NAG Library Routine Document

D02PSF

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

D02PSF computes the solution of a system of ordinary differential equations using interpolation anywhere on an integration step taken by D02PFF.

2 Specification

```
SUBROUTINE D02PSF (N, TWANT, IDERIV, NWANT, YWANT, YPWANT, F, WCOMM,       &
                   LWCOMM, IUSER, RUSER, IWSAV, RWSAV, IFAIL)

INTEGER            N, IDERIV, NWANT, LWCOMM, IUSER(*), IWSAV(130), IFAIL
REAL (KIND=nag_wp) TWANT, YWANT(NWANT), YPWANT(NWANT), WCOMM(LWCOMM),      &
                   RUSER(*), RWSAV(32*N+350)
EXTERNAL           F
```

3 Description

D02PSF and its associated routines (D02PFF, D02PQF, D02PRF, D02PTF and D02PUF) 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.

D02PFF computes the solution at the end of an integration step. Using the information computed on that step D02PSF computes the solution by interpolation at any point on that step. It cannot be used if METHOD $= 3$ or -3 was specified in the call to setup routine D02PQF.

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: N – INTEGER *Input*

On entry: n, the number of ordinary differential equations in the system to be solved by the integration routine.

Constraint: N ≥ 1.

2: TWANT – REAL (KIND=nag_wp) *Input*

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

3: IDERIV – INTEGER *Input*

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

IDERIV $= 0$
compute approximate solution.

IDERIV = 1
compute approximate first derivative.

IDERIV = 2
compute approximate solution and first derivative.

Constraint: IDERIV = 0, 1 or 2.

4: NWANT – INTEGER *Input*

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

Constraint: 1 ≤ NWANT ≤ N.

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

On exit: an approximation to the first NWANT components of the solution at TWANT if IDERIV = 0 or 2. Otherwise YWANT is not defined.

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

On exit: an approximation to the first NWANT components of the first derivative at TWANT if IDERIV = 1 or 2. Otherwise YPWANT is not defined.

7: 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 D02PFF.

The specification of F is:

```
SUBROUTINE F (T, N, Y, YP, IUSER, RUSER)

INTEGER             N, IUSER(*)
REAL (KIND=nag_wp) T, Y(N), YP(N), RUSER(*)
```

1: T – REAL (KIND=nag_wp) *Input*

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

2: N – INTEGER *Input*

On entry: n, the number of ordinary differential equations in the system to be solved.

3: Y(N) – REAL (KIND=nag_wp) array *Input*

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

4: YP(N) – REAL (KIND=nag_wp) array *Output*

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

5: IUSER($*$) – INTEGER array *User Workspace*
6: RUSER($*$) – REAL (KIND=nag_wp) array *User Workspace*

F is called with the parameters IUSER and RUSER as supplied to D02PSF. You are free to use the arrays IUSER and RUSER to supply information to F as an alternative to using COMMON global variables.

F must either be a module subprogram USEd by, or declared as EXTERNAL in, the (sub)program from which D02PSF is called. Parameters denoted as *Input* must **not** be changed by this procedure.

8: WCOMM(LWCOMM) – REAL (KIND=nag_wp) array *Communication Array*

On entry: this array stores information that can be utilized on subsequent calls to D02PSF.

9: LWCOMM – INTEGER *Input*

On entry: length of WCOMM.

METHOD $= 1$ or -1
LWCOMM must be at least 1.

METHOD $= 2$ or -2
LWCOMM must be at least $\mathrm{N} + \max(\mathrm{N}, 5 \times \mathrm{NWANT})$.

10: IUSER($*$) – INTEGER array *User Workspace*
11: RUSER($*$) – REAL (KIND=nag_wp) array *User Workspace*

IUSER and RUSER are not used by D02PSF, but are passed directly to F and may be used to pass information to this routine as an alternative to using COMMON global variables.

12: IWSAV(130) – INTEGER array *Communication Array*
13: RWSAV($32 \times \mathrm{N} + 350$) – REAL (KIND=nag_wp) array *Communication Array*

On entry: these must be the same arrays supplied in a previous call D02PFF. They must remain unchanged between calls.

On exit: information about the integration for use on subsequent calls to D02PFF, D02PSF or other associated routines.

14: 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 $= 1$

$|\mathrm{METHOD}| = 3$ in setup, but interpolation is not available for this method. Either use $|\mathrm{METHOD}| = 2$ in setup or use reset routine to force the integrator to step to particular points.

On entry, a previous call to the setup routine has not been made or the communication arrays have become corrupted, or a catastrophic error has already been detected elsewhere.
You cannot continue integrating the problem.

On entry, IDERIV $= \langle value \rangle$.
Constraint: IDERIV $= 0$, 1 or 2.

On entry, LWCOMM $= \langle value \rangle$, N $= \langle value \rangle$ and NWANT $= \langle value \rangle$.
Constraint: for $|\mathrm{METHOD}| = 2$, $\mathrm{LWCOMM} \geq \mathrm{N} + 5\mathrm{NWANT}$.

On entry, LWCOMM = ⟨*value*⟩.
Constraint: For |METHOD| = 1, LWCOMM $\geq$ 1.

On entry, N = ⟨*value*⟩, but the value passed to the setup routine was N = ⟨*value*⟩.

