1 SUMMARY

DC05A/AD is a package for solution of the system of ordinary differential equations (ODE) or differential algebraic equations (DAE)

$$\mathbf{F}_1(X,\mathbf{U}_1,\mathbf{U}_2)=\mathbf{0}\ , \tag{1a}$$
$$\mathbf{F}_2(X,\mathbf{U}_1,\mathbf{U}_2,\mathbf{U}'_2)=\mathbf{0}\ , \tag{1b}$$

where $\mathbf{F}_1$, $\mathbf{U}_1$ are vectors of length NALG, $\mathbf{F}_2$, $\mathbf{U}_2$ are vectors of length (NEQ-NALG), and $\mathbf{U}'_2=d\mathbf{U}_2/dX$. The solution is to start from given initial values $\mathbf{Y}$ of the $\mathbf{U}$ at $X=T$. It is numerically similar to DC03A/AD; but it presents a different user interface, possibly less convenient for ordinary use, intended for special-purpose packages. Function evaluation and linear algebra is done in the calling program by using ‘reverse communication’. Also, it can solve a wider range of problems, including implicit differential equations.

Components $\mathbf{U}_1$ of $\mathbf{U}$ are purely algebraic, their derivatives $\mathbf{U}'_1$ being absent from equations (1). The system must be of index no higher than 1, i.e. it must be non-singular in the sense that (1a) can in principle be solved for the $\mathbf{U}_1$ and (1b) for $\mathbf{U}'_2$. DC05 cannot solve DAE systems of index 2, which present added difficulties not present in those of index 1.

The form (1) includes, as the special (and common) case NALG=0, the pure ordinary differential equation (ODE) system (of index 0)

$$\mathbf{F}(X,\mathbf{U},\mathbf{U}')=\mathbf{0}\ , \tag{2}$$

which will often in practice take the simple explicit form

$$\mathbf{U}'-\mathbf{G}(X,\mathbf{U})=\mathbf{0}\ \cdot \tag{3}$$

Solution of DAE or stiff ODE problems is numerically demanding; if the single-precision version DC05A is used on 32-bit computers, e.g. IEEE single, it cannot be expected to be as reliable as on 64-bit computers, e.g. CRAY, or as the double-precision version DC05AD with IEEE.

ATTRIBUTES — **Version:** 1.0.0. **Types:** DC05A, DC05AD. **Calls:** None. **Original date:** July 1991. **Origin:** A.R.Curtis, Harwell.

2 HOW TO USE THE PACKAGE

2.1 Calling sequence and argument list

The single precision version

```
      CALL DC05A(T,Y,YDER,NEQ,TOL,INFO,IDID,FVAL,YTST,YLIM,
     *           CORR,WVEC,YNOR,MEQ,KD2,RW)
```

The double precision version

```
      CALL DC05AD(T,Y,YDER,NEQ,TOL,INFO,IDID,FVAL,YTST,YLIM,
     *           CORR,WVEC,YNOR,MEQ,KD2,RW)
```

T is a REAL (DOUBLE PRECISION for the D version) value of the independent variable X; supply initial value on first entry, DC05 returns output value.

Y is a REAL (DOUBLE PRECISION for the D version) array of length MEQ containing the components of the solution vector $\mathbf{U}$ at $X=T$. Supply initial values on first entry, DC05 returns output values on completion of a step. On return during a step it gives the values of $\mathbf{U}$ as given by equations (4) (in section 4).

`YDER` is a `REAL` (`DOUBLE PRECISION` for the D version) array of length `MEQ` containing the components of $\mathbf{U}' = d\mathbf{U}/dX$ at $X = T$. Supply initial guesses on first entry if equations are implicit and non-linear in $\mathbf{U}'$, or zero for components occurring linearly. `DC05` returns the output components of $\mathbf{U}'$ at $X = T$ on completion of a step. On return during a step it gives the values of $\mathbf{U}'$ as given by equations (4) (in section 4).

`NEQ` is an `INTEGER` set to the number of first order differential equations (plus algebraic equations, in a differential algebraic system) to be solved.

`TOL` is a `REAL` (`DOUBLE PRECISION` for the D version) value representing the accuracy required in the solution (changed by `DC05` only if out of range).

`INFO` is an `INTEGER` array of length 40 used to communicate how `DC05` should carry out its task (see section 2.2), and what it uses for workspace.

`IDID` is an `INTEGER` value reporting on return what the code did and what it now requires; it is used by `DC05` to decide what to do next (see also section 2.3.2). On return to the calling program, `IDID` may take the following values

≤ 0	Error return (see section 2.7).
1	New Jacobian matrices required (see section 2.4.1)
2	Decomposition of new Newton matrix required (see section 2.4.2)
3	Function evaluation and Newton correction (see section 2.4.3)
4	Apply Newton matrix (see section 2.4.4)
5	Step completed; test for output and/or end of run

`FVAL` is a `REAL` (`DOUBLE PRECISION` in the D version) array of length `MEQ`. The user must supply the vector $\mathbf{F}$ (or $\delta\mathbf{C}$, if the Newton equations (6) (section 4) are being solved), when `IDID=1, 2` or `3`.

`YTST` is a `REAL` (`DOUBLE PRECISION` in the D version) array of length `MEQ` used to hold comparison values against which error estimates on the components of `Y` are assessed. By default, it is initialised by `DC05,` and then updated at the end of each integration step in readiness for the next step. The user may (but is not advised) to supply comparison values and prevent control of them by `DC05.`

`YLIM` is a `REAL` (`DOUBLE PRECISION` in the D version) array of length `MEQ`, in which `DC05` maintains a record of the maximum absolute value attained by each component of `Y`, multiplied by the number `UMFAC`, for use in updating `YTST`, and in which the user may give information at the start of an integration about the expected sizes of the components of $\mathbf{U}$.

`CORR` is a `REAL` (`DOUBLE PRECISION` in the D version) array of length `MEQ` in which `DC05` returns the vector $\mathbf{C}$; it is also used as workspace.

