NAG Library Routine Document

D02PXF

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

D02PXF computes the solution of a system of ordinary differential equations using interpolation anywhere on an integration step taken by D02PDF.

2 Specification

```
SUBROUTINE D02PXF (TWANT, REQEST, NWANT, YWANT, YPWANT, F, WORK, WRKINT,        &
                   LENINT, IFAIL)

INTEGER            NWANT, LENINT, IFAIL
REAL (KIND=nag_wp) TWANT, YWANT(*), YPWANT(*), WORK(*), WRKINT(LENINT)
CHARACTER(1)       REQEST
EXTERNAL           F
```

3 Description

D02PXF and its associated routines (D02PDF, D02PVF, D02PWF, D02PYF and D02PZF) solve the initial value problem for a first-order system of ordinary differential equations. The routines, based on Runge–Kutta methods and derived from RKSUITE (see Brankin *et al.* (1991)), integrate

$$y' = f(t, y) \qquad \text{given} \qquad y(t_0) = y_0$$

where y is the vector of n solution components and t is the independent variable.

D02PDF computes the solution at the end of an integration step. Using the information computed on that step D02PXF computes the solution by interpolation at any point on that step. It cannot be used if METHOD $= 3$ was specified in the call to setup routine D02PVF.

4 References

Brankin R W, Gladwell I and Shampine L F (1991) RKSUITE: A suite of Runge–Kutta codes for the initial value problems for ODEs *SoftReport 91-S1* Southern Methodist University

5 Parameters

1: TWANT – REAL (KIND=nag_wp) *Input*

On entry: t, the value of the independent variable where a solution is desired.

2: REQEST – CHARACTER(1) *Input*

On entry: determines whether the solution and/or its first derivative are to be computed.

REQEST = 'S'
: Compute the approximate solution only.

REQEST = 'D'
: Compute the approximate first derivative of the solution only.

REQEST = 'B'
: Compute both the approximate solution and its first derivative.

Constraint: REQEST = 'S', 'D' or 'B'.

3: NWANT – INTEGER *Input*

On entry: the number of components of the solution to be computed. The first NWANT components are evaluated.

Constraint: $1 \leq \text{NWANT} \leq n$, where n is specified by NEQ in the prior call to D02PVF.

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

Note: the dimension of the array YWANT must be at least NWANT if REQEST = 'S' or 'B', and at least 1 otherwise.

On exit: an approximation to the first NWANT components of the solution at TWANT if REQEST = 'S' or 'B'. Otherwise YWANT is not defined.

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

Note: the dimension of the array YPWANT must be at least NWANT if REQEST = 'D' or 'B', and at least 1 otherwise.

On exit: an approximation to the first NWANT components of the first derivative at TWANT if REQEST = 'D' or 'B'. Otherwise YPWANT is not defined.

6: F – SUBROUTINE, supplied by the user. *External Procedure*

F must evaluate the functions f_i (that is the first derivatives y_i') for given values of the arguments t, y_i. It must be the same procedure as supplied to D02PDF.

The specification of F is:

```
SUBROUTINE F (T, Y, YP)
```

```
REAL (KIND=nag_wp) T, Y(*), YP(*)
```

In the description of the parameters of D02PXF below, n denotes the value of NEQ in the call of D02PVF.

1: T – REAL (KIND=nag_wp) *Input*

On entry: t, the current value of the independent variable.

2: Y($*$) – REAL (KIND=nag_wp) array *Input*

On entry: the current values of the dependent variables, y_i, for $i = 1, 2, \dots, n$.

3: YP($*$) – REAL (KIND=nag_wp) array *Output*

On exit: the values of f_i, for $i = 1, 2, \dots, n$.

F must either be a module subprogram USEd by, or declared as EXTERNAL in, the (sub)program from which D02PXF is called. Parameters denoted as *Input* must **not** be changed by this procedure.

7: WORK($*$) – REAL (KIND=nag_wp) array *Input/Output*

Note: the dimension of the array WORK must be at least LENWRK (see D02PVF).

On entry: this **must** be the same array as supplied to D02PDF and **must** remain unchanged between calls.

On exit: contains information about the integration for use on subsequent calls to D02PDF or other associated routines.

8: WRKINT(LENINT) – REAL (KIND=nag_wp) array *Input/Output*

On entry: must be the same array as supplied in previous calls, if any, and must remain unchanged between calls to D02PXF.

On exit: the contents are modified.

9: LENINT – INTEGER *Input*

On entry: the dimension of the array WRKINT as declared in the (sub)program from which D02PXF is called.

Constraints:

LENINT ≥ 1 if METHOD $= 1$ in the prior call to D02PVF;
LENINT $\geq n + 5 \times$ NWANT if METHOD $= 2$ and n is specified by NEQ in the prior call to D02PVF.

10: 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.

