

## NAG Library Function Document

### nag\_1d\_quad\_wt\_trig\_1 (d01snc)

#### 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

```
#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 (d01snc) 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

- Malcolm M A and Simpson R B (1976) Local versus global strategies for adaptive quadrature *ACM Trans. Math. Software* **1** 129–146
- Piessens R and Branders M (1975) Algorithm 002: computation of oscillating integrals *J. Comput. Appl. Math.* **1** 153–164
- Piessens R, de Doncker–Kapenga E, Überhuber C and Kahaner D (1983) *QUADPACK, A Subroutine Package for Automatic Integration* Springer–Verlag
- 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 *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: **omega** – double *Input*

*On entry:* the argument  $\omega$  in the weight function of the transform.

5: **wt\_func** – Nag\_TrigTransform *Input*

*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 *Input*

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

7: **epsrel** – double *Input*

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

8: **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**  $\geq 1$ .

- 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  $|I - \mathbf{result}|$ .
- 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 \* *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\_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** =  $\langle value \rangle$ .

**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. 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\_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: **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\_trig\_1 (d01snc) cannot guarantee, but in practice usually achieves, the following accuracy:

$$|I - \mathbf{result}| \leq tol$$

where

$$tol = \max\{|\mathbf{epsabs}|, |\mathbf{epsrel}| \times |I|\}$$

and **epsabs** and **epsrel** are user-specified absolute and relative error tolerances. Moreover it returns the quantity **abserr** which, in normal circumstances, satisfies

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

**8 Parallelism and Performance**

Not applicable.

**9 Further Comments**

The time taken by 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 `qp`. These contain the end-points of the sub-intervals used by `nag_ld_quad_wt_trig_1` (d01snc) along with the integral contributions and error estimates over the sub-intervals.

Specifically,  $i = 1, 2, \dots, n$ , 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** =  $\sum_{i=1}^n r_i$  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 `qp`→`num_subint`, and the values  $a_i$ ,  $b_i$ ,  $r_i$  and  $e_i$  are stored in the structure `qp` as

```

ai = qp→sub_int_beg_pts[i - 1],
bi = qp→sub_int_end_pts[i - 1],
ri = qp→sub_int_result[i - 1] and
ei = qp→sub_int_error[i - 1].

```

## 10 Example

This example computes

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

### 10.1 Program Text

```

/* nag_ld_quad_wt_trig_1 (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>

#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;
    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_ld_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_ld_quad_wt_trig_1 (d01snc).
 * One-dimensional adaptive quadrature, finite interval,
 * sine or cosine weight functions, thread-safe
 */
nag_ld_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\n", epsrel);
if (fail.code != NE_NOERROR)
    printf("Error from nag_ld_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_ld_quad_wt_trig_1 (d01snc) Example Program Results
(User-supplied callback g, first invocation.)
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
```

---