`WVEC` is a `REAL` (`DOUBLE PRECISION` in the D version) array of length `MEQ` used as workspace by `DC05.` It may also be used by the calling program during Jacobian evaluation.

`YNOR` is a `REAL` (`DOUBLE PRECISION` in the D version) array of dimensions (`MEQ,KD2`), used to hold coefficients of approximating polynomials to the solution; also, `DC05` supplies the vectors $\mathbf{A}$ and $\mathbf{B}$ in its first two columns for use in computing $\mathbf{U}$, $\mathbf{U}'$ for evaluation of $\mathbf{F}$.

`MEQ` is an `INTEGER` specifying the dimensions of arrays; it must be at least equal to `NEQ` in value.

`KD2` is an `INTEGER` specifying the second dimension of array `YNOR`. Its value is normally 7, and `DC05` cannot use integration formulas of order higher than `KMAX = (KD2 – 2)`; `KD2` must be at least 3, and for efficiency preferably at least 5.

`RW` is a `REAL` (or `DOUBLE PRECISION`) work array of length at least 65 which provides the code with storage space. At the start of a run, `RW(1)` must be set to a trial initial step size (see section 2.2). The scalars h, α, β are returned to the calling program in `RW(11)`, `RW(12)`, `RW(13)` for use in equations (4) (section 4).

2.2 Input on the first call

The first call is defined to be the start of each new problem.

Include code similar to the following in the calling program, inserting problem-dependent sections where indicated (see section 2.4 for more details). We show the calling sequence for `DC05A/AD` for definiteness:

```
C      INITIALISE FOR NEW PROBLEM

      (insertion A: initialise as described in section 2.2)

C      INTEGRATION STEP, AND RE-ENTRY WITHIN STEP

  100 CALL DC05AD(T, Y, YDER, NEQ, TOL, INFO, IDID, FVAL,
     1 YTST, YLIM, CORR, WVEC, YNOR, MEQ, KD2, RW)

C      DECIDE WHAT TO DO NEXT

      IF (IDID.GT.0) GO TO (110,120,130,140,150), IDID

C      NEGATIVE VALUE OF IDID INDICATES ERROR

      (insertion B: handle error (see section 2.3.3); to resume - )
      GO TO 100

C      NEW JACOBIAN MATRICES REQUIRED

  110 (insertion C: compute J, K at X = T, U = A, U' = B/h
                                             - see section 2.4.1)

C      DECOMPOSITION OF NEW NEWTON MATRIX REQUIRED

  120 (insertion D: compute and factorise W, saving J, K
                                             - see section 2.4.2)

C      FUNCTION EVALUATION AND NEWTON CORRECTION

  130 (insertion E: compute F at X = T, U, U' given by (4),
                               and set in FVAL - see section 2.4.3)

      IF (INFO(21).EQ.0) GO TO 100
C      APPLY NEWTON MATRIX
      (replace FVAL by DELTAC from (6) - see section 2.4.3)
      GO TO 100

C      STEP COMPLETED - TEST FOR OUTPUT AND/OR END OF RUN

  140 (insertion F: process any end of step output required;
                    test for output within step - see section 2.4.4
                    - and for end of range of integration)
  142 IF (no further output needed) GO TO 148
      T = next output point in step     (see section 2.4.4)
      INFO(1) = 2
      GO TO 100
  148 IF (end of range reached) GO TO 200
      GO TO 100

C      INTRA-STEP OUTPUT INTERPOLATED

  150 (insertion G: process output within step - see section 2.4.4)
      GO TO 142

C      END OF CALLING SEQUENCE

  200 (continue with further code)
```

The arguments should be set as follows:

`T` Set it to the initial value of *X*.

`Y` Set this array to the initial values of **U**. Dimension it as `MEQ` (see below).

`YDER` Set this array to initial guesses of **U**'=*d***U**/*dX*; zero is a good enough initial guess for any component which occurs linearly in equations (1). Initial guesses for the components $\mathbf{U}'_1$ are not required. Dimension it `MEQ`.

`NEQ` Set it to the number of equations (must be greater than 0).

`TOL` Specify the relative accuracy required in the solution components (actually, relative to comparison values `YTST(I)` whose default calculation is described below) by setting `TOL` to a suitable positive value. If the value specified is larger than 10^{-2} or smaller than a number `TOLMIN` (about 10^{-10} on most computers, but see section 3), it will be reset to the corresponding limit and the task will be suspended, the code returning to the calling program with `IDID=-2`. The user may alter `TOL` before re-entry, but in general the change should be accepted. `TOL` is used by the code in a local error test which requires that the error estimate on `Y(I)` satisfies

$$|\text{ local error }| \ \leq\ \texttt{TOL*YTST(I)}$$

where (at point `T`) `DC05A/AD` makes

$$\texttt{YTST(I)} \approx \max\,(\texttt{USIZ(I), UMFAC*(max(|Y(I,X)|, X} \leq \texttt{T))} + \texttt{URFAC*|Y(I,T)|}$$

for each component. This is approximate because of precautions taken to avoid zero values. `USIZ(I)` is taken as zero if it is not supplied by the user. For a component whose initial value and derivative are both zero, `YTST(I)` is kept very large until `|Y(I)|` grows to at least `USMALL`, when testing is introduced gradually on the component.

Default values of parameters are

$$\texttt{UMFAC}=10^{-10},\ \texttt{URFAC}=1,\ \texttt{USMALL}=10^{-30}.$$

These may be changed (see `INFO(5)` below), or even `YTST` removed from control by `DC05A/AD` and its values set by the user; although neither course is recommended.

`INFO` Use the `INFO` array to give more details about how the problem should be solved. This array should be of length 40, since `DC05A/AD` also uses it as workspace, but only the first 5 entries need to be set on entry.

Answer the following questions and set the elements of `INFO` accordingly. The simplest case corresponds to setting all the first 5 entries to 0.

`INFO(1)` Enables the code to initialise itself. You must set it to 0 to indicate the start of each new problem (this includes any change in the functions **F** or in any parameters used in evaluating them). `DC05A/AD` will set `INFO(1)` to 1 before return to indicate continuation of an old problem.

