NAG Library Function Document
nag_1d_quad_wt_alglog_1 (d01spc)

1 Purpose
nag_1d_quad_wt_alglog_1 (d01spc) is an adaptive integrator which calculates an approximation to the integral of a function $g(x)w(x)$ over a finite interval $[a, b]$:

$$I = \int_a^b g(x)w(x)\,dx$$

where the weight function $w$ has end-point singularities of algebraico-logarithmic type.

2 Specification

```c
#include <nag.h>
#include <nagd01.h>
void nag_1d_quad_wt_alglog_1 (double (*g)(double x, Nag_User *comm),
   double a, double b, double alfa, double beta, Nag_QuadWeight 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_alglog_1 (d01spc) is based upon the QUADPACK routine QAWSE (Piessens et al. (1983)) and integrates a function of the form $g(x)w(x)$, where the weight function $w(x)$ may have algebraico-logarithmic singularities at the end-points $a$ and/or $b$. The strategy is a modification of that in nag_1d_quad_osc_1 (d01skc). We start by bisecting the original interval and applying modified Clenshaw–Curtis integration of orders 12 and 24 to both halves. Clenshaw–Curtis integration is then used on all sub-intervals which have $a$ or $b$ as one of their end-points (Piessens et al. (1974)). On the other sub-intervals Gauss–Kronrod (7–15 point) integration is carried out.

A ‘global’ acceptance criterion (as defined by Malcolm and Simpson (1976)) is used. The local error estimation control is described by Piessens et al. (1983).

4 References
Piessens R, Mertens I and Branders M (1974) Integration of functions having end-point singularities Angew. Inf. 16 65–68

5 Arguments

1: g – function, supplied by the user

   g 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: \( \text{comm} \) – Nag_User *

Point to a structure of type Nag_User with the following member:

\( p \) – Pointer

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

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

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

2: \( a \) – double

*Input*

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

3: \( b \) – double

*Input*

\( On \ entry: \) the upper limit of integration, \( b \).

*Constraint:* \( b > a \).

4: \( \alpha \) – double

*Input*

\( On \ entry: \) the argument \( \alpha \) in the weight function.

*Constraint:* \( \alpha > 0 \).

5: \( \beta \) – double

*Input*

\( On \ entry: \) the argument \( \beta \) in the weight function.

*Constraint:* \( \beta > 0 \).

6: \( \text{wt_func} \) – Nag_QuadWeight

*Input*

\( On \ entry: \) indicates which weight function is to be used:

- if \( \text{wt_func} = \) Nag_Alg, \( w(x) = (x - a)^\alpha (b - x)^\beta \);
- if \( \text{wt_func} = \) Nag_Alg_loga, \( w(x) = (x - a)^\alpha (b - x)^\beta \ln(x - a) \);
- if \( \text{wt_func} = \) Nag_Alg_logb, \( w(x) = (x - a)^\alpha (b - x)^\beta \ln(b - x) \);
- if \( \text{wt_func} = \) Nag_Alg_loga_logb, \( w(x) = (x - a)^\alpha (b - x)^\beta \ln(x - a) \ln(b - x) \).

*Constraint:* \( \text{wt_func} = \) Nag_Alg, Nag_Alg_loga, Nag_Alg_logb or Nag_Alg_loga_logb.

7: \( \text{epsabs} \) – double

*Input*

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

8: \( \text{epsrel} \) – double

*Input*

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

9: \( \text{max_num_subint} \) – 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 \( \text{max_num_subint} \) should be.

*Constraint:* \( \text{max_num_subint} \geq 2 \).
10: `result` – double * 
   Output 
   *On exit*: the approximation to the integral \( I \).

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

12: `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_alglog_1 (d01spc).
   
   - `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, NE_REAL_ARG_LE, NE_2_REAL_ARG_LE 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_alglog_1 (d01spc) 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`.

13: `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 *.

14: `fail` – NagError * 
   Input/Output 
   The NAG error argument (see Section 3.6 in the Essential Introduction).

### 6 Error Indicators and Warnings

**NE_2_REAL_ARG_LE**

On entry, \( b = \langle \text{value} \rangle \) while \( a = \langle \text{value} \rangle \). These arguments must satisfy \( b > a \).

