NAG Library Routine Document

D02QZF

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

D02QZF interpolates components of the solution of a non-stiff system of first-order differential equations from information provided by the integrator routines D02QFF or D02QGF.

2 Specification

```
SUBROUTINE D02QZF (NEQF, TWANT, NWANT, YWANT, YPWANT, RWORK, LRWORK,        &
                   IWORK, LIWORK, IFAIL)

INTEGER             NEQF, NWANT, LRWORK, IWORK(LIWORK), LIWORK, IFAIL
REAL (KIND=nag_wp) TWANT, YWANT(NWANT), YPWANT(NWANT), RWORK(LRWORK)
```

3 Description

D02QZF evaluates the first NWANT components of the solution of a non-stiff system of first-order ordinary differential equations at any point using the method of Watts and Shampine (1986) and information generated by D02QFF or D02QGF. D02QZF should not normally be used to extrapolate outside the current range of the values produced by the integration routine.

4 References

Watts H A and Shampine L F (1986) Smoother interpolants for Adams codes *SIAM J. Sci. Statist. Comput.* **7** 334–345

5 Arguments

1: NEQF – INTEGER *Input*

On entry: the number of first-order ordinary differential equations being solved by the integration routine. It must contain the same value as the argument NEQF in a prior call to the setup routine D02QWF.

2: TWANT – REAL (KIND=nag_wp) *Input*

On entry: the point at which components of the solution and derivative are to be evaluated. TWANT should not normally be an extrapolation point, that is TWANT should satisfy

$$told \leq \text{TWANT} \leq \text{T},$$

or if integration is proceeding in the negative direction

$$told \geq \text{TWANT} \geq \text{T},$$

where $told$ is the previous integration point and is, to within rounding, TCURR – HLAST (see D02QXF). Extrapolation is permitted but not recommended and IFAIL $= 2$ is returned whenever extrapolation is attempted.

3: NWANT – INTEGER *Input*

On entry: the number of components of the solution and derivative whose values at TWANT are required. The first NWANT components are evaluated.

Constraint: $1 \leq \text{NWANT} \leq \text{NEQF}$.

4: YWANT(NWANT) – REAL (KIND=nag_wp) array *Output*

On exit: the calculated value of the ith component of the solution at TWANT, for $i = 1, 2, \ldots, \text{NWANT}$.

5: YPWANT(NWANT) – REAL (KIND=nag_wp) array *Output*

On exit: the calculated value of the ith component of the derivative at TWANT, for $i = 1, 2, \ldots, \text{NWANT}$.

6: RWORK(LRWORK) – REAL (KIND=nag_wp) array *Communication Array*

On entry: this **must** be the same argument RWORK as supplied to D02QWF and to D02QFF or D02QGF. It is used to pass information from these routines to D02QZF. Therefore its contents **must not** be changed before a call to D02QZF.

7: LRWORK – INTEGER *Input*

On entry: the dimension of the array RWORK as declared in the (sub)program from which D02QZF is called.

This must be the same argument LRWORK as supplied to D02QWF.

8: IWORK(LIWORK) – INTEGER array *Communication Array*

On entry: this **must** be the same argument IWORK as supplied to D02QWF and to D02QFF or D02QGF. It is used to pass information from these routines to D02QZF. Therefore its contents **must not** be changed before a call to D02QZF.

9: LIWORK – INTEGER *Input*

On entry: the dimension of the array IWORK as declared in the (sub)program from which D02QZF is called.

This must be the same argument LIWORK as supplied to D02QWF.

10: 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 $= 1$

An integration routine (D02QFF or D02QGF) has not been called, no integration steps have been taken since the last call to D02QWF with STATEF = 'S', one or more of the arguments LRWORK, LIWORK and NEQF does not match the same argument supplied to D02QWF, or NWANT does not satisfy $1 \leq \text{NWANT} \leq \text{NEQF}$.

IFAIL $= 2$

D02QZF has been called for extrapolation. The values of the solution and its derivative at TWANT have been calculated and placed in YWANT and YPWANT before returning with this warning (see Section 7).

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.

These error exits may be caused by overwriting elements of RWORK and IWORK.

7 Accuracy

The error in interpolation is of a similar order to the error arising from the integration. The same order of accuracy can be expected when extrapolating using D02QZF. However, the actual error in extrapolation will, in general, be much larger than for interpolation.

8 Parallelism and Performance

D02QZF 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.

D02QZF is not threaded in any implementation.

