

## NAG Toolbox

### nag\_inteq\_abel1\_weak (d05be)

#### 1 Purpose

`nag_inteq_abel1_weak` (d05be) computes the solution of a weakly singular nonlinear convolution Volterra–Abel integral equation of the first kind using a fractional Backward Differentiation Formulae (BDF) method.

#### 2 Syntax

```
[yn, work, ifail] = nag_inteq_abel1_weak(ck, cf, cg, initwt, tlim, yn, work,
'iorder', iorder, 'tolnl', tolnl, 'nmesh', nmesh)

[yn, work, ifail] = d05be(ck, cf, cg, initwt, tlim, yn, work, 'iorder', iorder,
'tolnl', tolnl, 'nmesh', nmesh)
```

**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_abel1_weak` (d05be) computes the numerical solution of the weakly singular convolution Volterra–Abel integral equation of the first kind

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

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

$$f(t) = t^\beta w(t) \quad \text{with} \quad \beta > -\frac{1}{2} \text{ and } w(t) \text{ smooth,} \quad (2)$$

then the solution  $y(t)$  is unique and has the form  $y(t) = t^{\beta-1/2} z(t)$ , (see Lubich (1987)). It is evident from (1) that  $f(0) = 0$ . You are required to provide the value of  $y(t)$  at  $t = 0$ . If  $y(0)$  is unknown, Section 9 gives a description of how an approximate value can be obtained.

The function uses a fractional BDF linear multi-step method selected by you to generate a family of quadrature rules (see `nag_inteq_abel_weak_weights` (d05by)). The BDF methods available in `nag_inteq_abel1_weak` (d05be) are of orders 4, 5 and 6 ( $= p$  say). For a description of the theoretical and practical background related to these methods we refer to Lubich (1987) 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  $2p-2$  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_abel1_weak` (d05be) is to be called several times (with the same value of  $N$ ) within the same program 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

Gorenflo R and Pfeiffer A (1991) On analysis and discretization of nonlinear Abel integral equations of first kind *Acta Math. Vietnam* **16** 211–262

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 (1987) Fractional linear multistep methods for Abel–Volterra integral equations of the first kind *IMA J. Numer. Anal* **7** 97–106

## 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' (Initial call).

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

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

**Note:** when nag\_inteq\_abell\_weak (d05be) is re-entered with a 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 × *machine precision*.

- 6: **yn(nmesh)** – REAL (KIND=nag\_wp) array

**yn**(1) must contain the value of  $y(t)$  at  $t = 0$  (see Section 9).

- 7: **work(lwk)** – REAL (KIND=nag\_wp) array

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

If **initwt** = 'S', **work** must contain fractional weights computed by a previous call of nag\_inteq\_abell\_weak (d05be) (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:*  $4 \leq \mathbf{iorder} \leq 6$ .

- 2: **tolnl** – REAL (KIND=nag\_wp)

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

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

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:*  $\mathbf{tolnl} > 10 \times \text{machine precision}$ .

3: **nmesh** – INTEGER

*Default:* the dimension of the array **yn**.

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

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

### 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  $i = 1, 2, \dots, \mathbf{nmesh}$ , where  $h = \mathbf{tlim}/(\mathbf{nmesh} - 1)$ .

2: **work**( $lwk$ ) – REAL (KIND=nag\_wp) array

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

Contains fractional weights which may be used by a subsequent call of nag\_inteq\_abel1\_weak (d05be).

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**  $\leq 10 \times \text{machine precision}$ ,  
 or **initwt**  $\neq$  'I' or 'S',  
 or **initwt** = 'S' on the first call to nag\_inteq\_abel1\_weak (d05be),  
 or **tolnl**  $\leq 10 \times \text{machine precision}$ ,  
 or **nmesh**  $\neq 2^m + 2 \times \mathbf{iorder} - 1, m \geq 1$ ,  
 or  $lwk < (2 \times \mathbf{iorder} + 6) \times \mathbf{nmesh} + 8 \times \mathbf{iorder}^2 - 16 \times \mathbf{iorder} + 1$ .

**ifail** = 2

The function cannot compute the  $2p - 2$  starting values due to an error in 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 single nonlinear equation (3). 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 (3) at each step of computation. This value can affect the accuracy of the solution. However, for most problems, the value of  $\sqrt{\epsilon}$ , where  $\epsilon$  is the *machine precision*, should be sufficient.

## 8 Further Comments

Also when solving (1) the initial value  $y(0)$  is required. This value may be computed from the limit relation (see Gorenflo and Pfeiffer (1991))

$$\frac{-2}{\sqrt{\pi}}k(0)g(0, y(0)) = \lim_{t \rightarrow 0} \frac{f(t)}{\sqrt{t}}. \quad (3)$$

If the value of the above limit is known then by solving the nonlinear equation (3) an approximation to  $y(0)$  can be computed. If the value of the above limit is not known, an approximation should be provided. Following the analysis presented in Gorenflo and Pfeiffer (1991), the following  $p$ th-order approximation can be used:

$$\lim_{t \rightarrow 0} \frac{f(t)}{\sqrt{t}} \simeq \frac{f(h^p)}{h^{p/2}}. \quad (4)$$

However, it must be emphasized that the approximation in (4) may result in an amplification of the rounding errors and hence you are advised (if possible) to determine  $\lim_{t \rightarrow 0} \frac{f(t)}{\sqrt{t}}$  by analytical methods.

