NAG Library Routine Document

D02PWF

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

D02PWF resets the end point in an integration performed by D02PDF.

2 Specification

```
SUBROUTINE D02PWF (TENDNU, IFAIL)
INTEGER             IFAIL
REAL (KIND=nag_wp) TENDNU
```

3 Description

D02PWF and its associated routines (D02PDF, D02PVF, D02PXF, 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

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

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

D02PWF 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 D02PDF) in the current direction of integration. It is much more efficient to use D02PWF for this purpose than to use D02PVF 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 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: 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 D02PVF and TNOW is returned by the preceding call to D02PDF. TENDNU must be distinguishable from TNOW for the method and the ***machine precision*** being used.

2: 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 TENDNU was detected or an invalid call to D02PWF was made, for example without a previous call to the integration routine D02PDF. You cannot continue integrating the problem.

IFAIL $= -99$

An unexpected error has been triggered by this routine. Please contact NAG.

See Section 3.8 in the Essential Introduction for further information.

IFAIL $= -399$

Your licence key may have expired or may not have been installed correctly.

See Section 3.7 in the Essential Introduction for further information.

IFAIL $= -999$

Dynamic memory allocation failed.

See Section 3.6 in the Essential Introduction for further information.

7 Accuracy

Not applicable.

8 Parallelism and Performance

Not applicable.

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 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 D02PWF 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. The value of π is obtained by using X01AAF.

Note that the length of TOL = 1.0E−4 and WORK is large enough for any valid combination of input arguments to D02PVF.