9 Further Comments

When interpolation for only a few components is required then it is more efficient to order the components of interest so that they are numbered first.

10 Example

This example solves the equation

$$y'' = -y, \quad y(0) = 0, \quad y'(0) = 1$$

reposed as

$$\begin{aligned} y_1' &= y_2 \\ y_2' &= -y_1 \end{aligned}$$

over the range $[0, \pi/2]$ with initial conditions $y_1 = 0$ and $y_2 = 1$ using vector error control (VECTOL = .TRUE.) and D02QFF in one-step mode (ONESTP = .TRUE.). D02QZF is used to provide solution values at intervals of $\pi/16$.

10.1 Program Text

```
!   D02QZF Example Program Text
!   Mark 26 Release. NAG Copyright 2016.

    Module d02qzfe_mod

!     D02QZF 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                             :: fcn
!     .. Parameters ..
      Integer, Parameter, Public         :: neqf = 2, neqg = 0, nin = 5,         &
                                            nout = 6
      Integer, Parameter, Public         :: latol = neqf
      Integer, Parameter, Public         :: liwork = 21 + 4*neqg
      Integer, Parameter, Public         :: lrtol = neqf
      Integer, Parameter, Public         :: lrwork = 23 + 23*neqf + 14*neqg
    Contains
      Subroutine fcn(neqf,x,y,f)

!       .. Scalar Arguments ..
        Real (Kind=nag_wp), Intent (In) :: x
        Integer, Intent (In)            :: neqf
!       .. Array Arguments ..
        Real (Kind=nag_wp), Intent (Out) :: f(neqf)
        Real (Kind=nag_wp), Intent (In) :: y(neqf)
!       .. Executable Statements ..
        f(1) = y(2)
        f(2) = -y(1)
        Return
      End Subroutine fcn
    End Module d02qzfe_mod

    Program d02qzfe

!     D02QZF Example Main Program

!     .. Use Statements ..
      Use nag_library, Only: d02qff, d02qfz, d02qwf, d02qzf, nag_wp
      Use d02qzfe_mod, Only: fcn, latol, liwork, lrtol, lrwork, neqf, neqg,    &
                             nin, nout
!     .. Implicit None Statement ..
      Implicit None
!     .. Local Scalars ..
      Real (Kind=nag_wp)                 :: hmax, t, tcrit, tinc, tout, tstart,  &
                                            twant
      Integer                            :: ifail, maxstp, nwant
      Logical                            :: alterg, crit, onestp, root, sophst,  &
                                            vectol
      Character (1)                      :: statef
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable    :: atol(:), rtol(:), rwork(:), y(:),    &
                                            ypwant(:), ywant(:)
      Integer, Allocatable               :: iwork(:)
!     .. Executable Statements ..
      Write (nout,*) 'D02QZF Example Program Results'
!     Skip heading in data file
      Read (nin,*)
      Allocate (atol(latol),rtol(lrtol),rwork(lrwork),y(neqf),ypwant(neqf),    &
        ywant(neqf),iwork(liwork))

      Read (nin,*) hmax, tstart
      Read (nin,*) tcrit, tinc
      Read (nin,*) statef
```

```
      Read (nin,*) vectol, onestp, crit
      Read (nin,*) maxstp
      Read (nin,*) rtol(1:neqf)
      Read (nin,*) atol(1:neqf)
      Read (nin,*) y(1:neqf)
      tout = tcrit
      t = tstart
      twant = tstart + tinc
      nwant = neqf

!     Set up integration.
      ifail = 0
      Call d02qwf(statef,neqf,vectol,atol,latol,rtol,lrtol,onestp,crit,tcrit,  &
        hmax,maxstp,neqg,alterg,sophst,rwork,lrwork,iwork,liwork,ifail)

      Write (nout,*)
      Write (nout,*) '  T         Y(1)     Y(2)'
      Write (nout,99999) t, y(1), y(2)

integ: Do While (t<tout)
        ifail = -1
        Call d02qff(fcn,neqf,t,y,tout,d02qfz,neqg,root,rwork,lrwork,iwork,     &
          liwork,ifail)

        If (ifail/=0) Then
          Exit integ
        End If

!       Interpolate at wanted time values up to time = t.
        Do While (twant<=t)
          ifail = 0
          Call d02qzf(neqf,twant,nwant,ywant,ypwant,rwork,lrwork,iwork,liwork, &
            ifail)
          Write (nout,99999) twant, ywant(1), ywant(2)
          twant = twant + tinc
        End Do
      End Do integ

99999 Format (1X,F7.4,2X,2(F7.4,2X))
    End Program d02qzfe
```