Is this the first call for this problem?

YES – set `INFO(1)=0`.
NO – not applicable here; on later entries, leave `INFO(1)` unchanged except to get extra output – see section 2.4.4.

`INFO(2)` It is occasionally necessary to specify a maximum step-size `HMAX` to be used in the integration, to prevent the code from ‘stepping over’ narrow features in the functions **F** without ‘seeing’ them. On most problems this is not necessary. Where it is, `INFO(2)` must be set equal to 1 to tell `DC05A/AD` to regard `RW(1)` as `HMAX`. However, it does this automatically on the first step only of a new problem, and a non-zero value of `INFO(2)` is not allowed on this call. At this point, therefore,

– set `INFO(2)=0`.

`INFO(3)` To give output efficiently at many *X* values, the code takes steps as large as possible, using an interpolation routine to give results where required within the last completed step. Thus it will normally integrate past the end of the range. Sometimes it may not be possible to integrate beyond a value `TSTOP` (because of an essential singularity there, such as a failure of definition of the functions **F** beyond that point). The code must be told this. There are better ways of dealing with ordinary discontinuities – see section 2.6.

Can the integration proceed without any upper limit on the independent variable?

YES – set `INFO(3)=0`.
NO – set `INFO(3)=1` and define `TSTOP` by setting `RW(2)=TSTOP`.

`INFO(4)` For a DAE problem, or an ODE one in which the derivatives $\mathbf{U}'_2$ occur implicitly and non-linearly, `DC05A/AD` must be informed by setting `INFO(4)` to a positive value. For a DAE problem, set it to the number `NALG` of algebraic components (always the first `NALG` components) of **U**, **F**. For a purely differential problem where the derivatives occur non-linearly, set it to (`NEQ+1`). If, on the other hand, your problem is neither stiff nor DAE, you may be able to omit linear algebra code in insertions C and D, and the second part of E. However, `DC05A/AD` must then be told not to use Newton iterations, in case it tries to.

Is your problem of the form of equation (1), rather than equation (2)?

YES – set `INFO(4)=NALG`.
NO – Answer the following question –

Do the derivatives **U'** occur nonlinearly?

YES – set `INFO(4)=NEQ+1`.
NO – Answer the following question –

Does the program include code for linear algebra?

YES – set `INFO(4)=0`.
NO – set `INFO(4)=-1`.

`INFO(5)` Allows the user (IF CERTAIN THIS IS REQUIRED) either: (a) to specify the expected sizes `USIZ(I)` of the components of the solution, and to supply own values for the parameters `URFAC`, `UMFAC` and `USMALL` (see under `TOL` above) used in accuracy testing, or (b) to switch off all `DC05A/AD` control of `YTST`, using instead values supplied by the user. There is no need to do either, even if some components of **U** are initially zero – the code copes well with this, and departing from the default may sacrifice much of the reliability built into it.

Is the default calculation of `YTST(I)` as given above required?

YES – set `INFO(5)=0`.
NO – set `INFO(5)=1` for (a) and set `RW(3)=URFAC`, `RW(4)=UMFAC`, `RW(5)=USMALL` and `YLIM(I)=USIZ(I), I=1` to `NEQ`.

`IDID` It is not necessary to set the value of `IDID` before first entry to `DC05A/AD,` and it should not be changed thereafter. It is used both by the calling program and by `DC05A/AD` to decide what to do next.

`FVAL` Dimension this array of size `MEQ`; it need not be set on first entry.

`YTST` Dimension this array of size `MEQ`; it need not be set on first entry, unless `INFO(5)=2` is set, when the user's error test comparison values must be set in it.

`YLIM` Dimension this array of size `MEQ`; it need not be set on first entry, unless `INFO(5)=1` is set, when the user's `USIZ` values must be set in it.

`CORR` Dimension this array of size `MEQ`; it need not be set on first entry.

`WVEC` Dimension this array of size `MEQ`; it need not be set on first entry.

`YNOR` Dimension this array `(MEQ,KD2)`; it need not be set on first entry.

`MEQ` Set this to the dimension of the above arrays; normally, `MEQ=NEQ` is required, but it may be larger if desired.

`KD2` Set this to 7 (normally; it may be reduced, but not below 3, to limit the maximum order of integration formula to `(KD2-2)`, if desired).

`RW` Dimension this array of length 65 in the calling program. If `INFO(3)` or `INFO(5)` is set to 1, the corresponding entries of RW must be set as prescribed. On first entry RW(1) must be set to a trial initial step size HTRY (which DC05A/AD will not exceed), as a guide to the scale of the independent variable X. A reasonable value might be TOL times the distance to the first output point, but even smaller values may give better results – DC05A/AD can increase its step very rapidly, if accuracy tests allow, at the start of a run.

2.3 Action on return from DC05A/AD

The code returns to the calling program with IDID set to indicate what it has done and what it requires next. Positive values of IDID indicate normal progress, negative values that the integration process was interrupted because of some major or minor error condition.

2.3.1 Output values

On return, parameters have values:

T the current value of X is the output value of T; the solution has reached this point if IDID=4 (completed step), but for smaller values of IDID failure of a subsequent error test may cause repetition of the step with smaller step size, and therefore smaller value of T.

Y if IDID=4 or 5, contains the current approximation to the solution **U** at $X=T$, obtained from interpolation polynomials. For other values of IDID, may be used to hold **U** computed from equations (4) (section 4).

YDER for IDID=4 or 5, holds the current approximation to the derivatives $\mathbf{U}'=d\mathbf{U}/dX$ at $X=T$, obtained by differentiating the interpolation formulas used for Y. For other values of IDID, may be used to hold $\mathbf{U}'$ computed from equations (4) (section 4).

FVAL holds no significant information.

YLIM holds the maximum absolute values taken so far by the Y array, multiplied by the constant UMFAC which occurs in the YTST calculation.

YTST holds the comparison values used in error and convergence testing on the components of Y.

