NAG Library Routine Document

D02PCF

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

D02PCF solves an initial value problem for a first-order system of ordinary differential equations using Runge–Kutta methods.

2 Specification

```
SUBROUTINE D02PCF (F, TWANT, TGOT, YGOT, YPGOT, YMAX, WORK, IFAIL)
INTEGER             IFAIL
REAL (KIND=nag_wp) TWANT, TGOT, YGOT(*), YPGOT(*), YMAX(*), WORK(*)
EXTERNAL            F
```

3 Description

D02PCF and its associated routines (D02PVF, 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

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

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

D02PCF is designed for the usual task, namely to compute an approximate solution at a sequence of points. You must first call D02PVF to specify the problem and how it is to be solved. Thereafter you call D02PCF repeatedly with successive values of TWANT, the points at which you require the solution, in the range from TSTART to TEND (as specified in D02PVF). In this manner D02PCF returns the point at which it has computed a solution TGOT (usually TWANT), the solution there (YGOT) and its derivative (YPGOT). If D02PCF encounters some difficulty in taking a step toward TWANT, then it returns the point of difficulty (TGOT) and the solution and derivative computed there (YGOT and YPGOT, respectively).

In the call to D02PVF you can specify either the first step size for D02PCF to attempt or that it compute automatically an appropriate value. Thereafter D02PCF estimates an appropriate step size for its next step. This value and other details of the integration can be obtained after any call to D02PCF 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 D02PCF by a call to D02PZF.

For more complicated tasks, you are referred to routines D02PDF, D02PWF and D02PXF, all of which are used by D02PCF.

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 D02PCF 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 D02PCF is called. Parameters denoted as *Input* must **not** be changed by this procedure.

2: TWANT – REAL (KIND=nag_wp) *Input*

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

Constraint: TWANT must be closer to TEND than the previous value of TGOT (or TSTART on the first call to D02PCF); see D02PVF for a description of TSTART and TEND. TWANT must not lie beyond TEND in the direction of integration.

3: TGOT – REAL (KIND=nag_wp) *Output*

On exit: t, the value of the independent variable at which a solution has been computed. On successful exit with IFAIL $= 0$, TGOT will equal TWANT. On exit with IFAIL > 1, a solution has still been computed at the value of TGOT but in general TGOT will not equal TWANT.

4: YGOT$(*)$ – REAL (KIND=nag_wp) array *Input/Output*

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

On entry: on the first call to D02PCF, YGOT need not be set. On all subsequent calls YGOT must remain unchanged.

On exit: an approximation to the true solution at the value of TGOT. At each step of the integration to TGOT, the local error has been controlled as specified in D02PVF. The local error has still been controlled even when TGOT $\neq$ TWANT, that is after a return with IFAIL > 1.

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

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

On exit: an approximation to the first derivative of the true solution at TGOT.

6: YMAX$(*)$ – REAL (KIND=nag_wp) array *Input/Output*

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

On entry: on the first call to D02PCF, YMAX need not be set. On all subsequent calls YMAX must remain unchanged.

On exit: YMAX(i) contains the largest value of $|y_i|$ computed at any step in the integration so far.

7: 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 D02PCF or other associated routines.

8: 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 input value for TWANT was detected or an invalid call to D02PCF was made, for example without a previous call to the setup routine D02PVF. You cannot continue integrating the problem.

IFAIL $= 2$

This return is possible only when METHOD $= 3$ has been selected in the preceding call of D02PVF. D02PCF is being used inefficiently because the step size has been reduced drastically many times to get answers at many values of TWANT. If you really need the solution at this many points, you should change to METHOD $= 2$ because it is (much) more efficient in this situation. To change METHOD, restart the integration from TGOT, YGOT by a call to D02PVF. If you wish to continue with METHOD $= 3$, just call D02PCF again without altering any of the arguments other than IFAIL. 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 the supplied routine 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, so TGOT is not equal to TWANT. If you wish to continue on towards TWANT, just call D02PCF again without altering any of the arguments other than IFAIL. The counter measuring work will be reset to zero automatically.

IFAIL $= 4$

It appears that this problem is stiff. The methods implemented in D02PCF 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 so TGOT is not equal to TWANT.

