NAG Toolbox for MATLAB

d01aq

1 Purpose

d01aq calculates an approximation to the Hilbert transform of a function $g(x)$ over $[a, b]$:

$$I = \int_a^b \frac{g(x)}{x - c} dx$$

for user-specified values of a, b and c.

2 Syntax

```
[result, abserr, w, iw, ifail] = d01aq(g, a, b, c, epsabs, epsrel, 'lw',
lw, 'liw', liw)
```

3 Description

d01aq is based on the QUADPACK routine QAWC (see Piessens *et al.* 1983) and integrates a function of the form $g(x)w(x)$, where the weight function

$$w(x) = \frac{1}{x - c}$$

is that of the Hilbert transform. (If $a < c < b$ the integral has to be interpreted in the sense of a Cauchy principal value.) It is an adaptive function which employs a 'global' acceptance criterion (as defined by Malcolm and Simpson 1976). Special care is taken to ensure that c is never the end point of a sub-interval (see Piessens *et al.* 1976). On each sub-interval (c_1, c_2) modified Clenshaw–Curtis integration of orders 12 and 24 is performed if $c_1 - d \le c \le c_2 + d$ where $d = (c_2 - c_1)/20$. Otherwise the Gauss 7-point and Kronrod 15-point rules are used. The local error estimation is described by Piessens *et al.* 1983.

4 References

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, Überhuber C and Kahaner D 1983 *QUADPACK, A Subroutine Package for Automatic Integration* Springer–Verlag

Piessens R, van Roy–Branders M and Mertens I 1976 The automatic evaluation of Cauchy principal value integrals *Angew. Inf.* **18** 31–35

5 Parameters

5.1 Compulsory Input Parameters

1: **g – string containing name of m-file**

g must return the value of the function g at a given point **x**.

Its specification is:

```
[result] = g(x)
```

Input Parameters

1: **x – double scalar**

The point at which the function g must be evaluated.

Output Parameters

1: **result – double scalar**

The result of the function.

2: **a – double scalar**

a, the lower limit of integration.

3: **b – double scalar**

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

4: **c – double scalar**

The parameter c in the weight function.

Constraint: **c** must not equal **a** or **b**.

5: **epsabs – double scalar**

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

6: **epsrel – double scalar**

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

5.2 Optional Input Parameters

1: **lw – int32 scalar**

Default: 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.

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

Default: 800

Constraint: **lw** ≥ 4.

2: **liw – int32 scalar**

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

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

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

Default: **lw**/4

Constraint: **liw** ≥ 1.

5.3 Input Parameters Omitted from the MATLAB Interface

None.

5.4 Output Parameters

1: **result – double scalar**

The approximation to the integral I.

2: **abserr – double scalar**

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

3: **w(lw) – double array**

Details of the computation, as described in Section 8.

4: **iw(liw) – int32 array**

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

5: **ifail – int32 scalar**

0 unless the function detects an error (see Section 6).

6 Error Indicators and Warnings

Note: d01aq may return useful information for one or more of the following detected errors or warnings.

ifail = 1

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 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

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

ifail = 3

Extremely bad local behaviour of $g(x)$ 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

On entry, **c** = **a** or **c** = **b**.

ifail = 5

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

7 Accuracy

d01aq cannot guarantee, but in practice usually achieves, the following accuracy:

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

where

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

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

The time taken by d01aq depends on the integrand and the accuracy required.

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 d01aq along with the integral contributions and error estimates over these 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, $\int_{a_i}^{b_i} g(x)w(x)\,dx \simeq r_i$ and $\textbf{result} = \sum_{i=1}^{n} r_i$. 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

```
d01aq_g.m

function result = d01aq_g(x)
  result = 1/(x^2+0.01^2);
```

```
a = -1;
b = 1;
c = 0.5;
epsabs = 0;
epsrel = 0.0001;
[result, abserr, w, iw, ifail] = d01aq('d01aq_g', a, b, c, epsabs, epsrel)
```

```
result =
 -628.4617
abserr =
    0.0132
w =
     array elided
iw =
     array elided
ifail =
           0
```