# NAG Library Routine Document

# D02MVF

**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

D02MVF is an integration method specific setup routine which must be called prior to linear algebra setup routines and integrators from the SPRINT suite of routines, if the DASSL implementation of Backward Differentiation Formulae (BDF) is to be used. Note that this method is also available, independent from the SPRINT suite, using D02NEF

## 2 Specification

```
SUBROUTINE D02MVF (NEQMAX, SDYSAV, MAXORD, CON, TCRIT, HMIN, HMAX, H0,     &
                   MAXSTP, MXHNIL, NORM, RWORK, IFAIL)

INTEGER             NEQMAX, SDYSAV, MAXORD, MAXSTP, MXHNIL, IFAIL
REAL (KIND=nag_wp) CON(3), TCRIT, HMIN, HMAX, H0, RWORK(50+4*NEQMAX)
CHARACTER(1)        NORM
```

## 3 Description

An integrator setup routine must be called before the call to any linear algebra setup routine or integrator from the SPRINT suite of routines in this sub-chapter. This setup routine, D02MVF, makes the choice of the DASSL integrator and permits you to define options appropriate to this choice. Alternative choices of integrator from this suite are the BDF method and the BLEND method which can be chosen by initial calls to D02NVF or D02NWF respectively.

## 4 References

See the D02M–N Sub-chapter Introduction.

## 5 Arguments

1: NEQMAX – INTEGER *Input*

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

*Constraint*: NEQMAX $\geq 1$.

2: SDYSAV – INTEGER *Input*

*On entry*: the second dimension of the array YSAV that will be supplied to the integrator, as declared in the (sub)program from which the integrator is called (e.g., see D02NBF).

*Constraint*: SDYSAV $\geq$ MAXORD $+ 3$.

3: MAXORD – INTEGER *Input*

*On entry*: the maximum order to be used for the BDF method. If MAXORD $= 0$ or MAXORD $> 5$ then MAXORD $= 5$ is assumed.

*Constraint*: MAXORD $\geq 0$.

4: CON(3) – REAL (KIND=nag_wp) array *Input/Output*

*On entry*: values to be used to control step size choice during integration. If any CON$(i) = 0.0$ on entry, it is replaced by its default value described below. In most cases this is the recommended setting.

CON(1), CON(2), and CON(3) are factors used to bound step size changes. If the current step size $h$ fails, then the modulus of the next step size is bounded by CON(1) $\times |h|$. The default value of CON(1) is 2.0. Note that the new step size may be used with a method of different order to the failed step. If the initial step size is $h$, then the modulus of the step size on the second step is bounded by CON(3) $\times |h|$. At any other stage in the integration, if the current step size is $h$, then the modulus of the next step size is bounded by CON(2) $\times |h|$. The default values are 10.0 for CON(2) and 1000.0 for CON(3).

*Constraints*:

These constraints must be satisfied after any zero values have been replaced by default values.

$0.0 <$ CON(1) $<$ CON(2) $<$ CON(3);
CON(2) $> 1.0$;
CON(3) $> 1.0$.

*On exit*: the values actually to be used by the integration routine.

5: TCRIT – REAL (KIND=nag_wp) *Input*

*On entry*: a point beyond which integration must not be attempted. The use of TCRIT is described under the argument ITASK in the specification for the integrator (e.g., see D02NBF). A value, 0.0 say, must be specified even if ITASK subsequently specifies that TCRIT will not be used.

6: HMIN – REAL (KIND=nag_wp) *Input*

*On entry*: the minimum absolute step size to be allowed. Set HMIN $= 0.0$ if this option is not required.

7: HMAX – REAL (KIND=nag_wp) *Input*

*On entry*: the maximum absolute step size to be allowed. Set HMAX $= 0.0$ if this option is not required.

8: H0 – REAL (KIND=nag_wp) *Input*

*On entry*: the step size to be attempted on the first step. Set H0 $= 0.0$ if the initial step size is calculated internally.

9: MAXSTP – INTEGER *Input*

*On entry*: the maximum number of steps to be attempted during one call to the integrator after which it will return with IFAIL $= 2$ (e.g., see D02NBF). Set MAXSTP $= 0$ if no limit is to be imposed.

10: MXHNIL – INTEGER *Input*

*On entry*: the maximum number of warnings printed (if ITRACE $\geq 0$, e.g., see D02NBF) per problem when $t + h = t$ on a step ($h =$ current step size). If MXHNIL $\leq 0$, a default value of 10 is assumed.

11: NORM – CHARACTER(1) *Input*

*On entry*: indicates the type of norm to be used.

NORM = 'M'
Maximum norm.

NORM = 'A'
Averaged L2 norm.

NORM = 'D'
Is the same as `A'.

If $vnorm$ denotes the norm of the vector $v$ of length $n$, then for the averaged L2 norm

$$vnormB = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\frac{v_i}{w_i}\right)^2},$$

while for the maximum norm

$$vnorm = \max_{1\leq i\leq n}\left|\frac{v_i}{w_i}\right|.$$

If you wish to weight the maximum norm or the L2 norm, then RTOL and ATOL should be scaled appropriately on input to the integrator (see under ITOL in the specification of the integrator for the formulation of the weight vector $w_i$ from RTOL and ATOL, e.g., see D02NBF).

