NAG Fortran Library Routine Document

D05BAF

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

D05BAF computes the solution of a nonlinear convolution Volterra integral equation of the second kind using a reducible linear multi-step method.

2 Specification

```
      SUBROUTINE D05BAF (CK, CG, CF, METHOD, IORDER, ALIM, TLIM, YN, ERREST,
     1                   NMESH, TOL, THRESH, WORK, LWK, IFAIL)
      INTEGER            IORDER, NMESH, LWK, IFAIL
      double precision   ALIM, TLIM, YN(NMESH), ERREST(NMESH), TOL, THRESH,
     1                   WORK(LWK)
      CHARACTER*1        METHOD
      EXTERNAL           CK, CG, CF
```

3 Description

D05BAF computes the numerical solution of the nonlinear convolution Volterra integral equation of the second kind

$$y(t) = f(t) + \int_a^t k(t-s)g(s,y(s))\,ds, \quad a \leq t \leq T. \tag{1}$$

It is assumed that the functions involved in (1) are sufficiently smooth. The routine uses a reducible linear multi-step formula selected by you to generate a family of quadrature rules. The reducible formulae available in D05BAF are the Adams–Moulton formulae of orders 3 to 6, and the backward differentiation formulae (BDF) of orders 2 to 5. For more information about the behaviour and the construction of these rules we refer to Lubich (1983) and Wolkenfelt (1982).

The algorithm is based on computing the solution in a step-by-step fashion on a mesh of equispaced points. The initial step size which is given by $(T-a)/N$, N being the number of points at which the solution is sought, is halved and another approximation to the solution is computed. This extrapolation procedure is repeated until successive approximations satisfy a user-specified error requirement.

The above methods require some starting values. For the Adams formula of order greater than 3 and the BDF of order greater than 2 we employ an explicit Dormand–Prince–Shampine Runge–Kutta method (see Shampine (1986)). The above scheme avoids the calculation of the kernel, $k(t)$, on the negative real line.

4 References

Lubich Ch (1983) On the stability of linear multi-step methods for Volterra convolution equations *IMA J. Numer. Anal.* **3** 439–465

Shampine L F (1986) Some practical Runge–Kutta formulas *Math. Comput.* **46 (173)** 135–150

Wolkenfelt P H M (1982) The construction of reducible quadrature rules for Volterra integral and integro-differential equations *IMA J. Numer. Anal.* **2** 131–152

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 D05BAF is called. Parameters denoted as *Input* must **not** be changed by this procedure.

2: 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 D05BAF is called. Parameters denoted as *Input* must **not** be changed by this procedure.

3: 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 D05BAF is called. Parameters denoted as *Input* must **not** be changed by this procedure.

4: METHOD – CHARACTER*1 *Input*

On entry: the type of method which you wish to employ.

METHOD = 'A'

For Adams type formulae.

METHOD = 'B'

For backward differentiation formulae.

Constraint: METHOD = 'A' or 'B'.

5: IORDER – INTEGER *Input*

On entry: the order of the method to be used.

Constraints:

if METHOD = 'A', $3 \leq \text{IORDER} \leq 6$;
if METHOD = 'B', $2 \leq \text{IORDER} \leq 5$.

6: ALIM – ***double precision*** *Input*

On entry: a, the lower limit of the integration interval.

Constraint: ALIM ≥ 0.0.

7: TLIM – ***double precision*** *Input*

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

Constraint: TLIM > ALIM.

8: YN(NMESH) – ***double precision*** array *Output*

On exit: YN(i) contains the approximate value of the true solution $y(t)$ at the specified point $t = \text{ALIM} + i \times H$, for $i = 1, 2, \ldots, \text{NMESH}$, where $H = (\text{TLIM} - \text{ALIM})/\text{NMESH}$.

9: ERREST(NMESH) – ***double precision*** array *Output*

On exit: ERREST(i) contains the estimated value of the relative error in the computed solution at the point $t = \text{ALIM} + i \times H$, for $i = 1, 2, \ldots, \text{NMESH}$, where $H = (\text{TLIM} - \text{ALIM})/\text{NMESH}$.

10: NMESH – INTEGER *Input*

On entry: the number of equidistant points at which the solution is sought.

Constraints:

if METHOD = 'A', $\text{NMESH} \geq \text{IORDER} - 1$;
if METHOD = 'B', $\text{NMESH} \geq \text{IORDER}$.

11: TOL – ***double precision*** *Input*

On entry: the relative accuracy required in the computed values of the solution.

Constraint: $\sqrt{\epsilon} < \text{TOL} < 1.0$, where ϵ is the ***machine precision***.

