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
REAL (KIND=nag_wp) 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 Arguments

1: TSOL – REAL (KIND=nag_wp) *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) – REAL (KIND=nag_wp) 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 argument LDYSAV when calling the integrator.

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

5: NEQ – INTEGER *Input*

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

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

6: YSAV(LDYSAV, SDYSAV) – REAL (KIND=nag_wp) 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 argument SDYSAV when calling the integrator.

8: RWORK($50 + 4 \times$ NEQ) – REAL (KIND=nag_wp) 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 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, 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.

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

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

8 Parallelism and Performance

D02MZF is not thread safe and should not be called from a multithreaded user program. Please see Section 3.12.1 in How to Use the NAG Library and its Documentation for more information on thread safety.

D02MZF is not threaded in any implementation.

9 Further Comments

None.

10 Example

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

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

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.

10.1 Program Text

```
!   D02MZF Example Program Text
!   Mark 26 Release. NAG Copyright 2016.

    Module d02mzfe_mod

!     D02MZF 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                           :: resid
!     .. Parameters ..
      Real (Kind=nag_wp), Parameter, Public :: tstep = 0.02_nag_wp
      Integer, Parameter, Public       :: iset = 1, neq = 3, nin = 5, nout = 6
      Integer, Parameter, Public       :: nrw = 50 + 4*neq
      Integer, Parameter, Public       :: nwkjac = neq*(neq+1)
      Integer, Parameter, Public       :: sdysav = 6
      Integer, Parameter, Public       :: ldysav = neq
    Contains
      Subroutine resid(neq,t,y,ydot,r,ires)

!       .. Parameters ..
        Real (Kind=nag_wp), Parameter  :: alpha = 0.04_nag_wp
```

```
        Real (Kind=nag_wp), Parameter  :: beta = 1.0E4_nag_wp
        Real (Kind=nag_wp), Parameter  :: gamma = 3.0E7_nag_wp
!       .. 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 ..
        r(1:3) = -ydot(1:3)
        If (ires==1) Then
          r(1) = -alpha*y(1) + beta*y(2)*y(3) + r(1)
          r(2) = alpha*y(1) - beta*y(2)*y(3) - gamma*y(2)*y(2) + r(2)
          r(3) = gamma*y(2)*y(2) + r(3)
        End If
        Return
      End Subroutine resid
    End Module d02mzfe_mod

    Program d02mzfe

!     D02MZF Example Main Program

!     .. Use Statements ..
      Use nag_library, Only: d02mzf, d02nby, d02ngf, d02ngz, d02nsf, d02nvf,   &
                             d02nyf, nag_wp, x04abf
      Use d02mzfe_mod, Only: iset, ldysav, neq, nin, nout, nrw, nwkjac, resid, &
                             sdysav, tstep
!     .. Implicit None Statement ..
      Implicit None
!     .. Local Scalars ..
      Real (Kind=nag_wp)              :: h, h0, hmax, hmin, hu, t, tcrit,     &
                                         tcur, tolsf, tout, xout
      Integer                         :: i, ifail, imxer, iout, itask, itol,  &
                                         itrace, maxord, maxstp, mxhnil,      &
                                         niter, nje, nq, nqu, nre, nst,       &
                                         outchn, pstat
      Logical                         :: petzld
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable  :: atol(:), rtol(:), rwork(:), sol(:),  &
