NAG Library Routine Document

D02NHF

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

D02NHF is a forward communication routine for integrating stiff systems of implicit ordinary differential equations coupled with algebraic equations when the Jacobian is a banded matrix.

2 Specification

```
SUBROUTINE D02NHF (NEQ, LDYSAV, T, TOUT, Y, YDOT, RWORK, RTOL, ATOL, ITOL,      &
                   INFORM, RESID, YSAV, SDYSAV, JAC, WKJAC, NWKJAC, JACPVT,     &
                   NJCPVT, MONITR, LDERIV, ITASK, ITRACE, IFAIL)

INTEGER            NEQ, LDYSAV, ITOL, INFORM(23), SDYSAV, NWKJAC,               &
                   JACPVT(NJCPVT), NJCPVT, ITASK, ITRACE, IFAIL
REAL (KIND=nag_wp) T, TOUT, Y(NEQ), YDOT(NEQ), RWORK(50+4*NEQ), RTOL(*),        &
                   ATOL(*), YSAV(LDYSAV,SDYSAV), WKJAC(NWKJAC)
LOGICAL            LDERIV(2)
EXTERNAL           RESID, JAC, MONITR
```

3 Description

D02NHF 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

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

It is designed specifically for the case where the resulting Jacobian is a banded matrix (see the description of JAC).

Both interval and step oriented modes of operation are available and also modes designed to permit intermediate output within an interval oriented mode.

An outline of a typical calling program for D02NHF is given below. It calls the banded matrix linear algebra setup routine D02NTF, the Backward Differentiation Formula (BDF) integrator setup routine D02NVF, and its diagnostic counterpart D02NYF.

```
!     Declarations

      EXTERNAL RESID, JAC, MONITR
          .
          .
          .
      IFAIL = 0
      CALL D02NVF(...,IFAIL)
      CALL D02NTF(NEQ, NEQMAX, JCEVAL, ML, MU, NWKJAC, NJCPVT, &
                  RWORK, IFAIL)
      IFAIL = -1
      CALL D02NHF(NEQ, NEQMAX, T, TOUT, Y, YDOT, RWORK, RTOL,  &
                  ATOL, ITOL, INFORM, RESID, YSAVE, NY2DIM,    &
         JAC, WKJAC, NWKJAC, JACPVT, NJCPVT, MONITR,  &
         LDERIV, ITASK, ITRACE, IFAIL)
      IF (IFAIL.EQ.1 .OR. IFAIL.GE.14) STOP
      IFAIL = 0
      CALL D02NYF(...)
          .
          .
          .
```

```
      STOP
      END
```

The linear algebra setup routine D02NTF and one of the integrator setup routines, D02MVF, D02NVF or D02NWF, must be called prior to the call of D02NHF. The integrator diagnostic routine D02NYF may be called after the call to D02NHF. There is also a routine, D02NZF, designed to permit you to change step size on a continuation call to D02NHF without restarting the integration process.

4 References

See the D02M–N sub-chapter Introduction.

5 Parameters

1: NEQ – INTEGER *Input*

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

Constraint: NEQ ≥ 1.

2: LDYSAV – INTEGER *Input*

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

Constraint: LDYSAV $\geq$ NEQ.

3: T – REAL (KIND=nag_wp) *Input/Output*

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

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

4: TOUT – REAL (KIND=nag_wp) *Input/Output*

On 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: normally unchanged. However, when ITASK $= 6$, then TOUT contains the value of T at which initial values have been computed without performing any integration. See descriptions of ITASK and LDERIV.

5: Y(NEQ) – REAL (KIND=nag_wp) array *Input/Output*

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

6: YDOT(NEQ) – REAL (KIND=nag_wp) array *Input/Output*

On entry: if LDERIV$(1) =$ .TRUE., YDOT must contain approximations to the time derivatives y' of the vector y.

If LDERIV$(1) =$ .FALSE., YDOT need not be set on entry.

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

7: RWORK$(50 + 4 \times \text{NEQ})$ – REAL (KIND=nag_wp) array *Communication Array*

8: RTOL$(*)$ – REAL (KIND=nag_wp) array *Input*

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

On entry: the relative local error tolerance.

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

9: ATOL$(*)$ – REAL (KIND=nag_wp) array *Input*

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

On entry: the absolute local error tolerance.

