NAG Toolbox

nag_quad_1d_fin_bad_vec (d01at)

1 Purpose

nag_quad_1d_fin_bad_vec (d01at) is a general purpose integrator which calculates an approximation to the integral of a function $f(x)$ over a finite interval $[a, b]$:

$$I = \int_a^b f(x)\,dx.$$

2 Syntax

```
[result, abserr, w, iw, ifail] = nag_quad_1d_fin_bad_vec(f, a, b, epsabs, epsrel, 'lw', lw, 'liw', liw)
[result, abserr, w, iw, ifail] = d01at(f, a, b, epsabs, epsrel, 'lw', lw, 'liw', liw)
```

3 Description

nag_quad_1d_fin_bad_vec (d01at) is based on the QUADPACK routine QAGS (see Piessens *et al.* (1983)). It is an adaptive function, using the Gauss 10-point and Kronrod 21-point rules. The algorithm, described in de Doncker (1978), incorporates a global acceptance criterion (as defined by Malcolm and Simpson (1976)) together with the ϵ-algorithm (see Wynn (1956)) to perform extrapolation. The local error estimation is described in Piessens *et al.* (1983).

The function is suitable as a general purpose integrator, and can be used when the integrand has singularities, especially when these are of algebraic or logarithmic type.

nag_quad_1d_fin_bad_vec (d01at) requires a function to evaluate the integrand at an array of different points and is therefore amenable to parallel execution. Otherwise the algorithm is identical to that used by nag_quad_1d_fin_bad (d01aj).

4 References

de Doncker E (1978) An adaptive extrapolation algorithm for automatic integration *ACM SIGNUM Newsl.* **13(2)** 12–18

Malcolm M A and Simpson R B (1976) Local versus global strategies for adaptive quadrature *ACM Trans. Math. Software* **1** 129–146

Piessens R, de Doncker–Kapenga E, Ûberhuber C and Kahaner D (1983) *QUADPACK, A Subroutine Package for Automatic Integration* Springer–Verlag

Wynn P (1956) On a device for computing the $e_m(S_n)$ transformation *Math. Tables Aids Comput.* **10** 91–96

5 Parameters

5.1 Compulsory Input Parameters

1: **f** – SUBROUTINE, supplied by the user.

f must return the values of the integrand f at a set of points.

```
[fv] = f(x, n)
```

Input Parameters

1: **x**(**n**) – REAL (KIND=nag_wp) array

The points at which the integrand f must be evaluated.

2: **n** – INTEGER

The number of points at which the integrand is to be evaluated. The actual value of **n** is always 21.

Output Parameters

1: **fv**(**n**) – REAL (KIND=nag_wp) array

fv(j) must contain the value of f at the point **x**(j), for $j = 1, 2, \ldots,$ **n**.

2: **a** – REAL (KIND=nag_wp)

a, the lower limit of integration.

3: **b** – REAL (KIND=nag_wp)

b, the upper limit of integration. It is not necessary that $a < b$.

4: **epsabs** – REAL (KIND=nag_wp)

The absolute accuracy required. If **epsabs** is negative, the absolute value is used. See Section 7.

5: **epsrel** – REAL (KIND=nag_wp)

The relative accuracy required. If **epsrel** is negative, the absolute value is used. See Section 7.

5.2 Optional Input Parameters

1: **lw** – INTEGER

Suggested value: **lw** = 800 to 2000 is adequate for most problems.

Default: 800

The dimension of the array **w**. the value of **lw** (together with that of **liw**) imposes a bound on the number of sub-intervals into which the interval of integration may be divided by the function. The number of sub-intervals cannot exceed **lw**/4. The more difficult the integrand, the larger **lw** should be.

Constraint: **lw** ≥ 4.

2: **liw** – INTEGER

Suggested value: **liw** = **lw**/4.

Default: **lw**/4

The dimension of the array **iw**. the number of sub-intervals into which the interval of integration may be divided cannot exceed **liw**.

Constraint: **liw** ≥ 1.

5.3 Output Parameters

1: **result** – REAL (KIND=nag_wp)

The approximation to the integral I.

2: **abserr** – REAL (KIND=nag_wp)

An estimate of the modulus of the absolute error, which should be an upper bound for $|I - \mathbf{result}|$.

3: **w**(**lw**) – REAL (KIND=nag_wp) array

Details of the computation see Section 9 for more information.

