NAG Fortran Library Routine Document

D02NBF

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

D02NBF is a forward communication routine for integrating stiff systems of explicit ordinary differential equations when the Jacobian is a full matrix.

2 Specification

```
 SUBROUTINE D02NBF(NEQ, NEQMAX, T, TOUT, Y, YDOT, RWORK, RTOL, ATOL,
1                  ITOL, INFORM, FCN, YSAVE, NY2DIM, JAC, WKJAC, NWKJAC,
2                  MONITR, ITASK, ITRACE, IFAIL)
 INTEGER           NEQ, NEQMAX, ITOL, INFORM(23), NY2DIM, NWKJAC, ITASK,
1                  ITRACE, IFAIL
 real              T, TOUT, Y(NEQMAX), YDOT(NEQMAX), RWORK(50+4*NEQMAX),
1                  RTOL(*), ATOL(*), YSAVE(NEQMAX,NY2DIM), WKJAC(NWKJAC)
 EXTERNAL          FCN, JAC, MONITR
```

3 Description

D02NBF is a general purpose routine for integrating the initial value problem for a stiff system of explicit ordinary differential equations,

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

It is designed specifically for the case where the Jacobian $\frac{\partial g}{\partial y}$ is a full matrix.

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 program calling D02NBF is given below. It calls the full matrix linear algebra setup routine D02NSF, and the Backward Differentiation Formula (BDF) integrator setup routine D02NVF, and its diagnostic counterpart D02NYF.

```
C
C     declarations
C
      EXTERNAL FCN, JAC, MONITR
      .
      .
      .
      IFAIL = 0
      CALL D02NVF(...,IFAIL)
      CALL D02NSF(NEQ, NEQMAX, JCEVAL, NWKJAC, RWORK, IFAIL)
      IFAIL = -1
      CALL D02NBF(NEQ, NEQMAX, T, TOUT, Y, YDOT, RWORK, RTOL,
     +  ATOL, ITOL, INFORM, FCN, YSAVE, NY2DIM, JAC, WKJAC, NWKJAC,
     +  MONITR, ITASK, ITRACE, IFAIL)
      IF (IFAIL.EQ.1. OR. IFAIL.GE.14) STOP
      IFAIL = 0
      CALL D02NYF(...)
      .
      .
      .
      STOP
      END
```

The linear algebra setup routine D02NSF and one of the integrator setup routines, D02NVF or D02NWF, must be called prior to the call of D02NBF. The integrator diagnostic routine D02NYF may be called after the call to D02NBF. There is also a routine, D02NZF, designed to permit the user to change step size on a continuation call to D02NBF without restarting the integration process.

4 References

None.

5 Parameters

1: NEQ – INTEGER *Input*

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

Constraint: NEQ $\geq$ 1.

2: NEQMAX – INTEGER *Input*

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

Constraint: NEQMAX $\geq$ NEQ.

3: T – ***real*** *Input/Output*

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

4: TOUT – ***real*** *Input*

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.

5: Y(NEQMAX) – ***real*** 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(NEQMAX) – ***real*** array *Output*

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

7: RWORK(50+4*NEQMAX) – ***real*** array *Workspace*

8: RTOL(*) – ***real*** array *Input*

Note: the dimension of the array RTOL must be at least 1 or NEQ (see ITOL).

On entry: the relative local error tolerance.

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

9: ATOL(*) – ***real*** array *Input*

Note: the dimension of the array ATOL must be at least 1 or NEQ (see ITOL).

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 D02NBF 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	$\mathrm{RTOL}(1) \times \lvert y_i \rvert + \mathrm{ATOL}(1)$
2	scalar	vector	$\mathrm{RTOL}(1) \times \lvert y_i \rvert + \mathrm{ATOL}(i)$
3	vector	scalar	$\mathrm{RTOL}(i) \times \lvert y_i \rvert + \mathrm{ATOL}(1)$
4	vector	vector	$\mathrm{RTOL}(i) \times \lvert y_i \rvert + \mathrm{ATOL}(i)$

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: $1 \leq \mathrm{ITOL} \leq 4$.

