NAG Fortran 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,
     1                   WORK, LWK, NCT, IFAIL)
      INTEGER            IORDER, NMESH, LWK, NCT(NMESH/32+1), IFAIL
      double precision   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

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

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 selected by you 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 theoretical and practical background related 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 Parameters

1: CK – ***double precision*** FUNCTION, supplied by the user *External Procedure*

CK must evaluate the kernel $k(t)$ of the integral equation (1).

Its specification is:

```
double precision FUNCTION CK (T)
double precision              T
```

1: T – ***double precision*** *Input*

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

CK must be declared as EXTERNAL in the (sub)program from which D05BDF is called. Parameters denoted as *Input* must **not** be changed by this procedure.

2: CF – ***double precision*** FUNCTION, supplied by the user *External Procedure*

CF must evaluate the function $f(t)$ in (1).

Its specification is:

```
double precision FUNCTION CF (T)
double precision              T
```

1: T – ***double precision*** *Input*

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

CF must be declared as EXTERNAL in the (sub)program from which D05BDF is called. Parameters denoted as *Input* must **not** be changed by this procedure.

3: CG – ***double precision*** FUNCTION, supplied by the user *External Procedure*

CG must evaluate the function $g(s, y(s))$ in (1).

Its specification is:

```
double precision FUNCTION CG (S, Y)
double precision              S, Y
```

1: S – ***double precision*** *Input*

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

2: Y – ***double precision*** *Input*

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

CG must be declared as EXTERNAL in the (sub)program from which D05BDF is called. Parameters 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), then 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.

Constraint: $4 \leq \text{IORDER} \leq 6$.

Suggested value: IORDER = 4.

6: TLIM – ***double precision*** *Input*

On entry: the final point of the integration interval, T.

Constraint: TLIM $> 10 \times$ ***machine precision***.

7: TOLNL – ***double precision*** *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 8).

Constraint: TOLNL $> 10 \times$ ***machine precision***.

Suggested value: TOLNL $= \sqrt{\epsilon}$ where ϵ is the ***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) – ***double precision*** 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 $i = 1, 2, \ldots, \text{NMESH}$, where $h = \text{TLIM}/(\text{NMESH} - 1)$.

10: WORK(LWK) – ***double precision*** array *Communication Array*
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 parameter you should refer to Chapter P01 for details.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

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.**

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 $\neq 2^m + 2 \times \mathrm{IORDER} - 1, m \geq 1$,
or LWK $< (2 \times \mathrm{IORDER} + 6) \times \mathrm{NMESH} + 8 \times \mathrm{IORDER}^2 - 16 \times \mathrm{IORDER} + 1$.

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 8 for further details).

IFAIL $= 3$

The routine cannot compute the solution at a specific step due to an error in the solution of single nonlinear equation (2). Relaxing the value of TOLNL and/or increasing the value of NMESH may overcome this problem (see Section 8 for further details).

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.

In general, for the choice of BDF method, you are recommended to use the fourth-order BDF formula (i.e., IORDER $= 4$).

8 Further Comments

In solving (1), initially, D05BDF computes the solution of a system of nonlinear equations for obtaining the $2p - 1$ starting values. C05NDF is used for this purpose. When a failure with IFAIL $= 2$ occurs (which corresponds to an error exit from C05NDF), 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

$$Y_n - \alpha g(t_n, Y_n) - \Psi_n = 0, \tag{2}$$

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.

