NAG Library Routine Document

D02NMF

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

D02NMF is a reverse communication routine for integrating stiff systems of explicit ordinary differential equations.

2 Specification

```
SUBROUTINE D02NMF (NEQ, LDYSAV, T, TOUT, Y, YDOT, RWORK, RTOL, ATOL,        &
                   ITOL, INFORM, YSAV, SDYSAV, WKJAC, NWKJAC, JACPVT,       &
                   NJCPVT, IMON, INLN, IRES, IREVCM, ITASK, ITRACE,         &
                   IFAIL)
INTEGER            NEQ, LDYSAV, ITOL, INFORM(23), SDYSAV, NWKJAC,           &
                   JACPVT(NJCPVT), NJCPVT, IMON, INLN, IRES, IREVCM,        &
                   ITASK, ITRACE, IFAIL
REAL (KIND=nag_wp) T, TOUT, Y(NEQ), YDOT(NEQ), RWORK(50+4*NEQ),             &
                   RTOL(*), ATOL(*), YSAV(LDYSAV,SDYSAV),                   &
                   WKJAC(NWKJAC)
```

3 Description

D02NMF is a general purpose routine for integrating the initial value problem for a stiff system of explicit ordinary differential equations,

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

An outline of a typical calling program is given below:

```
!     Declarations

      call linear algebra setup routine
      call integrator setup routine
      IREVCM=0
 1000 CALL D02NMF(NEQ, LDYSAV, T, TOUT, Y, YDOT, RWORK, RTOL,            &
                  ATOL, ITOL, INFORM, YSAVE, SDYSAV, WKJAC, NWKJAC,     &
        JACPVT, NJCPVT, IMON, INLN, IRES, IREVCM, ITASK,      &
    ITRACE, IFAIL)
        IF (IREVCM.GT.0) 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(IRECVM.EQ.1.OR.IREVCM.GE.3.AND.IREVCM.LE.5) THEN
            evaluate the derivative                                     (iii)
          ELSE IF(IREVCM.EQ.10) THEN
            indicates an unsuccessful step
          ENDIF
          GO TO 1000
        ENDIF

!     post processing (optional linear algebra diagnostic call
!     (sparse case only), optional integrator diagnostic call)

      STOP
      END
```

There are three major operations that may be required of the calling (sub)program on an intermeditate return (IREVCM $\neq 0$) from D02NMF; 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 JCEVAL = 'F' if using sparse matrix linear algebra) in a call to the linear algebra setup routine (see JCEVAL in D02NSF). 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 $\frac{d}{dy'}(\ ) = (hd)\frac{d}{dy}(\ )$.

The system of nonlinear equations that is solved has the form

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

but is solved in the form

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

where the function r is defined by

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

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

$$\frac{\partial r_i}{\partial y_j} = 1 - (hd)\frac{\partial g_i}{\partial y_j} \quad \text{if } i = j,$$

$$\frac{\partial r_i}{\partial y_j} = \quad -(hd)\frac{\partial g_i}{\partial y_j} \quad \text{otherwise,}$$

