

NAG Library Routine Document

D05BDF

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

D05BDF computes the solution of a weakly singular nonlinear convolution Volterra–Abel integral equation of the second kind using a fractional Backward Differentiation Formulae (BDF) method.

2 Specification

SUBROUTINE D05BDF (CK, CF, CG, INITWT, IORDER, TLIM, TOLNL, NMESH, YN, &
WORK, LWK, NCT, IFAIL)

INTEGER IORDER, NMESH, LWK, NCT(NMESH/32+1), IFAIL
REAL (KIND=nag_wp) CK, CF, CG, TLIM, TOLNL, YN(NMESH), WORK(LWK)
CHARACTER(1) INITWT
EXTERNAL CK, CF, CG

3 Description

D05BDF computes the numerical solution of the weakly singular convolution Volterra–Abel integral equation of the second kind

$$y(t) = f(t) + \frac{1}{\sqrt{\pi}} \int_0^t \frac{k(t-s)}{\sqrt{t-s}} g(s, y(s)) ds, \quad 0 \leq t \leq T. \quad (1)$$

Note the constant $\frac{1}{\sqrt{\pi}}$ in (1). It is assumed that the functions involved in (1) are sufficiently smooth.

The routine uses a fractional BDF linear multi-step method to generate a family of quadrature rules (see D05BYF). The BDF methods available in D05BDF are of orders 4, 5 and 6 (= p say). For a description of the theoretical and practical background to these methods we refer to Lubich (1985) and to Baker and Derakhshan (1987) and Hairer *et al.* (1988) respectively.

The algorithm is based on computing the solution $y(t)$ in a step-by-step fashion on a mesh of equispaced points. The size of the mesh is given by $T/(N-1)$, N being the number of points at which the solution is sought. These methods require $2p-1$ (including $y(0)$) starting values which are evaluated internally. The computation of the lag term arising from the discretization of (1) is performed by fast Fourier transform (FFT) techniques when $N > 32 + 2p - 1$, and directly otherwise. The routine does not provide an error estimate and you are advised to check the behaviour of the solution with a different value of N . An option is provided which avoids the re-evaluation of the fractional weights when D05BDF is to be called several times (with the same value of N) within the same program unit with different functions.

4 References

Baker C T H and Derakhshan M S (1987) FFT techniques in the numerical solution of convolution equations *J. Comput. Appl. Math.* **20** 5–24

Hairer E, Lubich Ch and Schlichte M (1988) Fast numerical solution of weakly singular Volterra integral equations *J. Comput. Appl. Math.* **23** 87–98

Lubich Ch (1985) Fractional linear multistep methods for Abel–Volterra integral equations of the second kind *Math. Comput.* **45** 463–469

5 Arguments

- 1: CK – REAL (KIND=nag_wp) FUNCTION, supplied by the user. *External Procedure*
 CK must evaluate the kernel $k(t)$ of the integral equation (1).

The specification of CK is:

```
FUNCTION CK (T)
REAL (KIND=nag_wp) CK
REAL (KIND=nag_wp) T
```

1: T – REAL (KIND=nag_wp)

Input

On entry: t , the value of the independent variable.

CK must either be a module subprogram USED by, or declared as EXTERNAL in, the (sub) program from which D05BDF is called. Arguments denoted as *Input* must **not** be changed by this procedure.

- 2: CF – REAL (KIND=nag_wp) FUNCTION, supplied by the user. *External Procedure*
 CF must evaluate the function $f(t)$ in (1).

The specification of CF is:

```
FUNCTION CF (T)
REAL (KIND=nag_wp) CF
REAL (KIND=nag_wp) T
```

1: T – REAL (KIND=nag_wp)

Input

On entry: t , the value of the independent variable.

CF must either be a module subprogram USED by, or declared as EXTERNAL in, the (sub) program from which D05BDF is called. Arguments denoted as *Input* must **not** be changed by this procedure.

- 3: CG – REAL (KIND=nag_wp) FUNCTION, supplied by the user. *External Procedure*
 CG must evaluate the function $g(s, y(s))$ in (1).

The specification of CG is:

```
FUNCTION CG (S, Y)
REAL (KIND=nag_wp) CG
REAL (KIND=nag_wp) S, Y
```

1: S – REAL (KIND=nag_wp)

Input

On entry: s , the value of the independent variable.

