton matrix and its subsequent LU decomposition; (5) error return from solving the correction equations.

These are distinguished internally (in the error message output routine `ALLERR`) by the local variable `ICODE` which runs from 1 to 5. Within each stage there are a number of messages, labelled by `IMESG`, which may be either FATAL in which case `DC06A/AD` will return to the calling program or WARNING MESSAGES in which case `DC06A/AD` proceeds after taking appropriate action and warning the user. FATAL error messages will appear on unit `LPS1` if `LPS10` and WARNING messages will appear on unit `LPO1` if `LPO10`. If an error can occur in a fatal and a non-fatal way, eg convergence failure, then these errors will share the same value of `IMESG`. The error return codes, `IDID`, can be derived from the two numbers `ICODE` and `IMESG` as follows:

```
IDID = -(100*ICODE+IMESG)      if the error is FATAL.
IDID = -(10*ICODE+IMESG)       if the error produces a WARNING only.
```

In addition in the case of fatal errors for stages 3, 4 and 5 it is intended that the second digit in the error return code will denote which linear algebra option was being used when the error was detected:

`0`	Message common to all linear algebra options.
`1`	Full linear algebra option.
`2`	Banded linear algebra option.

The error returns –1 to –9 are reserved for WARNING messages that are passed back to the calling program when some sensible modification has been made to the requested task but the program has otherwise been successful. A value of `IDID=1` always denotes a fully successful return.

2.5.1 Action to be taken on fatal error returns.

If a fatal error is due to the fact that `DC06A/AD` was called with an incorrect or inadmissible value for one of its arguments then this may be corrected and `DC06A/AD` called again if the user first resets `IDID` to 1. If `DC06A/AD` is called with `IDID` less than or equal to –100 then it will stop.

`IDID`	Error	`NEW`	`MODE`
–1	DOMAIN CHECKING ON: Solution was obtained by linear interpolation because ordinary interpolation gave results outside the domain of the derivative function.	2	3

The possible error returns are summarised in the following sections. The last two columns of each table show the possible values of `NEW` and `MODE` that may elicit the corresponding return. A hyphen, –, denotes the fact that any value of the parameter may appear in the argument list.

2.5.2 Failure to initialise or re-initialise the problem

`IDID`	Error	`NEW`	`MODE`
–11	Solution was interpolated rather than integrated because required end of interval was in the last completed step.		2
–101	`DC06A/AD` has been called in an unknown mode.	–	–
–102	`TSTOP` has been set less than the initial value `T` , or the value at the end of the last completed step.	–	–
–103	An element(s) of the array `INFORM` is outside its expected range.	0,1	1,2
–104	Attempt to interpolate on a NEW problem.	0,1	3
–105	Attempt to extrapolate on a NEW problem.	0,1	4
–106	Called in **interval integration mode** with stop value, `TEND`, set to a value that is less than the initial value, `T`.	0,1	2
–107	Called in **interval integration mode** with stop value, `TEND`, set to a value that is less than `T` at end of the last step.	2	2
–108	Program could not calculate a trial initial step size because no `HMAX` was set or `TEND` given in `MODE=2`.	0,1	1,2
–109	Attempt to integrate, interpolate or extrapolate a problem that has not been successfully initialised.	2	–
–110	Attempt to extrapolate a solution with the domain checking option turned on.	2	4
–111	Attempt to interpolate outside the last completed step with the domain checking option turned on.	2	3

