NAG Fortran Library Routine Document

D02NNF

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details.

1 Purpose

D02NNF is a reverse communication routine for integrating stiff systems of implicit ordinary differential equations coupled with algebraic equations.

2 Specification

```
 SUBROUTINE D02NNF (NEQ, NEQMAX, T, TOUT, Y, YDOT, RWORK, RTOL, ATOL,
1                   ITOL, INFORM, YSAVE, NY2DIM, WKJAC, NWKJAC, JACPVT,
2                   NJCPVT, IMON, INLN, IRES, IREVCM, LDERIV, ITASK,
3                   ITRACE, IFAIL)

 INTEGER            NEQ, NEQMAX, ITOL, INFORM(23), NY2DIM, NWKJAC,
1                   JACPVT(NJCPVT), NJCPVT, IMON, INLN, IRES, IREVCM,
2                   ITASK, ITRACE, IFAIL
 double precision   T, TOUT, Y(NEQMAX), YDOT(NEQMAX), RWORK(50+4*NEQMAX),
1                   RTOL(*), ATOL(*), YSAVE(NEQMAX,NY2DIM), WKJAC(NWKJAC)
 LOGICAL            LDERIV(2)
```

3 Description

D02NNF is a general purpose routine for integrating the initial value problem for a stiff system of implicit ordinary differential equations coupled with algebraic equations, written in the form

$$A(t,y)y' = g(t,y).$$

An outline of a typical calling program is given below:

```
C
C     declarations
C
      call linear algebra setup routine
      call integrator setup routine
      IREVCM=0
 1000 CALL D02NNF(NEQ, NEQMAX, T, TOUT, Y, YDOT, RWORK, RTOL,
       ATOL, ITOL, INFORM, YSAVE, NY2DIM, WKJAC, NWKJAC, JACPVT,
       NJCPVT, IMON, INLN, IRES, IREVCM, LDERIV,
       ITASK, ITRACE, IFAIL)

      IF (IREVCM.GT.0) THEN
        IF (IREVCM.GT.7 .AND. IREVCM.LT.11) THEN
          IF (IREVCM.EQ.8) THEN
            supply the Jacobian matrix                          (i)
          ELSE IF (IREVCM.EQ.9) THEN
            perform monitoring tasks requested by the user      (ii)
          ELSE IF (IREVCM.EQ.10) THEN
            indicates an unsuccessful step
          END IF
        ELSE
          evaluate the residual                                 (iii)
        ENDIF
        GO TO 1000
      END IF
C
C     post processing (optional linear algebra diagnostic call
C     (sparse case only), optional integrator diagnostic call)
C
      STOP
```

```
      END
```

There are three major operations that may be required of the calling (sub)program on an intermediate return (IREVCM $\neq 0$) from D02NNF; these are denoted (i), (ii) and (iii) above.

The following sections describe in greater detail exactly what is required of each of these operations.

(i) Supply the Jacobian matrix

You need only provide this facility if the parameter JCEVAL = 'A' (or 'F' if using sparse matrix linear algebra) in a call to the linear algebra setup routine. If the Jacobian matrix is to be evaluated numerically by the integrator, then the remainder of section (i) can be ignored.

We must define the system of nonlinear equations which is solved internally by the integrator. The time derivative, y', has the form

$$y' = (y - z)/(hd),$$

where h is the current step size and d is a parameter that depends on the integration method in use. The vector y is the current solution and the vector z depends on information from previous time steps. This means that $\frac{d}{dy'}(\ ) = \frac{1}{(hd)}\frac{d}{dy}(\ )$.

The system of nonlinear equations that is solved has the form

$$A(t,y)y' - g(t,y) = 0$$

but is solved in the form

$$r(t,y) = 0,$$

where r is the function defined by

$$r(t,y) = (hd)(A(t,y)(y - z)/(hd) - g(t,y)).$$

It is the Jacobian matrix $\dfrac{\partial r}{\partial y}$ that you must supply as follows:

$$\frac{\partial r_i}{\partial y_j} = a_{ij}(t,y) + hd\frac{\partial}{\partial y_j}\left(\sum_{k=1}^{\mathrm{NEQ}} a_{ik}(t,y)y'_k - g_i(t,y)\right),$$

where t, h and d are located in RWORK(19), RWORK(16) and RWORK(20) respectively and the arrays Y and YDOT contain the current solution and time derivatives respectively. Only the non-zero elements of the Jacobian need be set, since the locations where it is to be stored are preset to zero.

Hereafter in this document this operation will be referred to as JAC.

(ii) Perform tasks requested by you

This operation is essentially a monitoring function and additionally provides the opportunity of changing the current values of Y, YDOT, HNEXT (the step size that the integrator proposes to take on the next step), HMIN (the minimum step size to be taken on the next step), and HMAX (the maximum step size to be taken on the next step). The scaled local error at the end of a time step may be obtained by calling the ***double precision*** function D02ZAF as follows:

```
      IFAIL = 1
      ERRLOC = D02ZAF(NEQ,ROWK(51+NEQMAX),RWORK(51),IFAIL)
C     CHECK IFAIL BEFORE PROCEEDING
```

The following gives details of the location within the array RWORK of variables that may be of interest to you:

Variable	Specification	Location
TCURR	the current value of the independent variable	RWORK(19)
HLAST	last step size successfully used by the integrator	RWORK(15)
HNEXT	step size that the integrator proposes to take on the next step	RWORK(16)
HMIN	minimum step size to be taken on the next step	RWORK(17)
HMAX	maximum step size to be taken on the next step	RWORK(18)
NQU	the order of the integrator used on the last step	RWORK(10)

You are advised to consult the description of MONITR in D02NGF for details on what optional input can be made.

