NAG Library Function Document

nag_1d_quad_wt_cauchy_1 (d01sqc)

1 Purpose

nag_1d_quad_wt_cauchy_1 (d01sqc) 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_1 (
     double (*g)(double x, Nag_User *comm),
     double a, double b, double c, double epsabs, double epsrel,
     Integer max_num_subint, double *result, double *abserr,
     Nag_QuadProgress *qp, Nag_User *comm, NagError *fail)
```

3 Description

nag_1d_quad_wt_cauchy_1 (d01sqc) 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 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 (Piessens *et al.* (1976)). On each sub-interval (c_1, c_2) modified Clenshaw–Curtis integration of orders 12 and 24 is performed if $c_1 - d \leq c \leq c_2 + d$ 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, Ûberhuber 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 Arguments

1: **g** – function, supplied by the user *External Function*

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

The specification of **g** is:

```
double g (double x, Nag_User *comm)
```

1: **x** – double *Input*

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

2: **comm** – Nag_User *

Pointer to a structure of type Nag_User with the following member:

p – Pointer

On entry/exit: the pointer **comm**→**p** should be cast to the required type, e.g., `struct user *s = (struct user *)comm → p`, to obtain the original object's address with appropriate type. (See the argument **comm** below.)

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 argument c in the weight function.

Constraint: **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 7.

6: **epsrel** – double *Input*

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

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.

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 $|I - \textbf{result}|$.

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_1 (d01sqc).

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 9.

Before a subsequent call to nag_1d_quad_wt_cauchy_1 (d01sqc) is made, or when the information contained in these arrays is no longer useful, you should free the storage allocated by these pointers using the NAG macro `NAG_FREE`.

11: **comm** – Nag_User *

Pointer to a structure of type Nag_User with the following member:

p – Pointer

On entry/exit: the pointer **comm**→**p**, of type Pointer, allows you to communicate information to and from **g**(). An object of the required type should be declared, e.g., a structure, and its address assigned to the pointer **comm**→**p** by means of a cast to Pointer in the calling program, e.g., `comm.p = (Pointer)&s`. The type Pointer is `void *`.

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

The NAG error argument (see Section 2.7 in How to Use the NAG Library and its Documentation).

6 Error Indicators and Warnings

NE_2_REAL_ARG_EQ

On entry, $\mathbf{c} = \langle value \rangle$ while $\mathbf{a} = \langle value \rangle$. These arguments must satisfy $\mathbf{c} \neq \mathbf{a}$.

On entry, $\mathbf{c} = \langle value \rangle$ while $\mathbf{b} = \langle value \rangle$. These arguments must satisfy $\mathbf{c} \neq \mathbf{b}$.

NE_ALLOC_FAIL

Dynamic memory allocation failed.

NE_INT_ARG_LT

On entry, **max_num_subint** must not be less than 1: **max_num_subint** $= \langle value \rangle$.

NE_QUAD_BAD_SUBDIV

Extremely bad integrand behaviour occurs around the sub-interval $(\langle value \rangle, \langle value \rangle)$.
The same advice applies as in the case of NE_QUAD_MAX_SUBDIV.

NE_QUAD_MAX_SUBDIV

The maximum number of subdivisions has been reached: **max_num_subint** $= \langle value \rangle$.

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** $= \langle value \rangle$, **epsrel** $= \langle value \rangle$.
The error may be underestimated. Consider relaxing the accuracy requirements specified by **epsabs** and **epsrel**.

7 Accuracy

nag_1d_quad_wt_cauchy_1 (d01sqc) 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 Parallelism and Performance

nag_1d_quad_wt_cauchy_1 (d01sqc) is not threaded in any implementation.

9 Further Comments

The time taken by nag_1d_quad_wt_cauchy_1 (d01sqc) 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 you 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_1 (d01sqc) 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 **qp**→**num_subint**, and the values a_i, b_i, r_i and e_i are stored in the structure **qp** as

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

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

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

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

10 Example

This example computes

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

10.1 Program Text

```
/* nag_1d_quad_wt_cauchy_1 (d01sqc) Example Program.
 *
 * NAGPRODCODE Version.
 *
 * Copyright 2016 Numerical Algorithms Group.
 *
 * Mark 26, 2016.
 *
 */

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

#ifdef __cplusplus
extern "C"
{
#endif
  static double NAG_CALL g(double x, Nag_User *comm);
#ifdef __cplusplus
}
#endif

int main(void)
{
  static Integer use_comm[1] = { 1 };
  Integer exit_status = 0;
  double a, b, c;
  double epsabs, abserr, epsrel, result;
  Nag_QuadProgress qp;
  Integer max_num_subint;
  NagError fail;
  Nag_User comm;

  INIT_FAIL(fail);

  printf("nag_1d_quad_wt_cauchy_1 (d01sqc) Example Program Results\n");

  /* For communication with user-supplied functions: */
  comm.p = (Pointer) &use_comm;

  epsabs = 0.0;
  epsrel = 0.0001;
  a = -1.0;
  b = 1.0;
  c = 0.5;
  max_num_subint = 200;
  /* nag_1d_quad_wt_cauchy_1 (d01sqc).
   * One-dimensional adaptive quadrature, weight function
   * 1/(x-c), Cauchy principal value, thread-safe
   */
  nag_1d_quad_wt_cauchy_1(g, a, b, c, epsabs, epsrel, max_num_subint, &result,
                          &abserr, &qp, &comm, &fail);
  printf("a      - lower limit of integration = %10.4f\n", a);
  printf("b      - upper limit of integration = %10.4f\n", b);
  printf("epsabs - absolute accuracy requested = %11.2e\n", epsabs);
  printf("epsrel - relative accuracy requested = %11.2e\n\n", epsrel);
  printf("c      - parameter in the weight function = %11.2e\n", c);
  if (fail.code != NE_NOERROR)
    printf("Error from nag_1d_quad_wt_cauchy_1 (d01sqc) %s\n", fail.message);
  if (fail.code != NE_INT_ARG_LT && fail.code != NE_2_REAL_ARG_EQ &&
      fail.code != NE_ALLOC_FAIL && fail.code != NE_NO_LICENCE) {
    printf("result - approximation to the integral = %9.2f\n", result);
    printf("abserr - estimate of the absolute error = %11.2e\n", abserr);
    printf("qp.fun_count  - number of function evaluations = %4" NAG_IFMT
           "\n", qp.fun_count);
    printf("qp.num_subint  - number of subintervals used = %4" NAG_IFMT "\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);
  }
  else {
    exit_status = 1;
    goto END;
  }

END:
  return exit_status;
}

static double NAG_CALL g(double x, Nag_User *comm)
{
  double aa;
  Integer *use_comm = (Integer *) comm->p;

  if (use_comm[0]) {
    printf("(User-supplied callback g, first invocation.)\n");
    use_comm[0] = 0;
  }

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

10.2 Program Data

None.

10.3 Program Results

```
nag_1d_quad_wt_cauchy_1 (d01sqc) Example Program Results
(User-supplied callback g, first invocation.)
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
```