where t, h and d are located in RWORK(19), RWORK(16) and RWORK(20) respectively and the array Y contains the current values of the dependent variables. Only the nonzero 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, 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 timestep may be obtained by calling real function D02ZAF as follows:

```
      IFAIL = 1
      ERRLOC = D02ZAF(NEQ,RWORK(51+NEQ),RWORK(51),IFAIL)
!     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 D02NBF for details on what optional input can be made.

If Y is changed, then IMON must be set to 2 before return to D02NMF. If either of the values of HMIN or HMAX are changed, then IMON must be set ≥ 3 before return to D02NMF. If HNEXT is changed, then IMON must be set to 4 before return to D02NMF.

In addition you can force D02NMF to evaluate the residual vector

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

by setting IMON $= 0$ and INLN $= 3$ and then returning to D02NMF; on return to this monitoring operation the residual vector will be stored in RWORK$(50 + 2 \times \text{NEQ} + i)$, for $i = 1, 2, \ldots, \text{NEQ}$.

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

(iii) Evaluate the Derivative

This operation must evaluate the derivative vector for the explicit ordinary differential equation system defined by

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

where t is located in RWORK(19).

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

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 differential equations to be solved.

Constraint: NEQ ≥ 1.

2: LDYSAV – INTEGER *Input*

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

Constraint: LDYSAV $\geq$ NEQ.

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

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

4: TOUT – REAL (KIND=nag_wp) *Input*

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.

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

On intermediate re-entry: must be set to the derivatives as defined under the description of IREVCM.

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

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

On initial entry: must be the same array as used by one of the method setup routines D02MVF, D02NVF or D02NWF, and by one of the storage setup routines D02NTF, D02NUF or D02NVF. The contents of RWORK must not be changed between any call to a setup routine and the first call to D02NMF.

On intermediate re-entry: elements of RWORK must be set to quantities as defined under the description of IREVCM.

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

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 initial entry: the relative local error tolerance.

Constraint: RTOL(i) ≥ 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 initial entry: the absolute local error tolerance.

Constraint: ATOL(i) ≥ 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 D02NMF 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) $\times \lvert y_i \rvert +$ ATOL(1)
2	scalar	vector	RTOL(1) $\times \lvert y_i \rvert +$ ATOL(i)
3	vector	scalar	RTOL(i) $\times \lvert y_i \rvert +$ ATOL(1)
4	vector	vector	RTOL(i) $\times \lvert y_i \rvert +$ 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: YSAV(LDYSAV, SDYSAV) – REAL (KIND=nag_wp) array *Communication Array*

13: SDYSAV – INTEGER *Input*

On initial entry: the second dimension of the array YSAV as declared in the (sub)program from which D02NMF is called. An appropriate value for SDYSAV is described in the specifications of the integrator setup routines D02NVF and D02NWF. This value must be the same as that supplied to the integrator setup routine.

14: WKJAC(NWKJAC) – REAL (KIND=nag_wp) 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 D02NMF 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 D02NMF 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 D02NMF and the MONITR operation (see Section 3). With IREVCM $= 9$, IMON contains a flag indicating under what circumstances the return from D02NMF occurred:

IMON $= -2$
: Exit from D02NMF after IRES $= 4$ caused an early termination (this facility could be used to locate discontinuities).

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

IMON $= 0$
: Exit from D02NMF 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 D02NMF.

IMON $= -2$
: Integration is to be halted. A return will be made from D02NMF to the calling (sub)program with IFAIL $= 12$.

IMON $= -1$
Allow D02NMF 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 D02NMF 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 the MONITR operation (see Section 3), 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 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 D02NMF, the residual vector is evaluated and placed in RWORK$(50 + 2 \times$ NEQ $+ i)$, 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 or 5, IRES contains the value 1.

On intermediate re-entry: should be unchanged unless one of the following actions is required of D02NMF in which case IRES should be set accordingly.

IRES $= 2$
Indicates to D02NMF that control should be passed back immediately to the calling (sub)program with the error indicator set to IFAIL $= 11$.

IRES $= 3$
Indicates to D02NMF 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 D02NMF returns to the calling (sub)program with the error indicator set to IFAIL $= 7$.

IRES $= 4$
Indicates to D02NMF 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. The possible exit values of IREVCM are 1, 3, 4, 5, 8, 9 or 10, which should be interpreted as follows:

IREVCM = 1, 3, 4 and 5
: Indicates that an FCN operation (see Section 3) is required: $y' = g(t, y)$ must be supplied, where Y(i) is located in y_i, for $i = 1, 2, \ldots,$ NEQ.

For IREVCM = 1 or 3, y'_i should be placed in location RWORK($50 + 2 \times$ NEQ $+ i$), for $i = 1, 2, \ldots,$ NEQ.

For IREVCM = 4, y'_i should be placed in location RWORK($50 +$ NEQ $+ i$), for $i = 1, 2, \ldots,$ NEQ.

For IREVCM = 5, y'_i should be placed in location YDOT(i), for $i = 1, 2, \ldots,$ NEQ.

IREVCM = 8
: Indicates that a JAC operation (see Section 3) is required: the Jacobian matrix must be supplied.

If full matrix linear algebra is being used, then the (i, j)th element of the Jacobian must be stored in WKJAC($(j - 1) \times$ 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. If IPLACE = 1, then the (i, j)th element of the Jacobian must be stored in RWORK($50 + 2 \times$ NEQ $+ i$); otherwise it must be stored in RWORK($50 +$ NEQ $+ i$).

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 independent variable t, located in RWORK(19). No values must be changed before re-entering D02NMF; this facility enables you to determine the number of unsuccessful steps.

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

Constraint: IREVCM = 0, 1, 3, 4, 5, 8, 9 or 10.

22: ITASK – INTEGER *Input*

On initial 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 (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 $= 4$.

Constraint: $1 \leq \text{ITASK} \leq 5$.

23: 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.

24: IFAIL – INTEGER *Input/Output*

On initial 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 final 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 $= 1$

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 $\mathrm{Y}(1), \mathrm{Y}(2), \ldots, \mathrm{Y}(\mathrm{NEQ})$ 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 $= 7$

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

IFAIL $= 8$

Not used for this integrator.

IFAIL $= 9$

A singular Jacobian $\dfrac{\partial r}{\partial y}$ has been encountered. This error exit is unlikely to be taken when solving explicit ordinary differential equations. 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 FCN 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 D02NMF is unable to start the integration.

IFAIL $= -99$

An unexpected error has been triggered by this routine. Please contact NAG.

See Section 3.8 in the Essential Introduction for further information.

IFAIL $= -399$

Your licence key may have expired or may not have been installed correctly.

See Section 3.7 in the Essential Introduction for further information.

IFAIL $= -999$

Dynamic memory allocation failed.

See Section 3.6 in the Essential Introduction for further information.

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 Parallelism and Performance

D02NMF is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

D02NMF makes calls to BLAS and/or LAPACK routines, which may be threaded within the vendor library used by this implementation. Consult the documentation for the vendor library for further information.

Please consult the X06 Chapter Introduction for information on how to control and interrogate the OpenMP environment used within this routine. Please also consult the Users' Note for your implementation for any additional implementation-specific information.

9 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 9 of the documents for D02NBF (full matrix), D02NCF (banded matrix) or D02NDF (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 $\dfrac{\partial g}{\partial y}$ has complex eigenvalues near the imaginary axis for some part of the integration, you should try the BLEND option (setup routine D02NWF).

10 Example

This example solves the well-known stiff Robertson problem

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

over the range $[0, 10]$ with initial conditions $a = 1.0$ and $b = c = 0.0$ and with scalar error control (ITOL $= 1$). The integration proceeds until TOUT $= 10.0$ is passed, providing C^1 interpolation at intervals of 2.0 through a MONITR operation. The integration method used is the BDF method (setup

routine D02NVF) with a modified Newton method. The Jacobian is a full matrix, which is specified using the setup routine D02NSF; this Jacobian is to be calculated numerically.

10.1 Program Text