If either Y or YDOT are changed, then IMON must be set to 2 before return to D02NNF. If either of the values HMIN or HMAX are changed, then IMON must be set ≥ 3 before return to D02NNF. If HNEXT is changed, then IMON must be set to 4 before return to D02NNF.

In addition you can force D02NNF to evaluate the residual vector

$$A(t,y)y' - g(t,y)$$

by setting IMON $= 0$ and INLN $= 3$ and then returning to D02NNF; on return to this monitoring operation the residual vector will be stored in RWORK$(50 + 2 \times \text{NEQMAX} + i)$, for $i = 1, 2, \ldots, \text{NEQ}$.

Hereafter in this document this operation will be referred to as MONITR.

(iii) Evaluate the residual

This operation must evaluate the residual

$$r = g(t,y) - A(t,y)y' \tag{1}$$

in one case and

$$r = -A(t,y)y' \tag{2}$$

in another, where t is located in RWORK(19). The form of the residual that is returned is determined by the value of IRES returned by D02NNF. If IRES $= -1$, then the residual defined by equation (2) above must be returned; if IRES $= 1$, then the residual returned by equation (1) above must be returned.

Hereafter in this document this operation will be referred to as RESID.

4 References

See the D02M/N Sub-chapter Introduction.

5 Parameters

Note: this routine uses **reverse communication.** Its use involves an initial entry, intermediate exits and re-entries, and a final exit, as indicated by the **parameter IREVCM**. Between intermediate exits and re-entries, **all parameters other than YDOT, RWORK, WKJAC, IMON, INLN and IRES must remain unchanged**.

1: NEQ – INTEGER *Input*

On initial entry: the number of equations to be solved.

Constraint: NEQ ≥ 1.

2: NEQMAX – INTEGER *Input*

On initial entry: a bound on the maximum number of equations to be solved during the integration.

Constraint: NEQMAX $\geq$ NEQ.

3: T – ***double precision*** *Input/Output*

On initial entry: the value of the independent variable t. The input value of T is used only on the first call as the initial point of the integration.

On final exit: the value at which the computed solution y is returned (usually at TOUT).

4: TOUT – ***double precision*** *Input/Output*

On initial entry: the next value of t at which a computed solution is desired. For the initial t, the input value of TOUT is used to determine the direction of integration. Integration is permitted in either direction (see also ITASK).

Constraint: TOUT $\neq$ T.

On exit: is unaltered unless ITASK $= 6$ and LDERIV(2) = .TRUE. on entry (see also ITASK and LDERIV) in which case TOUT will be set to the result of taking a small step at the start of the integration.

5: Y(NEQMAX) – ***double precision*** array *Input/Output*

On initial entry: the values of the dependent variables (solution). On the first call the first NEQ elements of y must contain the vector of initial values.

On final exit: the computed solution vector evaluated at T (usually $t =$ TOUT).

6: YDOT(NEQMAX) – ***double precision*** array *Input/Output*

On initial entry: if LDERIV(1) = .TRUE., YDOT must contain approximations to the time derivatives y' of the vector y. If LDERIV(1) = .FALSE., then YDOT need not be set on entry.

On final exit: contains the time derivatives y' of the vector y at the last integration point.

7: RWORK$(50 + 4 \times$ NEQMAX$)$ – ***double precision*** array *Communication Array*

On initial entry: must be the same array as used by one of the method setup routines D02NVF, D02NWF or D02MVF, and by one of the storage setup routines D02NVF, D02NTF or D02NUF. The contents of RWORK must not be changed between any call to a setup routine and the first call to D02NNF.

On intermediate re-entry: must contain residual evaluations as described under the parameter IREVCM.

On intermediate exit: contains information for JAC, RESID and MONITR operations as described under Section 3 and the parameter IREVCM.

8: RTOL$(*)$ – ***double precision*** array *Input*

Note: the dimension of the array RTOL must be at least 1 if ITOL $= 1$ or ITOL $= 2$ and at least NEQ otherwise.

On initial entry: the relative local error tolerance.

Constraint: RTOL$(i) \geq 0.0$ for all relevant i (see ITOL).

9: ATOL$(*)$ – ***double precision*** array *Input*

Note: the dimension of the array ATOL must be at least 1 if ITOL $= 1$ or ITOL $= 3$ and at least NEQ otherwise.

On initial entry: the absolute local error tolerance.

Constraint: ATOL$(i) \geq 0.0$ for all relevant i (see ITOL).

10: ITOL – INTEGER *Input*

On initial entry: a value to indicate the form of the local error test. ITOL indicates to D02NNF whether to interpret either or both of RTOL or ATOL as a vector or a scalar. The error test to be satisfied is $\|e_i/w_i\| < 1.0$, where w_i is defined as follows:

ITOL	RTOL	ATOL	w_i
1	scalar	scalar	$\text{RTOL}(1) \times \lvert y_i \rvert + \text{ATOL}(1)$
2	scalar	vector	$\text{RTOL}(1) \times \lvert y_i \rvert + \text{ATOL}(i)$
3	vector	scalar	$\text{RTOL}(i) \times \lvert y_i \rvert + \text{ATOL}(1)$
4	vector	vector	$\text{RTOL}(i) \times \lvert y_i \rvert + \text{ATOL}(i)$

e_i is an estimate of the local error in y_i, computed internally, and the choice of norm to be used is defined by a previous call to an integrator setup routine.