12: THRESH – ***double precision*** *Input*

On entry: the threshold value for use in the evaluation of the estimated relative errors. For two successive meshes the following condition must hold at each point of the coarser mesh

$$\frac{|Y_1 - Y_2|}{\max(|Y_1|, |Y_2|, |\text{THRESH}|)} \leq \text{TOL},$$

where Y_1 is the computed solution on the coarser mesh and Y_2 is the computed solution at the corresponding point in the finer mesh. If this condition is not satisfied then the step size is halved and the solution is recomputed.

Note: THRESH can be used to effect a relative, absolute or mixed error test. If THRESH = 0.0 then pure relative error is measured and, if the computed solution is small and THRESH = 1.0, absolute error is measured.

13: WORK(LWK) – ***double precision*** array *Workspace*

14: LWK – INTEGER *Input*

On entry: the dimension of the array WORK as declared in the (sub)program from which D05BAF is called.

Constraint: $\text{LWK} \geq 10 \times \text{NMESH} + 6$.

Note: the above value of LWK is sufficient for D05BAF to perform only one extrapolation on the initial mesh as defined by NMESH. In general much more workspace is required and in the case when a large step size is supplied (i.e., NMESH is small), you must provide a considerably larger workspace.

15: 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, METHOD $\neq$ 'A' or 'B',
or IORDER < 2 or IORDER > 6,
or METHOD = 'A' and IORDER $= 2$,
or METHOD = 'B' and IORDER $= 6$,
or ALIM < 0,
or TLIM $\leq$ ALIM,
or TOL $< \sqrt{\epsilon}$ or TOL > 1.0, where ϵ is the ***machine precision***.

IFAIL $= 2$

On entry, NMESH $\leq$ IORDER $- 2$, when METHOD = 'A',
or NMESH $\leq$ IORDER $- 1$, when METHOD = 'B'.

IFAIL $= 3$

On entry, LWK $< 10 \times$ NMESH $+ 6$.

IFAIL $= 4$

The solution of the nonlinear equation (2) (see Section 8 for further details) could not be computed by C05AVF and C05AZF.

IFAIL $= 5$

The size of the workspace LWK is too small for the required accuracy. The computation has failed in its initial phase (see Section 8 for further details).

IFAIL = 6

The size of the workspace LWK is too small for the required accuracy on the interval [ALIM,TLIM] (see Section 8 for further details).

7 Accuracy

The accuracy depends on TOL, the theoretical behaviour of the solution of the integral equation, the interval of integration and on the method being used. It can be controlled by varying TOL and THRESH; you are recommended to choose a smaller value for TOL, the larger the value of IORDER.

You are warned not to supply a very small TOL, because the required accuracy may never be achieved. This will usually force an error exit with IFAIL = 5 or 6.

In general, the higher the order of the method, the faster the required accuracy is achieved with less workspace. For non-stiff problems (see Section 8) you are recommended to use the Adams method (METHOD = 'A') of order greater than 4 (IORDER > 4).

8 Further Comments

When solving (1), the solution of a nonlinear equation of the form

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

is required, where Ψ_n and α are constants. D05BAF calls C05AVF to find an interval for the zero of this equation followed by C05AZF to find its zero.

There is an initial phase of the algorithm where the solution is computed only for the first few points of the mesh. The exact number of these points depends on IORDER and METHOD. The step size is halved until the accuracy requirements are satisfied on these points and only then the solution on the whole mesh is computed. During this initial phase, if LWK is too small, D05BAF will exit with IFAIL = 5.

In the case IFAIL = 4 or 5, you may be dealing with a 'stiff' equation; an equation where the Lipschitz constant L of the function $g(t,y)$ in (1) with respect to its second argument is large, viz,

$$|g(t,u) - g(t,v)| \leq L|u - v|. \tag{3}$$

In this case, if a BDF method (METHOD = 'B') has been used, you are recommended to choose a smaller step size by increasing the value of NMESH, or provide a larger workspace. But, if an Adams method (METHOD = 'A') has been selected, you are recommended to switch to a BDF method instead.

In the case IFAIL = 6, the specified accuracy has not been attained but ERREST and YN contain the most recent approximation to the computed solution and the corresponding error estimate. In this case, the error message informs you of the number of extrapolations performed and the size of LWK required for the algorithm to proceed further.

On a successful exit, or with IFAIL = 6, you may wish to examine the contents of the workspace WORK. Specifically, for $i = 1, 2, \ldots, N$, where $N = \mathrm{int}(\mathrm{int}((\mathrm{LWK} - 6)/5)/2) + 1$, WORK$(i + N)$ and WORK(i) contain the computed approximation to the solution and its error estimate respectively at the point $t = \mathrm{ALIM} + \frac{(i-1)}{N} \times (\mathrm{TLIM} - \mathrm{ALIM})$.

