

NAG Toolbox

nag_inteq_volterra2 (d05ba)

1 Purpose

nag_inteq_volterra2 (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, work, ifail] = nag_inteq_volterra2(ck, cg, cf, method, iorder,
alim, tlim, nmesh, tol, thresh, lwk)

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

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

At Mark 23: **work** was made an output parameter.

3 Description

nag_inteq_volterra2 (d05ba) computes the numerical solution of the nonlinear convolution Volterra integral equation of the second kind

$$y(t) = f(t) + \int_a^t k(t-s)g(s, y(s)) ds, \quad a \leq t \leq T. \quad (1)$$

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 nag_inteq_volterra2 (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** – 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 the kernel $k(t)$ evaluated at **t**.

- 2: **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**.

- 3: **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**.

4: **method** – CHARACTER(1)

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

The order of the method to be used.

Constraints:

if **method** = 'A', $3 \leq \mathbf{iorder} \leq 6$;

if **method** = 'B', $2 \leq \mathbf{iorder} \leq 5$.

6: **alim** – REAL (KIND=nag_wp)

a , the lower limit of the integration interval.

Constraint: **alim** ≥ 0.0 .

7: **tlim** – REAL (KIND=nag_wp)

The final point of the integration interval, T .

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

8: **nmesh** – INTEGER

The number of equidistant points at which the solution is sought.

Constraints:

if **method** = 'A', **nmesh** $\geq \mathbf{iorder} - 1$;

if **method** = 'B', **nmesh** $\geq \mathbf{iorder}$.

9: **tol** – REAL (KIND=nag_wp)

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

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

10: **thresh** – REAL (KIND=nag_wp)

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

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

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** – INTEGER

The dimension of the array **work**.

Constraint: $\mathbf{lwk} \geq 10 \times \mathbf{nmesh} + 6$.

Note: the above value of **lwk** is sufficient for nag_inteq_volterra2 (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 Output Parameters

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

yn(i) contains the most recent approximation of the true solution $y(t)$ at the specified point $t = \mathbf{alim} + i \times H$, for $i = 1, 2, \dots, \mathbf{nmesh}$, where $H = (\mathbf{tlim} - \mathbf{alim})/\mathbf{nmesh}$.

2: **errest**(**nmesh**) – REAL (KIND=nag_wp) array

errest(i) contains the most recent approximation of the relative error in the computed solution at the point $t = \mathbf{alim} + i \times H$, for $i = 1, 2, \dots, \mathbf{nmesh}$, where $H = (\mathbf{tlim} - \mathbf{alim})/\mathbf{nmesh}$.

3: **work**(**lwk**) – REAL (KIND=nag_wp) array

If **ifail** = 5 or 6, **work**(1) contains the size of **lwk** required for the algorithm to proceed further.

4: **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, **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** < $10 \times \mathbf{nmesh} + 6$.

ifail = 4

The solution of the nonlinear equation (2) (see Section 9 for further details) could not be computed by nag_roots_contfn_interval_rcomm (c05av) and nag_roots_contfn_brent_rcomm (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 9 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 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 **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 9) 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

$$Y_n - \alpha g(t_n, Y_n) - \Psi_n = 0, \quad (2)$$

is required, where Ψ_n and α are constants. `nag_inteq_volterra2` (d05ba) calls `nag_roots_conf_n_interval_rcomm` (c05av) to find an interval for the zero of this equation followed by `nag_roots_conf_n_brent_rcomm` (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, `nag_inteq_volterra2` (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,

$$|g(t, u) - g(t, v)| \leq L|u - v|. \quad (3)$$

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

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

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

```
function d05ba_example

fprintf('d05ba example results\n\n');

ck = @(t) exp(-t);
cg = @(s, y) y + exp(-y);
cf = @(t) exp(-t);
method = 'A';
iorder = nag_int(6);
alim = 0;
tlim = 20;
nmesh = nag_int(6);
tol = 0.001;
thresh = x02aj;
lwk = nag_int(1000);
[yn, errest, work, ifail] = ...
    d05ba( ...
        ck, cg, cf, method, iorder, alim, tlim, nmesh, tol, thresh, lwk);

fprintf('Workspace requested = %4d\n', lwk);
fprintf('Tolerance           = %8.2e\n\n', tol);
fprintf(' t          approx. sol. true sol. est. error   actual error\n');
dt = (tlim-alim)/(double(nmesh));
t = [alim+dt:dt:tlim];
sol = log(t+exp(1))';
err = abs(yn-sol)./sol;
fprintf('%6.2f%14.5f%12.5f%15.5e%15.5e\n', [t' yn sol errest, err]');
```

9.2 Program Results

```
d05ba example results

Workspace requested = 1000
Tolerance           = 1.00e-03

 t          approx. sol. true sol. est. error   actual error
 3.33      1.80037    1.80033    8.03776e-05  2.38466e-05
 6.67      2.23916    2.23911    1.77740e-04  2.34772e-05
10.00      2.54310    2.54304    2.45949e-04  2.24564e-05
13.33      2.77587    2.77581    3.05737e-04  2.17430e-05
16.67      2.96456    2.96450    3.61698e-04  2.13821e-05
20.00      3.12324    3.12317    4.17127e-04  2.13102e-05
```