2: Y – REAL (KIND=nag_wp)

Input

On entry: the value of the solution y at the point S .

CG must either be a module subprogram USED by, or declared as EXTERNAL in, the (sub) program from which D05BDF is called. Arguments denoted as *Input* must **not** be changed by this procedure.

- 4: INITWT – CHARACTER(1) *Input*
On entry: if the fractional weights required by the method need to be calculated by the routine then set INITWT = 'I' (**I**nitial call).
 If INITWT = 'S' (**S**ubsequent call), the routine assumes the fractional weights have been computed on a previous call and are stored in WORK.
Constraint: INITWT = 'I' or 'S'.
Note: when D05BDF is re-entered with the value of INITWT = 'S', the values of NMESH, IORDER and the contents of WORK **must not** be changed.
- 5: IORDER – INTEGER *Input*
On entry: p , the order of the BDF method to be used.
Suggested value: IORDER = 4.
Constraint: $4 \leq \text{IORDER} \leq 6$.
- 6: TLIM – REAL (KIND=nag_wp) *Input*
On entry: the final point of the integration interval, T .
Constraint: TLIM > $10 \times$ *machine precision*.
- 7: TOLNL – REAL (KIND=nag_wp) *Input*
On entry: the accuracy required for the computation of the starting value and the solution of the nonlinear equation at each step of the computation (see Section 9).
Suggested value: TOLNL = $\sqrt{\epsilon}$ where ϵ is the *machine precision*.
Constraint: TOLNL > $10 \times$ *machine precision*.
- 8: NMESH – INTEGER *Input*
On entry: N , the number of equispaced points at which the solution is sought.
Constraint: NMESH = $2^m + 2 \times \text{IORDER} - 1$, where $m \geq 1$.
- 9: YN(NMESH) – REAL (KIND=nag_wp) array *Output*
On exit: YN(i) contains the approximate value of the true solution $y(t)$ at the point $t = (i - 1) \times h$, for $i = 1, 2, \dots, \text{NMESH}$, where $h = \text{TLIM}/(\text{NMESH} - 1)$.
- 10: WORK(LWK) – REAL (KIND=nag_wp) array *Communication Array*
On entry: if INITWT = 'S', WORK must contain fractional weights computed by a previous call of D05BDF (see description of INITWT).
On exit: contains fractional weights which may be used by a subsequent call of D05BDF.
- 11: LWK – INTEGER *Input*
On entry: the dimension of the array WORK as declared in the (sub)program from which D05BDF is called.
Constraint: LWK $\geq (2 \times \text{IORDER} + 6) \times \text{NMESH} + 8 \times \text{IORDER}^2 - 16 \times \text{IORDER} + 1$.
- 12: NCT(NMESH/32 + 1) – INTEGER array *Workspace*
- 13: 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, IORDER < 4 or IORDER > 6 ,
 or TLIM $\leq 10 \times$ *machine precision*,
 or INITWT \neq 'I' or 'S',
 or INITWT = 'S' on the first call to D05BDF,
 or TOLNL $\leq 10 \times$ *machine precision*,
 or NMESH $\neq 2^m + 2 \times$ IORDER $- 1$, $m \geq 1$,
 or LWK < $(2 \times$ IORDER $+ 6) \times$ NMESH $+ 8 \times$ IORDER² $- 16 \times$ IORDER $+ 1$.

IFAIL = 2

The routine cannot compute the $2p - 1$ starting values due to an error solving the system of nonlinear equations. Relaxing the value of TOLNL and/or increasing the value of NMESH may overcome this problem (see Section 9 for further details).

IFAIL = 3

The routine cannot compute the solution at a specific step due to an error in the solution of the nonlinear equation (2). Relaxing the value of TOLNL and/or increasing the value of NMESH may overcome this problem (see Section 9 for further details).

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

The accuracy depends on NMESH and TOLNL, the theoretical behaviour of the solution of the integral equation and the interval of integration. The value of TOLNL controls the accuracy required for computing the starting values and the solution of (2) at each step of computation. This value can affect the accuracy of the solution. However, for most problems, the value of $\sqrt{\epsilon}$, where ϵ is the *machine precision*, should be sufficient.