11: INFORM(23) – INTEGER array *Workspace*

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

FCN must evaluate the derivative vector for the explicit ordinary differential equation system, defined by $y' = g(t, y)$.

Its specification is:

```
SUBROUTINE FCN(NEQ, T, Y, F, IRES)
INTEGER          NEQ, IRES
real             T, Y(NEQ), F(NEQ)
```

1: NEQ – INTEGER *Input*

On entry: the number of differential equations being solved.

2: T – ***real*** *Input*

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

3: Y(NEQ) – ***real*** array *Input*

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

4: F(NEQ) – ***real*** array *Output*

On exit: the value y_i', given by $y_i' = g_i(t, y)$, for $i = 1, 2, \ldots, \mathrm{NEQ}$.

5: IRES – INTEGER *Input/Output*

On entry: IRES = 1.

On exit: the user may set IRES as follows to indicate certain conditions in FCN to the integrator:

IRES = 1

Indicates a normal return from FCN, that is IRES is not altered by the user and integration continues.

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 the MONITR routine immediately with parameter IMON = −2.

FCN must be declared as EXTERNAL in the (sub)program from which D02NBF is called. Parameters denoted as *Input* must **not** be changed by this procedure.

13: YSAVE(NEQMAX,NY2DIM) – ***real*** array *Workspace*

14: NY2DIM – INTEGER *Input*

On entry: the second dimension of the array YSAVE as declared in the (sub)program from which D02NBF is called. An appropriate value for NY2DIM is described in the specification of the integrator setup routines D02NVF and D02NWF. This value must be the same as that supplied to the integrator setup routine.

15: JAC – SUBROUTINE, supplied by 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 D02NBZ. (D02NBZ is included in the NAG Fortran Library and so need not be supplied by the user. Its name may be implementation dependent: see the Users' Note for your implementation for details.) The user indicates to the integrator whether this option is to be used by setting the parameter JCEVAL appropriately in a call to the linear algebra setup routine D02NSF.

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 $\frac{d}{dy'}() = \frac{1}{(hd)}\frac{d}{dy}()$. The system of nonlinear equations that is solved has the form

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

but it 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 $\frac{\partial r}{\partial y}$ that the user must supply in the routine JAC as follows:

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

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

Its specification is:

```
SUBROUTINE JAC(NEQ, T, Y, H, D, P)
INTEGER        NEQ
real           T, Y(NEQ), H, D, P(NEQ,NEQ)
```

1: NEQ – INTEGER *Input*

On entry: the number of differential equations being solved.

2: T – ***real*** *Input*

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

3: Y(NEQ) – ***real*** array *Input*

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

4: H – ***real*** *Input*

On entry: the current step size.

5: D – ***real*** *Input*

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

6: P(NEQ,NEQ) – ***real*** array *Output*

On exit: P(i, j) must contain $\frac{\partial r_i}{\partial y_j}$, for $i, j = 1, 2, \ldots,$ NEQ.

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

JAC must be declared as EXTERNAL in the (sub)program from which D02NBF is called. Parameters denoted as *Input* must **not** be changed by this procedure.

16: WKJAC(NWKJAC) – ***real*** array *Workspace*
17: NWKJAC – INTEGER *Input*

On entry: the dimension of the array WKJAC as declared in the (sub)program from which D02NBF is called. This value must be the same as that supplied to the linear algebra setup routine D02NSF.

Constraint: NWKJAC $\geq$ NEQMAX $\times$ (NEQMAX $+ 1$).

18: MONITR – SUBROUTINE, supplied by the user. *External Procedure*

MONITR performs tasks requested by the user. If this option is not required, then the actual argument for MONITR must be the dummy routine D02NBY. (D02NBY is included in the NAG Fortran Library and so need not be supplied by the user. Its name may be implementation dependent: see the Users' Note for your implementation for details.)