If you want to continue on towards TWANT, just call D02PCF again without altering any of the arguments other than IFAIL. 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 TGOT, YGOT 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 TGOT. 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 past TGOT and current values of the solution y. The integration cannot be continued. This return does not mean that you cannot integrate past TGOT, rather that you cannot do it with = .TRUE.. However, it may also indicate problems with the primary integration.

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

The accuracy of integration is determined by the parameters TOL and THRES in a prior call to D02PVF (see the routine document for D02PVF for further details and advice). 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 Parallelism and Performance

D02PCF is not threaded by NAG in any implementation.

D02PCF makes calls to BLAS and/or LAPACK routines, which may be threaded within the vendor library used by this implementation. Consult the documentation for the vendor library for further information.

Please consult the X06 Chapter Introduction for information on how to control and interrogate the OpenMP environment used within this routine. Please also consult the Users' Note for your implementation for any additional implementation-specific information.

9 Further Comments

If D02PCF 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 YGOT and YMAX should be monitored (or D02PDF should be used since this takes one integration step at a time) 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 D02PCF by a call to D02PYF. If ERRASS = .TRUE. in the call to D02PVF, global error assessment is available after any return from D02PCF (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 D02PCF returns with IFAIL $= 4$ then it is advisable to change to another code more suited to the solution of stiff problems. D02PCF 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.

10 Example

This example solves the equation

$$y'' = -y, \quad y(0) = 0, \quad y'(0) = 1$$

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 and compute the solution at intervals of length $\pi/4$ across the range. A low-order Runge–Kutta method (METHOD $= 1$, see D02PVF) is also used with tolerances TOL $= 1.0$E-3 and TOL $= 1.0$E-4 in turn so that the solutions can be compared. 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 Section 10 in D02PZF.

10.1 Program Text

```
!   D02PCF Example Program Text
!   Mark 25 Release. NAG Copyright 2014.

    Module d02pcfe_mod

!     D02PCF 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-3_nag_wp
      Real (Kind=nag_wp), Parameter, Public :: tol2 = 1.0E-4_nag_wp
      Integer, Parameter, Public            :: neq = 2, nin = 5, nout = 6,      &
```

```
                                          npts = 8
      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(*)
!       .. Executable Statements ..
        yp(1) = y(2)
        yp(2) = -y(1)
        Return
      End Subroutine f
    End Module d02pcfe_mod

    Program d02pcfe

!     D02PCF Example Main Program

!     .. Use Statements ..
      Use nag_library, Only: d02pcf, d02pvf, d02pyf, nag_wp
      Use d02pcfe_mod, Only: f, lenwrk, neq, nin, nout, npts, tol1, tol2
!     .. Implicit None Statement ..
      Implicit None
!     .. Local Scalars ..
      Real (Kind=nag_wp)                   :: hnext, hstart, tend, tgot, tinc, &
                                              tol, tstart, twant, waste
      Integer                              :: i, ifail, j, method, stpcst,     &
                                              stpsok, totf
      Logical                              :: errass
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable      :: thres(:), work(:), ygot(:),      &
                                              ymax(:), ypgot(:), ystart(:)
!     .. Intrinsic Procedures ..
      Intrinsic                            :: real
!     .. Executable Statements ..
      Write (nout,*) 'D02PCF Example Program Results'
!     Skip heading in data file
      Read (nin,*)
      Read (nin,*) method
      Allocate (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
      Read (nin,*) thres(1:neq)
      Read (nin,*) errass

!     Set control for output

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

loop: Do i = 1, 2
        If (i==1) Then
          tol = tol1
        Else
          tol = tol2
        End If

!       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)
```

