

## NAG C Library Function Document

### nag\_1d\_quad\_inf\_1 (d01smc)

#### 1 Purpose

nag\_1d\_quad\_inf\_1 (d01smc) calculates an approximation to the integral of a function  $f(x)$  over an infinite or semi-infinite interval  $[a, b]$ :

$$I = \int_a^b f(x) dx.$$

#### 2 Specification

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

void nag_1d_quad_inf_1 (double (*f)(double x, Nag_User *comm),
    Nag_BoundInterval boundinf, double bound, 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 on the QUADPACK routine QAGI (Piessens *et al.* (1983)). The entire infinite integration range is first transformed to  $[0, 1]$  using one of the identities

$$\int_{-\infty}^a f(x) dx = \int_0^1 f\left(a - \frac{1-t}{t}\right) \frac{1}{t^2} dt$$

$$\int_a^{\infty} f(x) dx = \int_0^1 f\left(a + \frac{1-t}{t}\right) \frac{1}{t^2} dt$$

$$\int_{-\infty}^{\infty} f(x) dx = \int_0^{\infty} (f(x) + f(-x)) dx = \int_0^1 \left[ f\left(\frac{1-t}{t}\right) + f\left(\frac{-1+t}{t}\right) \right] \frac{1}{t^2} dt$$

where  $a$  represents a finite integration limit. An adaptive procedure, based on the Gauss 7-point and Kronrod 15-point rules, is then employed on the transformed integral. The algorithm, described by De Doncker (1978), incorporates a global acceptance criterion (as defined by Malcolm and Simpson (1976)) together with the  $\epsilon$ -algorithm (Wynn (1956)) to perform extrapolation. The local error estimation is described by Piessens *et al.* (1983).

#### 4 Parameters

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

The function **f**, supplied by the user, must return the value of the integrand  $f$  at a given point.

The specification of **f** is:

```
double f(double x, Nag_User *comm)
```

1: **x** – double *Input*

*On entry:* the point at which the integrand  $f$  must be evaluated.

- |    |   |  |
|----|---|--|
| 2: | <p><b>comm</b> – Nag_User *</p> <p><i>On entry/on exit:</i> pointer to a structure of type <b>Nag_User</b> with the following member:</p> <p style="text-align: center;"><b>p</b> – Pointer <span style="float: right;"><i>Input/Output</i></span></p> <p><i>On entry/on exit:</i> the pointer <b>comm</b>→<b>p</b> should be cast to the required type, e.g.,<br/> <code>struct user *s = (struct user *)comm-&gt;p</code>, to obtain the original object's address with appropriate type. (See the argument <b>comm</b> below.)</p> |  |
|----|---|--|
- 2: **boundinf** – Nag\_BoundInterval *Input*
- On entry:* indicates the kind of integration interval:
- if **boundinf** = **Nag\_UpperSemiInfinite**, the interval is [**bound**,  $+\infty$ );
  - if **boundinf** = **Nag\_LowerSemiInfinite**, the interval is  $(-\infty, \mathbf{bound}]$ ;
  - if **boundinf** = **Nag\_Infinite**, the interval is  $(-\infty, +\infty)$ .
- Constraint:* **boundinf** = **Nag\_UpperSemiInfinite**, **Nag\_LowerSemiInfinite**, or **Nag\_Infinite**.
- 3: **bound** – double *Input*
- On entry:* the finite limit of the integration interval (if present). **bound** is not used if **boundinf** = **Nag\_Infinite**.
- 4: **epsabs** – double *Input*
- On entry:* the absolute accuracy required. If **epsabs** is negative, the absolute value is used. See Section 6.1.
- 5: **epsrel** – double *Input*
- On entry:* the relative accuracy required. If **epsrel** is negative, the absolute value is used. See Section 6.1.
- 6: **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.
- Suggested values:* a value in the range 200 to 500 is adequate for most problems.
- Constraint:* **max\_num\_subint**  $\geq 1$ .
- 7: **result** – double \* *Output*
- On exit:* the approximation to the integral  $I$ .
- 8: **abserr** – double \* *Output*
- On exit:* an estimate of the modulus of the absolute error, which should be an upper bound for  $|I - \mathbf{result}|$ .
- 9: **qp** – Nag\_QuadProgress \* *Output*
- 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\_inf\_1.