Its specification is:

```
      SUBROUTINE MONITR(NEQ, NEQMAX, T, HLAST, HNEXT, Y, YDOT, YSAVE, R,
     1                  ACOR, IMON, INLN, HMIN, HMAX, NQU)
      INTEGER           NEQ, NEQMAX, IMON, INLN, NQU
      real              T, HLAST, HNEXT, Y(NEQMAX), YDOT(NEQMAX),
     1                  YSAVE(NEQMAX,*), R(NEQMAX), ACOR(NEQMAX,2), HMIN,
     2                  HMAX
```

1: NEQ – INTEGER *Input*

On entry: the number of differential equations being solved.

2: NEQMAX – INTEGER *Input*

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

3: T – ***real*** *Input*

On entry: the current value of the independent variable.

4: HLAST – ***real*** *Input*

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

5: HNEXT – ***real*** *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(NEQMAX) – ***real*** array *Input/Output*

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

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

7: YDOT(NEQMAX) – ***real*** array *Input*

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

8: YSAVE(NEQMAX,*) – ***real*** array *Input*

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

9: R(NEQMAX) – ***real*** array *Input*

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

10: ACOR(NEQMAX,2) – ***real*** 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)
C     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 FCN) 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 below).

IMON $= 1$

The current step was successful.

On exit: IMON may be reset to determine subsequent action in D02NBF:

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 3 restarts unless IMON is set $\neq -1$ on exit.

IMON = 0

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

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 MONITR provided solution Y 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 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*** *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*** *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 the user to carry out interpolation using either of the routines D02XJF or D02XKF.

MONITR must be declared as EXTERNAL in the (sub)program from which D02NBF is called. Parameters denoted as *Input* must **not** be changed by this procedure.

19: ITASK – INTEGER *Input*

On 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 $t = \text{TOUT}$ (by overshooting and interpolating).

ITASK = 2

Take one step only and return.

ITASK = 3

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

ITASK = 4

Normal computation of output values of $y(t)$ at $t = \text{TOUT}$ but without overshooting $t = \text{TCRIT}$. 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 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.

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

20: 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. If ITRACE < -1, then -1 is assumed and similarly if ITRACE > 3, then 3 is assumed. If ITRACE $= -1$, no output is generated. If ITRACE $= 0$, only warning messages are printed on the current error message unit (see X04AAF). If ITRACE > 0, then 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.

21: IFAIL – INTEGER *Input/Output*

On entry: IFAIL must be set to 0, -1 or 1. Users who are unfamiliar with this parameter 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, 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.**

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 $\text{Y}(1), \text{Y}(2), \ldots, \text{Y}(\text{NEQ})$ 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 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 user-supplied subroutine FCN set its 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 $\frac{\partial r}{\partial y}$ has been encountered. This error exit is unlikely to be taken when solving explicit ordinary differential equations. The user 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 user-supplied subroutine FCN signalled the integrator to halt the integration and return (IRES $= 2$). Integration was successful as far as T.

IFAIL $= 12$

The user-supplied subroutine 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 the user is 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.

IFAIL $= 15$

The linear algebra setup routine D02NSF was not called prior to calling D02NBF.

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. Users 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 the user is 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 D02NBF the cost is proportional to NEQ^3, though for problems which are only mildly nonlinear the cost may be dominated by factors proportional to NEQ^2 except for very large problems.

In general the user is 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 $\frac{\partial g}{\partial y}$ has complex eigenvalues near the imaginary axis for some part of the integration, the user should try the BLEND option (setup routine D02NWF).

9 Example

We solve 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 $b = c = 0.0$ using scalar error contol (ITOL $= 1$) and computation of the solution at TOUT $= 10.0$ with TCRIT set to 10.0 (ITASK $= 4$). D02NBY is used for MONITR, we use a BDF integrator (setup routine D02NVF) and we select a modified Newton method. We illustrate the use of both a numerical and an analytical Jacobian.