Constraint: $1 \leq \text{ITOL} \leq 4$.

11: INFORM(23) – INTEGER array *Communication Array*

12: YSAVE(NEQMAX,NY2DIM) – ***double precision*** array *Communication Array*
13: NY2DIM – INTEGER *Input*

On initial entry: the second dimension of the array YSAVE as declared in the (sub)program from which D02NNF is called. An appropriate value for NY2DIM is described in the specifications of the integrator setup routines D02MVF, D02NVF and D02NWF. This value must be the same as that supplied to the integrator setup routine.

14: WKJAC(NWKJAC) – ***double precision*** array *Input/Output*

On intermediate re-entry: elements of the Jacobian as defined under the description of IREVCM. If a numerical Jacobian was requested then WKJAC is used for workspace.

On intermediate exit: the Jacobian is overwritten.

15: NWKJAC – INTEGER *Input*

On initial entry: the dimension of the array WKJAC as declared in the (sub)program from which D02NNF is called. The actual size depends on the linear algebra method used. An appropriate value for NWKJAC is described in the specifications of the linear algebra setup routines D02NSF, D02NTF and D02NUF for full, banded and sparse matrix linear algebra respectively. This value must be the same as that supplied to the linear algebra setup routine.

16: JACPVT(NJCPVT) – INTEGER array *Communication Array*
17: NJCPVT – INTEGER *Input*

On initial entry: the dimension of the array JACPVT as declared in the (sub)program from which D02NNF is called. The actual size depends on the linear algebra method used. An appropriate value for NJCPVT is described in the specifications of the linear algebra setup routines D02NTF and D02NUF for banded and sparse matrix linear algebra respectively. This value must be the same as that supplied to the linear algebra setup routine. When full matrix linear algebra is chosen, the array JACPVT is not used and hence NJCPVT should be set to 1.

18: IMON – INTEGER *Input/Output*

On intermediate exit: used to pass information between D02NNF and the MONITR operation (see Section 3). With IREVCM = 9, IMON contains a flag indicating under what circumstances the return from D02NNF occurred:

IMON = −2

Exit from D02NNF after IRES = 4 (set in the user-supplied (sub)program RESID operation (see Section 3) caused an early termination (this facility could be used to locate discontinuities).

IMON = −1

The current step failed repeatedly.

IMON = 0

Exit from D02NNF after a call to the internal nonlinear equation solver.

IMON $= 1$

The current step was successful.

On intermediate re-entry: may be reset to determine subsequent action in D02NNF.

IMON $= -2$

Integration is to be halted. A return will be made from D02NNF to the calling (sub)program with IFAIL $= 12$.

IMON $= -1$

Allow D02NNF to continue with its own internal strategy. The integrator will try up to three restarts unless IMON $\neq -1$.

IMON $= 0$

Return to the internal nonlinear equation solver, where the action taken is determined by the value of INLN.

IMON $= 1$

Normal exit to D02NNF to continue integration.

IMON $= 2$

Restart the integration at the current time point. The integrator will restart from order 1 when this option is used. The internal initialization module solves for new values of y and y' by using the values supplied in Y and YDOT by the MONITR operation (see Section 3) as initial estimates.

IMON $= 3$

Try to continue with the same step size and order as was to be used before entering the MONITR operation (see Section 3). HMIN and HMAX may be altered if desired.

IMON $= 4$

Continue the integration but using a new value of HNEXT and possibly new values of HMIN and HMAX.

19: INLN – INTEGER *Input/Output*

On intermediate re-entry: with IMON $= 0$ and IREVCM $= 9$, INLN specifies the action to be taken by the internal nonlinear equation solver. By setting INLN $= 3$ and returning to D02NNF, the residual vector is evaluated and placed in RWORK$(50 + 2 \times$ NEQMAX $+ i)$, for $i = 1, 2, \ldots,$ NEQ and then the MONITR operation (see Section 3) is invoked again. At present this is the only option available: INLN must not be set to any other value.

On intermediate exit: contains a flag indicating the action to be taken, if any, by the internal nonlinear equation solver.

20: IRES – INTEGER *Input/Output*

On intermediate exit: with IREVCM $= 1$, 2, 3, 4, 5, 6, 7 or 11, IRES specifies the form of the residual to be returned by the RESID operation (see Section 3).

If IRES $= 1$, then $r = g(t,y) - A(t,y)y'$ must be returned.

If IRES $= -1$, then $r = -A(t,y)y'$ must be returned.

On intermediate re-entry: should be unchanged unless one of the following actions is required of D02NNF in which case IRES should be set accordingly.

IRES $= 2$

Indicates to D02NNF that control should be passed back immediately to the calling (sub)program with the error indicator set to IFAIL $= 11$.

IRES = 3

Indicates to D02NNF that an error condition has occurred in the solution vector, its time derivative or in the value of t. The integrator will use a smaller time step to try to avoid this condition. If this is not possible D02NNF returns to the calling (sub)program with the error indicator set to IFAIL = 7.

