NAG C 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

```c
#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

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 $c_1 - d \leq c \leq c_2 + d$ 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 Parameters

1: $g$ – function supplied by user  

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

The specification of $g$ is:

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

1: $x$ – double  

*Input*

*On entry:* the point at which the function $g$ must be evaluated.
2: \textbf{comm} – Nag_User *  

*On entry/on exit:* pointer to a structure of type Nag_User with the following member:

\begin{itemize}
  \item \textbf{p} – Pointer \hspace{1cm} \textit{Input/Output}
\end{itemize}

*On entry/on exit:* the pointer \texttt{comm->p} should be cast to the required type, e.g.,

\begin{verbatim}
struct user *s = (struct user *)comm->p,
\end{verbatim}

to obtain the original object’s address with appropriate type. (See the argument \texttt{comm} below.)

2: \textbf{a} – double \hspace{1cm} \textit{Input}

*On entry:* the lower limit of integration, \(a\).

3: \textbf{b} – double \hspace{1cm} \textit{Input}

*On entry:* the upper limit of integration, \(b\). It is not necessary that \(a < b\).

4: \textbf{c} – double \hspace{1cm} \textit{Input}

*On entry:* the parameter \(c\) in the weight function.

\textbf{Constraints:} \texttt{c \neq a \text{ or } b}.

5: \textbf{epsabs} – double \hspace{1cm} \textit{Input}

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

6: \textbf{epsrel} – double \hspace{1cm} \textit{Input}

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

7: \textbf{max_num_subint} – Integer \hspace{1cm} \textit{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 \texttt{max_num_subint} should be.

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

\textbf{Constraint:} \texttt{max_num_subint \geq 1}.

8: \textbf{result} – double * \hspace{1cm} \textit{Output}

*On exit:* the approximation to the integral \(I\).

9: \textbf{abserr} – double * \hspace{1cm} \textit{Output}

*On exit:* an estimate of the modulus of the absolute error, which should be an upper bound for \(|I - \text{result}|\).

10: \textbf{qp} – Nag_QuadProgress *

Pointer to structure of type Nag_QuadProgress with the following members:

\begin{itemize}
  \item \textbf{num_subint} – Integer \hspace{1cm} \textit{Output}
\end{itemize}

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

\begin{itemize}
  \item \textbf{fun_count} – Integer \hspace{1cm} \textit{Output}
\end{itemize}

*On exit:* the number of function evaluations performed by nag_1d_quad_wt_cauchy_1.
sub_int_beg_pts – double *  
sub_int_end_pts – double *  
sub_int_result – double *  
sub_int_error – double *

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_1 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:  comm – Nag_User *

On entry/on exit: pointer to a structure of type Nag_User with the following member:

p – Pointer  

On entry/on exit: the pointer p, of type Pointer, allows the user to communicate information to and from the user-defined function f(). An object of the required type should be declared by the user, e.g., a structure, and its address assigned to the pointer 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 *

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 ≠ a.

On entry, c = <value> while b = <value>. These parameters must satisfy c ≠ 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.
6 Further Comments
The time taken by nag_1d_quad_wt_cauchy_1 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_1 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, \( \int_{a_i}^{b_i} g(x)w(x) \, dx \approx r_i \), and \( \text{result} = \sum_{i=1}^{n} r_i \).
The value of \( n \) is returned in \text{num_subint}, and the values \( a_i, b_i, r_i \) and \( e_i \) are stored in the structure \text{qp} as
\[
\begin{align*}
  a_i &= \text{sub_int_beg_pts}[i-1], \\
  b_i &= \text{sub_int_end_pts}[i-1], \\
  r_i &= \text{sub_int_result}[i-1] \quad \text{and} \\
  e_i &= \text{sub_int_error}[i-1].
\end{align*}
\]

6.1 Accuracy
The function cannot guarantee, but in practice usually achieves, the following accuracy:
\[
|I - \text{result}| \leq tol
\]
where
\[
tol = \max\{|\text{epsabs}|, |\text{epsrel}| \times |I|\}
\]
and \text{epsabs} and \text{epsrel} are user-specified absolute and relative error tolerances. Moreover it returns the quantity \text{abserr} which, in normal circumstances, satisfies
\[
|I - \text{result}| \leq \text{abserr} \leq tol.
\]

6.2 References
Math. Software 1 129–146
value integrals Angew. Inf. 18 31–35
Package for Automatic Integration Springer-Verlag

7 See Also
nag_1d_quad_gen_1 (d01sjc)

8 Example
To compute
\[
\int_{-1}^{1} \frac{dx}{(x^2 + 0.01^2)(x - 0.5^2)}.
\]
8.1 Program Text

/* nag_ld_quad_wt_cauchy_1(d0lsqc) Example Program
 * Mark 6 revised, 2000.
 */

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

static double g(double x, Nag_User *comm);

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

    Vprintf("d0lsqc Example Program Results\n");
    epsabs = 0.0;
    epsrel = 0.0001;
    a = -1.0;
    b = 1.0;
    c = 0.5;
    max_num_subint = 200;
    d0lsqc(g, a, b, c, epsabs, epsrel, max_num_subint, &result, &abserr,
          &qp, &comm, &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", 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);
}
static double g(double x, Nag_User *comm)
{
    double aa;
    aa = 0.01;
    return 1.0/(x*x+aa*aa);
}

8.2 Program Data
None.

8.3 Program Results

d01sqc 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