NAG Library Routine Document

D02PZF

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

D02PZF provides details about global error assessment computed during an integration with either D02PCF or D02PDF.

2 Specification

```
SUBROUTINE D02PZF (RMSERR, ERRMAX, TERRMX, WORK, IFAIL)

INTEGER             IFAIL
REAL (KIND=nag_wp) RMSERR(*), ERRMAX, TERRMX, WORK(*)
```

3 Description

D02PZF and its associated routines (D02PCF, D02PDF, D02PVF, D02PWF, D02PXF and D02PYF) 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.

After a call to D02PCF or D02PDF, D02PZF can be called for information about error assessment, if this assessment was specified in the setup routine D02PVF. A more accurate 'true' solution $\hat{y}$ is computed in a secondary integration. The error is measured as specified in D02PVF for local error control. At each step in the primary integration, an average magnitude μ_i of component y_i is computed, and the error in the component is

$$\frac{|y_i - \hat{y}_i|}{\max(\mu_i, \text{THRES}(i))}.$$

It is difficult to estimate reliably the true error at a single point. For this reason the RMS (root-mean-square) average of the estimated global error in each solution component is computed. This average is taken over all steps from the beginning of the integration through to the current integration point. If all has gone well, the average errors reported will be comparable to TOL (see D02PVF). The maximum error seen in any component in the integration so far and the point where the maximum error first occurred are also reported.

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

Note: the dimension of the array RMSERR must be at least n.

On exit: RMSERR(i) approximates the RMS average of the true error of the numerical solution for the ith solution component, for $i = 1, 2, \ldots, n$. The average is taken over all steps from the beginning of the integration to the current integration point.

2: ERRMAX – REAL (KIND=nag_wp) *Output*

On exit: the maximum weighted approximate true error taken over all solution components and all steps.

3: TERRMX – REAL (KIND=nag_wp) *Output*

On exit: the first value of the independent variable where an approximate true error attains the maximum value, ERRMAX.

4: WORK($*$) – REAL (KIND=nag_wp) array *Input*

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 D02PCF or D02PDF and **must** remain unchanged between calls.

5: 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: $\mathrm{IFAIL} = 0$ unless the routine detects an error or a warning has been flagged (see Section 6).

6 Error Indicators and Warnings

If on entry $\mathrm{IFAIL} = 0$ or -1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

Errors or warnings detected by the routine:

$\mathrm{IFAIL} = 1$

An invalid call to D02PZF has been made, for example without a previous call to D02PCF or D02PDF, or without error assessment having been specified in a call to D02PVF. You cannot continue integrating the problem.

7 Accuracy

Not applicable.

8 Further Comments

If the integration has proceeded 'well' and the problem is smooth enough, stable and not too difficult then the values returned in the arguments RMSERR and ERRMAX should be comparable to the value of TOL specified in the prior call to D02PVF.

9 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

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

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 $[0, 3\pi]$. Relative error control is used with threshold values of 1.0E$-$10 for each solution component and a high-order Runge–Kutta method (METHOD $= 3$) with tolerance TOL $=$ 1.0E$-$6. 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. Note also, for illustration purposes since it is not necessary for this problem, this example integrates to the end of the range regardless of efficiency concerns (i.e., returns from D02PCF with IFAIL $=$ 2, 3 or 4).

