NAG 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,          &
                   NMESH, TOL, THRESH, WORK, LWK, IFAIL)

INTEGER             IORDER, NMESH, LWK, IFAIL
REAL (KIND=nag_wp) CK, CG, CF, ALIM, TLIM, YN(NMESH), ERREST(NMESH), TOL,      &
                   THRESH, 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

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

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

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

3: 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 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', $3 \leq$ IORDER ≤ 6;
if METHOD = 'B', $2 \leq$ IORDER ≤ 5.

6: ALIM – REAL (KIND=nag_wp) *Input*

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

Constraint: ALIM ≥ 0.0.

7: TLIM – REAL (KIND=nag_wp) *Input*

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

Constraint: TLIM $>$ ALIM.

8: YN(NMESH) – REAL (KIND=nag_wp) array *Output*

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

9: ERREST(NMESH) – REAL (KIND=nag_wp) array *Output*

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

10: NMESH – INTEGER *Input*

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

Constraints:

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

11: TOL – REAL (KIND=nag_wp) *Input*

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

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

12: THRESH – REAL (KIND=nag_wp) *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) – REAL (KIND=nag_wp) array *Output*

14: LWK – INTEGER *Input*

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

Constraint: LWK $\geq 10 \times$ 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.

On exit: if IFAIL $= 5$ or 6, WORK(1) contains the size of LWK required for the algorithm to proceed further.

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 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, 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.**

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, 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 < 0.0,
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 $[\text{ALIM}, \text{TLIM}]$ (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 YN and ERREST 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. The latter quantity will also be available in WORK(1).

9 Example

Consider the following integral equation

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

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.E$-$4.

9.1 Program Text

```
!   D05BAF Example Program Text
!   Mark 24 Release. NAG Copyright 2012.
    Module d05bafe_mod

!     D05BAF Example Program Module:
!            Parameters and User-defined Routines

!     .. Use Statements ..
      Use nag_library, Only: nag_wp
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Integer, Parameter                   :: nmesh = 6, nout = 6
    Contains
      Function sol(t)

!       .. Function Return Value ..
        Real (Kind=nag_wp)                   :: sol
!       .. Scalar Arguments ..
        Real (Kind=nag_wp), Intent (In)      :: t
!       .. Intrinsic Procedures ..
        Intrinsic                            :: exp, log
!       .. Executable Statements ..
        sol = log(t+exp(1.0_nag_wp))

        Return

      End Function sol
      Function cf(t)

!       .. Function Return Value ..
        Real (Kind=nag_wp)                   :: cf
!       .. Scalar Arguments ..
        Real (Kind=nag_wp), Intent (In)      :: t
!       .. Intrinsic Procedures ..
        Intrinsic                            :: exp
!       .. Executable Statements ..
        cf = exp(-t)

        Return

      End Function cf
      Function ck(t)

!       .. Function Return Value ..
        Real (Kind=nag_wp)                   :: ck
!       .. Scalar Arguments ..
        Real (Kind=nag_wp), Intent (In)      :: t
!       .. Intrinsic Procedures ..
        Intrinsic                            :: exp
!       .. Executable Statements ..
        ck = exp(-t)

        Return

      End Function ck
      Function cg(s,y)

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

        Return
```

```
      End Function cg
    End Module d05bafe_mod
    Program d05bafe

!     D05BAF Example Main Program

!     .. Use Statements ..
      Use nag_library, Only: d05baf, nag_wp, x02ajf
      Use d05bafe_mod, Only: cf, cg, ck, nmesh, nout, sol
!     .. Implicit None Statement ..
      Implicit None
!     .. Local Scalars ..
      Real (Kind=nag_wp)                  :: alim, h, hi, si, thresh, tlim, tol
      Integer                             :: i, ifail, iorder, lwk
      Character (1)                       :: method
!     .. Local Arrays ..
      Real (Kind=nag_wp)                  :: errest(nmesh), yn(nmesh)
      Real (Kind=nag_wp), Allocatable     :: work(:)
!     .. Intrinsic Procedures ..
      Intrinsic                           :: abs, int, real
!     .. Executable Statements ..
      Write (nout,*) 'D05BAF Example Program Results'

      method = 'A'
      iorder = 6
      alim = 0.0_nag_wp
      tlim = 20.0_nag_wp
      h = (tlim-alim)/real(nmesh,kind=nag_wp)
      tol = 1.E-3_nag_wp
      thresh = x02ajf()
      lwk = 10*nmesh + 6
      Allocate (work(lwk))

!     Loop until the supplied workspace is big enough

loop: Do

        ifail = 1
        Call d05baf(ck,cg,cf,method,iorder,alim,tlim,yn,errest,nmesh,tol, &
          thresh,work,lwk,ifail)

        Select Case (ifail)
        Case (5,6)
          lwk = int(work(1))
          Deallocate (work)
          Allocate (work(lwk))
        Case Default
          Exit loop
        End Select

      End Do loop

      If (ifail/=0) Then
        Write (nout,99996) 'D05BAF exited with IFAIL =', ifail
        Go To 100
      End If

      Write (nout,*)
      Write (nout,99999) 'Size of workspace =', lwk
      Write (nout,99998) 'Tolerance         =', tol
      Write (nout,*)
      Write (nout,*) &
        '   T        Approx. Sol.  True Sol.    Est. Error    Actual Error'
      Do i = 1, nmesh
        hi = real(i,kind=nag_wp)*h
        si = sol(hi)
        Write (nout,99997) alim + hi, yn(i), si, errest(i), abs((yn(i)-si)/si)
      End Do

100   Continue
```

```
99999 Format (1X,A,I12)
99998 Format (1X,A,E12.4)
99997 Format (F7.2,2F14.5,2E15.5)
99996 Format (1X,A,I5)
    End Program d05bafe
```

9.2 Program Data

None.

9.3 Program Results

```
 D05BAF Example Program Results

 Size of workspace =         486
 Tolerance         =  0.1000E-02

    T        Approx. Sol.  True Sol.    Est. Error    Actual Error
   3.33        1.80037       1.80033    0.80378E-04    0.23847E-04
   6.67        2.23916       2.23911    0.17774E-03    0.23477E-04
  10.00        2.54310       2.54304    0.24595E-03    0.22456E-04
  13.33        2.77587       2.77581    0.30574E-03    0.21743E-04
  16.67        2.96456       2.96450    0.36170E-03    0.21382E-04
  20.00        3.12324       3.12317    0.41713E-03    0.21310E-04
```