NAG Fortran Library Routine Document

D05BEF

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

D05BEF computes the solution of a weakly singular nonlinear convolution Volterra–Abel integral equation of the first kind using a fractional Backward Differentiation Formulae (BDF) method.

2 Specification

```
 SUBROUTINE D05BEF(CK, CF, CG, INITWT, IORDER, TLIM, TOLNL, NMESH, YN,
1                  WORK, LWK, NCT, IFAIL)
 INTEGER           IORDER, NMESH, LWK, NCT(NMESH/32+1), IFAIL
 real              CK, CF, CG, TLIM, TOLNL, YN(NMESH), WORK(LWK)
 CHARACTER*1       INITWT
 EXTERNAL          CK, CF, CG
```

3 Description

D05BEF computes the numerical solution of the weakly singular convolution Volterra–Abel integral equation of the first kind

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

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

$$f(t) = t^{\beta} w(t) \quad \text{with} \quad \beta > -\tfrac{1}{2}, \tag{2}$$

then the solution $y(t)$ is unique and has the form $y(t) = t^{\beta-1/2} z(t)$, (see Lubich (1987)). It is evident from (1) that $f(0) = 0$. The user is required to provide the value of $y(t)$ at $t = 0$. If $y(0)$ is unknown, Section 8 gives a description of how an approximate value can be obtained.

The routine uses a fractional BDF linear multi-step method selected by the user to generate a family of quadrature rules (see D05BYF). The BDF methods available in D05BEF are of orders 4, 5 and 6 ($= p$ say). For a description of the theoretical and practical background related to these methods we refer to Lubich (1987), 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 $\mathrm{T}/(N-1)$, N being the number of points at which the solution is sought. These methods require $2p-2$ 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 users 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 D05BEF is to be called several times (with the same value of N) within the same program 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

Gorenflo R and Pfeiffer A (1991) On analysis and discretization of nonlinear Abel integral equations of first kind *Acta Math. Vietnam* **16** 211–262

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 (1987) Fractional linear multistep methods for Abel–Volterra integral equations of the first kind *IMA J. Numer. Anal* **7** 97–106

5 Parameters

1: CK – ***real*** FUNCTION, supplied by the user. *External Procedure*

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

Its specification is:

```
real FUNCTION CK(T)
real             T
```

1: T – ***real*** *Input*

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

CK must be declared as EXTERNAL in the (sub)program from which D05BEF is called. Parameters denoted as *Input* must **not** be changed by this procedure.

2: CF – ***real*** FUNCTION, supplied by the user. *External Procedure*

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

Its specification is:

```
real FUNCTION CF(T)
real             T
```

1: T – ***real*** *Input*

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

CF must be declared as EXTERNAL in the (sub)program from which D05BEF is called. Parameters denoted as *Input* must **not** be changed by this procedure.

3: CG – ***real*** FUNCTION, supplied by the user. *External Procedure*

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

Its specification is:

```
real FUNCTION CG(S, Y)
real             S, Y
```

1: S – ***real*** *Input*

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

2: Y – ***real*** *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 D05BEF 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 by a previous call and are stored in WORK.

Constraint: INITWT = 'I' or 'S'.

Note: When D05BEF is re-entered with a value of INITWT = 'S', the values of NMESH, IORDER and the contents of WORK **must** not be changed.

5: IORDER – INTEGER *Input*

On entry: the order of the BDF method to be used, p.

Constraint: $4 \leq$ IORDER ≤ 6.

Suggested value: IORDER = 4.

6: TLIM – ***real*** *Input*

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

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

7: TOLNL – ***real*** *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: the number of equispaced points, N, at which the solution is sought.

Constraint: NMESH $= 2^m + 2 \times$ IORDER $- 1$, where $m \geq 1$.

9: YN(NMESH) – ***real*** array *Input/Output*

On entry: YN(1) must contain the value of $y(t)$ at $t = 0$ (see Section 8).

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,...,NMESH, where $h =$ TLIM/(NMESH $- 1$).

10: WORK(LWK) – ***real*** array *Workspace*
11: LWK – INTEGER *Input*

On entry: the dimension of the array WORK as declared in the (sub)program from which D05BEF is called.

Constraint: LWK $\geq (2 \times$ IORDER $+ 6) \times$ NMESH $+ 8 \times$ IORDER$^2 - 16 \times$ 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. Users who are unfamiliar with this parameter 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, for users 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 D05BEF,
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 $+ 8 \times$ IORDER$^2 - 16 \times$ IORDER $+ 1$.

IFAIL $= 2$

The routine cannot compute the $2p - 2$ starting values due to an error in 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 the single nonlinear equation (3). 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 (3) 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, the user is recommended to use the fourth-order BDF formula (i.e., IORDER $= 4$).