9.1 Program Text

```
!   D02PZF Example Program Text
!   Mark 24 Release. NAG Copyright 2012.

    Module d02pzfe_mod

!     D02PZF Example Program Module:
!            Parameters and User-defined Routines

!     .. Use Statements ..
      Use nag_library, Only: nag_wp
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Integer, Parameter                   :: neq = 4, nin = 5, nout = 6
      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(*)
!       .. 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 d02pzfe_mod

    Program d02pzfe

!     D02PZF Example Main Program
!     .. Use Statements ..
      Use nag_library, Only: d02pcf, d02pvf, d02pyf, d02pzf, nag_wp
      Use d02pzfe_mod, Only: f, lenwrk, neq, nin, nout
!     .. Implicit None Statement ..
      Implicit None
!     .. Local Scalars ..
      Real (Kind=nag_wp)                    :: errmax, hnext, hstart, tend,     &
                                               terrmx, tgot, tol, tstart,       &
                                               twant, waste
      Integer                               :: ifail, method, stpcst, stpsok,   &
                                               totf
      Logical                               :: errass
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable       :: rmserr(:), thres(:), work(:),    &
                                               ygot(:), ymax(:), ypgot(:),      &
                                               ystart(:)
!     .. Executable Statements ..
      Write (nout,*) 'D02PZF Example Program Results'
!     Skip heading in data file
      Read (nin,*)
!     neq: number of differential equations
      Read (nin,*) method
      Allocate (rmserr(neq),thres(neq),work(lenwrk),ygot(neq),ymax(neq), &
        ypgot(neq),ystart(neq))

!     Set initial conditions and input for D02PVF

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

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

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

      twant = tend

integ: Do
        ifail = -1
        Call d02pcf(f,twant,tgot,ygot,ypgot,ymax,work,ifail)

        If (ifail<2 .Or. ifail>4) Then
          Exit integ
        End If

      End Do integ

      If (ifail==0) Then

!       Print solution.
        Write (nout,99997) tgot, ygot(1:neq)

!       Compute and print error estimates.
        ifail = 0
        Call d02pzf(rmserr,errmax,terrmx,work,ifail)

        Write (nout,99996) rmserr(1:neq)
        Write (nout,99995) errmax, terrmx

        ifail = 0
        Call d02pyf(totf,stpcst,waste,stpsok,hnext,ifail)
```

```
         Write (nout,99994) totf
      End If

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 (/' Componentwise error assessment'/9X,4(2X,E9.2))
99995 Format (/' Worst global error observed was ',E9.2, &
         ' - it occurred at T = ',F6.3)
99994 Format (/' Cost of the integration in evaluations of F is',I6)
    End Program d02pzfe
```

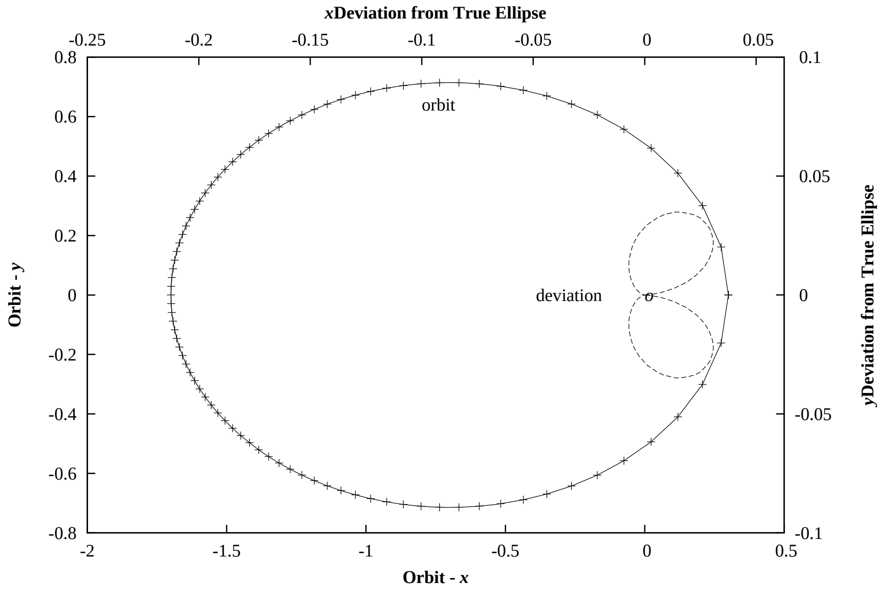
9.2 Program Data

```
D02PZF Example Program Data
   3                                         : neq, method
   0.0  9.42477796076937971538               : tstart, tend
   0.3  0.0  0.0  2.38047614284761666599     : ystart(1:neq)
   0.0  1.0E-6                               : hstart, tol
   1.0E-10  1.0E-10  1.0E-10  1.0E-10        : thres(1:neq)
   .TRUE.                                    : errass
```

9.3 Program Results

```
 D02PZF Example Program Results

 Calculation with TOL =  0.1E-05

    t         y1         y2         y3         y4

  0.000     0.3000     0.0000     0.0000     2.3805
  9.425    -1.7000     0.0000    -0.0000    -0.4201

 Componentwise error assessment
           0.38E-05   0.71E-05   0.69E-05   0.21E-05

 Worst global error observed was  0.34E-04 - it occurred at T =  6.302

 Cost of the integration in evaluations of F is  1361
```

Example Program
Solution to a Two-body Problem using High-order Runge-Kutta


*x*Deviation from True Ellipse
orbit
deviation
Orbit - *y*
*y*Deviation from True Ellipse
Orbit - *x*