9.1 Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     D02NBF Example Program Text
*     Mark 14 Revised.  NAG Copyright 1989.
*     .. Parameters ..
      INTEGER          NOUT
      PARAMETER        (NOUT=6)
      INTEGER          NEQ, NEQMAX, NRW, NINF, NWKJAC, MAXORD, NY2DIM,
     +                 MAXSTP, MXHNIL
      PARAMETER        (NEQ=3,NEQMAX=NEQ,NRW=50+4*NEQMAX,NINF=23,
     +                 NWKJAC=NEQMAX*(NEQMAX+1),MAXORD=5,
     +                 NY2DIM=MAXORD+1,MAXSTP=200,MXHNIL=5)
      real             H0, HMAX, HMIN
      PARAMETER        (H0=0.0e0,HMAX=10.0e0,HMIN=1.0e-10)
      LOGICAL          PETZLD
      PARAMETER        (PETZLD=.FALSE.)
*     .. Local Scalars ..
      real             H, HU, T, TCRIT, TCUR, TOLSF, TOUT
      INTEGER          I, IFAIL, IMXER, ITASK, ITOL, ITRACE, NITER, NJE,
     +                 NQ, NQU, NRE, NST
*     .. Local Arrays ..
      real             ATOL(NEQMAX), CONST(6), RTOL(NEQMAX), RWORK(NRW),
     +                 WKJAC(NWKJAC), Y(NEQMAX), YDOT(NEQMAX),
     +                 YSAVE(NEQMAX,NY2DIM)
      INTEGER          INFORM(NINF)
```

```
      LOGICAL          ALGEQU(NEQMAX)
*     .. External Subroutines ..
      EXTERNAL         D02NBF, D02NBY, D02NBZ, D02NSF, D02NVF, D02NYF,
     +                 FCN, JAC, X04ABF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'D02NBF Example Program Results'
      CALL X04ABF(1,NOUT)
*
*     First case. Integrate to TOUT without passing TOUT (set TCRIT to
*     TOUT and ITASK=4) using B.D.F formulae with a Newton method.
*     Default values for the array CONST are used. Employ scalar
*     tolerances and the Jacobian is evaluated internally.
*     MONITR subroutine replaced by NAG dummy routine D02NBY.
*
      T = 0.0e0
      TOUT = 10.0e0
      ITASK = 4
      Y(1) = 1.0e0
      Y(2) = 0.0e0
      Y(3) = 0.0e0
      ITOL = 1
      RTOL(1) = 1.0e-4
      ATOL(1) = 1.0e-7
      DO 20 I = 1, 6
         CONST(I) = 0.0e0
   20 CONTINUE
      TCRIT = TOUT
      IFAIL = 0
*
      CALL D02NVF(NEQMAX,NY2DIM,MAXORD,'Newton',PETZLD,CONST,TCRIT,HMIN,
     +            HMAX,H0,MAXSTP,MXHNIL,'Average-L2',RWORK,IFAIL)
      CALL D02NSF(NEQ,NEQMAX,'Numerical',NWKJAC,RWORK,IFAIL)
*
      WRITE (NOUT,*)
      WRITE (NOUT,*) '  Numerical Jacobian'
      WRITE (NOUT,*)
      WRITE (NOUT,*) '    X          Y(1)           Y(2)           Y(3)'
      WRITE (NOUT,99999) T, (Y(I),I=1,NEQ)
*
*     Soft fail and error messages only
      ITRACE = 0
      IFAIL = 1
*
      CALL D02NBF(NEQ,NEQMAX,T,TOUT,Y,YDOT,RWORK,RTOL,ATOL,ITOL,INFORM,
     +            FCN,YSAVE,NY2DIM,D02NBZ,WKJAC,NWKJAC,D02NBY,ITASK,
     +            ITRACE,IFAIL)
*
      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
         WRITE (NOUT,*)
      ELSE
         WRITE (NOUT,*)
         WRITE (NOUT,99998) 'Exit D02NBF with IFAIL = ', IFAIL,
     +     '  and T = ', T
      END IF
*
*     Second case. Integrate to TOUT without passing TOUT (set TCRIT to
*     TOUT and ITASK=4) using B.D.F formulae with a Newton method.
*     Default values for the array CONST are used. Employ scalar
```