CORR when IDID=1, 2 or 3, holds the current correction vector **C** for use in (4); this is always zero when IDID=1. For IDID=4 or 5, holds no significant values.

RW contains information, most of which is often of no interest to the user, but is necessary for subsequent calls and must be left undisturbed. Copies of some of this information, with mnemonic names, are held in COMMON block DC05Y/YD (see section 3). On return with IDID=1, 2 or 3, RW holds the scalars h, α, β needed for (4) (section 4).

The following values, whose mnemonic names are also shown, may be of interest on return:

RW(11) contains H $=h$, the length of the current or last completed step.

RW(12) contains ALFA $=\alpha$, used in constructing the vector **U**.

RW(13) contains BETA $=\beta$, used in constructing the vector $\mathbf{U}'$.

RW(14) contains RELERR, the ratio of error estimate on the last step to error tolerance.

RW(15) contains X0, the value of X at the start of the last completed step.

RW(16) contains XX, the current value of X, at the end of the last completed step.

RW(17) contains HNEW, the proposed length of the next step.

TOL remains unchanged unless IDID=2, when TOL has been altered to bring it within range.

YNOR contains coefficients of interpolation polynomials for the solution vector **U**. In particular, its first two columns hold the vectors **A**, **B** needed for (4) (section 4). Must not be changed.

INFO contains information, most of which is often of no interest to the user, but is necessary for subsequent calls and must be left undisturbed. Copies of some of this information, with mnemonic names, are held in `COMMON` block `DC05Y/YD` (see section 3).

The following values, whose mnemonic names are also shown, are or may be of interest on return:

`INFO(15)` contains `KDEQ`, the order of integration formula used on the last step.

`INFO(21)` contains `LNEWT`, which has the value 1 if solution of Newton equations is required after function evaluation, 0 if not.

`IDID` reports what the code did and what it now wants (see the following sections).

2.3.2 Task continuing normally reported by positive values of IDID

`IDID=1` The code requires evaluation of new Jacobian matrices J and K at the trial point $X = T$, $\mathbf{U} = \mathbf{A}$, $\mathbf{U}' = \mathbf{C}$ (either at the start of a DAE problem, or because convergence with the old matrix, or without Newton iterations, has been unsuccessful). See section 2.4.1. Always followed by Newton matrix evaluation and decomposition.

`IDID=2` The code requires evaluation and factorisation of a new Newton iteration matrix W using the current J, K (because stepsize or order has changed, or J, K have been updated). See section 2.4.2. Always followed by function evaluation.

`IDID=3` The code requires evaluation of the functions $\mathbf{F}$ at the trial point as above, in array `FVAL`. See section 2.4.3. Then, unless `INFO(21)=0`, the Newton equations must be solved, and the function vector in `FVAL` replaced by the solution of Newton equations (6). See sections 4 and 2.4.3. This second stage never occurs unless the Newton matrix has been factorised in preparation. Always followed by recall of `DC05A/AD`.

`IDID=4` A step was successfully completed and output values at end of step are returned in `T`, `Y`, `YDER`. Interpolated output at intermediate points within the step can be obtained by changing `T`, setting `INFO(1)=2` and re-entering; `DC05A/AD` then will return with `IDID=5` (see below and section 2.4.4). Followed by recall of `DC05A/AD` for intra-step output or to take a new step, or by branching to further code if the calling program finds that the end of the integration range has been reached.

`IDID=5` Interpolated output at an intermediate point in the step has been returned. `Y` and `YDER` will correspond to the altered `T`, and `INFO(1)` will have been reset to 1. Thus a further re-entry will take a new step, unless the user again sets `INFO(1)=2`. Always followed by a test for further output requirement, or for end of range.

2.3.3 Task interrupted, but resumable – reported by negative values of `IDID`.

If the user wishes to inform the code that the task should continue despite being interrupted, appropriate action should be taken and `INFO(1)` reset. If `INFO(1)`is not reset, the task will be terminated.

`IDID=-1` A large amount of work has been done (500 steps attempted). To enable the code to continue, set `INFO(1)=1` and call the code again. A further 500 steps will be allowed.

`IDID=-2` The error tolerance `TOL` has been changed to bring it within range. The user may change it but should be sure that the new value will work. To continue with the changed value, set `INFO(1)=1` and call the code again.

`IDID=-3` Not used.

`IDID=-4` The user asked for interpolated output (`INFO(1)=2`) at a point outside the last completed step, i.e. before `X0=RW(15)` or after `XX=RW(13)`. The values of `Y`, `YDER` returned are those obtained from the interpolation formula at the user's value of `T`. If the user believes that they may be seriously in error `INFO(1)=1` must be reset before calling `DC05A/AD` again, to make it take a new step; if instead further interpolated output is required, set `INFO(1)=2` and change `T`.

`IDID=-5` Ten Newton iterations at the start of a DAE problem have failed to achieve $\mathbf{F} = \mathbf{0}$ by converging the initial values of $\mathbf{U}$, $\mathbf{U}'$. If the values reached are an improvement, so that a further 10 iterations may be successful, set `INFO(1)=0` and call the code again.

`IDID=-6` `DC05A/AD` had repeated convergence test failures on the last attempted step. Inaccuracy of the Jacobian matrix may be the problem, or specifying no Jacobian (`INFO(4)=-1`) on a problem which really needs one. The calculation may be continued, perhaps at greatly reduced efficiency, by setting `INFO(1)=0` (which does a restart) and calling the code again.

`IDID=-7` `DC05A/AD` had repeated accuracy test failures on the last attempted step. There may be unrecognised discontinuities in the equations, or the **F** functions may be providing noisy derivative values (e.g. by failure to declare some variable `DOUBLE PRECISION` in the D version). The calculation may be continued by setting `INFO(1)=0` (which does a restart) and calling the code again.

`IDID=-8,...,-32` Not used.

2.3.4 Task terminated

`IDID=-33` The code has met trouble from which it cannot recover. A message is printed explaining the trouble, and control is returned to the calling program. For example, invalid input data has been detected. An attempt to continue will cause the job to be terminated by a Fortran `STOP` instruction.