Only the first character to the actual argument NORM is passed to D02MVF; hence it is permissible for the actual argument to be more descriptive, e.g., 'Maximum', 'Average L2' or 'Default' in a call to D02MVF.

*Constraint*: NORM = 'M', 'A' or 'D'.

12: RWORK(50 + 4 × NEQMAX) – REAL (KIND=nag_wp) array *Communication Array*

This must be the same workspace array as the array RWORK supplied to the integrator. It is used to pass information from the setup routine to the integrator and therefore the contents of this array must not be changed before calling the integrator.

13: IFAIL – INTEGER *Input/Output*

*On entry*: IFAIL must be set to 0, −1 or 1. If you are unfamiliar with this argument you should refer to Section 3.4 in How to Use the NAG Library and its Documentation 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, if you are not familiar with this argument, the recommended value is 0. **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

On entry, NEQMAX < 1,
or SDYSAV < MAXORD + 3,
or MAXORD < 0,
or MAXORD > 5,
or invalid value for element of the array CON,
or NORM ≠ 'M', 'A' or 'D'.

IFAIL $= -99$

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

See Section 3.9 in How to Use the NAG Library and its Documentation for further information.

IFAIL $= -399$

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

See Section 3.8 in How to Use the NAG Library and its Documentation for further information.

IFAIL $= -999$

Dynamic memory allocation failed.

See Section 3.7 in How to Use the NAG Library and its Documentation for further information.

## 7 Accuracy

Not applicable.

## 8 Parallelism and Performance

D02MVF is not threaded in any implementation.

## 9 Further Comments

None.

## 10 Example

This example solves the plane pendulum problem defined by the equations:

$$\begin{aligned} x' &= u \\ y' &= v \\ u' &= -\lambda x \\ v' &= -\lambda y - 1 \\ x^2 + y^2 &= 1 \end{aligned}$$

The additional algebraic constraint $xu + yv = 0$ can be derived, and after appropriate substitution and manipulation to avoid a singular Jacobian solves the equations:

$$\begin{aligned} y_1' &= y_3 - y_6 y_1 \\ y_2' &= y_4 - y_6 y_2 \\ y_3' &= -y_5 y_1 \\ y_4' &= -y_5 y_2 - 1 \\ 0 &= y_1 y_3 + y_2 y_4 \\ 0 &= y_1^2 + y_2^2 - 1 \end{aligned}$$

with given initial conditions and derivatives.

