NAG 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

```c
#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

nag_1d_quad_inf_1 (d01smc) 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 \( c \)-algorithm (Wynn (1956)) to perform extrapolation. The local error estimation is described by Piessens et al. (1983).

4 References


Wynn P (1956) On a device for computing the \( e_{n} (S_n) \) transformation Math. Tables Aids Comput. 10 91–96
5 Arguments

1: \texttt{f} – function, supplied by the user \hfill \textit{External Function}

\texttt{f} must return the value of the integrand \( f \) at a given point.

The specification of \( f \) is:
\[
\text{double } f \ (\text{double } x, \ 	ext{Nag_User } * \text{comm})
\]

1: \textit{x} – double

\textit{Input}

\textit{On entry}: the point at which the integrand \( f \) must be evaluated.

2: \texttt{comm} – Nag_User *

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

\textit{p} – Pointer

\textit{On entry/exit}: the pointer \( \text{comm} \rightarrow \text{p} \) should be cast to the required type, e.g.,
\[
\text{struct } \text{user } * \text{s} = (\text{struct } \text{user } *) \text{comm} \rightarrow \text{p},
\]

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

2: \texttt{boundinf} – Nag_BoundInterval

\textit{Input}

\textit{On entry}: indicates the kind of integration interval.

\texttt{boundinf} = Nag_UpperSemiInfinite

The interval is \((\text{bound}, +\infty)\).

\texttt{boundinf} = Nag_LowerSemiInfinite

The interval is \((-\infty, \text{bound})\).

\texttt{boundinf} = Nag_Infinite

The interval is \((-\infty, +\infty)\).

\textit{Constraint}: \texttt{boundinf} = Nag_UpperSemiInfinite, Nag_LowerSemiInfinite or Nag_Infinite.

3: \texttt{bound} – double

\textit{Input}

\textit{On entry}: the finite limit of the integration interval (if present). \texttt{bound} is not used if \texttt{boundinf} = Nag_Infinite.

4: \texttt{epsabs} – double

\textit{Input}

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

5: \texttt{epsrel} – double

\textit{Input}

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

6: \texttt{max_num_subint} – Integer

\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{Constraint}: \texttt{max_num_subint} \( \geq 1 \).

7: \texttt{result} – double *

\textit{Output}

\textit{On exit}: the approximation to the integral \( I \).
abserr – double *

Output

On exit: an estimate of the modulus of the absolute error, which should be an upper bound for \(|f - \text{result}|\).

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_inf_1 (d01smc).

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_inf_1 (d01smc) 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.

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 f(). 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 *.

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 boundinf had an illegal value.

**NE_INT_ARG_LT**

On entry, max_num_subint must not be less than 1: max_num_subint = (value).

**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_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.

NE_QUAD_MAX_SUBDIV
The maximum number of subdivisions has been reached: \(\text{max_num_subint} = <value>\).
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 \(\text{epsabs}\) and \(\text{epsrel}\), or increasing the value of \(\text{max_num_subint}\).

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: \(\text{epsabs} = <value>, \text{epsrel} = <value>\).
The error may be underestimated. Consider relaxing the accuracy requirements specified by \(\text{epsabs}\) and \(\text{epsrel}\).

7 Accuracy
\text{nag}_1d_quad_int_1 (d01smc) cannot guarantee, but in practice usually achieves, the following accuracy:
\[|I - \text{result}| \leq tol\]
where
\[tol = \max\{\lvert\text{epsabs}\rvert, \lvert\text{epsrel}\rvert \times |I|\}\]
and \(\text{epsabs}\) and \(\text{epsrel}\) are user-specified absolute and relative error tolerances. Moreover it returns the quantity \text{abserr} which, in normal circumstances, satisfies
\[|I - \text{result}| \leq \text{abserr} \leq tol.\]

8 Parallelism and Performance
Not applicable.

9 Further Comments
The time taken by \text{nag}_1d_quad_int_1 (d01smc) 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_1d_quad_inf_1 (d01smc) along with the integral contributions and error estimates over the 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, \(\int_a^b f(x) dx \simeq \sum_{i=1}^n r_i\) 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 \(qp->\text{num_subint}\), and the values \(a_i\), \(b_i\), \(r_i\) and \(e_i\) are stored in the structure \(qp\) as

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

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

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

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

10 Example

This example computes

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

10.1 Program Text

/* nag_1d_quad_inf_1 (d01smc) 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>

#ifdef __cplusplus
eextern "C" {
#endif
static double NAG_CALL f(double x, Nag_User *comm);
#ifdef __cplusplus
} 
#endif

int main(void)
{
    static Integer use_comm[1] = {1};
    Integer exit_status = 0;
    double a;
    double epsabs, abserr, epsrel, result;
    Nag_QuadProgress qp;
    Integer max_num_subint;
    NagError fail; 
    Nag_User comm;

    INIT_FAIL(fail);

    printf("nag_1d_quad_inf_1 (d01smc) Example Program Results\n");
/* For communication with user-supplied functions: */
comm.p = (Pointer)&use_comm;

epsabs = 0.0;
epsrel = 0.0001;
a = 0.0;
max_num_subint = 200;

/* nag_1d_quad_inf_1 (d01smc).
* One-dimensional adaptive quadrature over infinite or
* semi-infinite interval, thread-safe
*/

nag_1d_quad_inf_1(f, Nag_UpperSemiInfinite, a, 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 = infinity\n");
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_1d_quad_inf_1 (d01smc) %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 f(double x, Nag_User *comm)
{
    Integer *use_comm = (Integer *)comm->p;

    if (use_comm[0])
    {
        printf("(User-supplied callback f, first invocation.\n"");
        use_comm[0] = 0;
    }

    return 1.0/((x+1.0)*sqrt(x));
}

10.2 Program Data
None.
10.3 Program Results

nag_1d_quad_inf_1 (d01smc) Example Program Results
(User-supplied callback f, first invocation.)
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