**NE_ALLOC_FAIL**

Dynamic memory allocation failed.

**NE_BAD_PARAM**

On entry, argument `wt_func` had an illegal value.
On entry, `max_num_subint` must not be less than 2: `max_num_subint = \langle value\rangle`.

Extremely bad integrand behaviour occurs around the sub-interval \( \langle value\rangle,\langle value\rangle \). The same advice applies as in the case of \texttt{NE_QUAD_MAX_SUBDIV}.

The maximum number of subdivisions has been reached: `max_num_subint = \langle value\rangle`.

Round-off error prevents the requested tolerance from being achieved: `epsabs = \langle value\rangle`, `epsrel = \langle value\rangle`.

On entry, `alpha = \langle value\rangle`.
Constraint: `alpha \leq -1.0`.

On entry, `beta = \langle value\rangle`.
Constraint: `beta \leq -1.0`.

nag\_1d\_quad\_wt\_alglog\_1 (d01spc) 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 `abserr` which, in normal circumstances, satisfies

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

Not applicable.

The time taken by nag\_1d\_quad\_wt\_alglog\_1 (d01spc) 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}, \texttt{NE_REAL_ARG_LE}, \texttt{NE_2_REAL_ARG_LE} or \texttt{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\_1d\_quad\_wt\_alglog\_1 (d01spc) along with the integral contributions and error estimates over these 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, $\int_{a_i}^{b_i} g(x)w(x)dx \approx r_i$ and result $= \sum_{i=1}^{n} r_i$.

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

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

and

$$\int_0^1 \sin(10x) \sqrt{x(1-x)} dx.$$

10.1 Program Text

/* nag_1d_quad_wt_alglog_1 (d01spc) Example Program. */
* Copyright 2014 Numerical Algorithms Group.
* 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 f_sin(double x, Nag_User *comm);
static double NAG_CALL f_cos(double x, Nag_User *comm);

#ifdef __cplusplus
}
#endif

int main(void)
{
    /* Scalars */
    Integer exit_status = 0;
    Integer max_num_subint, wt_array_ind;
    int numfunc;
    double a, b, epsabs, abserr, epsrel, result;
    /* Arrays */
    static Integer use_comm[2] = {1, 1};
    static double alpha[2] = { 0.0, -0.5 };
    static double beta[2] = { 0.0, -0.5 };
    static const char *Nag_QuadWeight_array[] = { "Nag_Alg", "Nag_Alg_loga", ...
"Nag_Alg_logb", "Nag_Alg_loga_logb" );

/* Nag Types */
Nag_QuadProgress qp;
Nag_QuadWeight wt_func;
Nag_User comm;
NagError fail;

INIT_FAIL(fail);
printf("nag_1d_quad_wt_alglog_1 (d01spc) Example Program Results\n");

/* For communication with user-supplied functions: */
comm.p = (Pointer)&use_comm;

epsabs = 0.0;
epsrel = 0.0001;
a = 0.0;
b = 1.0;
max_num_subint = 200;
for (numfunc = 0; numfunc < 2; ++numfunc)
{
  switch (numfunc)
  {
  default:  case 0:  wt_func = Nag_Alg_loga;
            wt_array_ind = 1;
            /* nag_1d_quad_wt_alglog_1 (d01spc).
