NAG C Library Function Document

nag_1d_quad_wt_cauchy (d01aqc)

1 Purpose

nag_1d_quad_wt_cauchy (d01aqc) 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 Specification

```
#include <nag.h>
#include <nagd01.h>
```

```
void nag_1d_quad_wt_cauchy (double (*g)(double x),
          double a, double b, double c, double epsabs, double epsrel,
          Integer max_num_subint, double *result, double *abserr,
          Nag_QuadProgress *qp, NagError *fail)
```

3 Description

This function is based upon the QUADPACK routine QAWC (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 routine 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 (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 Parameters

1: **g** – function supplied by user *Function*

The function **g**, supplied by the user, must return the value of the function g at a given point.

The specification of **g** is:

```
double g(double x)
```

1: **x** – double *Input*

On entry: the point at which the function g must be evaluated.

2: **a** – double *Input*

On entry: the lower limit of integration, a.

3: **b** – double *Input*

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

4: **c** – double *Input*

On entry: the parameter c in the weight function.

Constraints: **c** $\neq$ **a** or **b**.

5: **epsabs** – double *Input*

On entry: the absolute accuracy required. If **epsabs** is negative, the absolute value is used. See Section 6.1.

6: **epsrel** – double *Input*

On entry: the relative accuracy required. If **epsrel** is negative, the absolute value is used. See Section 6.1.

7: **max_num_subint** – Integer *Input*

On entry: the upper bound on the number of sub-intervals into which the interval of integration may be divided by the function. The more difficult the integrand, the larger **max_num_subint** should be.

Suggested values: a value in the range 200 to 500 is adequate for most problems.

Constraint: **max_num_subint** ≥ 1.

8: **result** – double * *Output*

On exit: the approximation to the integral I.

9: **abserr** – double * *Output*

On exit: an estimate of the modulus of the absolute error, which should be an upper bound for $|I-$**result**$|$.

10: **qp** – Nag_QuadProgress *

Pointer to structure of type **Nag_QuadProgress** with the following members:

num_subint – Integer *Output*

On exit: the actual number of sub-intervals used.

fun_count – Integer *Output*

On exit: the number of function evaluations performed by nag_1d_quad_wt_cauchy.

sub_int_beg_pts – double * *Output*
sub_int_end_pts – double * *Output*
sub_int_result – double * *Output*
sub_int_error – double * *Output*

On exit: these pointers are allocated memory internally with **max_num_subint** elements. If an error exit other than **NE_INT_ARG_LT**, **NE_2_REAL_ARG_EQ** or **NE_ALLOC_FAIL** occurs, these arrays will contain information which may be useful. For details, see Section 6.

Before a subsequent call to nag_1d_quad_wt_cauchy is made, or when the information contained in these arrays is no longer useful, the user should free the storage allocated by these pointers using the NAG macro **NAG_FREE**.

11: **fail** – NagError * *Input/Output*

The NAG error parameter (see the Essential Introduction).

Users are recommended to declare and initialise **fail** and set **fail.print** = **TRUE** for this function.

5 Error Indicators and Warnings

NE_INT_ARG_LT

On entry, **max_num_subint** must not be less than 1: **max_num_subint** = <*value*>.

NE_2_REAL_ARG_EQ

On entry, **c** = <*value*> while **a** = <*value*>. These parameters must satisfy **c** $\neq$ **a**.
On entry, **c** = <*value*> while **b** = <*value*>. These parameters must satisfy **c** $\neq$ **b**.

NE_ALLOC_FAIL

Memory allocation failed.

NE_QUAD_MAX_SUBDIV

The maximum number of subdivisions has been reached: **max_num_subint** = <*value*>.
The maximum number of subdivisions has been reached without the accuracy requirements being achieved. Look at the integrand in order to determine the integration difficulties. 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 value of **max_num_subint**.

NE_QUAD_ROUNDOFF_TOL

Round-off error prevents the requested tolerance from being achieved: **epsabs** = <*value*>, **epsrel** = <*value*>.
The error may be underestimated. Consider relaxing the accuracy requirements specified by **epsabs** and **epsrel**.

NE_QUAD_BAD_SUBDIV

Extremely bad integrand behaviour occurs around the sub-interval (<*value*>, <*value*>).
The same advice applies as in the case of **NE_QUAD_MAX_SUBDIV**.

6 Further Comments

The time taken by nag_1d_quad_wt_cauchy depends on the integrand and the accuracy required.

