NAG Toolbox for MATLAB

d05ba

1 Purpose

d05ba computes the solution of a nonlinear convolution Volterra integral equation of the second kind using a reducible linear multi-step method.

2 Syntax

```
[yn, errest, ifail] = d05ba(ck, cg, cf, method, iorder, alim, tlim,
nmesh, tol, thresh, lwk)
```

3 Description

d05ba 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 function uses a reducible linear multi-step formula selected by you to generate a family of quadrature rules. The reducible formulae available in d05ba 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

5.1 Compulsory Input Parameters

1: **ck – string containing name of m-file**

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

Its specification is:

```
[result] = ck(t)
```

Input Parameters

1: **t – double scalar**

t, the value of the independent variable.

Output Parameters

1: **result – double scalar**

The result of the function.

2: **cg – string containing name of m-file**

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

Its specification is:

```
[result] = cg(s, y)
```

Input Parameters

1: **s – double scalar**

s, the value of the independent variable.

2: **y – double scalar**

The value of the solution y at the point **s**.

Output Parameters

1: **result – double scalar**

The result of the function.

3: **cf – string containing name of m-file**

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

Its specification is:

```
[result] = cf(t)
```

Input Parameters

1: **t – double scalar**

t, the value of the independent variable.

Output Parameters

1: **result – double scalar**

The result of the function.

4: **method – string**

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 – int32 scalar**

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 – double scalar**

a, the lower limit of the integration interval.

Constraint: **alim** ≥ 0.0.

7: **tlim – double scalar**

The final point of the integration interval, T.

Constraint: **tlim** $>$ **alim**.

8: **nmesh – int32 scalar**

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**.

9: **tol – double scalar**

The relative accuracy required in the computed values of the solution.

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

10: **thresh – double scalar**

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

$$\frac{|Y_1 - Y_2|}{\max(|Y_1|, |Y_2|, |\textbf{thresh}|)} \leq \textbf{tol},$$

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.

11: **lwk – int32 scalar**

Constraint: **lwk** $\geq 10 \times$ **nmesh** $+ 6$.

Note: the above value of **lwk** is sufficient for d05ba 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.

5.2 Optional Input Parameters

None.

5.3 Input Parameters Omitted from the MATLAB Interface

work

5.4 Output Parameters

1: **yn**(**nmesh**) – **double array**

yn(i) contains the approximate value of the true solution $y(t)$ at the specified point $t = \textbf{alim} + i \times H$, for $i = 1, 2, \ldots, \textbf{nmesh}$, where $H = (\textbf{tlim} - \textbf{alim})/\textbf{nmesh}$.

2: **errest**(**nmesh**) – **double array**

errest(i) contains the estimated value of the relative error in the computed solution at the point $t = \textbf{alim} + i \times H$, for $i = 1, 2, \ldots, \textbf{nmesh}$, where $H = (\textbf{tlim} - \textbf{alim})/\textbf{nmesh}$.

3: **ifail – int32 scalar**

0 unless the function detects an error (see Section 6).

6 Error Indicators and Warnings

Errors or warnings detected by the function:

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 c05av and c05az.

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. d05ba calls c05av to find an interval for the zero of this equation followed by c05az 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, d05ba 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 $N = \text{int}(\text{int}((\textbf{lwk} - 6)/5)/2) + 1$, **work**$(i + N)$ and **work**(i) contain the computed approximation to the solution and its error estimate respectively at the point $t = \textbf{alim} + \frac{(i-1)}{N} \times (\textbf{tlim} - \textbf{alim})$.

9 Example

d05ba_cf.m

```
function [result] = cf(t)
  result = exp(-t);
```

d05ba_cg.m

```
function [result] = cg(s, y)
```

```
  result = y + exp(-y);
```

d05ba_ck.m

```
function [result] = ck(t)
  result = exp(-t);
```

```
method = 'A';
iorder = int32(6);
alim = 0;
tlim = 20;
nmesh = int32(6);
tol = 0.001;
thresh = 1.111307226797642e-16;
lwk = int32(1000);
[yn, errest, ifail] = ...
       d05ba('d05ba_ck', 'd05ba_cg', 'd05ba_cf', method, iorder, alim,
tlim, nmesh, tol, thresh, lwk)
```

```
yn =
    1.8004
    2.2392
    2.5431
    2.7759
    2.9646
    3.1232
errest =
   1.0e-03 *
    0.0526
    0.1520
    0.2292
    0.2936
    0.3517
    0.4090
ifail =
           0
```