# NAG Library Routine Document

# D02MWF

**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

D02MWF is a setup routine which must be called prior to the integrator D02NEF, if the DASSL implementation of Backward Differentiation Formulae (BDF) is to be used.

## 2 Specification

```
SUBROUTINE D02MWF(NEQ, MAXORD, JCEVAL, HMAX, H0, ITOL, ICOM, LICOM, COM,
1                 LCOM, IFAIL)
INTEGER           NEQ, MAXORD, ITOL, ICOM(LICOM), LICOM, LCOM, IFAIL
double precision  HMAX, H0, COM(LCOM)
CHARACTER*1       JCEVAL
```

## 3 Description

This integrator setup routine must be called before the first call to the integrator D02NEF. This setup routine D02MWF permits you to define options for the DASSL integrator, such as: whether the Jacobian is to be provided or is to be approximated numerically by the integrator; the initial and maximum step-sizes for the integration; whether relative and absolute tolerances are system wide or per system equation; and the maximum order of BDF method permitted.

## 4 References

None.

## 5 Parameters

1: NEQ – INTEGER *Input*

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

*Constraint*: NEQ $\geq 1$.

2: MAXORD – INTEGER *Input*

*On entry*: the maximum order to be used for the BDF method. Orders up to 5th order are available; setting MAXORD $> 5$ means that the maximum order used will be 5.

*Constraint*: $1 \leq$ MAXORD.

3: JCEVAL – CHARACTER*1 *Input*

*On entry*: specifies the technique to be used to compute the Jacobian.

JCEVAL = 'N'
The Jacobian is to be evaluated numerically by the integrator.

JCEVAL = 'A'
You must supply a subroutine to evaluate the Jacobian on a call to the integrator.

Only the first character of the actual paramater JCEVAL is passed to D02MWF; hence it is permissible for the actual argument to be more descriptive, e.g., 'Numerical' or 'Analytical', on a call to D02MWF.

*Constraint*: JCEVAL = 'N' or 'A'.

4: HMAX – ***double precision*** *Input*

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

*Constraint*: HMAX $\geq$ 0.0.

5: H0 – ***double precision*** *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.

6: ITOL – INTEGER *Input*

*On entry*: a value to indicate the form of the local error test.

ITOL = 0
RTOL and ATOL are single element vectors.

ITOL = 1
RTOL and ATOL are vectors. This should be chosen if you want to apply different tolerances to each equation in the system.

See D02NEF.

**Note**: The tolerances must either both be single element vectors or both be vectors of length NEQ.

*Constraint*: ITOL = 0 or 1.

7: ICOM(LICOM) – INTEGER array *Communication Array*

*On exit*: used to communicate details of the task to be carried out to the integration routine D02NEF.

8: LICOM – INTEGER *Input*

*On entry*: the dimension of the array ICOM as declared in the (sub)program from which D02MWF is called.

*Constraint*: LICOM $\geq$ NEQ + 50.

9: COM(LCOM) – ***double precision*** array *Communication Array*

*On exit*: used to communicate problem parameters to the integration routine D02NEF. This must be the same communication array as the array COM supplied to D02NEF. In particular, the values of HMAX and H0 are contained in COM.

10: LCOM – INTEGER *Input*

*On entry*: the dimension of the array COM as declared in the (sub)program from which D02MWF is called.

*Constraints*:

the array COM must be large enough for the requirements of D02NEF. That is:

if the system Jacobian is dense, LCOM $\geq 40 + (\text{MAXORD} + 4) \times \text{NEQ} + \text{NEQ}^2$;
if the system Jacobian is banded,
LCOM $\geq 40 + (\text{MAXORD} + 4) \times \text{NEQ} + (2 \times \text{ML} + \text{MU} + 1) \times \text{NEQ} + 2 \times (\text{NEQ}/(\text{ML} + \text{MU} + 1) + 1)$.

Here ML and MU are the lower and upper bandwidths respectively that are to be specified in a subsequent call to D02NPF.

11: 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.

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

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

## 6 Error Indicators and Warnings

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

Errors or warnings detected by the routine:

IFAIL $= 1$

On entry, NEQ $< 1$.

IFAIL $= 2$

On entry, MAXORD $< 1$,
or MAXORD $> 5$.

IFAIL $= 3$

On entry, JCEVAL $\neq$ 'N' or 'A'.

IFAIL $= 4$

On entry, HMAX $< 0.0$.

IFAIL $= 6$

On entry, ITOL $\neq 0$ or 1.

IFAIL $= 8$

On entry, LICOM $<$ NEQ $+ 50$.

## 7 Accuracy

Not applicable.

## 8 Further Comments

None.

## 9 Example

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

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

Differentiating the algebraic constraint once, a new algebraic constraint is obtained

$$xu + yv = 0.$$

Differentiating the algebraic constraint one more time, substituting for $x'$, $y'$, $u'$, $v'$ and using $x^2 + y^2 - 1 = 0$, the corresponding DAE system includes the differential equations and the algebraic equation in $\lambda$:

$$u^2 + v^2 - \lambda - y = 0.$$

We solve the reformulated DAE system

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

For our experiments, we take consistent initial values

$$y_1(0) = 1,\ y_2(0) = 0,\ y_3(0) = 0,\ y_4(0) = 1 \text{ and } y_5(0) = 1$$

at $t = 0$.