Also when solving (1), initially, `nag_inteq_abell_weak` (d05be) computes the solution of a system of nonlinear equation for obtaining the  $2p - 2$  starting values. `nag_roots_sys_func_rcomm` (c05qd) is used for this purpose. If a failure with **ifail** = 2 occurs (corresponding 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

$$Y_n - \alpha g(t_n, Y_n) - \Psi_n = 0, \quad (5)$$

is required at each step of computation, where  $\Psi_n$  and  $\alpha$  are constants. `nag_inteq_abell_weak` (d05be) calls `nag_roots_contfn_cntin_rcomm` (c05ax) to find the root of this equation.

When a failure with **ifail** = 3 occurs (which corresponds to an error exit from `nag_roots_contfn_cntin_rcomm` (c05ax)), you are advised to either 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

We solve the following integral equations.

### Example 1

The density of the probability that a Brownian motion crosses a one-sided moving boundary  $a(t)$  before time  $t$ , satisfies the integral equation (see Hairer *et al.* (1988))

$$-\frac{1}{\sqrt{t}}\exp\left(\frac{1}{2}-\{a(t)\}^2/t\right) + \int_0^t \frac{\exp\left(-\frac{1}{2}\{a(t)-a(s)\}^2/(t-s)\right)}{\sqrt{t-s}} y(s) ds = 0, \quad 0 \leq t \leq 7.$$

In the case of a straight line  $a(t) = 1 + t$ , the exact solution is known to be

$$y(t) = \frac{1}{\sqrt{2\pi t^3}} \exp\left\{-\frac{(1+t)^2}{2t}\right\}$$

### Example 2

In this example we consider the equation

$$-\frac{2\log(\sqrt{1+t} + \sqrt{t})}{\sqrt{1+t}} + \int_0^t \frac{y(s)}{\sqrt{t-s}} ds = 0, \quad 0 \leq t \leq 5.$$

The solution is given by  $y(t) = \frac{1}{1+t}$ .

In the above examples, the fourth-order BDF is used, and **nmesh** is set to  $2^6 + 7$ .

## 9.1 Program Text

```
function d05be_example

fprintf('d05be example results\n\n');

ck = @(t) exp(-0.5*t);
cf = @(t) -exp(-0.5*(1+t)^2/t)/sqrt(pi*t);
cg = @(s, y) y;
initwt = 'Initial';
tlim = 7;
nmesh = 71;
yn = zeros(nmesh,1);
work = zeros(1059, 1);
[yn, work, ifail] = d05be( ...
    ck, cf, cg, initwt, tlim, yn, 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(6:5:nmesh,1:2);
fprintf('%8.4f%15.4f\n',solp');

sol1 = @(t)(exp(-(t+1).^2./(2*t))./(sqrt(2*pi)*t.^(3/2)));
[errmax,j] = max(yn - sol1(t)');
fprintf('\nThe maximum absolute error, %10.2e ',errmax);
fprintf(' occurred at t = %8.4f\nwith solution %8.4f\n\n',t(j), yn(j));

ck2 = @(t) sqrt(pi);
cf2 = @(t) -2*log(sqrt(1+t)+sqrt(t))/sqrt(1+t);
cg2 = @(s, y) y;
initwt = 'Subsequent';
tlim = 5;
yn(1) = 1;
[yn, work, ifail] = d05be( ...
    ck2, cf2, cg2, initwt, tlim, yn, 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(8:7:nmesh,1:2);
fprintf('%8.4f%15.4f\n',solp');
```

```

sol2 = @(t)(1./(1+t));
[errmax,j] = max(yn - sol2(t)');
fprintf('\nThe maximum absolute error, %10.2e ',errmax);
fprintf(' occurred at t = %8.4f\nwith solution %8.4f\n\n',t(j), yn(j));

```

## 9.2 Program Results

d05be example results

Example 1

The stepsize h = 0.1000

t	Approximate Solution
0.5000	0.1191
1.0000	0.0528
1.5000	0.0265
2.0000	0.0146
2.5000	0.0086
3.0000	0.0052
3.5000	0.0033
4.0000	0.0022
4.5000	0.0014
5.0000	0.0010
5.5000	0.0007
6.0000	0.0004
6.5000	0.0003
7.0000	0.0002

The maximum absolute error, 2.86e-03 occurred at t = 0.1000  
with solution 0.0326

Example 2

The stepsize h = 0.0714

t	Approximate Solution
0.5000	0.6667
1.0000	0.5000
1.5000	0.4000
2.0000	0.3333
2.5000	0.2857
3.0000	0.2500
3.5000	0.2222
4.0000	0.2000
4.5000	0.1818
5.0000	0.1667

The maximum absolute error, 3.17e-06 occurred at t = 0.0714  
with solution 0.9333

---