```
    Program d02nmfe

!     D02NMF Example Program Text

!     Mark 25 Release. NAG Copyright 2014.

!     .. Use Statements ..
      Use nag_library, Only: d02nmf, d02nsf, d02nvf, d02nyf, d02xkf, nag_wp,   &
                             x04abf
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Real (Kind=nag_wp), Parameter    :: tstep = 2.0E0_nag_wp
      Integer, Parameter               :: iset = 1, neq = 3, nin = 5,          &
                                          njcpvt = 1, nout = 6
      Integer, Parameter               :: nrw = 50 + 4*neq
      Integer, Parameter               :: nwkjac = neq*(neq+1)
      Integer, Parameter               :: sdysav = 6
      Integer, Parameter               :: ldysav = neq
!     .. Local Scalars ..
      Real (Kind=nag_wp)               :: h, h0, hlast, hmax, hmin, hnext, hu, &
                                          t, tc, tcrit, tcur, tolsf, tout, xout
      Integer                          :: i, ifail, iflag, imon, imxer, inln,  &
                                          ires, irevcm, itask, itol, itrace,   &
                                          lacor1, lacor2, lacor3, lacorb,      &
                                          lsavr1, lsavr2, lsavr3, lsavrb,      &
                                          maxord, maxstp, mxhnil, niter, nje,  &
                                          nq, nqu, nre, nst, outchn
      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(:)
!     .. Intrinsic Procedures ..
      Intrinsic                        :: int
!     .. Executable Statements ..
      Write (nout,*) 'D02NMF Example Program Results'
      Write (nout,*)
!     Skip heading in data file
      Read (nin,*)
!     neq: number of differential equations
      Allocate (atol(neq),rtol(neq),rwork(nrw),wkjac(nwkjac),y(neq),ydot(neq), &
        ysav(ldysav,sdysav),jacpvt(njcpvt),algequ(neq))


!     Integrate to tout by overshooting tout (itask=1) using B.D.F.
!     formulae with a Newton method. Default values for the array con
!     are used. Employ scalar tolerances and the Jacobian is evaluated
!     internally. On the reverse communication call equivalent to the
!     monitr call in forward communication routines carry out
!     interpolation using D02XKF.