8 Parallelism and Performance

D05BDF is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

D05BDF 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

In solving (1), initially, D05BDF computes the solution of a system of nonlinear equations for obtaining the $2p - 1$ starting values. C05QDF is used for this purpose. When a failure with IFAIL = 2 occurs (which corresponds to an error exit from C05QDF), you are advised to either relax the value of TOLNL or choose a smaller step size by increasing the value of NMESH. Once the starting values are computed successfully, the solution of a nonlinear equation of the form

$$Y_n - \alpha g(t_n, Y_n) - \Psi_n = 0, \quad (2)$$

is required at each step of computation, where Ψ_n and α are constants. D05BDF calls C05AXF to find the root of this equation.

If a failure with IFAIL = 3 occurs (which corresponds to an error exit from C05AXF), you are advised to relax the value of the TOLNL or choose a smaller step size by increasing the value of NMESH.

If a failure with IFAIL = 2 or 3 persists even after adjustments to TOLNL and/or NMESH then you should consider whether there is a more fundamental difficulty. For example, the problem is ill-posed or the functions in (1) are not sufficiently smooth.

10 Example

In this example we solve the following integral equations

$$y(t) = \sqrt{t} + \frac{3}{8}\pi t^2 - \int_0^t \frac{1}{\sqrt{t-s}} [y(s)]^3 ds, \quad 0 \leq t \leq 7,$$

with the solution $y(t) = \sqrt{t}$, and

$$y(t) = (3-t)\sqrt{t} - \int_0^t \frac{1}{\sqrt{t-s}} \exp\left(s(1-s)^2 - [y(s)]^2\right) ds, \quad 0 \leq t \leq 5,$$

with the solution $y(t) = (1-t)\sqrt{t}$. In the above examples, the fourth-order BDF is used, and NMESH is set to $2^6 + 7$.

10.1 Program Text

```
! D05BDF Example Program Text
! Mark 26 Release. NAG Copyright 2016.
! Module d05bdfe_mod

! D05BDF 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 :: cf1, cf2, cg1, cg2, ck1, ck2
! .. Parameters ..
```

```

Integer, Parameter, Public      :: iorder = 4
Integer, Parameter, Public      :: nmesh = 2**6 + 2*iorder - 1
Integer, Parameter, Public      :: nout = 6
Integer, Parameter, Public      :: lct = nmesh/32 + 1
Integer, Parameter, Public      :: lwk = (2*iorder+6)*nmesh + 8*iorder* &
                                iorder - 16*iorder + 1
Contains
  Function ck1(t)

!      .. Use Statements ..
      Use nag_library, Only: x01aaf
!      .. Function Return Value ..
      Real (Kind=nag_wp)          :: ck1
!      .. Scalar Arguments ..
      Real (Kind=nag_wp), Intent (In) :: t
!      .. Local Scalars ..
      Real (Kind=nag_wp)          :: pi
!      .. Intrinsic Procedures ..
      Intrinsic                   :: sqrt
!      .. Executable Statements ..
      ck1 = -sqrt(x01aaf(pi))

      Return

End Function ck1
Function cf1(t)

!      .. Use Statements ..
      Use nag_library, Only: x01aaf
!      .. Function Return Value ..
      Real (Kind=nag_wp)          :: cf1
!      .. Scalar Arguments ..
      Real (Kind=nag_wp), Intent (In) :: t
!      .. Local Scalars ..
      Real (Kind=nag_wp)          :: pi
!      .. Intrinsic Procedures ..
      Intrinsic                   :: sqrt
!      .. Executable Statements ..
      cf1 = sqrt(t) + (3.0_nag_wp/8.0_nag_wp)*t*t*x01aaf(pi)

      Return

End Function cf1
Function cg1(s,y)

!      .. Function Return Value ..
      Real (Kind=nag_wp)          :: cg1
!      .. Scalar Arguments ..
      Real (Kind=nag_wp), Intent (In) :: s, y
!      .. Executable Statements ..
      cg1 = y*y*y

      Return

End Function cg1
Function ck2(t)

!      .. Use Statements ..
      Use nag_library, Only: x01aaf
!      .. Function Return Value ..
      Real (Kind=nag_wp)          :: ck2
!      .. Scalar Arguments ..
      Real (Kind=nag_wp), Intent (In) :: t
!      .. Local Scalars ..
      Real (Kind=nag_wp)          :: pi
!      .. Intrinsic Procedures ..
      Intrinsic                   :: sqrt
!      .. Executable Statements ..
      ck2 = -sqrt(x01aaf(pi))

      Return

