NAG Library Routine Document

D02PDF

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

D02PDF is a one-step routine for solving an initial value problem for a first-order system of ordinary differential equations using Runge–Kutta methods.

2 Specification

```
SUBROUTINE D02PDF (F, TNOW, YNOW, YPNOW, WORK, IFAIL)

INTEGER             IFAIL
REAL (KIND=nag_wp) TNOW, YNOW(*), YPNOW(*), WORK(*)
EXTERNAL            F
```

3 Description

D02PDF and its associated routines (D02PVF, D02PWF, D02PXF, D02PYF and D02PZF) solve an 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 is designed to be used in complicated tasks when solving systems of ordinary differential equations. You must first call D02PVF to specify the problem and how it is to be solved. Thereafter you (repeatedly) call D02PDF to take one integration step at a time from TSTART in the direction of TEND (as specified in D02PVF). In this manner D02PDF returns an approximation to the solution YNOW and its derivative YPNOW at successive points TNOW. If D02PDF encounters some difficulty in taking a step, the integration is not advanced and the routine returns with the same values of TNOW, YNOW and YPNOW as returned on the previous successful step. D02PDF tries to advance the integration as far as possible subject to passing the test on the local error and not going past TEND.

In the call to D02PVF you can specify either the first step size for D02PDF to attempt or that it compute automatically an appropriate value. Thereafter D02PDF estimates an appropriate step size for its next step. This value and other details of the integration can be obtained after any call to D02PDF by a call to D02PYF. The local error is controlled at every step as specified in D02PVF. If you wish to assess the true error, you must set ERRASS = .TRUE. in the call to D02PVF. This assessment can be obtained after any call to D02PDF by a call to D02PZF.

If you want answers at specific points there are two ways to proceed:

(i) The more efficient way is to step past the point where a solution is desired, and then call D02PXF to get an answer there. Within the span of the current step, you can get all the answers you want at very little cost by repeated calls to D02PXF. This is very valuable when you want to find where something happens, e.g., where a particular solution component vanishes. You cannot proceed in this way with METHOD = 3.

(ii) The other way to get an answer at a specific point is to set TEND to this value and integrate to TEND. D02PDF will not step past TEND, so when a step would carry it past, it will reduce the step size so as to produce an answer at TEND exactly. After getting an answer there (TNOW = TEND), you can reset TEND to the next point where you want an answer, and repeat. TEND could be reset by a call to D02PVF, but you should not do this. You should use D02PWF instead because it is both easier to use and much more efficient. This way of getting answers at specific points can be used with

any of the available methods, but it is the only way with METHOD $= 3$. It can be inefficient. Should this be the case, the code will bring the matter to your attention.

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: 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.

The specification of F is:

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

In the description of the parameters of D02PDF 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 $i = 1, 2, \ldots, n$.

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

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

F must either be a module subprogram USEd by, or declared as EXTERNAL in, the (sub)program from which D02PDF is called. Parameters denoted as *Input* must **not** be changed by this procedure.

2: TNOW – REAL (KIND=nag_wp) *Output*

On exit: t, the value of the independent variable at which a solution has been computed.

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

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

On exit: an approximation to the solution at TNOW. The local error of the step to TNOW was no greater than permitted by the specified tolerances (see D02PVF).

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

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

On exit: an approximation to the derivative of the solution at TNOW.

5: 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 D02PVF. It **must** remain unchanged between calls.

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

6: 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, because for this routine the values of the output parameters may be useful even if IFAIL $\neq 0$ on exit, the recommended value is -1. **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 call to D02PDF was made, for example without a previous call to the setup routine D02PVF. You cannot continue integrating the problem.

IFAIL $= 2$

D02PDF is being used inefficiently because the step size has been reduced drastically many times to obtain answers at many points TEND. If you really need the solution at this many points, you should use D02PXF to obtain the answers inexpensively. If you need to change from METHOD $= 3$ to do this, restart the integration from TNOW, YNOW by a call to D02PVF. If you wish to continue as before, call D02PDF again. The monitor of this kind of inefficiency will be reset automatically so that the integration can proceed.

IFAIL $= 3$

A considerable amount of work has been expended in the (primary) integration. This is measured by counting the number of calls to F. At least 5000 calls have been made since the last time this counter was reset. Calls to F in a secondary integration for global error assessment (when ERRASS $=$ .TRUE. in the call to D02PVF) are not counted in this total. The integration was interrupted. If you wish to continue on towards TEND, just call D02PDF again. The counter measuring work will be reset to zero automatically.

IFAIL $= 4$

It appears that this problem is stiff. The methods implemented in D02PDF can solve such problems, but they are inefficient. You should change to another code based on methods appropriate for stiff problems. The integration was interrupted. If you want to continue on towards TEND, just call D02PDF again. The stiffness monitor will be reset automatically.

IFAIL $= 5$

It does not appear possible to achieve the accuracy specified by TOL and THRES in the call to D02PVF with the precision available on the computer being used and with this value of METHOD. You cannot continue integrating this problem. A larger value for METHOD, if possible, will permit greater accuracy with this precision. To increase METHOD and/or continue with larger values of TOL and/or THRES, restart the integration from TNOW, YNOW by a call to D02PVF.

