

NAG C 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

```
#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

This function 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 Parameters

1: **g** – function supplied by user *Function*

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, Nag_User *comm)
```

1: **x** – double *Input*

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

- | | | |
|----|---|--|
| 2: | <p>comm – Nag_User *</p> <p><i>On entry/on exit:</i> pointer to a structure of type Nag_User with the following member:</p> <p style="text-align: center;">p – Pointer <i>Input/Output</i></p> <p><i>On entry/on exit:</i> the pointer comm→p should be cast to the required type, e.g.,
 <code>struct user *s = (struct user *)comm->p</code>, to obtain the original object's address with appropriate type. (See the argument comm below.)</p> | |
|----|---|--|
-
- | | | |
|----|---|--------------|
| 2: | <p>a – double</p> <p><i>On entry:</i> the lower limit of integration, <i>a</i>.</p> | <i>Input</i> |
| 3: | <p>b – double</p> <p><i>On entry:</i> the upper limit of integration, <i>b</i>.</p> <p><i>Constraint:</i> b > a.</p> | <i>Input</i> |
| 4: | <p>alfa – double</p> <p><i>On entry:</i> the parameter α in the weight function.</p> <p><i>Constraint:</i> alfa > -1.0.</p> | <i>Input</i> |
| 5: | <p>beta – double</p> <p><i>On entry:</i> the parameter β in the weight function.</p> <p><i>Constraint:</i> beta > -1.0.</p> | <i>Input</i> |
| 6: | <p>wt_func – Nag_QuadWeight</p> <p><i>On entry:</i> indicates which weight function is to be used:</p> <p style="padding-left: 20px;">if wt_func = Nag_Alg, $w(x) = (x - a)^\alpha (b - x)^\beta$;</p> <p style="padding-left: 20px;">if wt_func = Nag_Alg_loga, $w(x) = (x - a)^\alpha (b - x)^\beta \ln(x - a)$;</p> <p style="padding-left: 20px;">if wt_func = Nag_Alg_logb, $w(x) = (x - a)^\alpha (b - x)^\beta \ln(b - x)$;</p> <p style="padding-left: 20px;">if wt_func = Nag_Alg_loga_logb, $w(x) = (x - a)^\alpha (b - x)^\beta \ln(x - a) \ln(b - x)$.</p> <p><i>Constraint:</i> wt_func = Nag_Alg, Nag_Alg_loga, Nag_Alg_logb, or Nag_Alg_loga_logb.</p> | <i>Input</i> |
| 7: | <p>epsabs – double</p> <p><i>On entry:</i> the absolute accuracy required. If epsabs is negative, the absolute value is used. See Section 6.1.</p> | <i>Input</i> |
| 8: | <p>epsrel – double</p> <p><i>On entry:</i> the relative accuracy required. If epsrel is negative, the absolute value is used. See Section 6.1.</p> | <i>Input</i> |
| 9: | <p>max_num_subint – Integer</p> <p><i>On entry:</i> 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.</p> <p><i>Suggested values:</i> a value in the range 200 to 500 is adequate for most problems.</p> <p><i>Constraint:</i> max_num_subint \geq 2.</p> | <i>Input</i> |

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 $|I - \mathbf{result}|$.

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.

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 6.

Before a subsequent call to nag_1d_quad_wt_alglog_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**.

13: **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 *`.

14: **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 2: **max_num_subint** = *<value>*.

NE_BAD_PARAM

On entry, parameter **wt_func** had an illegal value.

NE_REAL_ARG_LE

On entry, **alfa** must not be less than or equal to -1.0 : **alfa** = $\langle value \rangle$.

On entry, **beta** must not be less than or equal to -1.0 : **beta** = $\langle value \rangle$.

NE_2_REAL_ARG_LE

On entry, **b** = $\langle value \rangle$ while **a** = $\langle value \rangle$. These parameters must satisfy **b** > **a**.

NE_ALLOC_FAIL

Memory allocation failed.

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 discontinuity or a singularity of algebraico-logarithmic type within the interval can be determined, the interval must be split up at this point and the integrator called 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** = $\langle value \rangle$, **epsrel** = $\langle value \rangle$.

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 ($\langle value \rangle$, $\langle value \rangle$).