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

10: ITOL – INTEGER *Input*

On entry: a value to indicate the form of the local error test. ITOL indicates to D02NHF 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	RTOL$(1) \times \|y_i\| +$ ATOL(1)
2	scalar	vector	RTOL$(1) \times \|y_i\| +$ ATOL(i)
3	vector	scalar	RTOL$(i) \times \|y_i\| +$ ATOL(1)
4	vector	vector	RTOL$(i) \times \|y_i\| +$ 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: ITOL $= 1$, 2, 3 or 4.

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

12: RESID – SUBROUTINE, supplied by the user. *External Procedure*

RESID must evaluate the residual

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

in one case and

$$r = -A(t, y)y'$$

in another.

The specification of RESID is:

```
SUBROUTINE RESID (NEQ, T, Y, YDOT, R, IRES)

INTEGER            NEQ, IRES
REAL (KIND=nag_wp) T, Y(NEQ), YDOT(NEQ), R(NEQ)
```

1: NEQ – INTEGER *Input*

On entry: the number of equations being solved.

2: T – REAL (KIND=nag_wp) *Input*

On entry: t, the current value of the independent variable.

3: Y(NEQ) – REAL (KIND=nag_wp) array *Input*

On entry: the value of y_i, for $i = 1, 2, \ldots, \text{NEQ}$.

4: YDOT(NEQ) – REAL (KIND=nag_wp) array *Input*

On entry: the value of y'_i, for $i = 1, 2, \ldots, \text{NEQ}$, at t.

5: R(NEQ) – REAL (KIND=nag_wp) array *Output*

On exit: R(i) must contain the ith component of r, for $i = 1, 2, \ldots, \text{NEQ}$, where

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

or

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

and where the definition of r is determined by the input value of IRES.

6: IRES – INTEGER *Input/Output*

On entry: the form of the residual that must be returned in array R.

IRES $= -1$

The residual defined in equation (2) must be returned.

IRES $= 1$

The residual defined in equation (1) must be returned.

On exit: should be unchanged unless one of the following actions is required of the integrator, in which case IRES should be set accordingly.

IRES $= 2$

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

IRES $= 3$

Indicates to the integrator 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, the integrator returns to the calling (sub)program with the error indicator set to IFAIL $= 7$.

IRES $= 4$

Indicates to the integrator to stop its current operation and to enter MONITR immediately with parameter IMON $= -2$.

RESID must either be a module subprogram USEd by, or declared as EXTERNAL in, the (sub)program from which D02NHF is called. Parameters denoted as *Input* must **not** be changed by this procedure.

13: YSAV(LDYSAV,SDYSAV) – REAL (KIND=nag_wp) array *Communication Array*

14: SDYSAV – INTEGER *Input*

On entry: an appropriate value for SDYSAV 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.

15: JAC – SUBROUTINE, supplied by the NAG Library or the user. *External Procedure*

JAC must evaluate the Jacobian of the system. If this option is not required, the actual argument for JAC must be the dummy routine D02NHZ. (D02NHZ is included in the NAG Library.) You must indicate to the integrator whether this option is to be used by setting the parameter JCEVAL appropriately in a call to the banded linear algebra setup routine D02NTF.

First we must define the system of nonlinear equations which is solved internally by the integrator. The time derivative, y', generated internally, 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 $\frac{d}{dy'}(\ ) = (hd)\frac{d}{dy}(\ )$. The system of nonlinear equations that is solved has the form

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

but it 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 in JAC as follows:

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

The specification of JAC is:

```
SUBROUTINE JAC (NEQ, T, Y, YDOT, H, D, ML, MU, P)

INTEGER             NEQ, ML, MU
REAL (KIND=nag_wp) T, Y(NEQ), YDOT(NEQ), H, D, P(ML+MU+1,NEQ)
```

1: NEQ – INTEGER *Input*

On entry: the number of equations being solved.

2: T – REAL (KIND=nag_wp) *Input*

On entry: t, the current value of the independent variable.

3: Y(NEQ) – REAL (KIND=nag_wp) array *Input*

On entry: y_i, for $i = 1, 2, \ldots,$ NEQ, the current solution component.

4: YDOT(NEQ) – REAL (KIND=nag_wp) array *Input*

On entry: the derivative of the solution at the current point t.

5: H – REAL (KIND=nag_wp) *Input*

On entry: the current step size.

6: D – REAL (KIND=nag_wp) *Input*

On entry: the parameter d which depends on the integration method.