10.1 Program Text

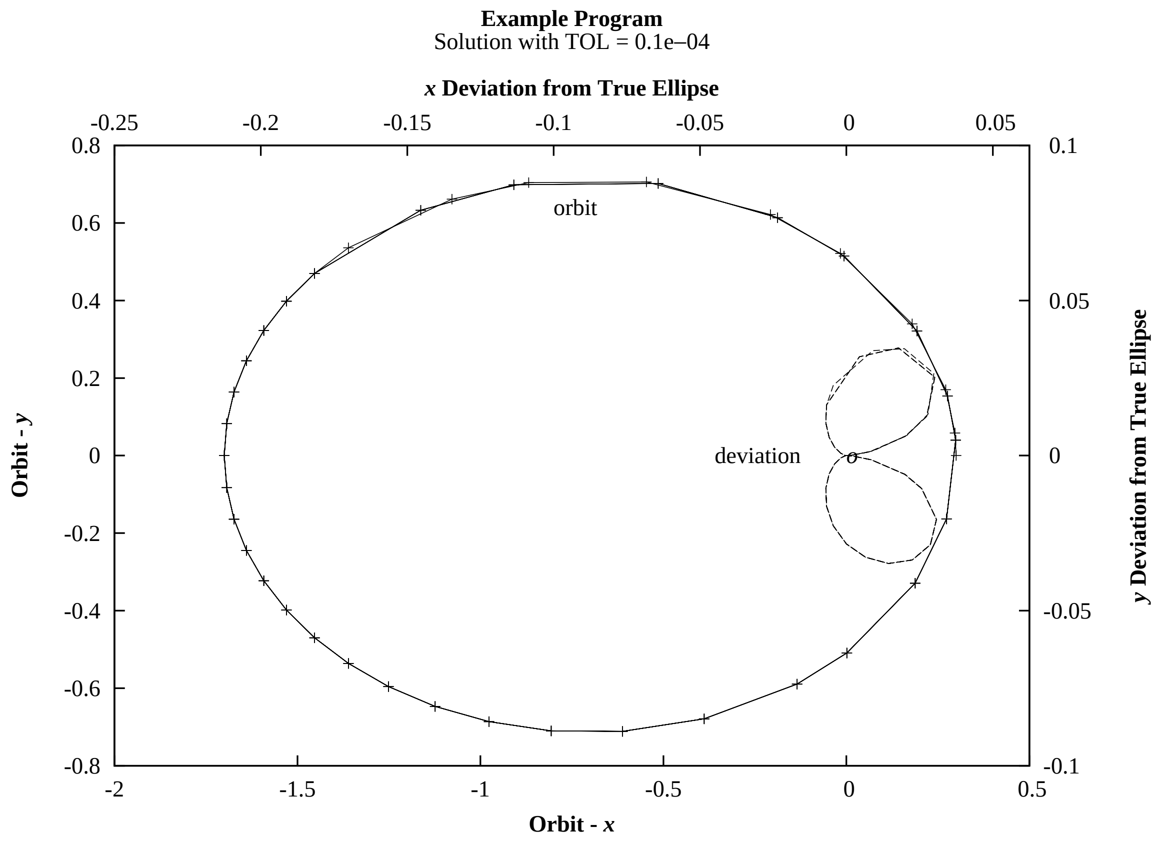
```
!   D02PWF Example Program Text
!   Mark 25 Release. NAG Copyright 2014.

    Module d02pwfe_mod

!     D02PWF 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 :: tol1 = 1.0E-4_nag_wp
      Real (Kind=nag_wp), Parameter, Public :: tol2 = 1.0E-5_nag_wp
      Integer, Parameter, Public           :: neq = 4, nin = 5, nout = 6,      &
                                              npts = 6
      Integer, Parameter, Public           :: 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(*)
!       .. 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 d02pwfe_mod

    Program d02pwfe

!     D02PWF Example Main Program

!     .. Use Statements ..
      Use nag_library, Only: d02pdf, d02pvf, d02pwf, d02pyf, nag_wp
      Use d02pwfe_mod, Only: f, lenwrk, neq, nin, nout, npts, tol1, tol2
!     .. Implicit None Statement ..
      Implicit None
!     .. Local Scalars ..
      Real (Kind=nag_wp)                  :: hnext, hstart, tendnu, tfinal,   &
                                             tinc, tnow, tol, tstart, waste
      Integer                             :: i, ifail, method, stpcst,        &
                                             stpsok, totf
      Logical                             :: errass
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable     :: thres(:), work(:), ynow(:),      &
                                             ypnow(:), ystart(:)
!     .. Intrinsic Procedures ..
      Intrinsic                           :: real
!     .. Executable Statements ..
      Write (nout,*) 'D02PWF Example Program Results'
!     Skip heading in data file
      Read (nin,*)
!     neq: number of differential equations
      Read (nin,*) method
      Allocate (thres(neq),work(lenwrk),ynow(neq),ypnow(neq),ystart(neq))

!     Set initial conditions and input for D02PVF

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

!     Set output control

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

      Do i = 1, 2
        If (i==1) tol = tol1
        If (i==2) tol = tol2
        tendnu = tstart + tinc

!       ifail: behaviour on error exit
!              =0 for hard exit, =1 for quiet-soft, =-1 for noisy-soft
        ifail = 0
        Call d02pvf(neq,tstart,ystart,tendnu,tol,thres,method,'Complex Task', &
          errass,hstart,work,lenwrk,ifail)

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

loop:   Do
          ifail = 0
          Call d02pdf(f,tnow,ynow,ypnow,work,ifail)

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

            If (tnow>=tfinal) Exit loop
            tendnu = tendnu + tinc
            Call d02pwf(tendnu,ifail)
          End If
        End Do loop

        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         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 d02pwfe
```

10.2 Program Data

```
D02PWF Example Program Data
   3                                                 : method
   0.0  1.88495559215387594307E1                     : tstart, tfinal
   0.3  0.0  0.0  2.38047614284761666599             : ystart
   0.0                                               : hstart
   1.0E-10  1.0E-10  1.0E-10  1.0E-10                : thres
   .FALSE.                                           : errass
```

10.3 Program Results

```
 D02PWF Example Program Results

 Calculation with TOL =  0.1E-03

    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 =  0.1E-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    -0.0000     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
```


Example Program
Solution with TOL = 0.1e–04
x Deviation from True Ellipse
-0.25
-0.2
-0.15
-0.1
-0.05
0
0.05
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
0.1
0.05
0
-0.05
-0.1
orbit
deviation
Orbit - *y*
y Deviation from True Ellipse
-2
-1.5
-1
-0.5
0
0.5
Orbit - *x*