```
*     tolerances and the Jacobian is evaluated by JAC.
*     MONITR subroutine replaced by NAG dummy routine D02NBY.
*
      T = 0.0e0
      Y(1) = 1.0e0
      Y(2) = 0.0e0
      Y(3) = 0.0e0
      IFAIL = 0
*
      CALL D02NVF(NEQMAX,NY2DIM,MAXORD,'Newton',PETZLD,CONST,TCRIT,HMIN,
     +            HMAX,H0,MAXSTP,MXHNIL,'Average-L2',RWORK,IFAIL)
*
      CALL D02NSF(NEQ,NEQMAX,'Analytical',NWKJAC,RWORK,IFAIL)
*
      WRITE (NOUT,*)
      WRITE (NOUT,*) '  Analytic Jacobian'
      WRITE (NOUT,*)
      WRITE (NOUT,*) '    X          Y(1)           Y(2)           Y(3)'
      WRITE (NOUT,99999) T, (Y(I),I=1,NEQ)
      IFAIL = 1
*
      CALL D02NBF(NEQ,NEQMAX,T,TOUT,Y,YDOT,RWORK,RTOL,ATOL,ITOL,INFORM,
     +            FCN,YSAVE,NY2DIM,JAC,WKJAC,NWKJAC,D02NBY,ITASK,ITRACE,
     +            IFAIL)
*
      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
         WRITE (NOUT,*)
      ELSE
         WRITE (NOUT,*)
         WRITE (NOUT,99998) 'Exit D02NBF with IFAIL = ', IFAIL,
     +     '  and T = ', T
      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)
      END
*
      SUBROUTINE FCN(NEQ,T,Y,R,IRES)
*     .. Scalar Arguments ..
      real              T
      INTEGER           IRES, NEQ
*     .. Array Arguments ..
      real              R(NEQ), Y(NEQ)
*     .. Executable Statements ..
      R(1) = -0.04e0*Y(1) + 1.0e4*Y(2)*Y(3)
      R(2) = 0.04e0*Y(1) - 1.0e4*Y(2)*Y(3) - 3.0e7*Y(2)*Y(2)
      R(3) = 3.0e7*Y(2)*Y(2)
      RETURN
      END
*
      SUBROUTINE JAC(NEQ,T,Y,H,D,P)
*     .. Scalar Arguments ..
      real              D, H, T
      INTEGER           NEQ
```

```
*     .. Array Arguments ..
      real              P(NEQ,NEQ), Y(NEQ)
*     .. Local Scalars ..
      real              HXD
*     .. Executable Statements ..
      HXD = H*D
      P(1,1) = 1.0e0 - HXD*(-0.04e0)
      P(1,2) = -HXD*(1.0e4*Y(3))
      P(1,3) = -HXD*(1.0e4*Y(2))
      P(2,1) = -HXD*(0.04e0)
      P(2,2) = 1.0e0 - HXD*(-1.0e4*Y(3)-6.0e7*Y(2))
      P(2,3) = -HXD*(-1.0e4*Y(2))
*     Do not need to set P(3,1) since Jacobian preset to zero
*     P(3,1) =       - HXD*(0.0E0)
      P(3,2) = -HXD*(6.0e7*Y(2))
      P(3,3) = 1.0e0 - HXD*(0.0e0)
      RETURN
      END
```

9.2 Program Data

None.

9.3 Program Results

```
 D02NBF Example Program Results

   Numerical Jacobian

     X          Y(1)           Y(2)           Y(3)
   0.000      1.00000        0.00000        0.00000
  10.000      0.84136        0.00002        0.15863

  HUSED =  0.51867E+00  HNEXT =  0.51867E+00  TCUR =  0.10000E+02
  NST =      55    NRE =    132    NJE =     17
  NQU =       3    NQ  =      3  NITER =     79
  Max Err Comp =    3


   Analytic Jacobian

     X          Y(1)           Y(2)           Y(3)
   0.000      1.00000        0.00000        0.00000
  10.000      0.84136        0.00002        0.15863

  HUSED =  0.51867E+00  HNEXT =  0.51867E+00  TCUR =  0.10000E+02
  NST =      55    NRE =     81    NJE =     17
  NQU =       3    NQ  =      3  NITER =     79
  Max Err Comp =    3
```