7: ML – INTEGER *Input*

8: MU – INTEGER *Input*

On entry: the number of subdiagonals and superdiagonals respectively in the band.

9: P(ML + MU + 1,NEQ) – REAL (KIND=nag_wp) array *Input/Output*

On entry: is set to zero.

On exit: elements of the Jacobian matrix $\dfrac{\partial r}{\partial y}$ stored as specified by the following pseudocode

```
DO 20 I = 1, NEQ
J1 = MAX(I-ML,1)
J2 = MIN(I+MU,NEQ)
DO 10 J = J1, J2
K = MIN(ML+1-I,0)+J
P(K,I) = δR/δY(I,J)
10 CONTINUE
20 CONTINUE
```

See also the routine document for F07BDF (DGBTRF).

Only nonzero elements of this array need be set, since it is preset to zero before the call to JAC.

JAC must either be a module subprogram USEd by, or declared as EXTERNAL in, the (sub)program from which D02NHF is called. Parameters denoted as *Input* must **not** be changed by this procedure.

16: WKJAC(NWKJAC) – REAL (KIND=nag_wp) array *Communication Array*
17: NWKJAC – INTEGER *Input*

On entry: the dimension of the array WKJAC as declared in the (sub)program from which D02NHF is called. This value must be the same as that supplied to the linear algebra setup routine D02NTF.

Constraint: NWKJAC $\geq (2m_L + m_U + 1) \times$ NEQ, where m_L and m_U are the number of subdiagonals and superdiagonals respectively in the band, defined by a call to D02NTF.

18: JACPVT(NJCPVT) – INTEGER array *Communication Array*
19: NJCPVT – INTEGER *Input*

On entry: the size of the array JACPVT. This value must be the same as that supplied to the linear algebra setup routine D02NTF.

Constraint: NJCPVT $\geq$ NEQ.

20: MONITR – SUBROUTINE, supplied by the NAG Library or the user. *External Procedure*

MONITR performs tasks requested by you. If this option is not required, then the actual argument for MONITR must be the dummy routine D02NBY. (D02NBY is included in the NAG Library.)

The specification of MONITR is:

```
SUBROUTINE MONITR (NEQ, LDYSAV, T, HLAST, HNEXT, Y, YDOT, YSAV, R,       &
                   ACOR, IMON, INLN, HMIN, HMAX, NQU)

INTEGER            NEQ, LDYSAV, IMON, INLN, NQU
REAL (KIND=nag_wp) T, HLAST, HNEXT, Y(NEQ), YDOT(NEQ),                   &
                   YSAV(LDYSAV,sdysav), R(NEQ), ACOR(NEQ,2), HMIN,       &
                   HMAX
```

where $sdysav$ is the numerical value of SDYSAV in the call of D02NHF.

1: NEQ – INTEGER *Input*

On entry: the number of equations being solved.

2: LDYSAV – INTEGER *Input*

On entry: an upper bound on the number of equations to be solved.

3: T – REAL (KIND=nag_wp) *Input*

On entry: the current value of the independent variable.

4: HLAST – REAL (KIND=nag_wp) *Input*

On entry: the last step size successfully used by the integrator.

5: HNEXT – REAL (KIND=nag_wp) *Input/Output*

On entry: the step size that the integrator proposes to take on the next step.

On exit: the next step size to be used. If this is different from the input value, then IMON must be set to 4.

6: Y(NEQ) – REAL (KIND=nag_wp) array *Input/Output*

On entry: y, the values of the dependent variables evaluated at t.

On exit: these values must not be changed unless IMON is set to 2.

7: YDOT(NEQ) – REAL (KIND=nag_wp) array *Input*

On entry: the time derivatives y' of the vector y.

8: YSAV(LDYSAV,$sdysav$) – REAL (KIND=nag_wp) array *Input*

On entry: workspace to enable you to carry out interpolation using either of the routines D02XJF or D02XKF.

9: R(NEQ) – REAL (KIND=nag_wp) array *Input*

On entry: if IMON $= 0$ and INLN $= 3$, then the first NEQ elements contain the residual vector $A(t,y)y' - g(t,y)$.