IRES = 4

Indicates to D02NNF to stop its current operation and to enter the MONITR operation (see Section 3) immediately.

21: IREVCM – INTEGER *Input/Output*

On initial entry: must contain 0.

On intermediate re-entry: should remain unchanged.

On intermediate exit: indicates what action you must take before re-entering D02NNF. The possible exit values of IREVCM are 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 or 11 which should be interpreted as follows:

IREVCM = 1, 2, 3, 4, 5, 6, 7 or 11

Indicates that a RESID operation (see Section 3) is required: you must supply the residual of the system. For each of these values of IREVCM, y_i is located in Y(i), for $i = 1, 2, \ldots, \text{NEQ}$.

For IREVCM = 1, 3, 6 or 11, y_i' is located in YDOT(i) and r_i should be stored in RWORK($50 + 2 \times \text{NEQMAX} + i$), for $i = 1, 2, \ldots, \text{NEQ}$.

For IREVCM = 2, y_i' is located in RWORK($50 + \text{NEQMAX} + i$) and r_i should be stored in RWORK($50 + 2 \times \text{NEQMAX} + i$), for $i = 1, 2, \ldots, \text{NEQ}$.

For IREVCM = 4 or 7, y_i' is located in YDOT(i) and r_i should be stored in RWORK($50 + \text{NEQMAX} + i$), for $i = 1, 2, \ldots, \text{NEQ}$.

For IREVCM = 5, y_i' is located in RWORK($50 + 2 \times \text{NEQMAX} + i$) and r_i should be stored in YDOT(i), for $i = 1, 2, \ldots, \text{NEQ}$.

IREVCM = 8

Indicates that a JAC operation (see Section 3) is required: you must supply the Jacobian matrix.

If full matrix linear algebra is being used, then the (i,j)th element of the Jacobian must be stored in WKJAC($(j-1) \times \text{NEQ} + i$).

If banded matrix linear algebra is being used, then the (i,j)th element of the Jacobian must be stored in WKJAC($(i-1) \times m_B + k$), where $m_B = m_L + m_U + 1$ and $k = \min(m_L - i + 1, 0) + j$; here m_L and m_U are the number of subdiagonals and superdiagonals, respectively, in the band.

If sparse matrix linear algebra is being used, then D02NRF must be called to determine which column of the Jacobian is required and where it should be stored.

```
CALL D02NRF(J, IPLACE, INFORM)
```

will return in J the number of the column of the Jacobian that is required and will set IPLACE = 1 or 2 (see D02NRF). If IPLACE = 1, then the (i,j)th element of the Jacobian must be stored in RWORK($50 + 2 \times \text{NEQMAX} + i$); otherwise it must be stored in RWORK($50 + \text{NEQMAX} + i$).

IREVCM = 9

Indicates that a MONITR operation (see Section 3) can be performed.

IREVCM = 10

Indicates that the current step was not successful, due to error test failure or convergence test failure. The only information supplied to you on this return is the current value of the

variable t, located in RWORK(19). No values must be changed before re-entering D02NNF; this facility enables you to determine the number of unsuccessful steps.

On final exit: IREVCM $= 0$ indicating that the user-specified task has been completed or an error has been encountered (see the descriptions for ITASK and IFAIL).

Constraint: $0 \leq$ IREVCM ≤ 11.

22: LDERIV(2) – LOGICAL array *Input/Output*

On initial entry: LDERIV(1) must be set to .TRUE. if you have supplied both an initial y and an initial y'. LDERIV(1) must be set to .FALSE. if only the initial y has been supplied.

LDERIV(2) must be set to .TRUE. if the integrator is to use a modified Newton method to evaluate the initial y and y'. Note that y and y', if supplied, are used as initial estimates. This method involves taking a small step at the start of the integration, and if ITASK $= 6$ on entry, T and TOUT will be set to the result of taking this small step. LDERIV(2) must be set to .FALSE. if the integrator is to use functional iteration to evaluate the initial y and y', and if this fails a modified Newton method will then be attempted. LDERIV(2) = .TRUE. is recommended if there are implicit equations or the initial y and y' are zero.

On final exit: LDERIV(1) is normally unchanged. However if ITASK $= 6$ and internal initialization was successful then LDERIV(1) = .TRUE..

LDERIV(2) = .TRUE., if implicit equations were detected. Otherwise LDERIV(2) = .FALSE..

23: ITASK – INTEGER *Input*

On initial entry: the task to be performed by the integrator. The permitted values for ITASK and their meanings are detailed below:

ITASK $= 1$

Normal computation of output values of $y(t)$ at $t =$ TOUT (by overshooting and interpolating).

ITASK $= 2$

Take one step only and return.

ITASK $= 3$

Stop at the first internal integration point at or beyond $t =$ TOUT and return.

ITASK $= 4$

Normal computation of output values of $y(t)$ at $t =$ TOUT but without overshooting $t =$ TCRIT. TCRIT must be specified as an option in one of the integrator setup routines prior to the first call to the integrator, or specified in the optional input routine prior to a continuation call. TCRIT (e.g., see D02NVF) may be equal to or beyond TOUT, but not before it in the direction of integration.

ITASK $= 5$

Take one step only and return, without passing TCRIT (e.g., see D02NVF). TCRIT must be specified under ITASK $= 4$.