On entry, NWANT = ⟨*value*⟩ and N = ⟨*value*⟩.
Constraint: 1 $\leq$ NWANT $\leq$ N.

You cannot call this routine after the integrator has returned an error.

You cannot call this routine before you have called the step integrator.

You cannot call this routine when you have specified, in the setup routine, that the range integrator will be used.

7 Accuracy

The computed values will be of a similar accuracy to that computed by D02PFF.

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. D02PFF is used to integrate the problem one step at a time and D02PSF 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 low order Runge–Kutta method (METHOD = −1) is also used with tolerances TOL = 1.0E−3 and TOL = 1.0E−4 in turn so that solutions may be compared.

9.1 Program Text

```
!   D02PSF Example Program Text
!   Mark 24 Release. NAG Copyright 2012.

    Module d02psfe_mod

!     D02PSF 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        :: tol0 = 1.0E-3_nag_wp
      Integer, Parameter                   :: n = 2, nin = 5, nout = 6,        &
                                              npts = 16, nwant = 1
      Integer, Parameter                   :: lrwsav = 350 + 32*n
    Contains
      Subroutine f(t,n,y,yp,iuser,ruser)

!       .. Scalar Arguments ..
```

```
        Real (Kind=nag_wp), Intent (In)      :: t
        Integer, Intent (In)                 :: n
!       .. Array Arguments ..
        Real (Kind=nag_wp), Intent (Inout)   :: ruser(*)
        Real (Kind=nag_wp), Intent (In)      :: y(n)
        Real (Kind=nag_wp), Intent (Out)     :: yp(n)
        Integer, Intent (Inout)              :: iuser(*)
!       .. Executable Statements ..
        yp(1) = y(2)
        yp(2) = -y(1)
        Return
      End Subroutine f
    End Module d02psfe_mod

    Program d02psfe

!     D02PSF Example Main Program

!     .. Use Statements ..
      Use nag_library, Only: d02pff, d02pqf, d02psf, d02ptf, nag_wp
      Use d02psfe_mod, Only: f, lrwsav, n, nin, nout, npts, nwant, tol0
!     .. Implicit None Statement ..
      Implicit None
!     .. Local Scalars ..
      Real (Kind=nag_wp)                     :: hnext, hstart, tend, tinc, tnow, &
                                                tol, tstart, twant, waste
      Integer                                :: fevals, i, ideriv, ifail,        &
                                                lwcomm, method, stepcost, stepsok
!     .. Local Arrays ..
      Real (Kind=nag_wp)                     :: ruser(1), thresh(n), yinit(n),   &
                                                ynow(n), ypnow(n),               &
                                                ypwant(nwant), ywant(nwant)
      Real (Kind=nag_wp), Allocatable        :: rwsav(:), wcomm(:)
      Integer                                :: iuser(1), iwsav(130)
!     .. Intrinsic Procedures ..
      Intrinsic                              :: real
!     .. Executable Statements ..
      Write (nout,*) 'D02PSF Example Program Results'
!     Skip heading in data file
      Read (nin,*)

      lwcomm = n + 5*nwant
      Allocate (rwsav(lrwsav),wcomm(lwcomm))

      wcomm(1:lwcomm) = 0.0_nag_wp

!     Set initial conditions and input for D02PQF

      Read (nin,*) method
      Read (nin,*) tstart, tend
      Read (nin,*) yinit(1:n)
      Read (nin,*) hstart
      Read (nin,*) thresh(1:n)

!     Set output control

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

      tol = tol0*10.0_nag_wp
      Do i = 1, 2
        tol = tol*0.1_nag_wp

!       Set up integration.
        ifail = 0
        Call d02pqf(n,tstart,tend,yinit,tol,thresh,method,hstart,iwsav,rwsav, &
          ifail)

        Write (nout,99999) tol
        Write (nout,99998)
        Write (nout,99997) tstart, yinit(1:n)
```

```
!       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.

integ:  Do While (tnow<tend)

!         Integrate one step to tnow.
          ifail = 0
          Call d02pff(f,n,tnow,ynow,ypnow,iuser,ruser,iwsav,rwsav,ifail)

!         Interpolate at required additional points up to tnow.
interp:   Do While (twant<=tnow)

!           Interpolate and print solution at t = twant.

            ideriv = 2
            ifail = 0
            Call d02psf(n,twant,ideriv,nwant,ywant,ypwant,f,wcomm,lwcomm, &
              iuser,ruser,iwsav,rwsav,ifail)
            Write (nout,99997) twant, ywant(1), ypwant(1)

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

        End Do integ

!       Get integration statistics.
        ifail = 0
        Call d02ptf(fevals,stepcost,waste,stepsok,hnext,iwsav,rwsav,ifail)

        Write (nout,99996) fevals

      End Do
99999 Format (/' Calculation with TOL = ',1P,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 d02psfe
```

9.2 Program Data

```
D02PSF Example Program Data
   2  -1                              : n, method
   0.0  6.28318530717958647692        : tstart, tend
   0.0  1.0                           : yinit(1:n)
   0.0                                : hstart
   1.0E-8  1.0E-8                     : thresh(1:n)
```

9.3 Program Results

```
 D02PSF Example Program Results

 Calculation with TOL =  1.0E-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     0.3826
  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   152

Calculation with TOL =  1.0E-04

   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   231
```