10: ACOR(NEQ,2) – REAL (KIND=nag_wp) array *Input*

On entry: with IMON $= 1$, ACOR$(i,1)$ contains the weight used for the ith equation when the norm is evaluated, and ACOR$(i,2)$ contains the estimated local error for the ith equation. The scaled local error at the end of a timestep may be obtained by calling the real function D02ZAF as follows:

```
IFAIL = 1
ERRLOC = D02ZAF(NEQ, ACOR(1,2), ACOR(1,1), IFAIL)
! CHECK IFAIL BEFORE PROCEEDING
```

11: IMON – INTEGER *Input/Output*

On entry: a flag indicating under what circumstances MONITR was called:

IMON $= -2$
: Entry from the integrator after IRES $= 4$ (set in RESID) caused an early termination (this facility could be used to locate discontinuities).

IMON $= -1$
: The current step failed repeatedly.

IMON $= 0$
: Entry after a call to the internal nonlinear equation solver (see INLN).

IMON $= 1$
: The current step was successful.

On exit: may be reset to determine subsequent action in D02NHF.

IMON $= -2$

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

IMON $= -1$

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

IMON $= 0$

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

IMON $= 1$

Normal exit to the integrator 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 solution Y, provided by MONITR, will be used for the initial conditions.

IMON $= 3$

Try to continue with the same step size and order as was to be used before the call to MONITR. 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.

12: INLN – INTEGER *Output*

On exit: the action to be taken by the internal nonlinear equation solver when MONITR is exited with IMON $= 0$. By setting INLN $= 3$ and returning to the integrator, the residual vector is evaluated and placed in the array R, and then MONITR is called again. At present this is the only option available: INLN must not be set to any other value.

13: HMIN – REAL (KIND=nag_wp) *Input/Output*

On entry: the minimum step size to be taken on the next step.

On exit: the minimum step size to be used. If this is different from the input value, then IMON must be set to 3 or 4.

14: HMAX – REAL (KIND=nag_wp) *Input/Output*

On entry: the maximum step size to be taken on the next step.

On exit: the maximum step size to be used. If this is different from the input value, then IMON must be set to 3 or 4. If HMAX is set to zero, no limit is assumed.

15: NQU – INTEGER *Input*

On entry: the order of the integrator used on the last step. This is supplied to enable you to carry out interpolation using either of the routines D02XJF or D02XKF.

MONITR must either be a module subprogram USEd by, or declared as EXTERNAL in, the (sub)program from which D02NHF is called. Parameters denoted as *Input* must **not** be changed by this procedure.

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

On 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 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..

22: ITASK – INTEGER *Input*

On entry: the task to be performed by the integrator.

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 before the first call to the integrator, or specified in the optional input routine before a continuation call. TCRIT 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. TCRIT must be specified as 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). 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 D02NHF is recalled with a different value of ITASK (and TOUT altered), then the initialization procedure is repeated, possibly leading to different initial conditions.

Constraint: $1 \leq$ ITASK ≤ 6.

23: ITRACE – INTEGER *Input*

On 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.

24: 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 Section 3.3 in the Essential Introduction for details.

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, because for this routine the values of the output parameters may be useful even if IFAIL $\neq 0$ on exit, the recommended value is -1. **When the value -1 or 1 is used it is essential to test the value of IFAIL on exit.**

On exit: IFAIL $= 0$ unless the routine detects an error or a warning has been flagged (see Section 6).

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 $= 1$

An illegal input was detected on entry, or after an internal call to MONITR. If ITRACE > -1, then 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 $\mathrm{Y}(1), \mathrm{Y}(2), \ldots, \mathrm{Y}(\mathrm{NEQ})$ 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 ($\mathrm{ATOL}(i) = 0.0$) was requested on a variable (the ith) which has now vanished. The integration was successful as far as T.

IFAIL $= 7$

RESID set its 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 $= 11$

RESID signalled the integrator to halt the integration and return (IRES $= 2$). Integration was successful as far as T.

IFAIL $= 12$

MONITR 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 D02NHF is unable to start the integration.

IFAIL $= 15$

The linear algebra setup routine D02NTF was not called before the call to D02NHF.

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. For D02NHF the cost is proportional to NEQ $\times$ (ML + MU + 1)^2, though for problems which are only mildly nonlinear the cost may be dominated by factors proportional to NEQ $\times$ (ML + MU + 1) except for very large problems.