ITASK $= 6$

The integrator will solve for the initial values of y and y' only and then return to the calling (sub)program without doing the integration. This option can be used to check the initial values of y and y'. Functional iteration or a 'small' backward Euler method used in conjunction with a damped Newton iteration is used to calculate these values (see LDERIV above). Note that if a backward Euler step is used then the value of t will have been advanced a short distance from the initial point.

Note: if D02NNF is recalled with a different value of ITASK (and TOUT altered) then the initialization procedure is repeated, possibly leading to different initial conditions.

Constraint: $1 \leq \text{ITASK} \leq 6$.

24: ITRACE – INTEGER *Input*

On initial entry: the level of output that is printed by the integrator. ITRACE may take the value -1, 0, 1, 2 or 3.

ITRACE < -1

-1 is assumed and similarly if ITRACE > 3, then 3 is assumed.

ITRACE $= -1$

No output is generated.

ITRACE $= 0$

Only warning messages are printed on the current error message unit (see X04AAF).

ITRACE > 0

Warning messages are printed as above, and on the current advisory message unit (see X04ABF) output is generated which details Jacobian entries, the nonlinear iteration and the time integration. The advisory messages are given in greater detail the larger the value of ITRACE.

25: IFAIL – INTEGER *Input/Output*

On entry: IFAIL must be set to 0, -1 or 1. If you are unfamiliar with this parameter you should refer to Chapter P01 for details.

On exit: IFAIL $= 0$ unless the routine detects an error (see Section 6).

For environments where it might be inappropriate to halt program execution when an error is detected, the value -1 or 1 is recommended. If the output of error messages is undesirable, then the value 1 is recommended. Otherwise, if you are not familiar with this parameter the recommended value is 0. **When the value -1 or 1 is used it is essential to test the value of IFAIL on exit.**

6 Error Indicators and Warnings

If on entry IFAIL $= 0$ or -1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

Errors or warnings detected by the routine:

IFAIL $= 1$

On entry, the integrator detected an illegal input or that a linear algebra and/or integrator setup routine has not been called prior to the call to the integrator. If ITRACE ≥ 0, the form of the error will be detailed on the current error message unit (see X04AAF).

IFAIL $= 2$

The maximum number of steps specified has been taken (see the description of optional inputs in the integrator setup routines and the optional input continuation routine, D02NZF).

IFAIL $= 3$

With the given values of RTOL and ATOL no further progress can be made across the integration range from the current point T. The components $\text{Y}(1), \text{Y}(2), \ldots, \text{Y}(\text{NEQ})$ contain the computed values of the solution at the current point T.

IFAIL = 4

There were repeated error test failures on an attempted step, before completing the requested task, but the integration was successful as far as T. The problem may have a singularity, or the local error requirements may be inappropriate.

IFAIL = 5

There were repeated convergence test failures on an attempted step, before completing the requested task, but the integration was successful as far as T. This may be caused by an inaccurate Jacobian matrix or one which is incorrectly computed.

IFAIL = 6

Some error weight w_i became zero during the integration (see the description of ITOL). Pure relative error control (ATOL$(i) = 0.0$) was requested on a variable (the ith) which has now vanished. The integration was successful as far as T.

IFAIL = 7

The RESID operation (see Section 3) set the error flag IRES = 3 continually despite repeated attempts by the integrator to avoid this.

IFAIL = 8

LDERIV(1) = .FALSE. on entry but the internal initialization routine was unable to initialize y' (more detailed information may be directed to the current error message unit, see X04AAF).

IFAIL = 9

A singular Jacobian $\dfrac{\partial r}{\partial y}$ has been encountered. You should check the problem formulation and Jacobian calculation.

IFAIL = 10

An error occurred during Jacobian formulation or back-substitution (a more detailed error description may be directed to the current error message unit, see X04AAF).

IFAIL = 11

The RESID operation (see Section 3) signalled the integrator to halt the integration and return by setting IRES = 2. Integration was successful as far as T.

IFAIL = 12

The MONITR operation (see Section 3) set IMON = -2 and so forced a return but the integration was successful as far as T.

IFAIL = 13

The requested task has been completed, but it is estimated that a small change in RTOL and ATOL is unlikely to produce any change in the computed solution. (Only applies when you are not operating in one step mode, that is when ITASK $\neq$ 2 or 5.)

IFAIL = 14

The values of RTOL and ATOL are so small that the routine is unable to start the integration.

7 Accuracy

The accuracy of the numerical solution may be controlled by a careful choice of the parameters RTOL and ATOL, and to a much lesser extent by the choice of norm. You are advised to use scalar error control unless the components of the solution are expected to be poorly scaled. For the type of decaying solution

typical of many stiff problems, relative error control with a small absolute error threshold will be most appropriate (that is, you are advised to choose ITOL = 1 with ATOL(1) small but positive).

8 Further Comments

The cost of computing a solution depends critically on the size of the differential system and to a lesser extent on the degree of stiffness of the problem; also on the type of linear algebra being used. For further details see Section 8 of the documents for D02NGF (full matrix), D02NHF (banded matrix) or D02NJF (sparse matrix).

In general, you are advised to choose the Backward Differentiation Formula option (setup routine D02NVF) but if efficiency is of great importance and especially if it is suspected that $\frac{\partial}{\partial y}(A^{-1}g)$ has complex eigenvalues near the imaginary axis for some part of the integration, you should try the BLEND option (setup routine D02NWF).