               * One-dimensional adaptive quadrature, weight function with
               * end-point singularities of algebraic-logarithmic type,
               * thread-safe
               */
            nag_1d_quad_wt_alglog_1(f_cos, a, b, alpha[numfunc], beta[numfunc],
                               wt_func, epsabs, epsrel, max_num_subint,
                               &result, &abserr, &qp, &comm, &fail);
            printf("Integral of cos(10*pi*x) on [a,b]\n");
            break;
  case 1:  wt_func = Nag_Alg;
            wt_array_ind = 0;
            nag_1d_quad_wt_alglog_1(f_sin, a, b, alpha[numfunc], beta[numfunc],
                               wt_func, epsabs, epsrel, max_num_subint,
                               &result, &abserr, &qp, &comm, &fail);
            printf("Integral of sin(10*x) on [a,b]\n");
          }
  printf("------------\n");
  printf("a - lower limit of integration = %9.4f\n", a);
  printf("b - upper limit of integration = %9.4f\n", b);
  printf("epsabs - absolute accuracy requested = %11.2e\n", epsabs);
  printf("epsrel - relative accuracy requested = %11.2e\n", epsrel);
  printf("alpha - weight function parameter = %9.4f\n",
          alpha[numfunc]);
  printf("beta - weight function parameter = %9.4f\n",
         beta[numfunc]);
  printf("wt_func - weight function used = %s\n",
         Nag_QuadWeight_array[wt_array_ind]);
  if (fail.code != NE_NOERROR) printf("%s\n", fail.message);

  if (fail.code == NE_NOERROR || fail.code == NE_QUAD_BAD_SUBDIV ||
      fail.code == NE_QUAD_MAX_SUBDIV || fail.code == NE_QUAD_ROUNDOFF_TOL)
  {
    printf("result - approximation to the integral = %10.5f\n",
           result);
    printf("abserr - estimate of absolute error = %11.2e\n",
           abserr);
    printf("qp.fun_count - function evaluations = %4"NAG_IFMT"\n",
           qp.fun_count);
    printf("qp.num_subint - 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 f_cos(double x, Nag_User *comm)
{
    double a;
    double pi;
    Integer *use_comm = (Integer *)comm->p;
    
    if (use_comm[0])
    {
        printf("(User-supplied callback f_cos, first invocation.)\n");
        use_comm[0] = 0;
    }

    /* nag_pi (x01aac). */
    pi = nag_pi;
    a = pi*10.0;
    return cos(a*x);
}

static double NAG_CALL f_sin(double x, Nag_User *comm)
{
    double omega;
    Integer *use_comm = (Integer *)comm->p;
    
    if (use_comm[1])
    {
        printf("(User-supplied callback f_sin, first invocation.)\n");
        use_comm[1] = 0;
    }

    omega = 10.0;
    return sin(omega*x);
}

10.2 Program Data
None.

10.3 Program Results
nag_1d_quad_wt_alglog_1 (d01spc) Example Program Results
(User-supplied callback f_cos, first invocation.)

<table>
<thead>
<tr>
<th>Integral of cos(10<em>pi</em>x) on [a,b]</th>
</tr>
</thead>
<tbody>
<tr>
<td>a - lower limit of integration</td>
</tr>
<tr>
<td>b - upper limit of integration</td>
</tr>
<tr>
<td>epsabs - absolute accuracy requested</td>
</tr>
<tr>
<td>epsrel - relative accuracy requested</td>
</tr>
<tr>
<td>alpha - weight function parameter</td>
</tr>
<tr>
<td>beta - weight function parameter</td>
</tr>
<tr>
<td>wt_func - weight function used</td>
</tr>
<tr>
<td>result - approximation to the integral</td>
</tr>
</tbody>
</table>
Integral of \( \sin(10x) \) on \([a,b]\)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a) - lower limit of integration</td>
<td>0.0000</td>
</tr>
<tr>
<td>(b) - upper limit of integration</td>
<td>1.0000</td>
</tr>
<tr>
<td>(\text{epsabs}) - absolute accuracy requested</td>
<td>0.00e+00</td>
</tr>
<tr>
<td>(\text{epsrel}) - relative accuracy requested</td>
<td>1.00e-04</td>
</tr>
<tr>
<td>(\alpha) - weight function parameter</td>
<td>-0.5000</td>
</tr>
<tr>
<td>(\beta) - weight function parameter</td>
<td>-0.5000</td>
</tr>
<tr>
<td>(\text{wt_func}) - weight function used</td>
<td>(\text{Nag_Alg})</td>
</tr>
<tr>
<td>(\text{result}) - approximation to the integral</td>
<td>0.53502</td>
</tr>
<tr>
<td>(\text{abserr}) - estimate of absolute error</td>
<td>1.94e-12</td>
</tr>
<tr>
<td>(\text{qp.fun_count}) - function evaluations</td>
<td>50</td>
</tr>
<tr>
<td>(\text{qp.num_subint}) - subintervals used</td>
<td>2</td>
</tr>
</tbody>
</table>