9 Example

In this example we solve the following integral equations

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

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

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

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

9.1 Program Text

```
*     D05BDF Example Program Text
*     Mark 16 Release. NAG Copyright 1992.
*     .. Parameters ..
      INTEGER          NOUT
      PARAMETER        (NOUT=6)
      INTEGER          IORDER, NMESH, LCT, LWK
      PARAMETER        (IORDER=4,NMESH=2**6+2*IORDER-1,LCT=NMESH/32+1,
     +                 LWK=(2*IORDER+6)*NMESH+8*IORDER*IORDER-16*IORDER+
     +                 1)
*     .. Local Scalars ..
      DOUBLE PRECISION ERR, ERRMAX, H, HI1, SOLN, T, TLIM, TOLNL
      INTEGER          I, IFAIL
*     .. Local Arrays ..
      DOUBLE PRECISION WORK(LWK), YN(NMESH)
      INTEGER          NCT(LCT)
*     .. External Functions ..
      DOUBLE PRECISION CF1, CF2, CG1, CG2, CK1, CK2, X02AJF
      EXTERNAL         CF1, CF2, CG1, CG2, CK1, CK2, X02AJF
*     .. External Subroutines ..
      EXTERNAL         D05BDF
*     .. Intrinsic Functions ..
      INTRINSIC        ABS, DBLE, MOD, SQRT
*     .. Executable Statements ..
      WRITE (NOUT,*) 'D05BDF Example Program Results'
      WRITE (NOUT,*)
      IFAIL = 0
      TLIM = 7.0D0
      TOLNL = SQRT(X02AJF())
      H = TLIM/(NMESH-1)
*
      CALL D05BDF(CK1,CF1,CG1,'Initial',IORDER,TLIM,TOLNL,NMESH,YN,WORK,
     +            LWK,NCT,IFAIL)
*
      WRITE (NOUT,*) 'Example 1'
      WRITE (NOUT,*)
      WRITE (NOUT,99997) H
      WRITE (NOUT,*)
      WRITE (NOUT,*) '     T        Approximate'
      WRITE (NOUT,*) '                Solution '
      WRITE (NOUT,*)
*
      ERRMAX = 0.0D0
      DO 20 I = 1, NMESH
         HI1 = DBLE(I-1)*H
         ERR = ABS(YN(I)-SQRT(HI1))
         IF (ERR.GT.ERRMAX) THEN
            ERRMAX = ERR
            T = HI1
            SOLN = YN(I)
         END IF
         IF (MOD(I,5).EQ.1) WRITE (NOUT,99998) HI1, YN(I)
   20 CONTINUE
```

```
      WRITE (NOUT,*)
      WRITE (NOUT,99999) ERRMAX, T, SOLN
*
*
      TLIM = 5.0D0
      H = TLIM/(NMESH-1)
*
      CALL D05BDF(CK2,CF2,CG2,'Subsequent',IORDER,TLIM,TOLNL,NMESH,YN,
     +            WORK,LWK,NCT,IFAIL)
*
      WRITE (NOUT,*)
      WRITE (NOUT,*) 'Example 2'
      WRITE (NOUT,*)
      WRITE (NOUT,99997) H
      WRITE (NOUT,*)
      WRITE (NOUT,*) '     T        Approximate'
      WRITE (NOUT,*) '                 Solution '
      WRITE (NOUT,*)
*
      ERRMAX = 0.0D0
      DO 40 I = 1, NMESH
         HI1 = DBLE(I-1)*H
         ERR = ABS(YN(I)-(1.0D0-HI1)*SQRT(HI1))
         IF (ERR.GT.ERRMAX) THEN
            ERRMAX = ERR
            T = HI1
            SOLN = YN(I)
         END IF
         IF (MOD(I,7).EQ.1) WRITE (NOUT,99998) HI1, YN(I)
   40 CONTINUE
      WRITE (NOUT,*)
      WRITE (NOUT,99999) ERRMAX, T, SOLN
*
      STOP
*
99999 FORMAT (' The maximum absolute error, ',E10.2,', occurred at T =',
     +       F8.4,/' with solution ',F8.4)
99998 FORMAT (1X,F8.4,F15.4)
99997 FORMAT (' The stepsize h = ',F8.4)
      END
*
*
      DOUBLE PRECISION FUNCTION CK1(T)
*     .. Scalar Arguments ..
      DOUBLE PRECISION                T
*     .. Local Scalars ..
      DOUBLE PRECISION                PI
*     .. External Functions ..
      DOUBLE PRECISION                X01AAF
      EXTERNAL                        X01AAF
*     .. Intrinsic Functions ..
      INTRINSIC                       SQRT
*     .. Executable Statements ..
      CK1 = -SQRT(X01AAF(PI))
      RETURN
      END
*
*
      DOUBLE PRECISION FUNCTION CF1(T)
*     .. Scalar Arguments ..
      DOUBLE PRECISION                T
*     .. Local Scalars ..
      DOUBLE PRECISION                PI
*     .. External Functions ..
      DOUBLE PRECISION                X01AAF
      EXTERNAL                        X01AAF
*     .. Intrinsic Functions ..
      INTRINSIC                       SQRT
*     .. Executable Statements ..
      CF1 = SQRT(T) + (3.0D0/8.0D0)*T*T*X01AAF(PI)
      RETURN
```

```
      END
*
*
      DOUBLE PRECISION FUNCTION CG1(S,Y)
*     .. Scalar Arguments ..
      DOUBLE PRECISION              S, Y
*     .. Executable Statements ..
      CG1 = Y*Y*Y
      RETURN
      END
*
*
      DOUBLE PRECISION FUNCTION CK2(T)
*     .. Scalar Arguments ..
      DOUBLE PRECISION              T
*     .. Local Scalars ..
      DOUBLE PRECISION              PI
*     .. External Functions ..
      DOUBLE PRECISION              X01AAF
      EXTERNAL                      X01AAF
*     .. Intrinsic Functions ..
      INTRINSIC                     SQRT
*     .. Executable Statements ..
      CK2 = -SQRT(X01AAF(PI))
      RETURN
      END
*
*
      DOUBLE PRECISION FUNCTION CF2(T)
*     .. Scalar Arguments ..
      DOUBLE PRECISION              T
*     .. Intrinsic Functions ..
*
      INTRINSIC                     SQRT
*     .. Executable Statements ..
      CF2 = (3.0D0-T)*SQRT(T)
      RETURN
      END
*
*
      DOUBLE PRECISION FUNCTION CG2(S,Y)
*     .. Scalar Arguments ..
      DOUBLE PRECISION              S, Y
*     .. Intrinsic Functions ..
      INTRINSIC                     EXP
*     .. Executable Statements ..
      CG2 = EXP(S*(1.0D0-S)*(1.0D0-S)-Y*Y)
      RETURN
      END
```

9.2 Program Data

None.

9.3 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

The maximum absolute error,   0.11E-07, occurred at T =  1.2000
with solution   1.0954

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

The maximum absolute error,   0.52E-07, occurred at T =  4.3571
with solution  -7.0076
```