9 Example

We solve the well-known stiff Robertson problem written as a differential system in implicit form

$$\begin{aligned} r_1 &= \left(a' + b' + c'\right) \\ r_2 &= 0.04a - 1.0\text{E}4bc - 3.0\text{E}7b^2 - b' \\ r_3 &= \qquad\qquad\qquad 3.0\text{E}7b^2 - c' \end{aligned}$$

over the range $[0, 10]$ with initial conditions $a = 1.0$ and $b = c = 0.0$ and with scalar error control (ITOL = 1). We integrate to the first internal integration point past TOUT = 10.0 (ITASK = 3), using a BDF method (setup routine D02NVF) and a modified Newton method. We treat the Jacobian as sparse (setup routine D02NUF) and we calculate it analytically. In this program we also illustrate the monitoring of step failures (IREVCM = 10) and the forcing of a return when the component falls below 0.9 in the evaluation of the residual by setting IRES = 2.

9.1 Program Text

```
*     D02NNF Example Program Text
*     Mark 14 Revised. NAG Copyright 1989.
*     .. Parameters ..
      INTEGER          NOUT
      PARAMETER        (NOUT=6)
      INTEGER          NEQ, NEQMAX, NRW, NINF, NJCPVT, NWKJAC, NIA, NJA,
     +                 MAXORD, NY2DIM, MAXSTP, MXHNIL
      PARAMETER        (NEQ=3,NEQMAX=NEQ,NRW=50+4*NEQMAX,NINF=23,
     +                 NJCPVT=150,NWKJAC=100,NIA=1,NJA=1,MAXORD=5,
     +                 NY2DIM=MAXORD+1,MAXSTP=200,MXHNIL=5)
      INTEGER          LACORB, LSAVRB
      PARAMETER        (LACORB=50+NEQMAX,LSAVRB=LACORB+NEQMAX)
      DOUBLE PRECISION H0, HMAX, HMIN, TCRIT
      PARAMETER        (H0=1.0D-4,HMAX=10.0D0,HMIN=1.0D-10,TCRIT=0.0D0)
      LOGICAL          PETZLD
      PARAMETER        (PETZLD=.TRUE.)
      DOUBLE PRECISION ETA, U, SENS
      PARAMETER        (ETA=1.0D-4,U=0.1D0,SENS=1.0D-6)
      LOGICAL          LBLOCK
      PARAMETER        (LBLOCK=.TRUE.)
*     .. Local Scalars ..
      DOUBLE PRECISION H, HU, HXD, T, TCUR, TOLSF, TOUT
      INTEGER          I, ICALL, IFAIL, IGROW, IMON, IMXER, INLN,
     +                 IPLACE, IRES, IREVCM, ISPLIT, ITASK, ITOL,
     +                 ITRACE, J, LACOR1, LACOR2, LACOR3, LIWREQ,
     +                 LIWUSD, LRWREQ, LRWUSD, LSAVR1, LSAVR2, LSAVR3,
     +                 NBLOCK, NFAILS, NGP, NITER, NJE, NLU, NNZ, NQ,
     +                 NQU, NRE, NST
*     .. Local Arrays ..
      DOUBLE PRECISION ATOL(NEQMAX), CONST(6), RTOL(NEQMAX), RWORK(NRW),
     +                 WKJAC(NWKJAC), Y(NEQMAX), YDOT(NEQMAX),
     +                 YSAVE(NEQMAX,NY2DIM)
```

```
      INTEGER          IA(NIA), INFORM(NINF), JA(NJA), JACPVT(NJCPVT)
      LOGICAL          ALGEQU(NEQMAX), LDERIV(2)
*     .. External Subroutines ..
      EXTERNAL         D02NNF, D02NRF, D02NUF, D02NVF, D02NXF, D02NYF,
     +                 X04ABF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'D02NNF Example Program Results'
      WRITE (NOUT,*)
      CALL X04ABF(1,NOUT)
*
*     Integrate towards TOUT stopping at the first mesh point beyond
*     TOUT (ITASK=3) using the B.D.F. formulae with a Newton method.
*     Employ scalar tolerances and the Jacobian is supplied, but its
*     structure is evaluated internally by calls to the Jacobian
*     forming part of the program (IREVCM=8). Default values for the
*     array CONST are used. Also count the number of step failures
*     (IREVCM=10).
*
      T = 0.0D0
      TOUT = 10.0D0
      ITASK = 3
      Y(1) = 1.0D0
      Y(2) = 0.0D0
      Y(3) = 0.0D0
      LDERIV(1) = .FALSE.
      LDERIV(2) = .FALSE.
      ITOL = 1
      RTOL(1) = 1.0D-4
      ATOL(1) = 1.0D-7
      DO 20 I = 1, 6
         CONST(I) = 0.0D0
   20 CONTINUE
      ISPLIT = 0
      NFAILS = 0
      IFAIL = 0
*
      CALL D02NVF(NEQMAX,NY2DIM,MAXORD,'Newton',PETZLD,CONST,TCRIT,HMIN,