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.**

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, an invalid input value for NWANT or LENINT was detected or an invalid call to D02PXF was made, for example without a previous call to the integration routine D02PDF, or after an error return from D02PDF, or if D02PDF was being used with METHOD $= 3$. You cannot continue integrating the problem.

7 Accuracy

The computed values will be of a similar accuracy to that computed by D02PDF.

8 Further Comments

None.

9 Example

This example solves the equation

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

reposed as

$$y_1' = y_2$$

$$y_2' = -y_1$$

over the range $[0, 2\pi]$ with initial conditions $y_1 = 0.0$ and $y_2 = 1.0$. Relative error control is used with threshold values of 1.0E-8 for each solution component. D02PDF is used to integrate the problem one step at a time and D02PXF is used to compute the first component of the solution and its derivative at intervals of length $\pi/8$ across the range whenever these points lie in one of those integration steps. A moderate order Runge–Kutta method (METHOD $= 2$) is also used with tolerances TOL $= 1.0\text{E}{-3}$ and TOL $= 1.0\text{E}{-4}$ in turn so that solutions may be compared. The value of π is obtained by using X01AAF.

Note that the length of WORK is large enough for any valid combination of input arguments to D02PVF and the length of WRKINT is large enough for any valid value of the parameter NWANT.

9.1 Program Text

```
!   D02PXF Example Program Text
!   Mark 24 Release. NAG Copyright 2012.

    Module d02pxfe_mod

!     D02PXF Example Program Module:
!            Parameters and User-defined Routines

!     .. Use Statements ..
      Use nag_library, Only: nag_wp
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Real (Kind=nag_wp), Parameter        :: tol1 = 1.0E-3_nag_wp
      Real (Kind=nag_wp), Parameter        :: tol2 = 1.0E-4_nag_wp
      Integer, Parameter                   :: neq = 2, nin = 5, nout = 6,      &
                                              npts = 16
      Integer, Parameter                   :: lenwrk = 32*neq
    Contains
      Subroutine f(t,y,yp)

!       .. Scalar Arguments ..
        Real (Kind=nag_wp), Intent (In)      :: t
!       .. Array Arguments ..
        Real (Kind=nag_wp), Intent (In)      :: y(*)
        Real (Kind=nag_wp), Intent (Out)     :: yp(*)
!       .. Executable Statements ..
        yp(1) = y(2)
        yp(2) = -y(1)
        Return
      End Subroutine f
    End Module d02pxfe_mod

    Program d02pxfe

!     D02PXF Example Main Program

!     .. Use Statements ..
      Use nag_library, Only: d02pdf, d02pvf, d02pxf, d02pyf, nag_wp
      Use d02pxfe_mod, Only: f, lenwrk, neq, nin, nout, npts, tol1, tol2
!     .. Implicit None Statement ..
      Implicit None
!     .. Local Scalars ..
      Real (Kind=nag_wp)                   :: hnext, hstart, tend, tinc, tnow, &
                                              tol, tstart, twant, waste
      Integer                              :: i, ifail, lenint, method, nwant, &
                                              stpcst, stpsok, totf
      Logical                              :: errass
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable      :: thres(:), work(:), wrkint(:),    &
                                              ynow(:), ypnow(:), ypwant(:),    &
                                              ystart(:), ywant(:)
!     .. Intrinsic Procedures ..
      Intrinsic                            :: real
!     .. Executable Statements ..
      Write (nout,*) 'D02PXF Example Program Results'
!     Skip heading in data file
      Read (nin,*)
!     neq: number of differential equations
```

```
      Read (nin,*) method, nwant
      lenint = neq + 5*nwant
      Allocate (thres(neq),work(lenwrk),wrkint(lenint),ynow(neq),ypnow(neq), &
        ypwant(nwant),ystart(neq),ywant(nwant))

!     Set initial conditions and input for D02PVF

      Read (nin,*) tstart, tend
      Read (nin,*) ystart(1:neq)
      Read (nin,*) hstart
      Read (nin,*) thres(1:neq)
      Read (nin,*) errass

!     Set output control

      tinc = (tend-tstart)/real(npts,kind=nag_wp)

      Do i = 1, 2
        If (i==1) tol = tol1
        If (i==2) tol = tol2

!       Set up integration.
        ifail = 0
        Call d02pvf(neq,tstart,ystart,tend,tol,thres,method,'Complex Task', &
          errass,hstart,work,lenwrk,ifail)

        Write (nout,99999) tol
        Write (nout,99998)
        Write (nout,99997) tstart, ystart(1:neq)

!       Set up first point at which solution is desired.
        twant = tstart + tinc
        tnow = tstart

!        Integrate by steps until tend is reached or error is encountered.

integr8: Do
          If (tnow>=tend) Exit integr8

!         Integrate one step to tnow.
          ifail = -1
          Call d02pdf(f,tnow,ynow,ypnow,work,ifail)
          If (ifail/=0) Exit integr8

!          Interpolate at required additional points up to tnow.
interpol8: Do
            If (twant>tnow) Exit interpol8

!           Interpolate and print solution at t = twant.
            ifail = 0
            Call d02pxf(twant,'Both',nwant,ywant,ypwant,f,work,wrkint,lenint, &
              ifail)
            Write (nout,99997) twant, ywant(1), ypwant(1)

!           Set next required solution point
            twant = twant + tinc
          End Do interpol8

        End Do integr8

!       Get integration statistics.
        ifail = 0
        Call d02pyf(totf,stpcst,waste,stpsok,hnext,ifail)

        Write (nout,99996) totf

      End Do
99999 Format (/' Calculation with TOL = ',E8.1)
```