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

9 Example

This example solves the well-known stiff Robertson problem written as an implicit differential system and in implicit form

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

exploiting the fact that we can show that $(a+b+c)' = 0$ for all time. Integration is over the range $[0, 10.0]$ with initial conditions $a = 1.0$ and $b = c = 0.0$ using scalar relative error control and vector absolute error control (ITOL $= 2$). We integrate using a BDF method (setup routine D02NVF) and a modified Newton method. The Jacobian is calculated numerically and we employ a default monitor, dummy routine D02NBY. We perform a normal integration (ITASK $= 1$) to obtain the value at TOUT $= 10.0$ by integrating past TOUT and interpolating. We also illustrate the use of ITASK $= 6$ to calculate initial values of y and y' and then return without integrating further.

9.1 Program Text

```
!   D02NHF Example Program Text
!   Mark 24 Release. NAG Copyright 2012.

    Module d02nhfe_mod

!     D02NHF Example Program Module:
!            Parameters and User-defined Routines

!     .. Use Statements ..
      Use nag_library, Only: nag_wp
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Real (Kind=nag_wp), Parameter        :: alpha = 0.04_nag_wp
      Real (Kind=nag_wp), Parameter        :: beta = 1.0E4_nag_wp
      Real (Kind=nag_wp), Parameter        :: gamma = 3.0E7_nag_wp
      Integer, Parameter                   :: iset = 1, itrace = 0, ml = 1,    &
                                              mu = 2, neq = 3, nin = 5
      Integer, Parameter                   :: njcpvt = neq
      Integer, Parameter                   :: nout = 6
      Integer, Parameter                   :: nrw = 50 + 4*neq
      Integer, Parameter                   :: nwkjac = neq*(2*ml+mu+1)
      Integer, Parameter                   :: ldysav = neq
    Contains
      Subroutine resid(neq,t,y,ydot,r,ires)

!       .. Scalar Arguments ..
        Real (Kind=nag_wp), Intent (In)      :: t
        Integer, Intent (Inout)              :: ires
        Integer, Intent (In)                 :: neq
!       .. Array Arguments ..
        Real (Kind=nag_wp), Intent (Out)     :: r(neq)
        Real (Kind=nag_wp), Intent (In)      :: y(neq), ydot(neq)
!       .. Executable Statements ..
        r(1) = -ydot(1) - ydot(2) - ydot(3)
        r(2) = -ydot(2)
        r(3) = -ydot(3)
        If (ires==1) Then
          r(1) = r(1)
          r(2) = alpha*y(1) - beta*y(2)*y(3) - gamma*y(2)*y(2) + r(2)
          r(3) = gamma*y(2)*y(2) + r(3)
        End If
        Return
      End Subroutine resid
    End Module d02nhfe_mod

    Program d02nhfe
```

```
!     D02NHF Example Main Program

!     .. Use Statements ..
      Use nag_library, Only: d02nby, d02nhf, d02nhz, d02ntf, d02nvf, d02nyf,   &
                             nag_wp, x04abf
      Use d02nhfe_mod, Only: iset, itrace, ldysav, ml, mu, neq, nin, njcpvt,   &
                             nout, nrw, nwkjac, resid
!     .. Implicit None Statement ..
      Implicit None
!     .. Local Scalars ..
      Real (Kind=nag_wp)                  :: h, h0, hmax, hmin, hu, t, tcrit, &
                                             tcur, tolsf, tout, tout1
      Integer                             :: i, ifail, imxer, itask, itol,    &
                                             maxord, maxstp, mxhnil, niter,   &
                                             nje, nq, nqu, nre, nst, outchn,  &
                                             sdysav
      Logical                             :: petzld
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable     :: atol(:), rtol(:), rwork(:),      &
                                             wkjac(:), y(:), ydot(:), ysav(:,:)
      Real (Kind=nag_wp)                  :: con(6)
      Integer                             :: inform(23)
      Integer, Allocatable                :: jacpvt(:)