### 10.1 Program Text

```
!   D02MVF Example Program Text
!   Mark 26 Release. NAG Copyright 2016.

    Module d02mvfe_mod

!     D02MVF Example Program Module:
!            Parameters and User-defined Routines

!     .. Use Statements ..
      Use nag_library, Only: nag_wp
!     .. Implicit None Statement ..
```

```
      Implicit None
!     .. Accessibility Statements ..
      Private
      Public                           :: jac, resid
!     .. Parameters ..
      Real (Kind=nag_wp), Parameter, Public :: one = 1.0_nag_wp
      Real (Kind=nag_wp), Parameter    :: two = 2.0_nag_wp
      Integer, Parameter, Public       :: iset = 1, neq = 6, nin = 5, nout = 6
      Integer, Parameter, Public       :: nrw = 50 + 4*neq
      Integer, Parameter, Public       :: nwkjac = neq*(neq+1)
      Integer, Parameter, Public       :: sdysav = 8
      Integer, Parameter, Public       :: 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 ..
        If (ires==-1) Then
          r(1) = -ydot(1)
          r(2) = -ydot(2)
          r(3) = -ydot(3)
          r(4) = -ydot(4)
          r(5:6) = 0.0E0_nag_wp
        Else
          r(1) = y(3) - y(6)*y(1) - ydot(1)
          r(2) = y(4) - y(6)*y(2) - ydot(2)
          r(3) = -y(5)*y(1) - ydot(3)
          r(4) = -y(5)*y(2) - one - ydot(4)
          r(5) = y(1)*y(3) + y(2)*y(4)
          r(6) = y(1)**2 + y(2)**2 - one
        End If
        Return
      End Subroutine resid

      Subroutine jac(neq,t,y,ydot,h,d,p)

!       .. Scalar Arguments ..
        Real (Kind=nag_wp), Intent (In) :: d, h, t
        Integer, Intent (In)           :: neq
!       .. Array Arguments ..
        Real (Kind=nag_wp), Intent (Inout) :: p(neq,neq)
        Real (Kind=nag_wp), Intent (In) :: y(neq), ydot(neq)
!       .. Local Scalars ..
        Real (Kind=nag_wp)             :: hxd
!       .. Executable Statements ..
        hxd = h*d
        p(1,1) = (one+hxd*y(6))
        p(1,3) = -hxd
        p(1,6) = hxd*y(1)
        p(2,2) = (one+hxd*y(6))
        p(2,4) = -hxd
        p(2,6) = hxd*y(2)
        p(3,1) = hxd*y(5)
        p(3,3) = one
        p(3,5) = hxd*y(1)
        p(4,2) = hxd*y(5)
        p(4,4) = one
        p(4,5) = hxd*y(2)
        p(5,1) = -hxd*y(3)
        p(5,2) = -hxd*y(4)
        p(5,3) = -hxd*y(1)
        p(5,4) = -hxd*y(2)
        p(6,1) = -two*hxd*y(1)
        p(6,2) = -two*hxd*y(2)
        Return
```

```
      End Subroutine jac
    End Module d02mvfe_mod

    Program d02mvfe

!     D02MVF Example Main Program

!     .. Use Statements ..
      Use nag_library, Only: d02mvf, d02nby, d02ngf, d02nsf, nag_wp, x04abf
      Use d02mvfe_mod, Only: iset, jac, ldysav, neq, nin, nout, nrw, nwkjac,   &
                             one, resid, sdysav
!     .. Implicit None Statement ..
      Implicit None
!     .. Local Scalars ..
      Real (Kind=nag_wp)              :: h0, hmax, hmin, t, tcrit, tout
      Integer                         :: i, ifail, itask, itol, itrace,       &
                                         maxord, maxstp, mxhnil, outchn
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable  :: atol(:), rtol(:), rwork(:),          &
                                         wkjac(:), y(:), ydot(:), ysav(:,:)
      Real (Kind=nag_wp)              :: con(3)
      Integer                         :: inform(23)
      Logical                         :: lderiv(2)
!     .. Executable Statements ..
      Write (nout,*) 'D02MVF Example Program Results'
!     Skip heading in data file
      Read (nin,*)
!     Allocations based on number of differential equations (neq)

      Allocate (atol(neq),rtol(neq),rwork(nrw),wkjac(nwkjac),y(neq),ydot(neq), &
        ysav(ldysav,sdysav))

!     Read algorithmic parameters
      Read (nin,*) maxord, maxstp, mxhnil
      Read (nin,*) hmin, hmax, h0
      Read (nin,*) rtol(1), atol(1)
      Read (nin,*) itrace, itol
      Read (nin,*) t, tout
      Read (nin,*) y(1:neq)
      Read (nin,*) itask

!     Set initial derivatives and other values
      outchn = nout
      Call x04abf(iset,outchn)
      con(1:3) = 0.0_nag_wp
      ydot(1) = y(3) - y(6)*y(1)
      ydot(2) = y(4) - y(6)*y(2)
      ydot(3) = -y(5)*y(1)
      ydot(4) = -y(5)*y(2) - one
      ydot(5) = -3.0_nag_wp*y(4)
      ydot(6) = 0.0_nag_wp
      tcrit = tout

!     ifail: behaviour on error exit
!             =0 for hard exit, =1 for quiet-soft, =-1 for noisy-soft
      ifail = 0
      Call d02mvf(neq,sdysav,maxord,con,tcrit,hmin,hmax,h0,maxstp,mxhnil,      &
        'AVERAGE-L2',rwork,ifail)

      Write (nout,*)
      Write (nout,99999) 'Pendulum problem with relative tolerance', rtol(1)
      Write (nout,99999) '                 and absolute tolerance', atol(1)
      Write (nout,99998)(i,i=1,neq)
      Write (nout,99997) t, y(1:neq)

      ifail = 0
      Call d02nsf(neq,neq,'ANALYTIC',nwkjac,rwork,ifail)

      lderiv(1) = .True.
      lderiv(2) = .True.
```

```
      ifail = 0
      Call d02ngf(neq,ldysav,t,tout,y,ydot,rwork,rtol,atol,itol,inform,resid,  &
        ysav,sdysav,jac,wkjac,nwkjac,d02nby,lderiv,itask,itrace,ifail)

      Write (nout,99997) t, y(1:neq)

99999 Format (1X,A40,' =',1P,E8.1)
99998 Format (/,1X,'  t ',3X,6('      y',I1))
99997 Format (1X,F7.4,2X,6(F8.4))
    End Program d02mvfe
```

## 10.2 Program Data

```
D02MVF Example Program Data
   5  5000  5                        : maxord, maxstp, mxhnil
   1.0E-10  0.0  0.0                 : hmin, hmax, h0
   1.0E-3 1.0E-6                     : rtol, atol
   0 1                               : itrace, itol
   0.0  3.14159265358979323846       : t, tout
   1.0  0.0  0.0  0.0  0.0  0.0      : initial y
   4                                 : itask
```

## 10.3 Program Results

```
 D02MVF Example Program Results

 Pendulum problem with relative tolerance = 1.0E-03
                  and absolute tolerance = 1.0E-06

   t          y1      y2      y3      y4      y5      y6
  0.0000    1.0000  0.0000  0.0000  0.0000  0.0000  0.0000
  3.1416   -0.9856 -0.1694 -0.0986  0.5736  0.5080  0.0000
```