1 Purpose

`nag_1d_quad_wt_trig_1` 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

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

void nag_1d_quad_wt_trig_1 (  
    double (*g)(double x, Nag_User *comm),  
    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, Nag_User *comm, NagError *fail)
```

3 Description

`nag_1d_quad_wt_trig_1` is based upon the QUADPACK routine QFOUR (Piessens et al. (1983)). It is an adaptive function, 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 \( l \leq 20 \). 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 References


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

1: \( g \) – function, supplied by the user

\textbf{External Function}

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

The specification of \( g \) is:

\[
\text{double } g \text{ (double } x, \text{ Nag_User *} \text{comm) }
\]

1: \( x \) – double \hspace{1cm} \textit{Input}

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

2: \( \text{comm} \) – Nag_User *

\textit{On entry/exit}: the pointer \( \text{comm} \rightarrow p \) should be cast to the required type, e.g.,

\[
\text{struct user *} \text{s} = (\text{struct user *} \text{comm}) \rightarrow p,
\]

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

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

\textit{On entry}: the lower limit of integration, \( a \).

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

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

4: \( \omega \) – double \hspace{1cm} \textit{Input}

\textit{On entry}: the argument \( \omega \) in the weight function of the transform.

5: \( \text{wt_func} \) – Nag_TrigTransform \hspace{1cm} \textit{Input}

\textit{On entry}: indicates which integral is to be computed:

\[
\begin{align*}
\text{if } \text{wt_func} &= \text{Nag_Cosine}, \ w(x) = \cos(\omega x); \\
\text{if } \text{wt_func} &= \text{Nag_Sine}, \ w(x) = \sin(\omega x).
\end{align*}
\]

\textit{Constraint}: \( \text{wt_func} = \text{Nag_Cosine} \) or \( \text{Nag_Sine} \).

6: \( \text{epsabs} \) – double \hspace{1cm} \textit{Input}

\textit{On entry}: the absolute accuracy required. If \( \text{epsabs} \) is negative, the absolute value is used. See Section 7.

7: \( \text{epsrel} \) – double \hspace{1cm} \textit{Input}

\textit{On entry}: the relative accuracy required. If \( \text{epsrel} \) is negative, the absolute value is used. See Section 7.

8: \( \text{max_num_subint} \) – Integer \hspace{1cm} \textit{Input}

\textit{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 \( \text{max_num_subint} \) should be.

\textit{Constraint}: \( \text{max_num_subint} \geq 1 \).
**d01 – Quadrature**

9: `result` – double *

*Output*

*On exit:* the approximation to the integral $I$.

10: `abserr` – double *

*Output*

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

11: `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_trig_1 (d01snc).

- `sub_int_beg_pts` – double *

- `sub_int_end_pts` – double *

- `sub_int_result` – double *

- `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_BAD_PARAM 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_trig_1 (d01snc) 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.

12: `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 *.

13: `fail` – NagError *

*Input/Output*

The NAG error argument (see Section 3.6 in the Essential Introduction).

### 6 Error Indicators and Warnings

**NE_ALLOC_FAIL**

Dynamic memory allocation failed.

**NE_BAD_PARAM**

On entry, argument `wt_func` had an illegal value.

**NE_INT_ARG_LT**

On entry, `max_num_subint` must not be less than 1: `max_num_subint = (value)`.
NE_QUAD_BAD_SUBDIV
Extremely bad integrand behaviour occurs around the sub-interval \((\text{value}), (\text{value})\).
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: \text{max\_num\_subint} = (\text{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 \text{epsabs} and \text{epsrel}, or increasing the value of \text{max\_num\_subint}.

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.

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_ROUNDOFF_TOL
Round-off error prevents the requested tolerance from being achieved: \text{epsabs} = (\text{value}), \text{epsrel} = (\text{value}).
The error may be underestimated. Consider relaxing the accuracy requirements specified by \text{epsabs} and \text{epsrel}.

7 Accuracy
\text{nag_1d_quad_wt_trig_1 (d01snc)} 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

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

8 Parallelism and Performance
Not applicable.

9 Further Comments
The time taken by \text{tnag_1d_quad_wt_trig_1 (d01snc)} 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 you may wish to examine the contents of the structure \( \text{qp} \). These contain the end-points of the sub-intervals used by \( \text{nag_1d_quad_wt_trig_1} \) \((\text{d01snc})\) along with the integral contributions and error estimates over the sub-intervals.

Specifically, \( i = 1, 2, \ldots, n \), 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, \( \frac{b^k}{a} g(x) w(x) dx \approx r_i \) 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, result (and abserr) are taken to be the values returned from the extrapolation process. The value of \( n \) is returned in \( \text{qp} \rightarrow \text{num_subint} \), and the values \( a_i, b_i, r_i \) and \( e_i \) are stored in the structure \( \text{qp} \) as

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

10 Example

This example computes

\[
\int_0^1 \ln x \sin(10\pi x) dx.
\]

10.1 Program Text

/* \text{nag_1d_quad_wt_trig_1} (\text{d01snc}) Example Program. */
/* Copyright 2014 Numerical Algorithms Group. */
/* Mark 5, 1998. */
/* Mark 6 revised, 2000. */
/* Mark 7 revised, 2001. */
/* */
#include <nag.h>
#include <stdio.h>
#include <nag_stdlib.h>
#include <math.h>
#include <nagd01.h>
#include <nagx01.h>

#define __cplusplus
extern "C" {
#define __cplusplus

static double NAG_CALL g(double x, Nag_User *comm);
#endif
###endif

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

  ...
INIT_FAIL(fail);

printf("nag_1d_quad_wt_trig_1 (d01snc) Example Program Results\n");

/* For communication with user-supplied functions: */
comm.p = (Pointer)&use_comm;
epsrel = 0.0001;
epsabs = 0.0;
a = 0.0;
b = 1.0;
/* nag_pi (x01aac).
*pi
*/
omega = nag_pi * 10.0;
wt_func = Nag_Sine;
max_num_subint = 200;
/* nag_1d_quad_wt_trig_1 (d01snc).
* One-dimensional adaptive quadrature, finite interval,
* sine or cosine weight functions, thread-safe
*/
nag_1d_quad_wt_trig_1(g, a, b, omega, wt_func, 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
", epsrel);
if (fail.code != NE_NOERROR)
printf("Error from nag_1d_quad_wt_trig_1 (d01snc) %s\n", fail.message);
if (fail.code != NE_INT_ARG_LT && fail.code != NE_BAD_PARAM &&
fail.code != NE_ALLOC_FAIL && fail.code != NE_NO_LICENCE)
{
printf("result - approximation to the integral = %9.5f\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)
{
Integer *use_comm = (Integer *)comm->p;
if (use_comm[0])
{
printf("(User-supplied callback g, first invocation.)\n");
use_comm[0] = 0;
}
return (x > 0.0)?log(x):0.0;
}
10.2 Program Data

None.

10.3 Program Results

nag_1d_quad_wt_trig_1 (d01snc) Example Program Results
(User-supplied callback g, first invocation.)

\( \text{a} \) - lower limit of integration = 0.0000
\( \text{b} \) - upper limit of integration = 1.0000
\( \text{epsabs} \) - absolute accuracy requested = 0.00e+00
\( \text{epsrel} \) - 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