4: **iw**(**liw**) – INTEGER array

iw(1) contains the actual number of sub-intervals used. The rest of the array is used as workspace.

5: **ifail** – INTEGER

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

6 Error Indicators and Warnings

Note: nag_quad_1d_fin_bad_vec (d01at) may return useful information for one or more of the following detected errors or warnings.

Errors or warnings detected by the function:

ifail = 1 (*warning*)

The maximum number of subdivisions allowed with the given workspace has been reached without the accuracy requirements being achieved. Look at the integrand in order to determine the integration difficulties. If the position of a local difficulty within the interval can be determined (e.g., a singularity of the integrand or its derivative, a peak, a discontinuity, etc.) you will probably gain from splitting up the interval at this point and calling the integrator on the subranges. If necessary, another integrator, which is designed for handling the type of difficulty involved, must be used. Alternatively, consider relaxing the accuracy requirements specified by **epsabs** and **epsrel**, or increasing the amount of workspace.

ifail = 2 (*warning*)

Round-off error prevents the requested tolerance from being achieved. Consider requesting less accuracy.

ifail = 3 (*warning*)

Extremely bad local integrand behaviour causes a very strong subdivision around one (or more) points of the interval. The same advice applies as in the case of **ifail** = 1.

ifail = 4 (*warning*)

The requested tolerance cannot be achieved because the extrapolation does not increase the accuracy satisfactorily; the returned result is the best which can be obtained. The same advice applies as in the case of **ifail** = 1.

ifail = 5 (*warning*)

The integral is probably divergent, or slowly convergent. Please note that divergence can occur with any nonzero value of **ifail**.

ifail = 6

On entry, **lw** < 4,
or **liw** < 1.

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

nag_quad_1d_fin_bad_vec (d01at) cannot guarantee, but in practice usually achieves, the following accuracy:

$$|I - \textbf{result}| \leq tol,$$

where

$$tol = \max\{|\textbf{epsabs}|, |\textbf{epsrel}| \times |I|\},$$

and **epsabs** and **epsrel** are user-specified absolute and relative error tolerances. Moreover, it returns the quantity **abserr** which, in normal circumstances, satisfies

$$|I - \textbf{result}| \leq \textbf{abserr} \leq tol.$$

8 Further Comments

If **ifail** $\neq 0$ on exit, then you may wish to examine the contents of the array **w**, which contains the end points of the sub-intervals used by nag_quad_1d_fin_bad_vec (d01at) along with the integral contributions and error estimates over the sub-intervals.

Specifically, for $i = 1, 2, \ldots, n$, let r_i denote the approximation to the value of the integral over the sub-interval $[a_i, b_i]$ in the partition of $[a, b]$ and e_i be the corresponding absolute error estimate. Then, $\int_{a_i}^{b_i} f(x)\,dx \simeq r_i$ and $\textbf{result} = \sum_{i=1}^{n} r_i$, unless nag_quad_1d_fin_bad_vec (d01at) terminates while testing for divergence of the integral (see Section 3.4.3 of Piessens *et al.* (1983)). In this case, **result** (and **abserr**) are taken to be the values returned from the extrapolation process. The value of n is returned in $\textbf{iw}(1)$, and the values a_i, b_i, e_i and r_i are stored consecutively in the array **w**, that is:

$a_i = \textbf{w}(i)$,

$b_i = \textbf{w}(n + i)$,

$e_i = \textbf{w}(2n + i)$ and

$r_i = \textbf{w}(3n + i)$.

9 Example

This example computes

$$\int_0^{2\pi} \frac{x \sin(30x)}{\sqrt{1 - (x/2\pi)^2}}\,dx.$$

9.1 Program Text

```
     function d01at_example

fprintf('d01at example results\n\n');

a = 0;
b = 2*pi;
epsabs = 0;
epsrel = 0.0001;
[result, abserr, w, iw, ifail] = ...
  d01at( ...
  @f, a, b, epsabs, epsrel);

fprintf('Result = %13.4f,  Standard error = %10.2e\n', result, abserr);

function [fv] = f(x,n)
  fv = zeros(n,1);
  for i = 1:n
    fv(i) = x(i)*sin(30*x(i))/sqrt(1-(x(i)/(2*pi))^2);
  end
```

9.2 Program Results

```
     d01at example results

Result =       -2.5433,  Standard error =   1.28e-05
```