If the function fails with an error exit other than **NE_INT_ARG_LT**, **NE_2_REAL_ARG_EQ** or **NE_ALLOC_FAIL**, then the user may wish to examine the contents of the structure **qp**. These contain the end-points of the sub-intervals used by nag_1d_quad_wt_cauchy 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, $\int_{a_i}^{b_i} g(x)w(x)\,dx \simeq r_i$ and **result** $= \sum_{i=1}^{n} r_i$. The value of n is returned in **num_subint**, and the values a_i, b_i, r_i and e_i are stored in the structure **qp** as

a_i = **sub_int_beg_pts**$[i-1]$,

b_i = **sub_int_end_pts**$[i-1]$,

r_i = **sub_int_result**$[i-1]$ and

e_i = **sub_int_error**$[i-1]$.

6.1 Accuracy

The function 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

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

6.2 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, Van Roy-Branders M and Mertens I (1976) The automatic evaluation of Cauchy principal value integrals *Angew. Inf.* **18** 31–35

Piessens R, De Doncker-Kapenga E, Überhuber C and Kahaner D (1983) *QUADPACK, A Subroutine Package for Automatic Integration* Springer-Verlag

7 See Also

nag_1d_quad_gen (d01ajc)

8 Example

To compute

$$\int_{-1}^{1} \frac{dx}{(x^2 + 0.01^2)(x - \frac{1}{2})}.$$

8.1 Program Text

```
/* nag_1d_quad_wt_cauchy(d01aqc) Example Program
 *
 * Copyright 1991 Numerical Algorithms Group.
 *
 * Mark 2, 1991.
 *
 * Mark 6 revised, 2000.
 */

#include <nag.h>
#include <stdio.h>
#include <nag_stdlib.h>
#include <nagd01.h>

static double g(double x);

main()
{
  double a, b, c;
  double epsabs, abserr, epsrel, result;
  Nag_QuadProgress qp;
  Integer max_num_subint;
  static NagError fail;
```

```
  Vprintf("d01aqc Example Program Results\n");
  epsabs = 0.0;
  epsrel = 0.0001;
  a = -1.0;
  b = 1.0;
  c = 0.5;
  max_num_subint = 200;
  d01aqc(g, a, b, c, epsabs, epsrel, max_num_subint, &result, &abserr,
         &qp, &fail);

  Vprintf("a      - lower limit of integration = %10.4f\n", a);
  Vprintf("b      - upper limit of integration = %10.4f\n", b);
  Vprintf("epsabs - absolute accuracy requested = %9.2e\n", epsabs);
  Vprintf("epsrel - relative accuracy requested = %9.2e\n\n", epsrel);
  Vprintf("c      - parameter in the weight function = %9.2e\n", c);
  if (fail.code != NE_NOERROR)
    Vprintf("%s\n", fail.message);
  if (fail.code != NE_INT_ARG_LT && fail.code != NE_2_REAL_ARG_EQ &&
      fail.code != NE_ALLOC_FAIL)
    {
      Vprintf("result - approximation to the integral = %9.2f\n", result);
      Vprintf("abserr - estimate of the absolute error = %9.2e\n", abserr);
      Vprintf("qp.fun_count  - number of function evaluations = %4ld\n",
              qp.fun_count);
      Vprintf("qp.num_subint  - number of subintervals used = %4ld\n",
              qp.num_subint);
      /* Free memory used by qp */
      NAG_FREE(qp.sub_int_beg_pts);
      NAG_FREE(qp.sub_int_end_pts);
      NAG_FREE(qp.sub_int_result);
      NAG_FREE(qp.sub_int_error);
      exit(EXIT_SUCCESS);
    }
  exit(EXIT_FAILURE);
}


static double g(double x)
{
  double aa;

  aa = 0.01;
  return 1.0/(x*x+aa*aa);
}
```

8.2 Program Data

None.

8.3 Program Results

```
d01aqc Example Program Results
a      - lower limit of integration =    -1.0000
b      - upper limit of integration =     1.0000
epsabs - absolute accuracy requested =  0.00e+00
epsrel - relative accuracy requested =  1.00e-04

c      - parameter in the weight function =  5.00e-01
result - approximation to the integral =   -628.46
abserr - estimate of the absolute error =  1.32e-02
qp.fun_count  - number of function evaluations =  255
qp.num_subint  - number of subintervals used =    8
```