9 Example

Consider the following integral equation

$$y(t) = e^{-t} + \int_0^t e^{-(t-s)} \left[y(s) + e^{-y(s)} \right] ds, \quad 0 \leq t \leq 20 \tag{4}$$

with the solution $y(t) = \ln(t + e)$. In this example, the Adams method of order 6 is used to solve this equation with TOL = 1.D − 4.

9.1 Program Text

```
*     D05BAF Example Program Text
*     Mark 14 Release. NAG Copyright 1989.
*     .. Parameters ..
      INTEGER          NOUT
      PARAMETER        (NOUT=6)
      INTEGER          LWK, NMESH
      PARAMETER        (LWK=1000,NMESH=6)
*     .. Local Scalars ..
      DOUBLE PRECISION ALIM, H, THRESH, TLIM, TOL
      INTEGER          I, IFAIL, IORDER
      CHARACTER        METHOD
*     .. Local Arrays ..
      DOUBLE PRECISION ERRST(NMESH), WORK(LWK), YN(NMESH)
*     .. External Functions ..
      DOUBLE PRECISION CF, CG, CK, SOL, X02AJF
      EXTERNAL         CF, CG, CK, SOL, X02AJF
*     .. External Subroutines ..
      EXTERNAL         D05BAF
*     .. Intrinsic Functions ..
      INTRINSIC        ABS
*     .. Executable Statements ..
      WRITE (NOUT,*) 'D05BAF Example Program Results'
      METHOD = 'A'
      IORDER = 6
      ALIM = 0.D0
      TLIM = 20.D0
      H = (TLIM-ALIM)/NMESH
      TOL = 1.D-3
      THRESH = X02AJF()
*
      WRITE (NOUT,*)
      WRITE (NOUT,99999) 'Size of workspace =', LWK
      WRITE (NOUT,99998) 'Tolerance         =', TOL
      WRITE (NOUT,*)
      IFAIL = 0
*
      CALL D05BAF(CK,CG,CF,METHOD,IORDER,ALIM,TLIM,YN,ERRST,NMESH,TOL,
     +            THRESH,WORK,LWK,IFAIL)
*
      IF (IFAIL.EQ.0) THEN
         WRITE (NOUT,*)
     +'   T        Approx. Sol.  True Sol.    Est. Error    Actual Error
     +'
         WRITE (NOUT,99997) (ALIM+I*H,YN(I),SOL(I*H),ERRST(I),ABS((YN(I)
     +     -SOL(I*H))/SOL(I*H)),I=1,NMESH)
      END IF
      STOP
*
99999 FORMAT (1X,A,I12)
99998 FORMAT (1X,A,E12.4)
99997 FORMAT (F7.2,2F14.5,2E15.5)
      END
*
      DOUBLE PRECISION FUNCTION SOL(T)
*     .. Scalar Arguments ..
      DOUBLE PRECISION             T
*     .. Intrinsic Functions ..
      INTRINSIC                    EXP, LOG
*     .. Executable Statements ..
      SOL = LOG(T+EXP(1.D0))
      RETURN
      END
*
      DOUBLE PRECISION FUNCTION CF(T)
*     .. Scalar Arguments ..
      DOUBLE PRECISION            T
*     .. Intrinsic Functions ..
      INTRINSIC                   EXP
*     .. Executable Statements ..
```

```
      CF = EXP(-T)
      RETURN
      END
*
      DOUBLE PRECISION FUNCTION CK(T)
*     .. Scalar Arguments ..
      DOUBLE PRECISION            T
*     .. Intrinsic Functions ..
      INTRINSIC                   EXP
*     .. Executable Statements ..
      CK = EXP(-T)
      RETURN
      END
*
      DOUBLE PRECISION FUNCTION CG(S,Y)
*     .. Scalar Arguments ..
      DOUBLE PRECISION            S, Y
*     .. Intrinsic Functions ..
      INTRINSIC                   EXP
*     .. Executable Statements ..
      CG = Y + EXP(-Y)
      RETURN
      END
```

9.2 Program Data

None.

9.3 Program Results

```
 D05BAF Example Program Results

 Size of workspace =        1000
 Tolerance         =  0.1000E-02

    T        Approx. Sol.  True Sol.    Est. Error    Actual Error
   3.33       1.80037       1.80033    0.52609E-04    0.23847E-04
   6.67       2.23916       2.23911    0.15199E-03    0.23477E-04
  10.00       2.54310       2.54304    0.22922E-03    0.22456E-04
  13.33       2.77587       2.77581    0.29359E-03    0.21743E-04
  16.67       2.96456       2.96450    0.35172E-03    0.21382E-04
  20.00       3.12324       3.12317    0.40905E-03    0.21310E-04
```