NAG Toolbox for MATLAB

d05bd

1 Purpose

d05bd 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 Syntax

```
[yn, work, ifail] = d05bd(ck, cf, cg, initwt, tlim, nmesh, work,
'iorder', iorder, 'tolnl', tolnl, 'lwk', lwk)
```

3 Description

d05bd computes the numerical solution of the weakly singular convolution Volterra–Abel integral equation of the second kind

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

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

The function uses a fractional BDF linear multi-step method selected by you to generate a family of quadrature rules (see d05by). The BDF methods available in d05bd 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 function 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 d05bd 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

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: **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.

3: **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.

4: **initwt – string**

If the fractional weights required by the method need to be calculated by the function, then set **initwt** = 'I' (**I**nitial call).

If **initwt** = 'S' (**S**ubsequent call), then the function assumes the fractional weights have been computed on a previous call and are stored in **work**.

Constraint: **initwt** = 'I' or 'S'.

Note: when d05bd is re-entered with the value of **initwt** = 'S', the values of **nmesh**, **iorder** and the contents of **work must** not be changed

5: **tlim – double scalar**

The final point of the integration interval, T.

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

6: **nmesh – int32 scalar**

N, the number of equispaced points at which the solution is sought.

Constraint: **nmesh** $= 2^m + 2 \times$ **iorder** $- 1$, where $m \geq 1$.

7: **work(lwk) – double array**

Constraint: **lwk** $\geq (2 \times$ **iorder** $+ 6) \times$ **nmesh** $+ 8 \times$ **iorder**$^2 - 16 \times$ **iorder** $+ 1$.

5.2 Optional Input Parameters

1: **iorder – int32 scalar**

p, the order of the BDF method to be used.

Constraint: $4 \leq$ **iorder** ≤ 6.

Suggested value: **iorder** $= 4$.

Default: 4

2: **tolnl – double scalar**

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***.

Default: $\sqrt{}$

3: **lwk – int32 scalar**

Default: The dimension of the array **work**.

Constraint: **lwk** $\geq (2 \times$ **iorder** $+ 6) \times$ **nmesh** $+ 8 \times$ **iorder**$^2 - 16 \times$ **iorder** $+ 1$.

5.3 Input Parameters Omitted from the MATLAB Interface

nct

5.4 Output Parameters

1: **yn(nmesh) – double array**

yn(i) contains the approximate value of the true solution $y(t)$ at the point $t = (i-1) \times h$, for $i = 1, 2, \ldots,$ **nmesh**, where $h =$ **tlim**$/($**nmesh** $- 1)$.

2: **work**(**lwk**) – **double array**

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, **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 d05bd,
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 function 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 function 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, d05bd computes the solution of a system of nonlinear equations for obtaining the $2p - 1$ starting values. c05nd is used for this purpose. When a failure with **ifail** $= 2$ occurs (which corresponds to an error exit from c05nd), 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. d05bd calls c05ax to find the root of this equation.

If a failure with **ifail** $= 3$ occurs (which corresponds to an error exit from c05ax), 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

d05bd_cf.m

```
function [result] = cf(t)
  result = sqrt(t) + (3.0d0/8.0d0)*t*t*pi;
```

d05bd_cg.m

```
function [result] = cg(s, y)
  result = y^3;
```

d05bd_ck.m

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

```
initwt = 'Initial';
tlim = 7;
nmesh = int32(71);
work = zeros(1059, 1);
[yn, workOut, ifail] = d05bd('d05bd_ck', 'd05bd_cf', 'd05bd_cg', initwt,
tlim, nmesh, work)
```

```
yn =
     array elided
workOut =
     array elided
ifail =
           0
```