2.4 Preparing the code insertions

Insertion A has been described in section 2.2, and insertion B in section 2.3.3. The remaining insertions are now described, in the sections below, in the order in which they appear in section 2.2. The code in these insertions should not change any quantity not specifically permitted.

2.4.1 Jacobian calculation (insertion C)

`IDID=1` If any special action (for example determining the sparsity pattern) is required on the first Jacobian evaluation, code must be written to detect the first return with `IDID=1` and take the necessary action. Storage must also be provided as required for the matrices J, K and for sparsity pattern description. Evaluation by finite differencing is recommended, except perhaps for linear problems where the Jacobians are needed in any case for evaluation of **F**. However, the calling program is responsible for choosing good increments for differencing. For simple problems of the form (3), K is trivial and does not need explicit evaluation. If sparsity allows differencing for more than one column at a time, the user is responsible for finding and implementing a suitable column grouping.

Having computed J, K their values must be retained for later use, as well as supplied immediately to the following insertion. This insertion may use arrays `WVEC`, `CORR`, `Y` and `YDER` as workspace. (Space to be occupied later by the Newton matrix may also be used as workspace.)

2.4.2 Newton matrix generation and decomposition (insertion D)

`IDID=2` The values of h, α, β needed for W will be found in `RW(11)`, `RW(12)` and `RW(13)`, as for equations (4). The precise nature of the process referred to as factorisation or decomposition of W is up to the user; all that `DC05A/AD` needs is that W is stored in some form which enables the user to solve equations (5) in section 4 efficiently for many successive right-hand sides. The user is responsible for providing necessary storage for this form of W and for any indexing information needed.

If any special action (for example analysis of sparsity pattern to choose pivotal sequence) is required on the first decomposition of W, the user must write code to detect the first occasion and take the necessary action. Since this occasion will follow on from the first return with `IDID=1`, detection can be shared with the insertion above. This insertion may use arrays `Y`, `YDER`, `CORR`, `WVEC` as workspace, but must restore zero values in `CORR` if it uses it.

2.4.3 Function or correction calculation (insertion E)

`IDID=3` This is the insertion which defines the problem; in it the functions **F** must be evaluated at the current trial point $X = T$, with **U**, **U'** given by (4), and the values put in `FVAL`. `CORR`, `WVEC` or `YNOR` must not be altered. In practice, it is probably best to call a subroutine to evaluate **F**, since the same code will be needed in computing Jacobians (by finite differences) when IDID=1.

Unless INFO(21)=0, the user must then solve the equations (5) for δ**C**, and return the solution in FVAL. T, CORR, WVEC or YNOR, must not be changed, but Y, YDER may be used as workspace. The coding needed to solve (5) will, of course, depend on the chosen decomposition of *W*; if an approximate method of solution is chosen, it must be accurate enough not to upset the convergence test in DC05A/AD, which is to an accuracy of 0.1*TOL , relative to the components of YTST, so δ**C** should be found to rather better accuracy than this.

2.4.4 End of integration step, output handling (insertions F, G)

IDID=4,5 If output is required only at the end of each integration step, it need only be processed when IDID=4 and tested for the end of the integration range having been reached, returning to call DC05A/AD again for a further step if it has not. If however output is required at one or more intermediate points in the step just completed, set T to the next value of *X* at which it is wanted, change INFO(1) to 2, and re-enter. DC05A/AD will interpolate the values of **U** and **U'**, returning them in arrays Y, YDER, with IDID=5 and INFO(1) reset to 1. This may be done as often as required.

There is an alternative way of obtaining interpolated output, which does not test that the specified value $X = T$ is within the last completed step, and interrupt the task if not. Include, in insertion G, code of the form

```
T = next output point in step
FRAC = (T - RW(16)) / RW(11)
CALL DC05CD(T,Y,YNOR,YDER,FRAC,MEQ,KD2)
```

and on return from DC05CD the interpolated values will be found in Y, YDER. As many calls of this kind as required may be included.

It is safe to test whether any required output point has been passed, and to iterate if necessary to find it, if it is determined by a function of *X*,**U**, **U'**, since the interpolated values in Y, YDER are continuous functions of T

See section 2.6 for information about more advanced techniques, including the handling of discontinuities in the definition of **F**. Arrays FVAL, CORR, WVEC may be used as workspace in this insertion. There is no need to restore end-of-step Y and YDER values before re-entering for a new step. However, if a new problem is started by re-entering with INFO(1)=0, it will of course start from the current contents of Y, YDER.

2.5 How to continue the integration by steps after the first

The code is organised so that steps after the first involve little (if any) additional effort on the part of the user. On return with IDID=4, if output is required at points other than the end of the step, set T to the required value and change INFO(1) to 2 before re-entering DC05A/AD to get the solution at the output point; there is no need to restore end-of-step values before re-entering for a new step. (If however the user interpolates and then does a restart by setting INFO(1)=0, the restart will be from the interpolated point.) To carry out another integration step, re-enter DC05A/AD without altering any quantity not specifically permitted below.

In particular, do not change NEQ, YNOR, MEQ or KD2 (or T, Y, YDER in case of a restart), or the differential equations as specified in insertion E. In principle, a change in any numerical parameter counts as a change to the differential equations, but see section 2.6. Do not change INFO or RW except as described in the following paragraphs. Any such change constitutes a new problem, and should be treated as such by restarting with a new first call, even if in fact the same problem is being continued after a discontinuity. (However, in practice DC05 can often cope quite well with ordinary discontinuities in **F**, at little greater cost than if a restart is forced).

The values USIZ(I), if used (INFO(2)=1), are referenced only on the first call, and so cannot effectively be changed; do not attempt to change URFAC, UMFAC, USMALL by altering RW(3), RW(4), or RW(5). However, TOL can be changed at any time; increasing it may make the equations easier to integrate, but decreasing it makes them harder to integrate and so should usually be avoided. The user may switch to his own control of YTST by setting INFO(5)=2;

but cannot thereafter switch back again.