```

```

End Function ck2
Function cf2(t)

! .. Function Return Value ..
Real (Kind=nag_wp)          :: cf2
! .. Scalar Arguments ..
Real (Kind=nag_wp), Intent (In) :: t
! .. Intrinsic Procedures ..
Intrinsic                   :: sqrt
! .. Executable Statements ..
cf2 = (3.0_nag_wp-t)*sqrt(t)

Return

End Function cf2
Function cg2(s,y)

! .. Function Return Value ..
Real (Kind=nag_wp)          :: cg2
! .. Scalar Arguments ..
Real (Kind=nag_wp), Intent (In) :: s, y
! .. Intrinsic Procedures ..
Intrinsic                   :: exp
! .. Executable Statements ..
cg2 = exp(s*(1.0_nag_wp-s)*(1.0_nag_wp-s)-y*y)

Return

End Function cg2
End Module d05bdfe_mod
Program d05bdfe

! D05BDF Example Main Program

! .. Use Statements ..
Use nag_library, Only: d05bdf, nag_wp, x02ajf
Use d05bdfe_mod, Only: cf1, cf2, cg1, cg2, ck1, ck2, iorder, lct, lwk, &
    nmesh, nout

! .. Implicit None Statement ..
Implicit None
! .. Local Scalars ..
Real (Kind=nag_wp)          :: h, t, tlim, tolnl
Integer                     :: i, ifail
! .. Local Arrays ..
Real (Kind=nag_wp)          :: work(lwk), yn(nmesh)
Integer                     :: nct(lct)
! .. Intrinsic Procedures ..
Intrinsic                   :: mod, real, sqrt
! .. Executable Statements ..
Write (nout,*) 'D05BDF Example Program Results'

tlim = 7.0_nag_wp
tolnl = sqrt(x02ajf())
h = tlim/real(nmesh-1,kind=nag_wp)

ifail = 0
Call d05bdf(ck1,cf1,cg1,'Initial',iorder,tlim,tolnl,nmesh,yn,work,lwk, &
    nct,ifail)

Write (nout,*)
Write (nout,*) 'Example 1'
Write (nout,*)
Write (nout,99998) h
Write (nout,*)
Write (nout,*) '          T          Approximate'
Write (nout,*) '          Solution '
Write (nout,*)

Do i = 1, nmesh
    t = real(i-1,kind=nag_wp)*h

```

```

      If (mod(i,5)==1) Then
        Write (nout,99999) t, yn(i)
      End If

End Do

tlim = 5.0_nag_wp
h = tlim/real(nmesh-1,kind=nag_wp)

ifail = 0
Call d05bdf(ck2,cf2,cg2,'Subsequent',iorder,tlim,toln1,nmesh,yn,work,
           lwk,nct,ifail)

Write (nout,*)
Write (nout,*) 'Example 2'
Write (nout,*)
Write (nout,99998) h
Write (nout,*)
Write (nout,*) '      T      Approximate'
Write (nout,*) '      Solution '
Write (nout,*)

Do i = 1, nmesh
  t = real(i-1,kind=nag_wp)*h

  If (mod(i,7)==1) Then
    Write (nout,99999) t, yn(i)
  End If

End Do

99999 Format (1X,F8.4,F15.4)
99998 Format (' The stepsize h = ',F8.4)
End Program d05bdf

```

10.2 Program Data

None.

10.3 Program Results

D05BDF Example Program Results

Example 1

The stepsize h = 0.1000

T	Approximate Solution
0.0000	0.0000
0.5000	0.7071
1.0000	1.0000
1.5000	1.2247
2.0000	1.4142
2.5000	1.5811
3.0000	1.7321
3.5000	1.8708
4.0000	2.0000
4.5000	2.1213
5.0000	2.2361
5.5000	2.3452
6.0000	2.4495
6.5000	2.5495
7.0000	2.6458

Example 2

The stepsize h = 0.0714

T	Approximate Solution
0.0000	0.0000
0.5000	0.3536
1.0000	0.0000
1.5000	-0.6124
2.0000	-1.4142
2.5000	-2.3717
3.0000	-3.4641
3.5000	-4.6771
4.0000	-6.0000
4.5000	-7.4246
5.0000	-8.9443