2.5.3 Error returns from DC05AD

IDID	Error	NEW	MODE
–21	**Restart** performed because **500 unsuccessful attempts** have been made on the current step but the total successive number of such returns is less than a given cut-off value of N500A.	2	1,2
–22	Relative error tolerance has been changed by DC05A/AD and accepted by DC06A/AD.	–	–
–24	Values were interpolated rather than extrapolated because the given point, T, lies within the last completed step.	2	4
–25	Initial values of **U** and **U'** have failed to converge after 10 iterations but the residual is still reducing therefore **restarting** and trying another 10 iterations.	2	1,2
–26	**Restart** performed because a **repeated convergence failure** has been encountered but the total successive number of such returns on the current step is less than a given cut-off value of NRCFA.	2	1,2
–27	**Restart** performed because a **repeated accuracy failure** has been encountered but the total successive number of such returns on the current step is less than a given cut-off value of NRAFA.	2	1,2
–28	**Restart** performed because a **repeated convergence failure** has been encountered but the total successive number of such returns on the current step is less than a given cut-off value of NRCFA. Failure is due solely to the corrected values lying outside the domain of definition of the dependent variables.	2	1,2
–201	More than N500A lots of 500 unsuccessful attempts on the current step.	2	1,2
–203	Unknown error from DC05A/AD (should never happen).	–	–
–204	Returned values were extrapolated rather than interpolated because the given point, T, lies outside the last completed step. As a consequence they may be **grossly inaccurate**.	2	3
–205	Program has stopped because the initial values of **U** and **U'** have failed to converge and the residual is no longer decreasing.	2	1,2
–206	More than NRCFA successive repeated convergence failures on the current step.	2	1,2
–207	More than NRAFA successive repeated accuracy failures on the current step.	2	1,2
–208	More than NRCFA successive repeated accuracy failures on the current step: – failure is solely due to the corrected values lying outside the domain of definition of the dependent variables.	2	1,2
–209	The iteration of the initial conditions has gone outside the domain of the derivative function.	0,1	1,2
–210	The final converged values of the initial conditions lie outside the domain of the derivative function.	0,1	1,2
–211	DC05A/AD has reduced its initial stepsize to less than HDMIN in an attempt to satisfy the criteria that the variables lie in the domain of the derivative function.	–	1,2
–212	Fatal error from DC05A/AD.	–	–

Default values: The cut-off parameters have the following default values;

```
N500A=3
NRCFA=10
NRAFA=10
```

These values are set in the the common block initialisation routine CMNINT.

2.5.4 Error messages from DC05A/AD

In addition error messages produced by `DC05A/AD` will appear on unit 6. The user is referred to the documentation for `DC05` for further details of such messages.

2.5.5 Error returns from the Jacobian/K-Jacobian calculation

Recall that the error messages from this stage are divided into two types:

`300-309`	Messages common to any linear algebra option.
`310-319`	Full linear algebra option.
`320-329`	Banded linear algebra option.

2.5.5.1 Messages common to any linear algebra option

`IDID`	Error
–301	Not enough room in the `INTEGER` workspace for pivot indices.
–305	Not enough room in the `REAL` workspace for Jacobian.
–306	Not enough room in the `REAL` workspace for K-Jacobian.
–307	Not enough room in the `REAL` workspace for LU decomposition.

2.5.6 Error return from the formation of the Newton matrix and LU decomposition

Recall that the error messages from this stage are divided into three types:

`400-409`	Messages common to all linear algebra options.
`410-419`	Full linear algebra option.
`420-429`	Banded linear algebra option.

2.5.6.1 Full linear algebra

`IDID`	Error
–411	A pivot in the Gaussian elimination of the Newton iteration matrix was zero.

2.5.6.2 Banded linear algebra

`IDID`	Error
–421	A pivot in the Gaussian elimination of the Newton iteration matrix was zero.

3 GENERAL INFORMATION

Use of common: `COMMON` blocks `DC06W/WD`, `DC06X/XD`, `DC06Y/YD`, `DC06Z/ZD`.

Workspace: As described.

Other subroutines: Calls `DGBFA/SGBFA`, `DGBSL/SGBSL`, `DGEFA/SGEFA`, `DGESL/SGESL`, `DC05A/AD`, and user routines `FCN`, `JAC`, `KAC`, `JPT`, `KPT`.