      Read (nin,*) maxord, maxstp, mxhnil
      Read (nin,*) hmin, hmax, h0, tcrit
      Read (nin,*) petzld
      Read (nin,*) t, tout
      Read (nin,*) itol
      Read (nin,*) y(1:neq)
      Read (nin,*) rtol(1), atol(1)


      outchn = nout
```

```
      Call x04abf(iset,outchn)
      itask = 1
      xout = tstep
      con(1:6) = 0.0E0_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 d02nsf(neq,neq,'Numerical',nwkjac,rwork,ifail)

      lacorb = 50 + neq
      lacor1 = lacorb + 1
      lacor2 = lacorb + 2
      lacor3 = lacorb + 3
      lsavrb = lacorb + neq
      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)

!     Soft fail and error messages only
      irevcm = 0
      itrace = 0

steps: Do
        ifail = 1
        Call d02nmf(neq,ldysav,t,tout,y,ydot,rwork,rtol,atol,itol,inform,ysav, &
          sdysav,wkjac,nwkjac,jacpvt,njcpvt,imon,inln,ires,irevcm,itask, &
          itrace,ifail)

        Select Case (irevcm)
        Case (0)
          Exit steps
        Case (1,3)
!         Equivalent to fcn evaluation in forward communication
!         routines
          rwork(lsavr1) = -0.04E0_nag_wp*y(1) + 1.0E4_nag_wp*y(2)*y(3)
          rwork(lsavr2) = 0.04E0_nag_wp*y(1) - 1.0E4_nag_wp*y(2)*y(3) - &
            3.0E7_nag_wp*y(2)*y(2)
          rwork(lsavr3) = 3.0E7_nag_wp*y(2)*y(2)
        Case (4)
!         Equivalent to fcn evaluation in forward communication
!         routines
          rwork(lacor1) = -0.04E0_nag_wp*y(1) + 1.0E4_nag_wp*y(2)*y(3)
          rwork(lacor2) = 0.04E0_nag_wp*y(1) - 1.0E4_nag_wp*y(2)*y(3) - &
            3.0E7_nag_wp*y(2)*y(2)
          rwork(lacor3) = 3.0E7_nag_wp*y(2)*y(2)
        Case (5)
!         Equivalent to fcn evaluation in forward communication
!         routines
          ydot(1) = -0.04E0_nag_wp*y(1) + 1.0E4_nag_wp*y(2)*y(3)
          ydot(2) = 0.04E0_nag_wp*y(1) - 1.0E4_nag_wp*y(2)*y(3) - &
            3.0E7_nag_wp*y(2)*y(2)
          ydot(3) = 3.0E7_nag_wp*y(2)*y(2)
        Case (9)
!         Equivalent to monitr call in forward communication routines
          If (imon==1) Then
            tc = rwork(19)
            hlast = rwork(15)
            hnext = rwork(16)
            nqu = int(rwork(10))
interp:     Do
              If (tc-hlast<xout .And. xout<=tc) Then
                iflag = 1

                Call d02xkf(xout,rwork(lsavr1),neq,ysav,ldysav,sdysav, &
                  rwork(lacor1),neq,tc,nqu,hlast,hnext,iflag)
```

```
              If (iflag/=0) Then
                imon = -2
                Exit interp
              Else
                Write (nout,99999) xout, (rwork(lsavrb+i),i=1,neq)
                xout = xout + tstep
                If (xout>tout) Exit interp
              End If
            Else
              Exit interp
            End If
          End Do interp
        End If
      Case (2,6:8)
        Write (nout,*)
        Write (nout,99994) 'Illegal value of IREVCM = ', irevcm
        Exit steps
      End Select
    End Do steps
    If (ifail==0) Then

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

      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
      Write (nout,*)
    Else
      Write (nout,*)
      Write (nout,99998) 'Exit D02NMF with IFAIL = ', ifail, '  and T = ', t
    End If

99999 Format (1X,F8.3,3(F13.5,2X))
99998 Format (1X,A,I2,A,E12.5)
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)
    End Program d02nmfe
```

10.2 Program Data

```
D02NMF Example Program Data
   5  200  5                 : maxord, maxstp, mxhnil
   1.0E-10  10.0  0.0  0.0    : hmin, hmax, h0, tcrit
   .FALSE.                   : petzld
   0.0  10.0                 : t, tout
   1                         : itol
   1.0  0.0  0.0             : y
   1.0E-4  1.0E-7            : rtol, atol
```

10.3 Program Results

```
 D02NMF Example Program Results

     X          Y(1)          Y(2)          Y(3)
    0.000      1.00000       0.00000       0.00000
    2.000      0.94161       0.00003       0.05836
    4.000      0.90551       0.00002       0.09446
    6.000      0.87926       0.00002       0.12072
    8.000      0.85854       0.00002       0.14144
   10.000      0.84135       0.00002       0.15863
```

```
HUSED =  0.90178E+00  HNEXT =  0.90178E+00  TCUR =  0.10766E+02
NST =     55    NRE =    128    NJE =     16
NQU =      4    NQ  =      4  NITER =     78
Max err comp =    3
```