NAG Library Routine Document

D02MZF

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

D02MZF interpolates components of the solution of a system of first-order differential equations from information provided by those integrators in sub-chapter D02M–N using methods set up by calls to D02MVF, D02NVF or D02NWF.

2 Specification

```
SUBROUTINE D02MZF(TSOL, SOL, M, LDYSAV, NEQ, YSAV, SDYSAV, RWORK, IFAIL)
INTEGER           M, LDYSAV, NEQ, SDYSAV, IFAIL
double precision  TSOL, SOL(M), YSAV(LDYSAV,SDYSAV), RWORK(50+4*NEQ)
```

3 Description

D02MZF evaluates the first M components of the solution of a system of ordinary differential equations at any point using natural polynomial interpolation based on information generated by the integrator. This information must be passed unchanged to D02MZF. D02MZF should not normally be used to extrapolate outside the range of values obtained from the above routine.

4 References

See the D02M–N sub-chapter Introduction.

5 Parameters

1: TSOL – ***double precision*** *Input*

On entry: the point at which the first M components of the solution are to be evaluated. TSOL should not normally be an extrapolation point. Extrapolation is permitted but not recommended.

2: SOL(M) – ***double precision*** array *Output*

On exit: the calculated value of the solution at TSOL.

3: M – INTEGER *Input*

On entry: the number of components of the solution whose values are required.

Constraint: $1 \leq \mathrm{M} \leq \mathrm{NEQ}$.

4: LDYSAV – INTEGER *Input*

On entry: the value used for the parameter LDYSAV when calling the integrator.

Constraint: $\mathrm{LDYSAV} \geq 1$.

5: NEQ – INTEGER *Input*

On entry: the value used for the parameter NEQ when calling the integrator.

Constraint: $1 \leq \mathrm{NEQ} \leq \mathrm{LDYSAV}$.

6: YSAV(LDYSAV,SDYSAV) – ***double precision*** array *Input*

On entry: the values provided in the array YSAV on return from the integrator.

7: SDYSAV – INTEGER *Input*

On entry: the value used for the parameter SDYSAV when calling the integrator.

8: RWORK$(50 + 4 \times \text{NEQ})$ – ***double precision*** array *Input*

On entry: the values provided in the array RWORK on return from the integrator.

9: 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, M < 1,
or LDYSAV < 1,
or NEQ < 1,
or M $>$ NEQ,
or NEQ $>$ LDYSAV.

IFAIL $= 2$

On entry, when accessing an element of the array RWORK an unexpected quantity was found. You have not passed the correct array to D02MZF or has overwritten elements of this array.

IFAIL $= 3$

On entry, D02MZF has been called for extrapolation. Before returning with this error exit, the value of the solution at TSOL is calculated and placed in SOL.

7 Accuracy

The solution values returned will be of a similar accuracy to those computed by the integrator.

8 Further Comments

You are recommended to employ the interpolant provided by D02XKF if using the backward differentiation integrator specified by calling setup routine D02NVF with the parameter PETZLD set to .FALSE..

9 Example

This example solves the well-known stiff Robertson problem written in implicit form

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

with initial conditions $a = 1.0$ and $b = c = 0.0$ over the range $[0, 0.1]$ with vector error control (ITOL = 4), the BDF method (setup routine D02NVF) and functional iteration. The Jacobian is calculated numerically if the functional iteration encounters difficulty and the integration is in one-step mode (ITASK = 2), with natural interpolation to calculate the solution at intervals of 0.02 using D02MZF externally. D02NBY is used for MONITR.