4 METHOD

DC06A/AD uses the stiff differential equation integrator DC05A/AD and the linear algebra solvers. It co-ordinates the solution of the problem by calling routines to perform function evaluations, integrate the problem and solve the resulting linear algebra. It has been written so as to try and handle the less serious error returns from DC05A/AD and only to stop for the fatal or serious errors. It can be called in two fundamentally different modes of operation. In the first mode it will **integrate** a given set of equations either over a specified interval of the independent variable *t*, usually time, or for one *t* step of a length deemed appropriate by DC05A/AD. In the second mode it can be used to **obtain output** by interpolation or extrapolation over a previously integrated region.

5 EXAMPLE OF USE

The example below shows a simple program that may be used to call DC06A/AD so as to get output at a set of equally spaced time points.

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
C TITLE:  TEST                                                         C
C ------                                                               C
C AUTHOR: R.G.Myhill                                                   C
C -------                                                              C
C DATE:   July, 1991.                                                  C
C -----                                                                C
C DESCRIPTION:  This is a routine to integrate test problems over a    C
C ------------  specified interval using DC06AD. It prints out the     C
C               results at a user specified number of equally spaced   C
C               time points. It also calls a user subroutine to        C
C               evaluate the solution at the same points if such a     C
C               routine is available.                                  C
C                                                                      C
C METHOD:  The program uses DC06AD to integrate the problem one step   C
C ------   at a time, checking for output at the end of each step.     C
C                                                                      C
C                      INITIALISE                                      C
C                          |<---------------------                     C
C                     TAKE ONE STEP               |                    C
C           -------------->|                  yes |                    C
C          |       TIME < NEXT OUTPUT TIME?  -----                     C
C          |               | no                                        C
C          |      INTERPOLATE AT OUTPUT TIME                           C
C          |               |                  yes                      C
C          |      TIME >= FINAL OUTPUT TIME? ---------> STOP           C
C          |               | no                                        C
C           ---------------                                            C
C                                                                      C
C       Where TIME = time at end of step.                              C
C                                                                      C
C ARGUMENTS:    NONE                                                   C
C ----------                                                           C
C                                                                      C
C INPUT/OUTPUT:                                                        C
C -------------                                                        C
C                                                                      C
C   INPUT:    IIN1  (=5) = Problem specification file                  C
C  OUTPUT:    IOUT1 (=6) = Error messages                              C
C             IOUT2      = Integrated values                           C
C                                                                      C
C  Problem specification file:                                         C
C  +++++++++++++++++++++++++++                                         C
C                                                                      C
C     NEQ, NALGB                  - No. of eqns, no. of algebraics     C
C     T, TEND                     - Start time, finish time            C
C     (YVAL(I), I=1,NEQ)          - Initial values of variables        C
C     (YDER(I), I=1,NEQ)          - Initial values of derivatives      C
```