The value of HMAX can be changed, if INFO(3)=1 is set and the required value put in RW(1). However, if TSTOP (INFO(4)=1) has been set, the code will refuse to integrate past the current TSTOP. Once it has been reached, RW(2) must be changed or INFO(4) set to 0 in order to continue. This may be done at any time, but whenever you set INFO(4)=1 TSTOP must be supplied in RW(2).

The parameter INFO(1) determines whether the call is the first of a new problem (including a restart after a discontinuity) or a continuation call. INFO(1) must be set to 0 to start a new problem; on return, INFO(1) will have the value 1, to indicate a continuation call, unless the task was interrupted. The user can therefore continue a problem without changing INFO(1), but must reset it to continue an interrupted task, or to start a new problem. As described above, it can be set to 2, when IDID=4 or 5, to ask for output at an intermediate point. DC05 will set it back to 1, to continue with another step.

2.5.1 To continue a task after a completed step:

If IDID=4 or 5 Call the code again to continue the integration, or first call it with INFO(1)=2 and T changed (see above) to get interpolated output within the step just completed.

2.6 Advanced output testing; handling discontinuities

The user may want output when some function of the solution **U**, **U'** and the independent variable *X* takes a certain value, e.g. when $P(X, \mathbf{U}, \mathbf{U}') = 0$ for a known function *P*. As an example, suppose output is required at $U_1 = X U_3$. Start integrating and, at each return with IDID=4, monitor P(T, Y, YDER) = Y(1) – T*Y(3) to see if it has changed sign; if not, call the code again. When *P* has changed sign in the most recent step, carry out an inverse interpolation to find a value *T* at which $P = 0$. This has to be done iteratively. When the point has been found to sufficient accuracy (for example to within TOL*H), the necessary output action can be taken. We discuss below how to do the inverse interpolation.

This technique may be used to handle ordinary discontinuities in the functions; by ordinary discontinuities we mean those where a smooth continuation does exist, but is not followed because of a change of definition or of some parameter value. A switching function $P(X, \mathbf{U}, \mathbf{U}')$ is constructed, such that the discontinuity occurs at P=0. Using the old **F** coding (including the smooth continuation past the switching point), locate the latter by the above technique. Then make the necessary discontinuous change, set INFO(1)=0 to force a re-start, and call the code again; the restart will occur from the point in the smooth solution at which the switching function becomes zero, which is exactly what is required. This is the most efficient means of dealing with this kind of discontinuity, but it is by no means necessary to use it in all cases. DC05 can in fact integrate through ordinary discontinuities or near-discontinuities without serious loss of efficiency in most cases.

To do the necessary inverse interpolation we can use Newton's method. Consider our example function, $P = U_1 - X U_3$. We can evaluate

$$\frac{dP}{dX} = \frac{dU_1}{dX} - U_3 - X \frac{dU_3}{dX} \ .$$

As applied to interpolated values within the current step, the values of dU_i / dX returned in YDER are exact; thus we can use them in the above expression, calculating (YDER(1) – Y(3) – T*YDER(3)). The Newton estimate for *X* where $P = 0$ is

```
      X = T - (P/(dP/dX)) evaluated at X=T
        = T - (Y(1)-T*Y(3))/(YDER(1)-Y(3)-T*YDER(3))
```

in our example. If this lies within the current step, given by the range RW(15) to RW(16), we set it as a new T, set INFO(1)=2, and re-enter to get interpolated values there, iterating until the change is within tolerance. If we find that the above *X* is outside the current step, we carry out a binary subdivision of either (RW(15), T) or (T, RW(16)), choosing the sub-interval in which *P* changes sign, until we find a point at which the Newton iterate is within the step, or until the subdivision is already fine enough to be within tolerance.

2.7 Error messages and other communication

Explanatory messages are put out before task termination, and also on failure to update TOUT. This is done on the Fortran unit specified by integer LP in COMMON block DC05Z/ZD, whose default value is 6; setting LP to zero switches off these messages. All the task termination messages finish with

```
     ***** DC05A TASK TERMINATED FOR ABOVE REASON
```

The messages which precede the above termination message are largely self-explanatory, but are listed here with brief comments:

```
***** RE-ENTRY WITH ILLEGALLY CHANGED IDID OR INFO
```

The user has attempted some illegal change.

```
***** NEGATIVE VALUE      (value) SPECIFIED FOR URFAC
***** NEGATIVE VALUE      (value) SPECIFIED FOR UMFAC
***** NEGATIVE VALUE      (value) SPECIFIED FOR USMALL
***** HMAX SPECIFIED LESS THAN ZERO     (value)
```

The user has entered the negative value which is printed.

```
***** ILLEGAL ENTRY WITH NEQ =   (value)
```

The user has specified a negative or zero number of equations.

```
***** RE-ENTRY AFTER RETURN WITH IDID =   (value)
       WITHOUT RESETTING INFO(1)
```

The user has failed to reset INFO(1) after task suspension.

```
***** ILLEGAL INPUT VALUE   (value) FOR INFO(nn)
```

The user's value for the designated component of INFO is illegal.

```
***** NO CONVERGENCE OF INITIAL VALUES IN 10 NEWTON ITERATIONS
```

Preliminary iterations have not converged (IDID = –5).

```
***** INITIAL STEP ESTIMATE      (value) IN RW(1) IS NOT POSITIVE
```

The user entered the negative or zero estimate which is printed.

```
***** TSTOP =      (value) IS LESS THAN T =      (value)
```

The user tried to set TSTOP less than the current T.

```
***** REQUESTED TOLERANCE      (value) IS NOT POSITIVE
```

The user entered the negative or zero value printed.

```
***** INTEGRATION REQUESTED BEYOND TSTOP =      (value)
```

The user tried to resume without resetting or cancelling TSTOP.

```
***** SPECIFIED ARRAY SIZE (value) IS LESS THAN NEQ = (value)
```

If MEQ is set smaller than NEQ.