## 9.1 Program Text

```
*     NAG D02MWF Example Program Text
*     Mark 22 Release. NAG Copyright 2008.
*     .. Parameters ..
      INTEGER          NOUT
      PARAMETER        (NOUT=6)
      INTEGER          NEQ, MAXORD, LICOM, LCOM
      PARAMETER        (NEQ=5,MAXORD=5,LICOM=50+NEQ,LCOM=40+(MAXORD+4)
     +                 *NEQ+NEQ**2)
      INTEGER          IJAC
      PARAMETER        (IJAC=1)
*     .. Local Scalars ..
      DOUBLE PRECISION G1, G2, H0, HMAX, T, TOUT
      INTEGER          I, IFAIL, ITASK, ITOL, NADV
      CHARACTER*8      JCEVAL
*     .. Local Arrays ..
      DOUBLE PRECISION ATOL(NEQ), COM(LCOM), RTOL(NEQ), RUSER(1),
     +                 Y(NEQ), YDOT(NEQ)
      INTEGER          ICOM(LICOM), IUSER(1)
*     .. External Subroutines ..
      EXTERNAL         D02MWF, D02NEF, JAC, RES, X04ABF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'D02MWF Example Program Results'
      WRITE (NOUT,*)
      NADV = NOUT
      CALL X04ABF(1,NADV)
      T = 0.0D0
      TOUT = 1.0D0
      ITOL = 1
      DO I = 1, NEQ
         RTOL(I) = 1.0D-8
         ATOL(I) = 1.0D-8
      END DO
*     Set initial values
      Y(1) = 1.0D0
      Y(2) = 0.0D0
      Y(3) = 0.0D0
      Y(4) = 1.0D0
      Y(5) = 1.0D0
      IF (IJAC.EQ.1) THEN
         JCEVAL = 'Analytic'
      ELSE
         JCEVAL = 'Numeric'
      END IF
      HMAX = 0.0D0
```

```
      H0 = 0.0D0
      IFAIL = 1
      CALL D02MWF(NEQ,MAXORD,JCEVAL,HMAX,H0,ITOL,ICOM,LICOM,COM,LCOM,
     +            IFAIL)
      IF (IFAIL.EQ.0) THEN
         WRITE (NOUT,*)
     +     't           y(1)       y(2)       y(3)       y(4)      y(5)'
         WRITE (NOUT,99998) T, (Y(I),I=1,NEQ)
         DO I = 1, NEQ
            YDOT(I) = 0.0D0
         END DO
         ITASK = 0
   20    CALL D02NEF(NEQ,T,TOUT,Y,YDOT,RTOL,ATOL,ITASK,RES,JAC,ICOM,COM,
     +              LCOM,IUSER,RUSER,IFAIL)
         WRITE (*,99998) T, (Y(I),I=1,NEQ)
         WRITE (NOUT,*)
         WRITE (NOUT,99999) ITASK
         WRITE (NOUT,*)
         IF ((ITASK.GE.0) .AND. (ITASK.LE.3)) THEN
            IF (T.LT.TOUT) GO TO 20
            G1 = Y(1)**2 + Y(2)**2 - 1
            G2 = Y(1)*Y(3) + Y(2)*Y(4)
            WRITE (NOUT,99997) G1
            WRITE (NOUT,99996) G2
         END IF
      ELSE
         WRITE (NOUT,99995) IFAIL
      END IF
*
99999 FORMAT (1X,'D02NEF returned with ITASK = ',I4)
99998 FORMAT (1X,F6.4,5(F11.6))
99997 FORMAT (1X,'The position-level constraint G1 = ',E12.4)
99996 FORMAT (1X,'The velocity-level constraint G2 = ',E12.4)
99995 FORMAT (1X,' ** D02MWF returned with IFAIL = ',I5)
      END
*
      SUBROUTINE RES(NEQ,T,Y,YDOT,R,IRES,IUSER,RUSER)
*     .. Scalar Arguments ..
      DOUBLE PRECISION T
      INTEGER          IRES, NEQ
*     .. Array Arguments ..
      DOUBLE PRECISION R(NEQ), RUSER(1), Y(NEQ), YDOT(NEQ)
      INTEGER          IUSER(1)
*     .. Executable Statements ..
      R(1) = Y(3) - YDOT(1)
      R(2) = Y(4) - YDOT(2)
      R(3) = -Y(5)*Y(1) - YDOT(3)
      R(4) = -Y(5)*Y(2) - 1.D0 - YDOT(4)
      R(5) = Y(3)**2 + Y(4)**2 - Y(5) - Y(2)
      RETURN
      END
*
      SUBROUTINE JAC(NEQ,T,Y,YDOT,PD,CJ,IUSER,RUSER)
*     .. Scalar Arguments ..
      DOUBLE PRECISION CJ, T
      INTEGER          NEQ
*     .. Array Arguments ..
      DOUBLE PRECISION PD(NEQ,NEQ), RUSER(1), Y(NEQ), YDOT(NEQ)
      INTEGER          IUSER(1)
*     .. Executable Statements ..
      PD(1,1) = -CJ
      PD(1,3) = 1.0D0
      PD(2,2) = -CJ
      PD(2,4) = 1.0D0
      PD(3,1) = -Y(5)
      PD(3,3) = -CJ
      PD(3,5) = -Y(1)
      PD(4,2) = -Y(5)
      PD(4,4) = -CJ
      PD(4,5) = -Y(2)
      PD(5,2) = -1.0D0
```

```
      PD(5,3) = 2.0D0*Y(3)
      PD(5,4) = 2.0D0*Y(4)
      PD(5,5) = -1.0D0
      RETURN
      END
```

## 9.2 Program Data

None.

## 9.3 Program Results

```
 D02MWF Example Program Results

 t           y(1)       y(2)       y(3)       y(4)       y(5)
 0.0000   1.000000   0.000000   0.000000   1.000000   1.000000
 1.0000   0.867349   0.497701  -0.033748   0.058813  -0.493103

 D02NEF returned with ITASK =    3

 The position-level constraint G1 =  -0.8580E-08
 The velocity-level constraint G2 =  -0.3005E-07
```