```
C     TSTOP, LSTOP                 - Time beyond which eqns may not be  C
C                                    integrated and a flag to show      C
C                                    whether such a time has been set   C
C     LDMAIN                       - Whether domain checking is on/off  C
C     TOL                          - Relative tolerance for integration C
C     HMAX                         - Maximum step size                  C
C     (INFORM(I), I=1,5)           - Information array for DC06AD       C
C     NRPAR, (RPAR(I), I=1,NRPAR) - No. of optional real parameters     C
C                                    and values                         C
C     NIPAR, (IPAR(I), I=1,NIPAR) - No. of optional integer parameters C
C                                    and values                         C
C     LLP,LLP1,IOUT2               - Output units:                      C
C                                   LLP  = Standard output unit: DC06AD C
C                                   LLP1 = Optional output unit: DC06AD C
C                                   IOUT2= Output values file           C
C     NPR                          - Number of print intervals          C
C     IDEBUG                       - Level of monitor/debug output      C
C                                                                       C
C  Output values file:                                                  C
C  ++++++++++++++++++                                                   C
C                                                                       C
C    Time1 YVAL(1)   AYVAL(1)   YDER(1)   AYDER(1)                      C
C    ..... ......... .......... ......... ..........                    C
C    Time1 YVAL(NEQ) AYVAL(NEQ) YDER(NEQ) AYDER(NEQ)                    C
C    Time2 YVAL(1)   AYVAL(1)   YDER(1)   AYDER(1)                      C
C    ..... ......... .......... ......... ..........                    C
C    Time3 YVAL(NEQ) AYVAL(NEQ) YDER(NEQ) AYDER(NEQ)                    C
C    ..... ......... .......... ......... ..........                    C
C    ..... ......... .......... ......... ..........                    C
C                                                                       C
C AYVAL(i) = Analytic value of YVAL(i) where it exists and 0 otherwise C
C AYDER(i) = Analytic value of YDER(i) where it exists and 0 otherwise C
C                                                                       C
C EXTERNAL FUNCTIONS:  FCN = Derivative function                        C
C -------------------  JAC = User   Jacobian evaluation function        C
C                      KAC = User K-Jacobian evaluation function        C
C                      JPT = User   Jacobian sparsity pattern function C
C                      KPT = User K-Jacobian sparsity pattern function C
C COMMON:      NONE                                                     C
C -------                                                               C
C CALLED BY:   This is the MAIN routine.                                C
C ----------                                                            C
C CALLS:       DC06AD                (see DC06AD for its calling tree) C
C ------                                                                C
C                                                                       C
C ERROR RETURNS:  IDID has the value set by DC06AD. The user should    C
C --------------  therefore consult the comments in DC06AD for further C
C                 information on the possible error returns.           C
C                                                                       C
C LOCAL VARIABLES:  All variables                                       C
C ----------------                                                      C
C                                                                       C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      IMPLICIT INTEGER (I-N)
C
C Note this is important
C
      LOGICAL  FCN,JAC,KAC,JPT,KPT
      EXTERNAL FCN,JAC,KAC,JPT,KPT
C
      LOGICAL LSTOP, LDMAIN
C
      PARAMETER(IIN1=5,IOUT1=6)
C
      PARAMETER(LRW=10000,LIW=10000,MAXNEQ=50,MAXPAR=50)
C
```

```
      DIMENSION RWORK(LRW),IWORK(LIW), RPAR(MAXPAR), IPAR(MAXPAR)
C
      DIMENSION  YVAL(MAXNEQ), YDER(MAXNEQ), INFORM(5)
      DIMENSION AYVAL(MAXNEQ),AYDER(MAXNEQ)
C
C **************************
C *     INITIALISATION     *
C **************************
C
      MODE=1
      NEW=0
      READ(IIN1,*) NEQ, NALGB
      READ(IIN1,*) T, TEND
      READ(IIN1,*) (YVAL(I), I=1,NEQ)
      READ(IIN1,*) (YDER(I), I=1,NEQ)
      READ(IIN1,*) TSTOP, LSTOP
      READ(IIN1,*) LDMAIN
      READ(IIN1,*) TOL
      READ(IIN1,*) HMAX
      READ(IIN1,*) (INFORM(I), I=1,5)
      READ(IIN1,*) NRPAR, (RPAR(I), I=1,NRPAR)
      READ(IIN1,*) NIPAR, (IPAR(I), I=1,NIPAR)
      READ(IIN1,*) LLP,LLP1,IOUT2
      READ(IIN1,*) NPR
      READ(IIN1,*) IDEBUG
C
      HMAXX = TOL*(TEND-T)
C
      TINT = (TEND-T)/NPR
      TNXT = T
C
C ***********************
C *     TAKE A STEP     *
C ***********************
C
  500 CALL DC06AD(MODE,NEW,JAC,KAC,JPT,KPT,FCN,RPAR,IPAR,NEQ,
     1            NALGB,T,TEND,TSTOP,LSTOP,LDMAIN,YVAL,YDER,TOL,
     2            HMAXX,INFORM,RVALS,IVALS,RWORK,LRW,IWORK,LIW,
     3            LLP,LLP1,IDEBUG,IDID)
      IF(IDID.LE.-100) THEN
        WRITE(IOUT1,'(\,A,I5,\)')
     1  '***** TEST: Stopping because DC06AD returned IDID = ',IDID
        STOP
      ENDIF
      NEW=2
      HMAXX=HMAX
C
C **************************************************************
C *     STEP COMPLETED - TEST FOR OUTPUT AND/OR END OF RUN     *
C **************************************************************
C
C Interpolate at an active output point
C
  600 IF (T.GE.TNXT) THEN
        MODE=3
        CALL DC06AD(MODE,NEW,JAC,KAC,JPT,KPT,FCN,RPAR,IPAR,NEQ,
     1              NALGB,TNXT,TEND,TSTOP,LSTOP,LDMAIN,YVAL,YDER,TOL,
     2              HMAX,INFORM,RVALS,IVALS,RWORK,LRW,IWORK,LIW,
     3              LLP,LLP1,IDEBUG,IDID)
        IF(IDID.LE.-100) THEN
        WRITE(IOUT1,'(\,A,I5,\)')
     1  '***** TEST: Stopping because DC06AD returned IDID = ',IDID
          STOP
        ENDIF
C
        CALL ANL(TNXT,AYVAL,AYDER,RPAR,IPAR,FVAL)
C
C Write out current values and set next output time
```