9.1 Program Text

```
*     D02MZF Example Program Text
*     Mark 22 Release. NAG Copyright 2008.
*     .. Parameters ..
      INTEGER          NOUT, PSTAT
      PARAMETER        (NOUT=6,PSTAT=0)
      INTEGER          NEQ, NEQMAX, NRW, NINF, NWKJAC, MAXORD, SDYSAV,
     +                 MAXSTP, MXHNIL, LDYSAV
      PARAMETER        (NEQ=3,NEQMAX=NEQ,NRW=50+4*NEQMAX,NINF=23,
     +                 NWKJAC=NEQMAX*(NEQMAX+1),MAXORD=5,
     +                 SDYSAV=MAXORD+1,MAXSTP=200,MXHNIL=5,LDYSAV=NEQ)
      DOUBLE PRECISION H0, HMAX, HMIN, TCRIT
      PARAMETER        (H0=0.0D0,HMAX=10.0D0,HMIN=1.0D-10,TCRIT=0.0D0)
      LOGICAL          PETZLD
      PARAMETER        (PETZLD=.FALSE.)
*     .. Local Scalars ..
      DOUBLE PRECISION H, HU, T, TCUR, TOLSF, TOUT, XOUT
      INTEGER          I, IFAIL, IMXER, IOUT, ITASK, ITOL, ITRACE,
     +                 NITER, NJE, NQ, NQU, NRE, NST, OUTCHN
*     .. Local Arrays ..
      DOUBLE PRECISION ATOL(NEQ), CONST(6), RTOL(NEQ), RWORK(NRW),
     +                 SOL(NEQ), WKJAC(NWKJAC), Y(NEQ), YDOT(NEQ),
     +                 YSAV(LDYSAV,SDYSAV)
      INTEGER          INFORM(NINF)
      LOGICAL          ALGEQU(NEQ), LDERIV(2)
*     .. External Subroutines ..
      EXTERNAL         D02MZF, D02NBY, D02NGF, D02NGZ, D02NSF, D02NVF,
     +                 D02NYF, RESID, X04ABF
*     .. Intrinsic Functions ..
      INTRINSIC        DBLE
*     .. Executable Statements ..
      WRITE (NOUT,*) 'D02MZF Example Program Results'
      OUTCHN = NOUT
      CALL X04ABF(1,OUTCHN)
*
*     Integrate to TOUT by overshooting TOUT in one step mode (ITASK=2)
*     using B.D.F formulae with a functional iteration method.
*     Default values for the array CONST are used. Employ vector
*     tolerances and the Jacobian is evaluated internally, if necessary.
*     MONITR subroutine replaced by NAG dummy routine D02NBY.
*     Interpolation outside D02NGF using D02MZF.
*
      T = 0.0D0
      TOUT = 0.1D0
      ITASK = 2
      Y(1) = 1.0D0
      Y(2) = 0.0D0
      Y(3) = 0.0D0
      LDERIV(1) = .FALSE.
      LDERIV(2) = .FALSE.
      ITOL = 4
      RTOL(1) = 1.0D-4
      RTOL(2) = 1.0D-3
      RTOL(3) = 1.0D-4
```

```
      ATOL(1) = 1.0D-7
      ATOL(2) = 1.0D-8
      ATOL(3) = 1.0D-7
      DO 20 I = 1, 6
         CONST(I) = 0.0D0
   20 CONTINUE
      IFAIL = 1
*
      CALL D02NVF(NEQMAX,SDYSAV,MAXORD,'Functional-iteration',PETZLD,
     +            CONST,TCRIT,HMIN,HMAX,H0,MAXSTP,MXHNIL,'Average-L2',
     +            RWORK,IFAIL)
      IF (IFAIL.NE.0) THEN
         WRITE (NOUT,99993) IFAIL
         GO TO 80
      END IF
*     Linear algebra setup required (in case functional iteration
*     encounters any difficulty).
      IFAIL = 0
      CALL D02NSF(NEQ,NEQMAX,'Numerical',NWKJAC,RWORK,IFAIL)
*
      XOUT = 0.02D0
      IOUT = 1
*
      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
   40 IFAIL = 1
*
      CALL D02NGF(NEQ,LDYSAV,T,TOUT,Y,YDOT,RWORK,RTOL,ATOL,ITOL,INFORM,
     +            RESID,YSAV,SDYSAV,D02NGZ,WKJAC,NWKJAC,D02NBY,LDERIV,
     +            ITASK,ITRACE,IFAIL)
*
      IF (IFAIL.EQ.0) THEN
*
         IFAIL = 0
         CALL D02NYF(NEQ,NEQMAX,HU,H,TCUR,TOLSF,RWORK,NST,NRE,NJE,NQU,
     +               NQ,NITER,IMXER,ALGEQU,INFORM,IFAIL)
*
   60    CONTINUE
         IF (TCUR-HU.LT.XOUT .AND. XOUT.LE.TCUR) THEN
            IFAIL = 0
*           Natural interpolation
            CALL D02MZF(XOUT,SOL,NEQ,LDYSAV,NEQ,YSAV,SDYSAV,RWORK,IFAIL)
*
            WRITE (NOUT,99999) XOUT, (SOL(I),I=1,NEQ)
            IOUT = IOUT + 1
            XOUT = DBLE(IOUT)*0.02D0
            IF (IOUT.LT.6) THEN
               GO TO 60
            ELSE IF (PSTAT.EQ.1) THEN
               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
            END IF
         ELSE
            GO TO 40
         END IF
      ELSE
         WRITE (NOUT,*)
         WRITE (NOUT,99998) 'Exit D02NGF with IFAIL = ', IFAIL,
     +     '  and T = ', T
      END IF
   80 CONTINUE
*
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)
99993 FORMAT (1X,/1X,' ** D02NVF returned with IFAIL = ',I5)
      END
*
      SUBROUTINE RESID(NEQ,T,Y,YDOT,R,IRES)
*     .. Scalar Arguments ..
      DOUBLE PRECISION T
      INTEGER          IRES, NEQ
*     .. Array Arguments ..
      DOUBLE PRECISION R(NEQ), Y(NEQ), YDOT(NEQ)
*     .. Executable Statements ..
      R(1) = -YDOT(1)
      R(2) = -YDOT(2)
      R(3) = -YDOT(3)
*
      IF (IRES.EQ.1) THEN
         R(1) = -0.04D0*Y(1) + 1.0D4*Y(2)*Y(3) + R(1)
         R(2) = 0.04D0*Y(1) - 1.0D4*Y(2)*Y(3) - 3.0D7*Y(2)*Y(2) + R(2)
         R(3) = 3.0D7*Y(2)*Y(2) + R(3)
      END IF
      RETURN
      END
```

9.2 Program Data

None.

9.3 Program Results

```
 D02MZF Example Program Results

     X          Y(1)           Y(2)           Y(3)
    0.000      1.00000        0.00000        0.00000
    0.020      0.99920        0.00004        0.00076
    0.040      0.99841        0.00004        0.00155
    0.060      0.99763        0.00004        0.00234
    0.080      0.99685        0.00004        0.00311
    0.100      0.99608        0.00004        0.00389
```