Informative messages are also put out on a unit specified by LP1 (if greater than zero) in COMMON block DC05Z/ZD. Non-diagnostic messages are given on the following occasions: when DC05 has chosen a starting step size (necessarily not exceeding the user's HTRY); if the size chosen has to be further reduced on the first step because of accuracy tests; if a very small lower bound on step-size causes TOL relaxation; if a subsequent increase in step-size removes TOL relaxation. These are all self-explanatory. Diagnostic messages are given on the following occasions when DC05 experiences difficulty; they do not necessarily indicate an error condition, merely that the integration stepsize has had to be reduced. When the corrections on the first corrector iteration are so large that it is unlikely that the accuracy test will be passed; when convergence failure occurs in spite of new Jacobian matrices; and when accuracy test failure occurs so many times on the same step that DC05 reduces the order of integration formula. The default value of LP1 is zero.

2.8 Over-riding the defaults

The default error and convergence testing, with USIZ=0, has proved very reliable over a wide range of problems, and the user should think carefully before deciding to change it. It has built-in refinements to deal with cases where a component of **U** has an initial value of zero or passes through zero during integration; this is why the value given for YTST(I) in section 4 is only approximate. For a component whose initial value is zero, UMFAC is used with the initial stepsize times its initial rate of change; if the latter is also zero, testing on the component is suspended until it becomes non-zero, and is then brought in gradually, relative to a size which at first is at least USMALL.

3 GENERAL INFORMATION

Use of common: COMMON blocks DC05Y/YD, DC05Z/ZD, of which the latter is initialised in BLOCK DATA.

Workspace: As described.

Other subroutines: None

3.1 Subroutines and COMMON blocks

All subprograms in the double precision version have

```
IMPLICIT DOUBLE PRECISION (A-H,O-Z)
```

immediately after the heading. There is a block data subprogram headed

```
BLOCK DATA DC05BD
```

which sets values for variables in COMMON block DC05Z (see below). The main routine, which takes an integration step and optionally provides output within a completed step, is headed

```
      SUBROUTINE DC05AD(T,Y,YDER,NEQ,TOL,INFO,IDID,FVAL
     1,YTST,YLIM,CORR,WVEC,YNOR,MEQ,KD2,RW)
```

The routine headed

```
SUBROUTINE DC05CD(TIME,YVAL,ZNOR,YDER,XINT,MY,KY)
```

interpolates the solution to a point fractional distance XINT (which should be between –1 and 0) from the end of the last completed step, returning values in TIME and in arrays YVAL and YDER.

The subroutine headed

```
SUBROUTINE DC05ED(TIME,ZNOR,MY,KY)
```

resets TIME and the Nordsieck array, which it is passed in ZNOR, when it is necessary to repeat a step.

The subroutine headed

```
      SUBROUTINE DC05FD
```

limits the integration stepsize `H` to lie between `HMIN` (a very small number) and `HMAX`, if the user has specified the latter. It prints informative messages on unit `LP1` if that has been set to a positive number.

The subroutine headed

```
     SUBROUTINE DC05GD(L)
```

sets up coefficients for a Gear type integration formula of order L.

Finally, the subroutine headed

```
      SUBROUTINE DC05HD(JUMP,U,V,UUU,YVAL,YTST)
```

is responsible for the diagnostic messages on unit `LP1`, if this has been set to a positive value. It is unlikely that the user will need to concern himself with these.

The following `COMMON` blocks and related declarations are included in the routines of the package:

```
      COMMON /DC05ZD/ ZERO,ONE,TWO,TEN,PT1,PT5,ROUND,TOLMIN
     1,TOLMAX,VARMIN,VSMAL,VARMAX,EALFA,CTOL,CTOL1
     2,BIASD,BIASL,BIASU,CNVFAC,RMINIT,RMCONV
     3,RATMIN,HFINIT,HFRUN,RHOMAX,RHOFAC,RHOJAC,TSTINT
     4,KMAXX,NRETRY,LP,LP1

      COMMON /DC05YD/
     X HMAX0,TSTOP,URFAC,UMFAC,USMALL
     Y,SPARE6,A0HJAC,HFAC,HMIN,HMAX
     1,H,ALFA,BETA,RELERR,X0
     2,XX,HNEW,HMCONV,HOK,RATMAX,ACTOL,EBETA,HDOWN,HSAME,HUP
     3,A0H,A0,FKDP1,PD,QD,PL,QL,PU,QU,ECON
     4,ECON1,TSTFAC,ERNORM,VECK(7)
     5,CTEST,RHO,CRATE,CNORM,DNORM,ENORM,UNORM,HOLD,FRACT,SNORM
     6,CFAC(10)

      COMMON /DC05YD/ INFO1,INFO2,INFO3,INFO4,INFO5
     Y,NY1,KUPSWT,KDNSWT,NSTEP,NCG
     X,KDP1,LSTRT,NCORR,KNT500,KDEQ
     1,IUPH,ICORR,NSCONV,LJAC,LLUD
     2,LNEWT,NY,KMAX,KMAX1,IFAIL
     3,ICNT,ICNTM,IRMX,ITRATE,KC
     4,KNEW,KNTFL,LNEWTN,NALG,NALG1
     5,INFSAV(5)

      DIMENSION IWSAVE(40), RWSAVE(65)
      EQUIVALENCE (HMAX0,RWSAVE(1))
      EQUIVALENCE (INFO1,IWSAVE(1))
```

Block `DC05Y` contains workspace; its contents are saved in argument arrays `RW`, `INFO` on return to the calling program. Therefore, it is possible with care to nest calls to `DC05`, e.g. to solve an inner ODE problem as part of the function evaluation for an outer one.