      Logical, Allocatable                :: algequ(:)
      Logical                             :: lderiv(2)
!     .. Executable Statements ..
      Write (nout,*) 'D02NHF Example Program Results'
      Flush (nout)

!     Skip heading in data file
      Read (nin,*)
      Read (nin,*) maxord, maxstp, mxhnil
      sdysav = maxord + 1
      Allocate (atol(neq),rtol(neq),rwork(nrw),wkjac(nwkjac),y(neq),ydot(neq), &
        ysav(ldysav,sdysav),jacpvt(njcpvt),algequ(neq))

      outchn = nout
      Call x04abf(iset,outchn)

!     Set itask=6 to provide initial estimates of solution and its
!     time derivative. Default values for the array con are used.
!     Use the B.D.F. formulae with a Newton method.
!     Employ scalar relative tolerance and vector absolute tolerance.
!     The Jacobian is evaluated internally.
!     monitr subroutine replaced by NAG dummy routine D02NBY.

      Read (nin,*) hmin, hmax, h0, tcrit
      Read (nin,*) petzld
      Read (nin,*) t, tout1
      Read (nin,*) y(1:neq)
      Read (nin,*) lderiv(1:2)
      Read (nin,*) itol
      Read (nin,*) rtol(1)
      Read (nin,*) atol(1:neq)
      itask = 6
      tout = tout1
      con(1:6) = 0.0_nag_wp

      ifail = 0
      Call d02nvf(neq,sdysav,maxord,'Newton',petzld,con,tcrit,hmin,hmax,h0, &
        maxstp,mxhnil,'Average-L2',rwork,ifail)

      ifail = 0
      Call d02ntf(neq,neq,'Numerical',ml,mu,nwkjac,njcpvt,rwork,ifail)

!     Hard fail on initial and subsequent tasks.
      ifail = 0
      Call d02nhf(neq,ldysav,t,tout,y,ydot,rwork,rtol,atol,itol,inform,resid, &
        ysav,sdysav,d02nhz,wkjac,nwkjac,jacpvt,njcpvt,d02nby,lderiv,itask, &
        itrace,ifail)
```

```
      Write (nout,*)
      Write (nout,99999) ' Initial    Y : ', y(1:neq)
      Write (nout,99999) ' Initial YDOT : ', ydot(1:neq)
      Flush (nout)

!     Use these initial estimates and integrate to tout (overshoot and
!     interpolate)
      itask = 1
      tout = tout1

      ifail = 0
      Call d02nhf(neq,ldysav,t,tout,y,ydot,rwork,rtol,atol,itol,inform,resid, &
        ysav,sdysav,d02nhz,wkjac,nwkjac,jacpvt,njcpvt,d02nby,lderiv,itask, &
        itrace,ifail)

      Write (nout,99993)(i,i=1,neq)
      Write (nout,99998) tout, y(1:neq)

      ifail = 0
      Call d02nyf(neq,neq,hu,h,tcur,tolsf,rwork,nst,nre,nje,nqu,nq,niter, &
        imxer,algequ,inform,ifail)

      Write (nout,*)
      Write (nout,99997) hu, h, tcur
      Write (nout,99996) nst, nre, nje
      Write (nout,99995) nqu, nq, niter
      Write (nout,99994) ' Max err comp = ', imxer

99999 Format (1X,A,3(F11.4,2X))
99998 Format (1X,F8.3,3(F13.5,2X))
99997 Format (1X,' HUSED = ',E12.5,'  HNEXT = ',E12.5,'  TCUR = ',E12.5)
99996 Format (1X,' NST = ',I6,'    NRE = ',I6,'    NJE = ',I6)
99995 Format (1X,' NQU = ',I6,'    NQ  = ',I6,'  NITER = ',I6)
99994 Format (1X,A,I4)
99993 Format (/1X,'    X ',3('         Y(',I1,')  '))
    End Program d02nhfe
```

9.2 Program Data

```
D02NHF Example Program Data
   5  200  5                    : maxord, maxstp, mxhnil
   1.0E-10  10.0  1.0E-4  0.0   : hmin, hmax, h0, tcrit
   .FALSE.                     : petzld
   0.0  10.0                   : t, tout
   1.0  0.0  0.0               : y
   .FALSE. .FALSE.             : lderiv
   2                           : itol
   1.0E-4                      : rtol
   1.0E-6 1.0E-7 1.0E-6        : atol
```

9.3 Program Results

```
 D02NHF Example Program Results
 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

  Initial    Y :      1.0000       0.0000       0.0000
  Initial YDOT :     -0.0400       0.0400       0.0000
 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

     X          Y(1)          Y(2)          Y(3)
   10.000      0.84135       0.00002       0.15863
```

```
 HUSED =  0.91752E+00  HNEXT =  0.91752E+00  TCUR =  0.10885E+02
 NST =     51    NRE =    118    NJE =     14
 NQU =      4    NQ  =      4  NITER =     68
 Max err comp =    3
```