```
C
  700   WRITE(IOUT2,'(5E12.4)')
     1  (TNXT,YVAL(I),AYVAL(I),YDER(I),AYDER(I), I=1,NEQ)
        TNXT = TNXT + TINT
      ENDIF
C
C Is the next active output point within the last completed step
C
      IF((TNXT.LT.T).AND.(TNXT.LE.TEND)) GOTO 600
C
C If still in the range of integration then take another step else stop
C
      IF(T.LT.TEND) THEN
        MODE=1
        GOTO 500
      ENDIF
C
  800 CONTINUE
      STOP
      END
```

The example below is of a derivative function as discussed in section 2.4

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                       C
C  DESCRIPTION: Example derivative function.                            C
C  ============                                                         C
C                                                                       C
C INPUT:                                                                C
C                                                                       C
C       IDM = 0 Function evaluation + Check variables are in domain    C
C             1 Check variables are in domain only                      C
C         T = Time                                                      C
C   YVAL(*) = Variable values                                           C
C   YDER(*) = Derivative values                                         C
C   RPAR(*) = User's DOUBLE PRECISION parameters                        C
C   IPAR(*) = User's INTEGER parameters                                 C
C                                                                       C
C OUTPUT:                                                               C
C                                                                       C
C   FVAL(*) = Values of the derivative function                         C
C                                                                       C
C  If the equations are IMPLICIT then:                                  C
C                                                                       C
C  FVAL(I) = F(i)    see section 1.1                                    C
C                                                                       C
C  If the equations are EXPLICIT then:                                  C
C                                                                       C
C  FVAL(I) = G(i)    see section 1.1.1                                  C
C                                                                       C
C  In this case: F(i) = Y'(i) - G(i)                                    C
C                     = Y'(i) - FVAL(I)                                 C
C                                                                       C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
      LOGICAL FUNCTION FCN(IDMN,T,YVAL,YDER,RPAR,IPAR,FVAL)
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      IMPLICIT INTEGER (I-N)
C
      DIMENSION RPAR(*), IPAR(*), YVAL(*), YDER(*) ,FVAL(*)
C
C Check supplied variables are within the domain
C
      FCN = .TRUE.
      DO 10 I=1,2
        IF(YVAL(I).LT.0.0) FCN=.FALSE.
```

```
   10 CONTINUE
C
      IF((IDMN.EQ.0).AND.FCN) THEN
C
C Derivative function evaluation
C
        FVAL(1) = RPAR(1)*YVAL(1)+RPAR(2)*YVAL(2)+COS(RPAR(3)*T)
        FVAL(2) =         YVAL(1)+        YVAL(2)
C
      ENDIF
C
      RETURN
      END
```