NAG Library Function Document
nag_1d_quad_osc_1 (d01skc)

1 Purpose
nag_1d_quad_osc_1 (d01skc) is an adaptive integrator, especially suited to oscillating, nonsingular integrands, which calculates an approximation to the integral of a function \( f(x) \) over a finite interval \([a, b]\):
\[
I = \int_a^b f(x) \, dx.
\]

2 Specification
#include <nag.h>
#include <nagd01.h>
void nag_1d_quad_osc_1 (double (*f)(double x, Nag_User *comm), double a, double b, 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_osc_1 (d01skc) is based upon the QUADPACK routine QAG (Piessens et al. (1983)). It is an adaptive function, using the Gauss 30-point and Kronrod 61-point rules. A ‘global’ acceptance criterion (as defined by Malcolm and Simpson (1976)) is used. The local error estimation is described by Piessens et al. (1983).

As this function is based on integration rules of high order, it is especially suitable for nonsingular oscillating integrands.

This function requires you to supply a function to evaluate the integrand at a single point.

4 References
Piessens R (1973) An algorithm for automatic integration Angew. Inf. 15 399–401

5 Arguments
1: \( f \) – function, supplied by the user

\( f \) 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

On entry: the point at which the integrand \( f \) 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: **epsabs** – double  
   *Input*
   
   *On entry:* the absolute accuracy required. If **epsabs** is negative, the absolute value is used. See Section 7.

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

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.

   **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 \(|I - \text{result}|\).

9: **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_osc_1 (d01skc).

   **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 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_osc_1 (d01skc) 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.

10: **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 *.

11: **fail** – NagError *

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_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_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_RONDOFF_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**.

7 Accuracy

nag_1d_quad_osc_1 (d01skc) cannot guarantee, but in practice usually achieves, the following accuracy:

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

where

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

Mark 25
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

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

8 Parallelism and Performance

Not applicable.

9 Further Comments

The time taken by nag_1d_quad_osc_1 (d01skc) depends on the integrand and the accuracy required. If the function fails with an error exit other than \( \text{NE_INT_ARG_LT} \) or \( \text{NE_ALLOC_FAIL} \), then you may wish to examine the contents of the structure \( \text{qp} \). These contain the end-points of the sub-intervals used by nag_1d_quad_osc_1 (d01skc) 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, \( \int_{a_i}^{b_i} f(x) \, dx \approx r_i \) and \( \text{result} = \sum_{i=1}^{n} r_i \). 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

\[
\int_0^{2\pi} \sin(30x) \cos x \, dx.
\]

10.1 Program Text

/* nag_1d_quad_osc_1 (d01skc) 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>

#if defined __cplusplus
extern "C" {
#endif
static double NAG_CALL f(double x, Nag_User *comm);
#endif
}
int main(void) {
    static Integer use_comm[1] = {1};
    Integer exit_status = 0;
    double a, b;
    double epsabs, abserr, epsrel, result;
    Nag_QuadProgress qp;
    Integer max_num_subint;
    NagError fail;
    /* nag_pi (x01aac). */
    *pi
    */
    double pi = nag_pi;
    Nag_User comm;

    INIT_FAIL(fail);
    printf("nag_1d_quad_osc_1 (d01skc) Example Program Results\n");
    /* For communication with user-supplied functions: */
    comm.p = (Pointer)&use_comm;

    epsabs = 0.0;
    epsrel = 0.001;
    a = 0.0;
    b = pi * 2.0;
    max_num_subint = 200;

    /* nag_1d_quad_osc_1 (d01skc).
    * One-dimensional adaptive quadrature, suitable for
    * oscillating functions, thread-safe
    */
    nag_1d_quad_osc_1(f, a, b, 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", epsrel);
    if (fail.code != NE_NOERROR) {
        printf("Error from nag_1d_quad_osc_1 (d01skc) %s\n", fail.message);
        if (fail.code != NE_INT_ARG_LT && 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;
        }
    } 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 x*sin(x*30.0)*cos(x);

10.2 Program Data

None.

10.3 Program Results

nag_1d_quad_osc_1 (d01skc) Example Program Results
(User-supplied callback f, first invocation.)

a - lower limit of integration = 0.0000
b - upper limit of integration = 6.2832
epsabs - absolute accuracy requested = 0.00e+00
epsrel - relative accuracy requested = 1.00e-03

result - approximation to the integral = -0.20967
abserr - estimate of the absolute error = 4.48e-14
qp.fun_count - number of function evaluations = 427
qp.num_subint - number of subintervals used = 4