Block `DC05Z` contains values built permanently into the package at compilation time by the `BLOCK DATA` subprogram, which contains the `DATA` statements and comments:

```
      DATA BIASD /0.5D0/, BIASL /0.5D0/, BIASU /0.2D0/
     1,CNVFAC/1.D-3/,RMINIT/1.D3/,EALFA/0.D0/,CTOL/0.3D0/,CTOL1/0.2D0/
     2,RMCONV/1.25D0/,RATMIN/1.1D0/,HFINIT/5.D-2/,HFRUN/5.D-1/
     3,RHOMAX/5.D-1/,RHOFAC/0.75D0/,RHOJAC/3.D-1/,TSTINT/1.D10/
     4,TOLMIN/1.D-10/,TOLMAX/1.D-2/
     5,ROUND/1.D-15/,VARMIN/1.D-70/,VSMAL/1.D-30/,VARMAX/1.D65/
      DATA KMAXX/5/, NRETRY/3/, LP/6/, LP1/0/
C   BIASx CONTROL SAFETY FACTORS FOR ORDERS 1 DOWN, SAME, 1 UP.
C   CNVFAC CONTROLS LEVEL AT WHICH CORRECTIONS REGARDED AS NOISE.
C   RMINIT IS INITIAL MAXIMUM STEPSIZE INCREASE RATIO, RATMIN IS
C    MINIMUM RATIO MAKING INCREASE WORTH WHILE; RMCONV IS MAXIMUM
C    RATIO ALLOWED AFTER REDUCTION FOR CONVERGENCE FAILURE.
C   EALFA ALLOWS SOME SMOOTHING OF ERROR NORM FROM STEP TO STEP;
C    VALUES FROM ZERO (NO SMOOTHING) TO 0.5 COULD BE TRIED.
C   CTOL, CTOL1 ARE USED TO PREDICT WHETHER NEW NEWTON MATRIX
C    WILL BE NEEDED AFTER CHANGE OF H OR ORDER.
C   HFINIT IS INITIAL STEPSIZE REDUCTION FACTOR ON CONVERGENCE
C    FAILURE, HFRUN IS FACTOR AFTER FIRST SUCCESSFUL STEP.
C   RHOFAC, RHOJAC ARE LEVELS OF TOLERANCE ON CONVERGENCE FACTOR,
C    RHOFAC IS MAX. ALLOWED, RHOJAC STIMULATES NEW JACOBIAN.
C   TSTINT CONTROLS GENTLE APPLICATION OF ACCURACY TESTING TO
C    COMPONENTS INITIALLY ZERO, AFTER THEY REACH VSMAL.
C   TOLMIN, TOLMAX ARE BOUNDS ON RELATIVE ACCURACY ACTOL.
C   KMAXX IS MAXIMUM ORDER ALLOWED, MUST BE IN [1,5].
C   LP IS NORMAL MESSAGE OUTPUT UNIT, LP1 FOR DIAGNOSTICS.
C
C   OTHER METHOD-DEFINING VALUES INITIALISED ON EACH NEW PROBLEM
C
C     HMIN = VARMIN
C     URFAC = ONE
C     UMFAC = TOLMIN
C
C   ABSOLUTE CONSTANTS
      DATA ZERO /0.D0/, ONE /1.D0/, TWO /2.D0/, TEN /10.D0/
     1, PT1 /.1D0/, PT5 /.5D0/
```

4 METHOD

The method used is especially efficient on stiff problems defined by equation (2) or the more general DAE problems of equation (1). The package defines the values of vectors **U**, **U'** to the calling program by means of scalars h, and vectors **A**, **B**, **C** such that

$$\mathbf{U}_1 = \mathbf{A}_1 - \mathbf{C}_1 \ , \qquad (4a)$$

$$\mathbf{U}_2 = \mathbf{A}_2 + \alpha \mathbf{C}_2 \ , \qquad (4b)$$

$$\mathbf{U}_2' = (\mathbf{B}_2 + \beta \mathbf{C}_2) \ /h \ \cdot \qquad (4c)$$

Normally, $\alpha = -1$ and h is the current integration step size, but during preliminary iterations to get initial values $\alpha = 0$ and $h = 1$. The scalars are supplied by `DC05` as elements of its work array `RW`; **A** and **B** are held in the first two columns of array `YNOR`, and **C** in array `CORR` (see section 3). For convenience, the vectors **U** and **U'** are returned to the calling program, so that it need not actually contain code to evaluate (4).

Then the Newton iteration matrix

$$W = \frac{\partial \mathbf{F}}{\partial \mathbf{C}} ,$$

$$= \begin{pmatrix} W_{11} & W_{12} \\ W_{21} & W_{22} \end{pmatrix} = \begin{pmatrix} -J_{11} & \alpha J_{12} \\ -J_{21} & \alpha J_{22} + \beta K_{22} / h \end{pmatrix} , \tag{5}$$

$$= \begin{pmatrix} -\frac{\partial \mathbf{F}_1}{\partial \mathbf{U}_1} & \alpha \frac{\partial \mathbf{F}_1}{\partial \mathbf{U}_2} \\ -\frac{\partial \mathbf{F}_2}{\partial \mathbf{U}_1} & \alpha \frac{\partial \mathbf{F}_2}{\partial \mathbf{U}_2} + \frac{\beta}{h} \frac{\partial \mathbf{F}_2}{\partial \mathbf{U}'_2} \end{pmatrix} .$$

of the equations (1) is used in solving such problems, from the beginning for a DAE or implicit problem or after a possible initial transient for an explicit stiff problem. Here we have written J_{ik} for the Jacobian matrix $\partial \mathbf{F}_i / \partial \mathbf{U}_k$ and K_{ik} for $\partial \mathbf{F}_i / \partial \mathbf{U}'_k$. The matrix W is used in Newton iterations, solving the correction equations

$$W \delta \mathbf{C} + \mathbf{F} = \mathbf{0} , \tag{6}$$

for a correction vector $\delta \mathbf{C}$ to be added to $\mathbf{C}$. Normally, the calling program must return $\delta \mathbf{C}$, but during an initial transient of a stiff explicit problem (3), or throughout a non-stiff explicit problem, Newton iteration may not be needed. In such cases, **F** should be returned until `DC05` specifically requests Newton iteration; it will do this from the beginning in DAE problems, but may never do so on a non-stiff ODE problem.