NAG Toolbox

nag_inteq_abel2_weak (d05bd)

1 Purpose

nag_inteq_abel2_weak (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] = nag_inteq_abel2_weak(ck, cf, cg, initwt, tlim, nmesh, work, 'iorder', iorder, 'tolnl', tolnl)
[yn, work, ifail] = d05bd(ck, cf, cg, initwt, tlim, nmesh, work, 'iorder', iorder, 'tolnl', tolnl)
```

Note: the interface to this routine has changed since earlier releases of the toolbox:

At Mark 22: *lwk* was removed from the interface.

3 Description

nag_inteq_abel2_weak (d05bd) computes the numerical solution of the weakly singular convolution Volterra–Abel integral equation of the second kind

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

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

The function uses a fractional BDF linear multi-step method to generate a family of quadrature rules (see nag_inteq_abel_weak_weights (d05by)). The BDF methods available in nag_inteq_abel2_weak (d05bd) are of orders 4, 5 and 6 ($= p$ say). For a description of the theoretical and practical background 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 nag_inteq_abel2_weak (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** – REAL (KIND=nag_wp) FUNCTION, supplied by the user.

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

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

Input Parameters

1: **t** – REAL (KIND=nag_wp)

t, the value of the independent variable.

Output Parameters

1: **result**

The value of $k(t)$ evaluated at **t**.

2: **cf** – REAL (KIND=nag_wp) FUNCTION, supplied by the user.

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

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

Input Parameters

1: **t** – REAL (KIND=nag_wp)

t, the value of the independent variable.

Output Parameters

1: **result**

The value of $f(t)$ evaluated at **t**.

3: **cg** – REAL (KIND=nag_wp) FUNCTION, supplied by the user.

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

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

Input Parameters

1: **s** – REAL (KIND=nag_wp)

s, the value of the independent variable.

2: **y** – REAL (KIND=nag_wp)

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

Output Parameters

1: **result**

The value of $g(s, y(s))$ evaluated at **s** and **y**.

4: **initwt** – CHARACTER(1)

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), 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 nag_inteq_abel2_weak (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** – REAL (KIND=nag_wp)

The final point of the integration interval, T.

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

6: **nmesh** – INTEGER

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) – REAL (KIND=nag_wp) array

lwk, the dimension of the array, must satisfy the constraint $lwk \geq (2 \times$ **iorder** $+ 6) \times$ **nmesh** $+ 8 \times$ **iorder**$^2 - 16 \times$ **iorder** $+ 1$.

If **initwt** = 'S', **work** must contain fractional weights computed by a previous call of nag_inteq_abel2_weak (d05bd) (see description of **initwt**).

5.2 Optional Input Parameters

1: **iorder** – INTEGER

Suggested value: **iorder** = 4.

Default: 4

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

Constraint: 4 $\leq$ **iorder** $\leq$ 6.

2: **tolnl** – REAL (KIND=nag_wp)

Suggested value: **tolnl** $= \sqrt{\epsilon}$ where ϵ is the ***machine precision***.

Default: $\sqrt{$***machine precision***$}$

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 9).

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

5.3 Output Parameters

1: **yn**(**nmesh**) – REAL (KIND=nag_wp) 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) – REAL (KIND=nag_wp) array

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

Contains fractional weights which may be used by a subsequent call of nag_inteq_abel2_weak (d05bd).

3: **ifail** – INTEGER

ifail $= 0$ unless the function detects an error (see Section 5).

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 nag_inteq_abel2_weak (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 9 for further details).

ifail $= 3$

The function cannot compute the solution at a specific step due to an error in the solution of the nonlinear equation (2). Relaxing the value of **tolnl** and/or increasing the value of **nmesh** may overcome this problem (see Section 9 for further details).

ifail $= -99$

An unexpected error has been triggered by this routine. Please contact NAG.

ifail $= -399$

Your licence key may have expired or may not have been installed correctly.

ifail $= -999$

Dynamic memory allocation failed.

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.

8 Further Comments

In solving (1), initially, nag_inteq_abel2_weak (d05bd) computes the solution of a system of nonlinear equations for obtaining the $2p - 1$ starting values. nag_roots_sys_func_rcomm (c05qd) is used for this purpose. When a failure with **ifail** $= 2$ occurs (which corresponds to an error exit from nag_roots_sys_func_rcomm (c05qd)), 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. nag_inteq_abel2_weak (d05bd) calls nag_roots_contfn_cntin_rcomm (c05ax) to find the root of this equation.

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

In this example we solve the following integral equations

$$y(t) = \sqrt{t} + \tfrac{3}{8}\pi t^2 - \int_0^t \frac{1}{\sqrt{t-s}}[y(s)]^3\,ds, \quad 0 \le t \le 7,$$

with the solution $y(t) = \sqrt{t}$, and

$$y(t) = (3-t)\sqrt{t} - \int_0^t \frac{1}{\sqrt{t-s}}\exp\Big(s(1-s)^2 - [y(s)]^2\Big)ds, \quad 0 \le t \le 5,$$

with the solution $y(t) = (1-t)\sqrt{t}$. In the above examples, the fourth-order BDF is used, and **nmesh** is set to $2^6 + 7$.

9.1 Program Text

```
function d05bd_example

fprintf('d05bd example results\n\n');

ck = @(t) -sqrt(pi);
cf = @(t) sqrt(t) + (3/8)*t*t*pi;
cg = @(s, y) y^3;
initwt = 'Initial';
tlim = 7;
nmesh = nag_int(71);
work = zeros(1059, 1);
[yn, work, ifail] = d05bd( ...
                          ck, cf, cg, initwt, tlim, nmesh, work);

h = tlim/double(nmesh-1);
fprintf('Example 1\n\n The stepsize h = %8.4f\n\n',h);
fprintf('     t        Approximate\n');
fprintf('                Solution\n');
t = [0:h:tlim];
sol = [t' yn];
solp = sol(1:5:nmesh,1:2);
fprintf('%8.4f%15.4f\n',solp');

ck2 = @(t) -sqrt(pi);
cf2 = @(t) (3-t)*sqrt(t);
cg2 = @(s, y) exp(s*(1-s)^2-y^2);
initwt = 'Subsequent';
tlim = 5;
[yn, work, ifail] = d05bd( ...
                          ck2, cf2, cg2, initwt, tlim, nmesh, work);
h = tlim/double(nmesh-1);

fprintf('\n\nExample 2\n\n The stepsize h = %8.4f\n\n',h);
fprintf('     t        Approximate\n');
```

```
fprintf('                Solution\n');
t = [0:h:tlim];
sol = [t' yn];
solp = sol(1:7:nmesh,1:2);
fprintf('%8.4f%15.4f\n',solp');
```

9.2 Program Results

```
     d05bd example results

Example 1

 The stepsize h =   0.1000

     t        Approximate
                Solution
  0.0000         0.0000
  0.5000         0.7071
  1.0000         1.0000
  1.5000         1.2247
  2.0000         1.4142
  2.5000         1.5811
  3.0000         1.7321
  3.5000         1.8708
  4.0000         2.0000
  4.5000         2.1213
  5.0000         2.2361
  5.5000         2.3452
  6.0000         2.4495
  6.5000         2.5495
  7.0000         2.6458

Example 2

 The stepsize h =   0.0714

     t        Approximate
                Solution
  0.0000         0.0000
  0.5000         0.3536
  1.0000         0.0000
  1.5000        -0.6124
  2.0000        -1.4142
  2.5000        -2.3717
  3.0000        -3.4641
  3.5000        -4.6771
  4.0000        -6.0000
  4.5000        -7.4246
  5.0000        -8.9443
```