                                         wkjac(:), y(:), ydot(:), ysav(:,:)
      Real (Kind=nag_wp)              :: con(6)
      Integer                         :: inform(23)
      Logical, Allocatable            :: algequ(:)
      Logical                         :: lderiv(2)
!     .. Executable Statements ..
      Write (nout,*) 'D02MZF 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),sol(neq),wkjac(nwkjac),y(neq),  &
        ydot(neq),ysav(ldysav,sdysav),algequ(neq))

!     Read algorithmic parameters
      Read (nin,*) maxord, maxstp, mxhnil, pstat
      Read (nin,*) petzld
      Read (nin,*) hmin, hmax, h0, tcrit
      Read (nin,*) t, tout
      Read (nin,*) itol
      Read (nin,*) y(1:neq)
      Read (nin,*) lderiv(1:2)
      Read (nin,*) rtol(1:neq)
      Read (nin,*) atol(1:neq)

      outchn = nout
      Call x04abf(iset,outchn)
      con(1:6) = 0.0_nag_wp
      itask = 2
      itrace = 0
```

```
!     ifail: behaviour on error exit
!             =0 for hard exit, =1 for quiet-soft, =-1 for noisy-soft
      ifail = 0
      Call d02nvf(neq,sdysav,maxord,'Functional-iteration',petzld,con,tcrit,   &
        hmin,hmax,h0,maxstp,mxhnil,'Average-L2',rwork,ifail)

!     Linear algebra setup.
      ifail = 0
      Call d02nsf(neq,neq,'Numerical',nwkjac,rwork,ifail)

      Write (nout,99994)(i,i=1,neq)
      Write (nout,99999) t, y(1:neq)

!     First value of t to interpolate and print solution at.
      iout = 1
      xout = tstep

!      Integrate one step at a time and overshoot tout (itask=2).
steps: Do
        ifail = 0
        Call d02ngf(neq,ldysav,t,tout,y,ydot,rwork,rtol,atol,itol,inform,      &
          resid,ysav,sdysav,d02ngz,wkjac,nwkjac,d02nby,lderiv,itask,itrace,    &
          ifail)

!       Get Current value of t (tcur) and last step used (hu)
        Call d02nyf(neq,neq,hu,h,tcur,tolsf,rwork,nst,nre,nje,nqu,nq,niter,    &
          imxer,algequ,inform,ifail)

interp: Do
          If (tcur-hu<xout .And. xout<=tcur) Then
!           xout is in interval of last step, so interpolate.
            ifail = 0
            Call d02mzf(xout,sol,neq,ldysav,neq,ysav,sdysav,rwork,ifail)

            Write (nout,99999) xout, sol(1:neq)
            iout = iout + 1
            If (iout>=6) Then
!             Final solution point printed. Print stats if required.
              If (pstat==1) Then
                Write (nout,*)
                Write (nout,99998) hu, h, tcur
                Write (nout,99997) nst, nre, nje
                Write (nout,99996) nqu, nq, niter
                Write (nout,99995) ' Max err comp = ', imxer
              End If
              Exit steps
            End If
!           Set next xout. This might also be in last step, so keep
!           looping for interpolation.
            xout = xout + tstep
            Cycle interp
          Else
!           Take another step.
            Cycle steps
          End If
        End Do interp
      End Do steps

99999 Format (1X,F8.3,3(F13.5,2X))
99998 Format (1X,' HUSED = ',E12.5,'  HNEXT = ',E12.5,'  TCUR = ',E12.5)
99997 Format (1X,' NST = ',I6,'    NRE = ',I6,'    NJE = ',I6)
99996 Format (1X,' NQU = ',I6,'    NQ  = ',I6,'  NITER = ',I6)
99995 Format (1X,A,I4)
99994 Format (/,1X,'    X ',3('         Y(',I1,')  '))
    End Program d02mzfe
```

10.2 Program Data

```
D02MZF Example Program Data
   5  200  5  0                    : maxord, maxstp, mxhnil, pstat
   .FALSE.                         : petzld
   1.0E-10  10.0  0.0  0.0         : hmin, hmax, h0, tcrit
   0.0  0.1                        : t, tout
   4                               : itol
   1.0  0.0  0.0                   : y
   .FALSE. .FALSE.                 : lderiv
   1.0E-4  1.0E-3  1.0E-4          : rtol
   1.0E-7  1.0E-8  1.0E-7          : atol
```

10.3 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
```