NAG C Library Function Document

nag_1d_quad_wt_trig (d01anc)

1 Purpose

nag_1d_quad_wt_trig (d01anc) calculates an approximation to the sine or the cosine transform of a function \( g \) over \( [a, b] \):

\[
I = \int_a^b g(x) \sin(\omega x) \, dx \quad \text{or} \quad I = \int_a^b g(x) \cos(\omega x) \, dx
\]

(for a user-specified value of \( \omega \)).

2 Specification

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

void nag_1d_quad_wt_trig (double (*g)(double x),
                        double a, double b, double omega, Nag_TrigTransform wt_func,
                        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 QFOUR (Piessens et al. (1983)). It is an adaptive routine, designed to integrate a function of the form \( g(x)w(x) \), where \( w(x) \) is either \( \sin(\omega x) \) or \( \cos(\omega x) \). If a sub-interval has length

\[
L = |b - a|2^{-l}
\]

then the integration over this sub-interval is performed by means of a modified Clenshaw-Curtis procedure (Piessens and Branders (1975)) if \( L\omega > 4 \) and \( l < 20 \). In this case a Chebyshev-series approximation of degree 24 is used to approximate \( g(x) \), while an error estimate is computed from this approximation together with that obtained using Chebyshev-series of degree 12. If the above conditions do not hold then Gauss 7-point and Kronrod 15-point rules are used. The algorithm, described in Piessens et al. (1983), incorporates a global acceptance criterion (as defined in Malcolm and Simpson (1976)) together with the \( \epsilon \)-algorithm (Wynn (1956)) to perform extrapolation. The local error estimation is described in 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:

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

1: \( x \) – double

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

2: \( a \) – double

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

4:  omega – double
   On entry: the parameter ω in the weight function of the transform.

5:  wt_func – Nag_PolyTransform
   On entry: indicates which integral is to be computed:
     if wt_func = Nag_Cosine, w(x) = \cos(\omega x);
     if wt_func = Nag_Sine, w(x) = \sin(\omega x).
   Constraint: wt_func = Nag_Cosine or Nag_Sine.

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

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

8:  max_num_subint – Integer
   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.

9:  result – double *
   On exit: the approximation to the integral I.

10: abserr – double *
    On exit: an estimate of the modulus of the absolute error, which should be an upper bound for |I − result|.

11: qp – Nag_PolyTransform *
    Pointer to structure of type Nag_PolyTransform with the following members:

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

        fun_count – Integer
       On exit: the number of function evaluations performed by nag_1d_quad_wt_poly.

        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_BAD_PARAM 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_trig 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.

12: 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_BAD_PARAM
On entry, parameter wt_func had an illegal value.

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. 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 sub-intervals. 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 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.

NE_QUAD_ROUNDOFF_EXTRAPL
Round-off error is detected during extrapolation. The requested tolerance cannot be achieved, because the extrapolation does not increase the accuracy satisfactorily; the returned result is the best that can be obtained. The same advice applies as in the case of NE_QUAD_MAX_SUBDIV.

NE_QUAD_NO_CONV
The integral is probably divergent or slowly convergent. Please note that divergence can also occur with any error exit other than NE_INT_ARG_LT, NE_BAD_PARAM or NE_ALLOC_FAIL.
6 Further Comments

The time taken by nag_1d_quad_wt_trig depends on the integrand and the accuracy required.

If the function fails with an error exit other than NE_INT_ARG_LT, NE_BAD_PARAM 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_trig 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, \( \int_a^b g(x)w(x) \, dx \simeq r_i \) and \( \text{result} = \sum_{i=1}^{n} r_i \) unless the function terminates while testing for
divergence of the integral (see Section 3.4.3 of Piessens et al. (1983)). In this case, \( \text{result} \) (and \( \text{abser} \))
are taken to be the values returned from the extrapolation process. 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

\[
\begin{align*}
  a_i &= \text{sub_int_beg_pts}[i - 1], \\
  b_i &= \text{sub_int_end_pts}[i - 1], \\
  r_i &= \text{sub_int_result}[i - 1] \text{ and} \\
  e_i &= \text{sub_int_error}[i - 1].
\end{align*}
\]

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{abser} which, in normal circumstances, satisfies

\[
|I - \text{result}| \leq \text{abser} \leq tol.
\]

6.2 References

Math. Software 1 129–146

Math. 1 153–164

Package for Automatic Integration Springer-Verlag

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

7 See Also

nag_1d_quad_gen (d01ajc)

8 Example

To compute

\[
\int_0^1 \ln x \sin(10\pi x) \, dx.
\]
8.1 Program Text

/* nag ld quad wt trig(d01nc) Example Program */
* * Mark 2, 1991.
* * Mark 6 revised, 2000.
*/

#include <nag.h>
#include <stdio.h>
#include <nag_stdbl.h>
#include <math.h>
#include <nagd01.h>
#include <nagx01.h>

static double g(double x);

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

  Vprintf("d01nc Example Program Results\n");
  epsrel = 0.0001;
  epsabs = 0.0;
  a = 0.0;
  b = 1.0;
  omega = X01AAC * 10.0;
  wt_func = Nag_Sine;
  max_num_subint = 200;
  d01nc(g, a, b, omega, wt_func, 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", epsrel);
  if (fail.code != NE_NOERROR)
    Vprintf("%s\n", fail.message);
  if (fail.code != NE_INT_ARG_LT && fail.code != NE_BAD_PARAM &&
      fail.code != NE_ALLOC_FAIL)
  {
    Vprintf("result - approximation to the integral = %9.5f\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);
d01anc

NAG_FREE(qp.sub_int_error);
exit(EXIT_SUCCESS);
}
exit(EXIT_FAILURE);
}

static double g(double x)
{
    return (x>0.0) ? log(x) : 0.0;
}

8.2 Program Data

None.

8.3 Program Results

d01anc Example Program Results
a    - lower limit of integration =  0.0000
b    - upper limit of integration =  1.0000
epsabs - absolute accuracy requested =  0.00e+00
epsrel - relative accuracy requested =  1.00e-04

result - approximation to the integral =  -0.12814
abserr - estimate of the absolute error =  3.58e-06
qp.fun_count - number of function evaluations =  275
qp.num_subint - number of subintervals used =  8