<b>sub_int_beg_pts</b> – double *	<i>Output</i>
<b>sub_int_end_pts</b> – double *	<i>Output</i>
<b>sub_int_result</b> – double *	<i>Output</i>
<b>sub_int_error</b> – double *	<i>Output</i>

*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 6.

Before a subsequent call to `nag_1d_quad_inf_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**.

10: **comm** – Nag\_User \*

*On entry/on exit:* pointer to a structure of type **Nag\_User** with the following member:

<b>p</b> – Pointer	<i>Input/Output</i>
--------------------	---------------------

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

11: **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 **boundinf** 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 **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** = *<value>*, **epsrel** = *<value>*.

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 (*<value>*, *<value>*).  
The same advice applies as in the case of **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 **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 **NE\_INT\_ARG\_LT**, **NE\_BAD\_PARAM** or **NE\_ALLOC\_FAIL**.

**NE\_QUAD\_BAD\_SUBDIV\_INTS**

Extremely bad integrand behaviour occurs around one of the sub-intervals (*<value>*, *<value>*) or (*<value>*, *<value>*).  
The same advice applies as in the case of **NE\_QUAD\_MAX\_SUBDIV**.

**6 Further Comments**

The time taken by `nag_1d_quad_inf_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** 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_inf_1` along with the integral contributions and error estimates over the 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,  $\int_{a_i}^{b_i} f(x) dx \simeq r_i$  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 **num\_subint**, and the values  $a_i$ ,  $b_i$ ,  $r_i$  and  $e_i$  are stored in the structure **qp** as

$$\begin{aligned} a_i &= \text{sub\_int\_beg\_pts}[i - 1], \\ b_i &= \text{sub\_int\_end\_pts}[i - 1], \\ r_i &= \text{sub\_int\_result}[i - 1] \text{ and} \\ e_i &= \text{sub\_int\_error}[i - 1]. \end{aligned}$$

**6.1 Accuracy**

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

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

where

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

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

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

## 6.2 References

De Doncker E (1978) An adaptive extrapolation algorithm for automatic integration *ACM SIGNUM Newsl.* **13 (2)** 12–18

Malcolm M A and Simpson R B (1976) Local versus global strategies for adaptive quadrature *ACM Trans. Math. Software* **1** 129–146

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

## 7 See Also

nag\_ld\_quad\_gen\_1 (d01sjc)

## 8 Example

To compute

$$\int_0^{\infty} \frac{1}{(x+1)\sqrt{x}} dx.$$

### 8.1 Program Text

```

/* nag_ld_quad_inf_1(d01smc) 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>

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

main()
{
    double a;
    double epsabs, abserr, epsrel, result;
    Nag_QuadProgress qp;
    Integer max_num_subint;
    static NagError fail;
    Nag_User comm;

    Vprintf("d01smc Example Program Results\n");
    epsabs = 0.0;
    epsrel = 0.0001;
    a = 0.0;
    max_num_subint = 200;

    d01smc(f, Nag_UpperSemiInfinite, a, 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 = infinity\n");
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 = %4ld\n",
        qp.fun_count);
    Vprintf("qp.num_subint - number of subintervals used = %4ld\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 f(double x, Nag_User *comm)
{
    return 1.0/((x+1.0)*sqrt(x));
}

```

## 8.2 Program Data

None.

## 8.3 Program Results

```

d01smc Example Program Results
a      - lower limit of integration =      0.0000
b      - upper limit of integration = infinity
epsabs - absolute accuracy requested =  0.00e+00
epsrel - relative accuracy requested =  1.00e-04

result - approximation to the integral =   3.14159
abserr - estimate of the absolute error =  2.65e-05
qp.fun_count - number of function evaluations =  285
qp.num_subint - number of subintervals used =   10

```

---