1 Purpose

nag_1d_quad_wt_trig_1 (d01snc) 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 *absewr, Nag_QuadProgress *qp,
    Nag_User *comm, 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 \( l \leq 20 \). 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 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 *

\textit{On entry/on exit:} pointer to a structure of type \texttt{Nag_User} with the following member:

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

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

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

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

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

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

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

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

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

\textit{On entry:} the parameter $\omega$ in the weight function of the transform.

5: \texttt{wt_func} – Nag_TrigTransform \hspace{1cm} \textit{Input}

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

\begin{itemize}
  \item if \texttt{wt_func} = \texttt{Nag_Cosine}, $w(x) = \cos(\omega x)$;
  \item if \texttt{wt_func} = \texttt{Nag_Sine}, $w(x) = \sin(\omega x)$.
\end{itemize}

\textit{Constraint:} \texttt{wt_func} = \texttt{Nag_Cosine} or \texttt{Nag_Sine}.

6: \texttt{epsabs} – double \hspace{1cm} \textit{Input}

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

7: \texttt{epsrel} – double \hspace{1cm} \textit{Input}

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

8: \texttt{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 \texttt{max_num_subint} should be.

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

\textit{Constraint:} \texttt{max_num_subint} $\geq$ 1.

9: \texttt{result} – double * \hspace{1cm} \textit{Output}

\textit{On exit:} the approximation to the integral $I$.

10: \texttt{abserr} – double * \hspace{1cm} \textit{Output}

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

11: 

qp – Nag_QuadProgress *

Pointer to structure of type Nag_QuadProgress 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_ld_quad_wt_trig_1.

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_INTEGRATIONunky 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_ld_quad_wt_trig_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.

12: 

comm – Nag_User *

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

p – Pointer

Input/Output

On entry/on exit: the pointer p, of type Pointer, allows the user to communicate information to and from the user-defined function g(). 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 *.

13: 

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 \texttt{epsabs} and \texttt{epsrel}, or increasing the value of
\texttt{max_num_subint}.

**NE QUAD ROUNDOFF TOL**
Round-off error prevents the requested tolerance from being achieved: \texttt{epsabs} = \textlt{value}\textgt,\
\texttt{epsrel} = \textlt{value}\textgt.
The error may be underestimated. Consider relaxing the accuracy requirements specified by \texttt{epsabs}
and \texttt{epsrel}.

**NE QUAD BAD_SUBDIV**
Extremely bad integrand behaviour occurs around the sub-interval \textlt{value}\textgt, \textlt{value}\textgt.
The same advice applies as in the case of \texttt{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 \texttt{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 \texttt{NE_INT_ARG_LT},
\texttt{NE_BAD_PARAM} or \texttt{NE_ALLOC_FAIL}.

6  Further Comments
The time taken by \texttt{tnag_1d_quad_wt_trig_1} depends on the integrand and the accuracy required.
If the function fails with an error exit other than \texttt{NE_INT_ARG_LT}, \texttt{NE_BAD_PARAM} or
\texttt{NE_ALLOC_FAIL}, then the user may wish to examine the contents of the structure \texttt{qp}. These contain
the end-points of the sub-intervals used by \texttt{tnag_1d_quad_wt_trig_1} along with the integral contributions
and error estimates over the sub-intervals.
Specifically, for \texttt{i} = 1, 2, \ldots, \texttt{n}, let \texttt{r_i} denote the approximation to the value of the integral over the sub-
interval \texttt{[a_i, b_i]} in the partition of \texttt{[a, b]} and \texttt{e_i} be the corresponding absolute error estimate.

Then, \[
\int_{a}^{b} g(x)w(x) \, dx \approx r_i
\]
and \texttt{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 \textit{et al.} (1983)). In this case, \texttt{result} (and \texttt{abserr})
are taken to be the values returned from the extrapolation process. The value of \texttt{n} is returned in \texttt{num_subint},
and the values \texttt{a_i}, \texttt{b_i}, \texttt{r_i} and \texttt{e_i} are stored in the structure \texttt{qp} as
\[
\begin{align*}
\texttt{a_i} & = \texttt{sub_int_beg_pts[i-1]}, \\
\texttt{b_i} & = \texttt{sub_int_end_pts[i-1]}, \\
\texttt{r_i} & = \texttt{sub_int_result[i-1]} \text{ and} \\
\texttt{e_i} & = \texttt{sub_int_error[i-1]};
\end{align*}
\]

6.1  Accuracy
The function cannot guarantee, but in practice usually achieves, the following accuracy:

\[
|I - \texttt{result}| \leq \texttt{tol}
\]
where
\[
\texttt{tol} = \max\{|\texttt{epsabs}|, |\texttt{epsrel}| \times |I|\}
\]
and \texttt{epsabs} and \texttt{epsrel} are user-specified absolute and relative error tolerances. Moreover it returns the
quantity abserr which, in normal circumstances, satisfies

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

### 6.2 References


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

### 7 See Also

nag_1d_quad_gen_1 (d0lsjc)

### 8 Example

To compute

$$\int_0^1 \ln x \sin(10\pi x) \, dx.$$

#### 8.1 Program Text

```c
/* nag_1d_quad_wt_trig_1(d0lsnc) Example Program */
/* Copyright 1998 Numerical Algorithms Group. */
/* Mark 5, 1998. */
/* Mark 6 revised, 2000. */
*/
#include <nag.h>
#include <stdio.h>
#include <nag_stdlib.h>
#include <math.h>
#include <nagd01.h>
#include <nagx01.h>

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

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;
    Nag_User comm;
```
Vprintf("d01snc 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;
d01snc(g, a, b, omega, wt_func, 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\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 = %4lld\n",
        qp.fun_count);
    Vprintf("qp.num_subint - number of subintervals used = %4lld\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, Nag_User *comm)
{
    return (x>0.0) ? log(x) : 0.0;
}

8.2 Program Data
None.

8.3 Program Results
d01snc 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