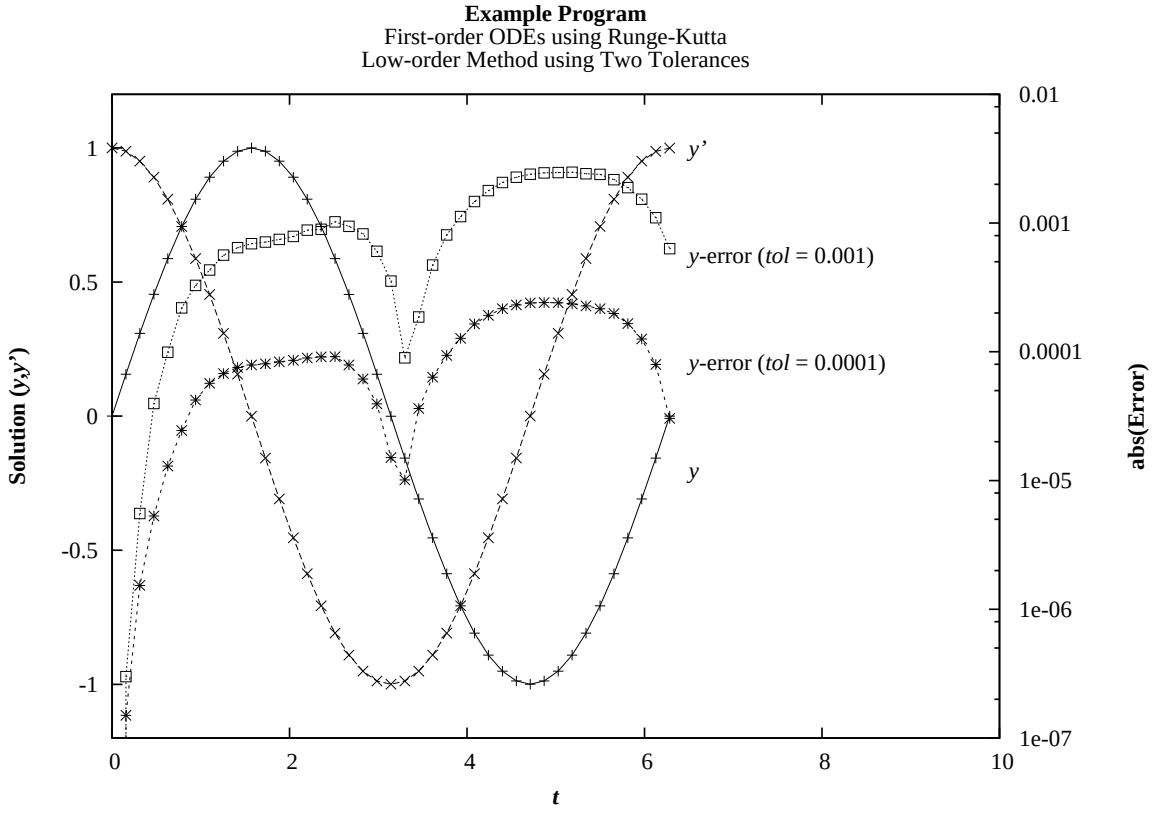
```
        Write (nout,99999) tol
        Write (nout,99998)
        Write (nout,99997) tstart, ystart(1:neq)
        twant = tstart
        Do j = 1, npts
          twant = twant + tinc

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

          Write (nout,99997) tgot, ygot(1:neq)
        End Do

        ifail = 0
        Call d02pyf(totf,stpcst,waste,stpsok,hnext,ifail)
        Write (nout,99996) totf

      End Do loop

99999 Format (/' Calculation with TOL = ',E8.1)
99998 Format (/'    t         y1        y2'/)
99997 Format (1X,F6.3,2(3X,F7.3))
99996 Format (/' Cost of the integration in evaluations of F is',I6)
    End Program d02pcfe
```

10.2 Program Data

```
D02PCF Example Program Data
   1                                : 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
```

10.3 Program Results

```
 D02PCF Example Program Results

 Calculation with TOL =  0.1E-02

    t         y1        y2

  0.000     0.000     1.000
  0.785     0.707     0.707
  1.571     0.999    -0.000
  2.356     0.706    -0.706
  3.142    -0.000    -0.999
  3.927    -0.706    -0.706
  4.712    -0.998     0.000
  5.498    -0.705     0.706
  6.283     0.001     0.997

 Cost of the integration in evaluations of F is   124

 Calculation with TOL =  0.1E-03

    t         y1        y2

  0.000     0.000     1.000
  0.785     0.707     0.707
  1.571     1.000    -0.000
  2.356     0.707    -0.707
  3.142    -0.000    -1.000
  3.927    -0.707    -0.707
  4.712    -1.000     0.000
  5.498    -0.707     0.707
  6.283     0.000     1.000

 Cost of the integration in evaluations of F is   235
```


Example Program
First-order ODEs using Runge-Kutta
Low-order Method using Two Tolerances
Solution (y,y')
abs(Error)
t
y'
y-error (tol = 0.001)
y-error (tol = 0.0001)
y
1
0.5
0
-0.5
-1
0.01
0.001
0.0001
1e-05
1e-06
1e-07
0
2
4
6
8
10