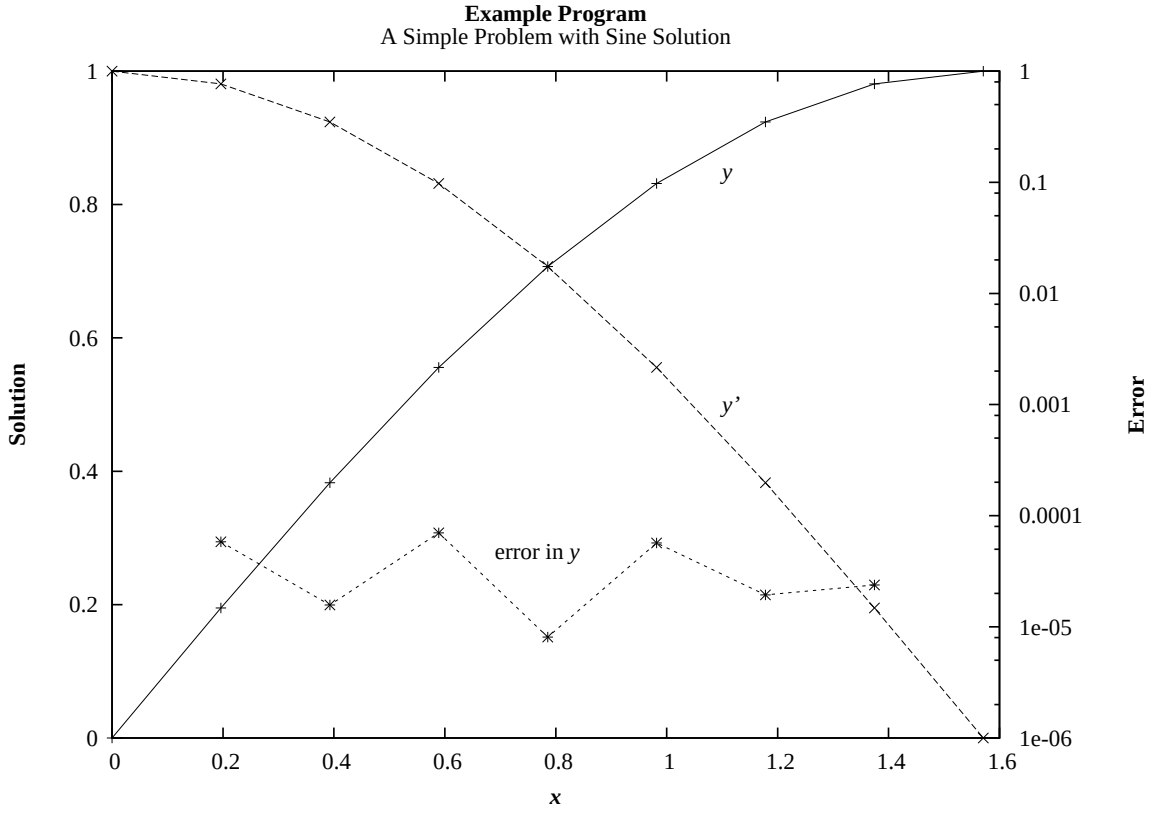
10.2 Program Data

```
D02QZF Example Program Data
   2.0  0.0                                                : hmax, tstart
   1.5707963267948966192  1.9634954084936207740E-1        : tcrit, tinc
   S                                                       : statef
   .TRUE. .TRUE. .TRUE.                                    : vectol, onestp, crit
   500                                                     : maxstp
   1.0E-4  1.0E-4                                          : rtol
   1.0E-8  1.0E-8                                          : atol
   0.0  1.0                                                : y
```

10.3 Program Results

```
 D02QZF Example Program Results

   T         Y(1)     Y(2)
  0.0000   0.0000   1.0000
  0.1963   0.1951   0.9808
  0.3927   0.3827   0.9239
  0.5890   0.5556   0.8315
  0.7854   0.7071   0.7071
  0.9817   0.8315   0.5556
  1.1781   0.9239   0.3827
  1.3744   0.9808   0.1951
  1.5708   1.0000  -0.0000
```


Example Program
A Simple Problem with Sine Solution
Solution
Error
x
y
y'
error in y
1
0.8
0.6
0.4
0.2
0
0.2
0.4
0.6
0.8
1.2
1.4
1.6
0.1
0.01
0.001
0.0001
1e-05
1e-06