8 Further Comments

Also when solving (1) the initial value $y(0)$ is required. This value may be computed from the limit relation (see Gorenflo and Pfeiffer (1991))

$$\frac{-2}{\sqrt{\pi}} k(0) g(0, y(0)) = \lim_{t \to 0} \frac{f(t)}{\sqrt{t}}. \tag{3}$$

If the value of the above limit is known then by solving the nonlinear equation (3) an approximation to $y(0)$ can be computed. If the value of the above limit is not known, an approximation should be provided. Following the analysis presented in Gorenflo and Pfeiffer (1991), the following pth-order approximation can be used:

$$\lim_{t \to 0} \frac{f(t)}{\sqrt{t}} \simeq \frac{f(h^P)}{h^{P/2}}. \tag{4}$$

However, it must be emphasized that the approximation in (4) may result in an amplification of the rounding errors and hence users are advised (if possible) to determine $\lim_{t \to 0} \frac{f(t)}{\sqrt{t}}$ by analytical methods.

Also when solving (1), initially, D05BEF computes the solution of a system of nonlinear equation for obtaining the $2p-2$ starting values. C05NDF is used for this purpose. If a failure with IFAIL $= 2$ occurs (corresponding to an error exit from C05NDF), users 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

$$Y_n - \alpha g(t_n, Y_n) - \Psi_n = 0, \tag{5}$$

is required at each step of computation, where Ψ_n and α are constants. D05BEF calls C05AXF to find the root of this equation.

When a failure with IFAIL $= 3$ occurs (which corresponds to an error exit from C05AXF), users are advised to either 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 the user 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

We solve the following integral equations.

Example 1:

The density of the probability that a Brownian motion crosses a one-sided moving boundary $a(t)$ before time t, satisfies the integral equation (see Hairer *et al.* (1988))

$$-\frac{1}{\sqrt{t}}\exp\Big(\tfrac{1}{2} - \{a(t)\}^2/t\Big) + \int_0^t \frac{\exp\Big(-\tfrac{1}{2}\{a(t)-a(s)\}^2/(t-s)\Big)}{\sqrt{t-s}} y(s)\,ds = 0, \quad 0 \le t \le 7.$$

In the case of a straight line $a(t) = 1 + t$, the exact solution is known to be

$$y(t) = \frac{1}{\sqrt{2\pi t^3}}\exp\{-(1+t)^2/2t\}$$

Example 2:

In this example we consider the equation

$$-\frac{2\log(\sqrt{1+t}+\sqrt{t})}{\sqrt{1+t}} + \int_0^t \frac{y(s)}{\sqrt{t-s}}ds = 0, \quad 0 \le t \le 5.$$

The solution is given by $y(t) = \frac{1}{1+t}$.

In the above examples, the fourth-order BDF is used, and NMESH is set to $2^6 + 7$.

9.1 Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     D05BEF 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 ..
      real             ERR, ERRMAX, H, HI1, SOLN, T, TLIM, TOLNL
      INTEGER          I, IFAIL
*     .. Local Arrays ..
      real             WORK(LWK), YN(NMESH)
```