IFAIL $= 6$

(This error exit can only occur if ERRASS = .TRUE. in the call to D02PVF.) The global error assessment may not be reliable beyond the current integration point TNOW. This may occur because either too little or too much accuracy has been requested or because $f(t, y)$ is not smooth enough for values of t just beyond TNOW and current values of the solution y. The integration cannot be continued. This return does not mean that you cannot integrate past TNOW, rather that you cannot do it with ERRASS = .TRUE.. However, it may also indicate problems with the primary integration.

7 Accuracy

The accuracy of integration is determined by the parameters TOL and THRES in a prior call to D02PVF. Note that only the local error at each step is controlled by these parameters. The error estimates obtained are not strict bounds but are usually reliable over one step. Over a number of steps the overall error may accumulate in various ways, depending on the properties of the differential system.

8 Further Comments

If D02PDF returns with IFAIL $= 5$ and the accuracy specified by TOL and THRES is really required then you should consider whether there is a more fundamental difficulty. For example, the solution may contain a singularity. In such a region the solution components will usually be large in magnitude. Successive output values of YNOW should be monitored with the aim of trapping the solution before the singularity. In any case numerical integration cannot be continued through a singularity, and analytical treatment may be necessary.

Performance statistics are available after any return from D02PDF (except when IFAIL $= 1$) by a call to D02PYF. If ERRASS = .TRUE. in the call to D02PVF, global error assessment is available after any return from D02PDF (except when IFAIL $= 1$) by a call to D02PZF.

After a failure with IFAIL $= 5$ or 6 the diagnostic routines D02PYF and D02PZF may be called only once.

If D02PDF returns with IFAIL $= 4$ then it is advisable to change to another code more suited to the solution of stiff problems. D02PDF will not return with IFAIL $= 4$ if the problem is actually stiff but it is estimated that integration can be completed using less function evaluations than already computed.

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 $y_2 = 1.0$. We use relative error control with threshold values of 1.0E$-$8 for each solution component and print the solution at each integration step across the range. We use a medium order Runge–Kutta method (METHOD $= 2$) with tolerances TOL $=$ 1.0E$-$4 and TOL $=$ 1.0E$-$5 in turn so that we may compare the solutions. 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.

See also the example programs for D02PWF and D02PXF.

9.1 Program Text

```
!   D02PDF Example Program Text
!   Mark 24 Release. NAG Copyright 2012.

    Module d02pdfe_mod

!     D02PDF 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-4_nag_wp
      Real (Kind=nag_wp), Parameter        :: tol2 = 1.0E-5_nag_wp
      Integer, Parameter                   :: neq = 2, 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(*)
!       .. Executable Statements ..
        yp(1) = y(2)
        yp(2) = -y(1)
        Return
      End Subroutine f
    End Module d02pdfe_mod

    Program d02pdfe

!     D02PDF Example Main Program

!     .. Use Statements ..
      Use nag_library, Only: d02pdf, d02pvf, d02pyf, nag_wp
      Use d02pdfe_mod, Only: f, lenwrk, neq, nin, nout, tol1, tol2
!     .. Implicit None Statement ..
      Implicit None
!     .. Local Scalars ..
      Real (Kind=nag_wp)                   :: hnext, hstart, tend, 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(:)
!     .. Executable Statements ..
      Write (nout,*) 'D02PDF Example Program Results'
!     Skip heading in data file
      Read (nin,*)
      Read (nin,*) method
      Allocate (thres(neq),work(lenwrk),ynow(neq),ypnow(neq),ystart(neq))

!     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

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

```
!       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,'Complex Task', &
          errass,hstart,work,lenwrk,ifail)

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

loop:   Do
          ifail = -1
          Call d02pdf(f,tnow,ynow,ypnow,work,ifail)

          If (ifail==0) Then
            Write (nout,99997) tnow, ynow(1:neq)
            If (tnow>=tend) Exit loop
          Else
            Exit loop
          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'/)
99997 Format (1X,F6.3,2(3X,F8.4))
99996 Format (/' Cost of the integration in evaluations of F is',I6)
    End Program d02pdfe
```

9.2 Program Data

```
D02PDF Example Program Data
   2                              : method
   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

```
 D02PDF Example Program Results

 Calculation with TOL =  0.1E-03

    t         y1        y2

  0.000     0.0000     1.0000
  0.785     0.7071     0.7071
  1.519     0.9987     0.0513
  2.282     0.7573    -0.6531
  2.911     0.2285    -0.9735
  3.706    -0.5348    -0.8450
  4.364    -0.9399    -0.3414
  5.320    -0.8209     0.5710
  5.802    -0.4631     0.8863
  6.283     0.0000     1.0000

 Cost of the integration in evaluations of F is    78

 Calculation with TOL =  0.1E-04

    t         y1        y2
```

```
 0.000     0.0000     1.0000
 0.393     0.3827     0.9239
 0.785     0.7071     0.7071
 1.416     0.9881     0.1538
 1.870     0.9557    -0.2943
 2.204     0.8062    -0.5916
 2.761     0.3711    -0.9286
 3.230    -0.0880    -0.9961
 3.587    -0.4304    -0.9026
 4.022    -0.7710    -0.6368
 4.641    -0.9974    -0.0717
 5.152    -0.9049     0.4256
 5.521    -0.6903     0.7235
 5.902    -0.3718     0.9283
 6.283     0.0000     1.0000

 Cost of the integration in evaluations of F is   118
```