     +            HMAX,H0,MAXSTP,MXHNIL,'Average-l2',RWORK,IFAIL)
      CALL D02NUF(NEQ,NEQMAX,'Analytical',NWKJAC,IA,NIA,JA,NJA,JACPVT,
     +            NJCPVT,SENS,U,ETA,LBLOCK,ISPLIT,RWORK,IFAIL)
*
*     Soft fail and error messages only
      IREVCM = 0
      IFAIL = 1
      ITRACE = 0
*
      LACOR1 = LACORB + 1
      LACOR2 = LACORB + 2
      LACOR3 = LACORB + 3
      LSAVR1 = LSAVRB + 1
      LSAVR2 = LSAVRB + 2
      LSAVR3 = LSAVRB + 3
      WRITE (NOUT,*) '    X          Y(1)           Y(2)           Y(3)'
      WRITE (NOUT,99999) T, (Y(I),I=1,NEQ)
*
   40 CONTINUE
*
      CALL D02NNF(NEQ,NEQMAX,T,TOUT,Y,YDOT,RWORK,RTOL,ATOL,ITOL,INFORM,
     +            YSAVE,NY2DIM,WKJAC,NWKJAC,JACPVT,NJCPVT,IMON,INLN,
     +            IRES,IREVCM,LDERIV,ITASK,ITRACE,IFAIL)
*
      IF (IREVCM.GT.0) THEN
         IF (IREVCM.EQ.1 .OR. IREVCM.EQ.3 .OR. IREVCM.EQ.6 .OR.
     +       IREVCM.EQ.11) THEN
*           Equivalent to RESID evaluation in forward communication
*           routines
            RWORK(LSAVR1) = -YDOT(1) - YDOT(2) - YDOT(3)
            RWORK(LSAVR2) = -YDOT(2)
            RWORK(LSAVR3) = -YDOT(3)
            IF (IRES.EQ.1) THEN
               RWORK(LSAVR1) = 0.0D0 + RWORK(LSAVR1)
```

```
               RWORK(LSAVR2) = 0.04D0*Y(1) - 1.0D4*Y(2)*Y(3) -
     +                         3.0D7*Y(2)*Y(2) + RWORK(LSAVR2)
               RWORK(LSAVR3) = 3.0D7*Y(2)*Y(2) + RWORK(LSAVR3)
            END IF
         ELSE IF (IREVCM.EQ.2) THEN
*           Equivalent to RESID evaluation in forward communication
*           routines
            RWORK(LSAVR1) = -RWORK(LACOR1) - RWORK(LACOR2) -
     +                      RWORK(LACOR3)
            RWORK(LSAVR2) = -RWORK(LACOR2)
            RWORK(LSAVR3) = -RWORK(LACOR3)
         ELSE IF (IREVCM.EQ.4 .OR. IREVCM.EQ.7) THEN
*           Equivalent to RESID evaluation in forward communication
*           routines
            RWORK(LACOR1) = -YDOT(1) - YDOT(2) - YDOT(3)
            RWORK(LACOR2) = -YDOT(2)
            RWORK(LACOR3) = -YDOT(3)
            IF (IRES.EQ.1) THEN
               RWORK(LACOR1) = 0.0D0 + RWORK(LACOR1)
               RWORK(LACOR2) = 0.04D0*Y(1) - 1.0D4*Y(2)*Y(3) -
     +                         3.0D7*Y(2)*Y(2) + RWORK(LACOR2)
               RWORK(LACOR3) = 3.0D7*Y(2)*Y(2) + RWORK(LACOR3)
            END IF
         ELSE IF (IREVCM.EQ.5) THEN
*           Equivalent to RESID evaluation in forward communication
*           routines
            YDOT(1) = 0.0D0 - RWORK(LSAVR1) - RWORK(LSAVR2) -
     +                RWORK(LSAVR3)
            YDOT(2) = 0.04D0*Y(1) - 1.0D4*Y(2)*Y(3) - 3.0D7*Y(2)*(2) -
     +                RWORK(LSAVR2)
            YDOT(3) = 3.0D7*Y(2)*Y(2) - RWORK(LSAVR3)
         ELSE IF (IREVCM.EQ.8) THEN
*           Equivalent to JAC evaluation in forward communication
*           routines
            CALL D02NRF(J,IPLACE,INFORM)
*
            HXD = RWORK(16)*RWORK(20)
*
            IF (IPLACE.LT.2) THEN
               IF (J.LT.2) THEN
                  RWORK(LSAVR1) = 1.0D0 - HXD*(0.0D0)
                  RWORK(LSAVR2) = 0.0D0 - HXD*(0.04D0)
*                 RWORK(LSAVR3) = 0.0 - HXD*(0.0)
               ELSE IF (J.EQ.2) THEN
                  RWORK(LSAVR1) = 1.0D0 - HXD*(0.0D0)
                  RWORK(LSAVR2) = 1.0D0 - HXD*(-1.0D4*Y(3)-6.0D7*Y(2))
                  RWORK(LSAVR3) = 0.0D0 - HXD*(6.0D7*Y(2))
               ELSE IF (J.GT.2) THEN
                  RWORK(LSAVR1) = 1.0D0 - HXD*(0.0D0)
                  RWORK(LSAVR2) = 0.0D0 - HXD*(-1.0D4*Y(2))
                  RWORK(LSAVR3) = 1.0D0 - HXD*(0.0D0)
               END IF
            ELSE
               IF (J.LT.2) THEN
                  RWORK(LACOR1) = 1.0D0 - HXD*(0.0D0)
                  RWORK(LACOR2) = 0.0D0 - HXD*(0.04D0)