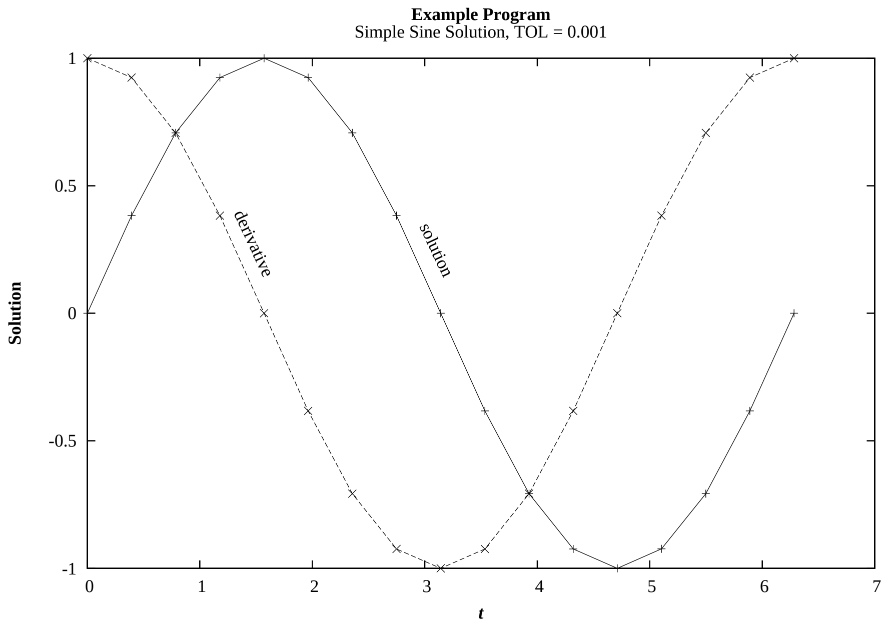
```
99998 Format (/'    t         y1         y1'''/)
99997 Format (1X,F6.3,2(3X,F8.4))
99996 Format (/' Cost of the integration in evaluations of F is',I6)

    End Program d02pxfe
```

9.2 Program Data

```
D02PXF Example Program Data
   2  1                               : neq, method, nwant
   0.0  6.28318530717958647692        : tstart, tend
   0.0  1.0                           : ystart(1:neq)
   0.0                                : hstart
   1.0E-8  1.0E-8                     : thres(1:neq)
   .FALSE.                            : errass
```

9.3 Program Results

```
 D02PXF Example Program Results

 Calculation with TOL =  0.1E-02

    t         y1         y1'

  0.000     0.0000     1.0000
  0.393     0.3827     0.9239
  0.785     0.7071     0.7071
  1.178     0.9239     0.3826
  1.571     1.0000    -0.0001
  1.963     0.9238    -0.3828
  2.356     0.7070    -0.7073
  2.749     0.3825    -0.9240
  3.142    -0.0002    -0.9999
  3.534    -0.3829    -0.9238
  3.927    -0.7072    -0.7069
  4.320    -0.9239    -0.3823
  4.712    -0.9999     0.0004
  5.105    -0.9236     0.3830
  5.498    -0.7068     0.7073
  5.890    -0.3823     0.9239

 Cost of the integration in evaluations of F is    68

 Calculation with TOL =  0.1E-03

    t         y1         y1'

  0.000     0.0000     1.0000
  0.393     0.3827     0.9239
  0.785     0.7071     0.7071
  1.178     0.9239     0.3827
  1.571     1.0000    -0.0000
  1.963     0.9239    -0.3827
  2.356     0.7071    -0.7071
  2.749     0.3827    -0.9239
  3.142    -0.0000    -1.0000
  3.534    -0.3827    -0.9239
  3.927    -0.7071    -0.7071
  4.320    -0.9239    -0.3827
  4.712    -1.0000     0.0000
  5.105    -0.9238     0.3827
  5.498    -0.7071     0.7071
  5.890    -0.3826     0.9239

 Cost of the integration in evaluations of F is   105
```


Example Program
Simple Sine Solution, TOL = 0.001
Solution
t
derivative
solution
1
0.5
0
-0.5
-1
0
1
2
3
4
5
6
7