NAG Library Routine Document

D02PRF

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

D02PRF resets the end point in an integration performed by D02PFF and D02PGF.

2 Specification

```
SUBROUTINE D02PRF (TENDNU, IWSAV, RWSAV, IFAIL)
INTEGER            IWSAV(130), IFAIL
REAL (KIND=nag_wp) TENDNU, RWSAV(350)
```

3 Description

D02PRF and its associated routines (D02PFF, D02PGF, D02PHF, D02PJF, D02PQF, D02PSF, 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

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

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

D02PRF is used to reset the final value of the independent variable, t_f, when the integration is already underway. It can be used to extend or reduce the range of integration. The new value must be beyond the current value of the independent variable (as returned in TNOW by D02PFF or D02PGF) in the current direction of integration. It is much more efficient to use D02PRF for this purpose than to use D02PQF which involves the overhead of a complete restart of the integration.

If you want to change the direction of integration then you must restart by a call to 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 Arguments

1: TENDNU – REAL (KIND=nag_wp) *Input*

On entry: the new value for t_f.

Constraint: sign(TENDNU – TNOW) = sign(TEND – TSTART), where TSTART and TEND are as supplied in the previous call to D02PQF and TNOW is returned by the preceding call to D02PFF or D02PGF (i.e., integration must proceed in the same direction as before). TENDNU must be distinguishable from TNOW for the method and the ***machine precision*** being used.

2: IWSAV(130) – INTEGER array *Communication Array*

3: RWSAV(350) – REAL (KIND=nag_wp) array *Communication Array*

Note: the communication array RWSAV used by the other routines in the suite must be used here however, only the first 350 elements will be referenced.

On entry: these must be the same arrays supplied in a previous call to D02PFF or D02PGF. They must remain unchanged between calls.

On exit: information about the integration for use on subsequent calls to D02PFF or D02PGF or other associated routines.

4: 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, 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, TENDNU is not beyond TNOW (step integrator) in the direction of integration. The direction is negative, TENDNU $= \langle value \rangle$ and TNOW $= \langle value \rangle$.

On entry, TENDNU is not beyond TNOW (step integrator) in the direction of integration. The direction is positive, TENDNU $= \langle value \rangle$ and TNOW $= \langle value \rangle$.

On entry, TENDNU is too close to TNOW (step integrator). Their difference is $\langle value \rangle$, but this quantity must be at least $\langle value \rangle$.

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

You cannot call this routine before you have called the setup routine.

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

You cannot call this routine when the range integrator has been used.

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

Not applicable.

8 Parallelism and Performance

D02PRF is not threaded in any implementation.

9 Further Comments

None.

10 Example

This example integrates a two body problem. The equations for the coordinates $(x(t), y(t))$ of one body as functions of time t in a suitable frame of reference are

$$x'' = -\frac{x}{r^3}$$

$$y'' = -\frac{y}{r^3}, \quad r = \sqrt{x^2 + y^2}.$$

The initial conditions

$$x(0) = 1 - \epsilon, \quad x'(0) = 0$$
$$y(0) = 0, \qquad y'(0) = \sqrt{\frac{1+\epsilon}{1-\epsilon}}$$

lead to elliptic motion with $0 < \epsilon < 1$. $\epsilon = 0.7$ is selected and the system of ODEs is reposed as

$$y_1' = y_3$$

$$y_2' = y_4$$

$$y_3' = -\frac{y_1}{r^3}$$

$$y_4' = -\frac{y_2}{r^3}$$

over the range $[0, 6\pi]$. Relative error control is used with threshold values of 1.0E−10 for each solution component and compute the solution at intervals of length π across the range using D02PRF to reset the end of the integration range. A high-order Runge–Kutta method (METHOD $= -3$) is also used with tolerances TOL $=$ 1.0E−4 and TOL $=$ 1.0E−5 in turn so that the solutions may be compared.