```
      INTEGER          NCT(LCT)
*     .. External Functions ..
      real             CF1, CF2, CG1, CG2, CK1, CK2, SOL1, SOL2, X02AJF
      EXTERNAL         CF1, CF2, CG1, CG2, CK1, CK2, SOL1, SOL2, X02AJF
*     .. External Subroutines ..
      EXTERNAL         D05BEF
*     .. Intrinsic Functions ..
      INTRINSIC        ABS, real, MOD, SQRT
*     .. Executable Statements ..
      WRITE (NOUT,*) 'D05BEF Example Program Results'
      WRITE (NOUT,*)
      IFAIL = 0
      TLIM = 7.0e0
      TOLNL = SQRT(X02AJF())
      H = TLIM/(NMESH-1)
*
      YN(1) = 0.0e0
*
      CALL D05BEF(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 = 0.0e0
      DO 20 I = 2, NMESH
         HI1 = real(I-1)*H
         ERR = ABS(YN(I)-SOL1(HI1))
         IF (ERR.GT.ERRMAX) THEN
            ERRMAX = ERR
            T = HI1
            SOLN = YN(I)
         END IF
         IF (I.GT.5 .AND. MOD(I,5).EQ.1) WRITE (NOUT,99998) HI1, YN(I)
   20 CONTINUE
      WRITE (NOUT,*)
      WRITE (NOUT,99999) ERRMAX, T, SOLN
*
      WRITE (NOUT,*)
*
      TLIM = 5.0e0
      H = TLIM/(NMESH-1)
      YN(1) = 1.0e0
*
      CALL D05BEF(CK2,CF2,CG2,'Subsequent',IORDER,TLIM,TOLNL,NMESH,YN,
     +            WORK,LWK,NCT,IFAIL)
*
      WRITE (NOUT,*) 'Example 2'
      WRITE (NOUT,*)
      WRITE (NOUT,99997) H
      WRITE (NOUT,*)
      WRITE (NOUT,*) '     T        Approximate'
      WRITE (NOUT,*) '                Solution '
      WRITE (NOUT,*)
*
      ERRMAX = 0.0e0
      DO 40 I = 1, NMESH
         HI1 = real(I-1)*H
         ERR = ABS(YN(I)-SOL2(HI1))
         IF (ERR.GT.ERRMAX) THEN
            ERRMAX = ERR
            T = HI1
            SOLN = YN(I)
         END IF
         IF (I.GT.7 .AND. 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
*
*
      real FUNCTION CK1(T)
*     .. Scalar Arguments ..
      real                T
*     .. Intrinsic Functions ..
      INTRINSIC           EXP
*     .. Executable Statements ..
      CK1 = EXP(-0.5e0*T)
      RETURN
      END
*
*
      real FUNCTION CF1(T)
*     .. Scalar Arguments ..
      real                T
*     .. Local Scalars ..
      real                A, PI, T1
*     .. External Functions ..
      real                X01AAF
      EXTERNAL            X01AAF
*     .. Intrinsic Functions ..
      INTRINSIC           EXP, SQRT
*     .. Executable Statements ..
      T1 = 1.0e0 + T
      A = 1.0e0/SQRT(X01AAF(PI)*T)
      CF1 = -A*EXP(-0.5e0*T1*T1/T)
      RETURN
      END
*
*
      real FUNCTION CG1(S,Y)
*     .. Scalar Arguments ..
      real                S, Y
*     .. Executable Statements ..
      CG1 = Y
      RETURN
      END
*
*
      real FUNCTION SOL1(T)
*     .. Scalar Arguments ..
      real                  T
*     .. Local Scalars ..
      real                  C, PI, T1
*     .. External Functions ..
      real                  X01AAF
      EXTERNAL              X01AAF
*     .. Intrinsic Functions ..
      INTRINSIC             EXP, SQRT
*     .. Executable Statements ..
*
      T1 = 1.0e0 + T
      C = 1.0e0/SQRT(2.0e0*X01AAF(PI))
      SOL1 = C*(1.0e0/(T**1.5e0))*EXP(-T1*T1/(2.0e0*T))
      RETURN
      END
*
*
      real FUNCTION CK2(T)
*     .. Scalar Arguments ..
```

```
      real                T
*     .. Local Scalars ..
      real                PI
*     .. External Functions ..
      real                X01AAF
      EXTERNAL            X01AAF
*     .. Intrinsic Functions ..
      INTRINSIC           SQRT
*     .. Executable Statements ..
      CK2 = SQRT(X01AAF(PI))
      RETURN
      END
*
*
      real  FUNCTION CF2(T)
*     .. Scalar Arguments ..
      real                T
*     .. Local Scalars ..
      real                ST1
*     .. Intrinsic Functions ..
      INTRINSIC           LOG, SQRT
*     .. Executable Statements ..
      ST1 = SQRT(1.0e0+T)
      CF2 = -2.0e0*LOG(ST1+SQRT(T))/ST1
      RETURN
      END
*
*
      real  FUNCTION CG2(S,Y)
*     .. Scalar Arguments ..
      real                S, Y
*     .. Executable Statements ..
      CG2 = Y
      RETURN
      END
*
*
      real  FUNCTION SOL2(T)
*     .. Scalar Arguments ..
      real                 T
*     .. Executable Statements ..
      SOL2 = 1.0e0/(1.0e0+T)
      RETURN
      END
```

9.2 Program Data

None.

9.3 Program Results

```
 D05BEF Example Program Results

 Example 1

 The stepsize h =   0.1000

      T        Approximate
                 Solution

   0.5000         0.1191
   1.0000         0.0528
   1.5000         0.0265
   2.0000         0.0146
   2.5000         0.0086
   3.0000         0.0052
   3.5000         0.0033
   4.0000         0.0022
   4.5000         0.0014
   5.0000         0.0010
```

```
   5.5000          0.0007
   6.0000          0.0004
   6.5000          0.0003
   7.0000          0.0002

 The maximum absolute error,   0.29E-02, occurred at T =  0.1000
 with solution   0.0326


 Example 2

 The stepsize h =   0.0714

      T        Approximate
                 Solution

   0.5000          0.6667
   1.0000          0.5000
   1.5000          0.4000
   2.0000          0.3333
   2.5000          0.2857
   3.0000          0.2500
   3.5000          0.2222
   4.0000          0.2000
   4.5000          0.1818
   5.0000          0.1667

 The maximum absolute error,   0.32E-05, occurred at T =  0.0714
 with solution   0.9333
```