The same advice applies as in the case of **NE_QUAD_MAX_SUBDIV**.

6 Further Comments

The time taken by `nag_1d_quad_wt_alglog_1` 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**, **NE_REAL_ARG_LE**, **NE_2_REAL_ARG_LE** 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_alglog_1` along with the integral contributions and error estimates over these sub-intervals.

Specifically, for $i = 1, 2, \dots, n$, 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** = $\sum_{i=1}^n r_i$.

The value of n is returned in **num_subint**, and the values a_i , b_i , r_i and e_i are stored in the structure **qp** as

$$a_i = \text{sub_int_beg_pts}[i - 1],$$

$$b_i = \text{sub_int_end_pts}[i - 1],$$

$$r_i = \text{sub_int_result}[i - 1] \text{ and}$$

$$e_i = \text{sub_int_error}[i - 1].$$

6.1 Accuracy

The function 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

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

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

Piessens R, De Doncker-Kapenga E, Überhuber C and Kahaner D (1983) *QUADPACK, A Subroutine Package for Automatic Integration* Springer-Verlag

7 See Also

nag_1d_quad_gen_1 (d01sjc)

8 Example

To compute

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

and

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

8.1 Program Text

```

/* nag_1d_quad_wt_alglog_1(d01spc) 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 f_sin(double x, Nag_User *comm);
static double f_cos(double x, Nag_User *comm);

```

```

main()
{
    static double alfa[2] = {0.0, -0.5};
    static double beta[2] = {0.0, -0.5};
    Nag_QuadWeight wt_func;
    double a, b;
    double epsabs, abserr, epsrel, result;
    static NagError fail;
    Nag_QuadProgress qp;
    Integer max_num_subint;
    int numfunc;
    NAG_D01SPC_FUN g;
    static char *Nag_QuadWeight_array[] =
    { "Nag_Alg", "Nag_Alg_loga", "Nag_Alg_logb", "Nag_Alg_loga_logb"};
    Boolean success = TRUE;
    Nag_User comm;
    Integer wt_array_ind;

    Vprintf("d01spc Example Program Results\n");
    epsabs = 0.0;
    epsrel = 0.0001;
    a = 0.0;
    b = 1.0;
    max_num_subint = 200;
    for (numfunc=0; numfunc < 2; ++numfunc)
    {
        switch (numfunc)
        {
            case 0:
                g = f_cos;
                wt_func = Nag_Alg_loga;
                wt_array_ind = 1;
                break;
            case 1:
                g = f_sin;
                wt_func = Nag_Alg;
                wt_array_ind = 0;
        }

        d01spc(g, a, b, alfa[numfunc], beta[numfunc],
            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);
        Vprintf("alfa   - parameter in the weight function = %10.4f\n",
            alfa[numfunc]);
        Vprintf("beta   - parameter in the weight function = %10.4f\n",
            beta[numfunc]);
        Vprintf("wt_func - denotes weight function to be \
used = %s\n", Nag_QuadWeight_array[wt_array_ind]);
        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_REAL_ARG_LE && fail.code != NE_2_REAL_ARG_LE &&
            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\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
        success = FALSE;
}
if (success)
    exit(EXIT_SUCCESS);
else
    exit(EXIT_FAILURE);
}

static double f_cos(double x, Nag_User *comm)
{
    double a;
    double pi;

    pi = X01AAC;
    a = pi*10.0;
    return cos(a*x);
}

static double f_sin(double x, Nag_User *comm)
{
    double omega;

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

```

8.2 Program Data

None.

8.3 Program Results

d01spc 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

alfa   - parameter in the weight function =      0.0000
beta   - parameter in the weight function =      0.0000
wt_func - denotes weight function to be used = Nag_Alg_loga
result - approximation to the integral = -0.04899
abserr - estimate of the absolute error = 1.14e-07
qp.fun_count - number of function evaluations = 110

```

```
qp.num_subint - number of subintervals used = 4

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

alfa - parameter in the weight function = -0.5000
beta - parameter in the weight function = -0.5000
wt_func - denotes weight function to be used = Nag_Alq
result - approximation to the integral = 0.53502
abserr - estimate of the absolute error = 1.94e-12
qp.fun_count - number of function evaluations = 50
qp.num_subint - number of subintervals used = 2
```