10.1 Program Text

```
!   D02PRF Example Program Text
!   Mark 26 Release. NAG Copyright 2016.

    Module d02prfe_mod

!     D02PRF 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                             :: f
!     .. Parameters ..
      Real (Kind=nag_wp), Parameter, Public :: tol0 = 1.0E-4_nag_wp
```

```
      Integer, Parameter, Public       :: n = 4, nin = 5, nout = 6, npts = 6
      Integer, Parameter, Public       :: 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(*)
!       .. Local Scalars ..
        Real (Kind=nag_wp)             :: r
!       .. Intrinsic Procedures ..
        Intrinsic                      :: sqrt
!       .. Executable Statements ..
        r = sqrt(y(1)**2+y(2)**2)
        yp(1) = y(3)
        yp(2) = y(4)
        yp(3) = -y(1)/r**3
        yp(4) = -y(2)/r**3
        Return
      End Subroutine f
    End Module d02prfe_mod

    Program d02prfe

!     D02PRF Example Main Program

!     .. Use Statements ..
      Use nag_library, Only: d02pff, d02pqf, d02prf, d02ptf, nag_wp
      Use d02prfe_mod, Only: f, lrwsav, n, nin, nout, npts, tol0
!     .. Implicit None Statement ..
      Implicit None
!     .. Local Scalars ..
      Real (Kind=nag_wp)               :: hnext, hstart, tendnu, tfinal, tinc, &
                                          tnow, tol, tstart, waste
      Integer                          :: fevals, i, ifail, method, stepcost,  &
                                          stepsok
!     .. Local Arrays ..
      Real (Kind=nag_wp)               :: ruser(1)
      Real (Kind=nag_wp), Allocatable  :: rwsav(:), thresh(:), yinit(:),       &
                                          ynow(:), ypnow(:)
      Integer                          :: iuser(1)
      Integer, Allocatable             :: iwsav(:)
!     .. Intrinsic Procedures ..
      Intrinsic                        :: real
!     .. Executable Statements ..
      Write (nout,*) 'D02PRF Example Program Results'
!     Skip heading in data file
      Read (nin,*)
!     n: number of differential equations
      Allocate (thresh(n),iwsav(130),rwsav(lrwsav),ynow(n),ypnow(n),yinit(n))

!     Set initial conditions and input for D02PQF

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

!     Set output control

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

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

```
        tendnu = tstart + tinc

!       ifail: behaviour on error exit
!              =0 for hard exit, =1 for quiet-soft, =-1 for noisy-soft
        ifail = 0
        Call d02pqf(n,tstart,tendnu,yinit,tol,thresh,method,hstart,iwsav,      &
          rwsav,ifail)

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

tstep:  Do
          ifail = 0
          Call d02pff(f,n,tnow,ynow,ypnow,iuser,ruser,iwsav,rwsav,ifail)

          If (tnow==tendnu) Then
            Write (nout,99997) tnow, ynow(1:n)

            If (tnow>=tfinal) Then
              Exit tstep
            End If
            tendnu = tendnu + tinc
            Call d02prf(tendnu,iwsav,rwsav,ifail)
          End If

        End Do tstep

        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         y2         y3         y4',/)
99997 Format (1X,F6.3,4(3X,F8.4))
99996 Format (/,' Cost of the integration in evaluations of F is',I6)

    End Program d02prfe
```

10.2 Program Data

```
D02PRF Example Program Data
  -3                                                    : method
   0.0                        18.84955592153875943      : tstart, tfinal
   0.3      0.0      0.0      2.3804761284761666599     : yinit
   0.0                                                   : hstart
   1.0E-10  1.0E-10  1.0E-10  1.0E-10                    : thresh
```

10.3 Program Results

```
 D02PRF Example Program Results

 Calculation with TOL =  1.0E-04

    t         y1         y2         y3         y4

  0.000     0.3000     0.0000     0.0000     2.3805
  3.142    -1.7000     0.0000    -0.0000    -0.4201
  6.283     0.3000    -0.0000     0.0001     2.3805
  9.425    -1.7000     0.0000    -0.0000    -0.4201
 12.566     0.3000    -0.0003     0.0016     2.3805
 15.708    -1.7001     0.0001    -0.0001    -0.4201
 18.850     0.3000    -0.0010     0.0045     2.3805

 Cost of the integration in evaluations of F is   571

 Calculation with TOL =  1.0E-05
```

```
   t          y1          y2          y3          y4

 0.000      0.3000      0.0000      0.0000      2.3805
 3.142     -1.7000     -0.0000      0.0000     -0.4201
 6.283      0.3000      0.0000     -0.0000      2.3805
 9.425     -1.7000      0.0000     -0.0000     -0.4201
12.566      0.3000     -0.0001      0.0004      2.3805
15.708     -1.7000      0.0000     -0.0000     -0.4201
18.850      0.3000     -0.0003      0.0012      2.3805

Cost of the integration in evaluations of F is   748
```