*                 RWORK(LACOR3) = 0.0 - HXD*(0.0)
               ELSE IF (J.EQ.2) THEN
                  RWORK(LACOR1) = 1.0D0 - HXD*(0.0D0)
                  RWORK(LACOR2) = 1.0D0 - HXD*(-1.0D4*Y(3)-6.0D7*Y(2))
                  RWORK(LACOR3) = 0.0D0 - HXD*(6.0D7*Y(2))
               ELSE IF (J.GT.2) THEN
                  RWORK(LACOR1) = 1.0D0 - HXD*(0.0D0)
                  RWORK(LACOR2) = 0.0D0 - HXD*(-1.0D4*Y(2))
                  RWORK(LACOR3) = 1.0D0 - HXD*(0.0D0)
               END IF
            END IF
*           Step failure
         ELSE IF (IREVCM.EQ.10) THEN
            NFAILS = NFAILS + 1
         END IF
```

```
         GO TO 40
       ELSE
         IF (IFAIL.EQ.0) THEN
            WRITE (NOUT,99999) T, (Y(I),I=1,NEQ)
*
            CALL D02NYF(NEQ,NEQMAX,HU,H,TCUR,TOLSF,RWORK,NST,NRE,NJE,
     +                  NQU,NQ,NITER,IMXER,ALGEQU,INFORM,IFAIL)
*
            WRITE (NOUT,*)
            WRITE (NOUT,99997) ' HUSED = ', HU, '  HNEXT = ', H,
     +        '  TCUR = ', TCUR
            WRITE (NOUT,99996) ' NST = ', NST, '    NRE = ', NRE,
     +        '    NJE = ', NJE
            WRITE (NOUT,99996) ' NQU = ', NQU, '    NQ  = ', NQ,
     +        '  NITER = ', NITER
            WRITE (NOUT,99995) ' Max err comp = ', IMXER,
     +        '   No. of failed steps = ', NFAILS
            ICALL = 0
*
            CALL D02NXF(ICALL,LIWREQ,LIWUSD,LRWREQ,LRWUSD,NLU,NNZ,NGP,
     +                  ISPLIT,IGROW,LBLOCK,NBLOCK,INFORM)
*
            WRITE (NOUT,*)
            WRITE (NOUT,99994) ' NJCPVT (required ', LIWREQ, '  used ',
     +        LIWUSD, ')'
            WRITE (NOUT,99994) ' NWKJAC (required ', LRWREQ, '  used ',
     +        LRWUSD, ')'
            WRITE (NOUT,99993) ' No. of LU-decomps ', NLU,
     +        '  No. of nonzeros ', NNZ
            WRITE (NOUT,99995) ' No. of FCN calls to form Jacobian ',
     +        NGP, '  Try ISPLIT ', ISPLIT
            WRITE (NOUT,99992) ' Growth est ', IGROW,
     +        '  No. of blocks on diagonal ', NBLOCK
         ELSE IF (IFAIL.EQ.10) THEN
            ICALL = 1
*
            CALL D02NXF(ICALL,LIWREQ,LIWUSD,LRWREQ,LRWUSD,NLU,NNZ,NGP,
     +                  ISPLIT,IGROW,LBLOCK,NBLOCK,INFORM)
*
            WRITE (NOUT,*)
            WRITE (NOUT,99994) ' NJCPVT (required ', LIWREQ, '  used ',
     +        LIWUSD, ')'
            WRITE (NOUT,99994) ' NWKJAC (required ', LRWREQ, '  used ',
     +        LRWUSD, ')'
         ELSE
            WRITE (NOUT,*)
            WRITE (NOUT,99998) 'Exit D02NNF with IFAIL = ', IFAIL,
     +        '  and T = ', T
         END IF
       END IF
       STOP
*
99999 FORMAT (1X,F8.3,3(F13.5,2X))
99998 FORMAT (1X,A,I2,A,E12.5)
99997 FORMAT (1X,A,E12.5,A,E12.5,A,E12.5)
99996 FORMAT (1X,A,I6,A,I6,A,I6)
99995 FORMAT (1X,A,I4,A,I4)
99994 FORMAT (1X,A,I8,A,I8,A)
99993 FORMAT (1X,A,I4,A,I8)
99992 FORMAT (1X,A,I8,A,I4)
      END
```

9.2 Program Data

None.

9.3 Program Results

```
D02NNF Example Program Results

    X          Y(1)           Y(2)           Y(3)
   0.000      1.00000        0.00000        0.00000
WARNING... EQUATION(=I1) AND POSSIBLY OTHER EQUATIONS ARE
IMPLICIT AND IN CALCULATING THE INITIAL VALUES THE EQNS
WILL BE TREATED AS IMPLICIT.
IN ABOVE MESSAGE I1 =         1
  10.488      0.83759        0.00002        0.16239

 HUSED =  0.60471E+00  HNEXT =  0.60471E+00  TCUR =  0.10488E+02
 NST =     65    NRE =    163    NJE =     14
 NQU =      3    NQ  =      3  NITER =    154
 Max err comp =    3   No. of failed steps =    0

 NJCPVT (required       74  used      150)
 NWKJAC (required       16  used       77)
 No. of LU-decomps   14  No. of nonzeros        5
 No. of FCN calls to form Jacobian    0  Try ISPLIT   73
 Growth est      862  No. of blocks on diagonal    3
```