

## nag\_1d\_quad\_osc (d01akc)

### 1. Purpose

**nag\_1d\_quad\_osc (d01akc)** is an adaptive integrator, especially suited to oscillating, non-singular 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(double (*f)(double x), double a, double b,
                    double epsabs, double epsrel, Integer max_num_subint,
                    double *result, double *abserr, Nag_QuadProgress *qp,
                    NagError *fail)
```

### 3. Description

**nag\_1d\_quad\_osc** 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 non-singular oscillating integrands.

**nag\_1d\_quad\_osc** requires the user to supply a function to evaluate the integrand at a single point.

### 4. Parameters

**f**

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

The specification of **f** is:

<pre>double f(double x)      x</pre>
--------------------------------------

Input: the point at which the integrand  $f$  must be evaluated.

**a**

Input: the lower limit of integration,  $a$ .

**b**

Input: the upper limit of integration,  $b$ . It is not necessary that  $a < b$ .

**epsabs**

Input: the absolute accuracy required. If **epsabs** is negative, the absolute value is used. See Section 6.1.

**epsrel**

Input: the relative accuracy required. If **epsrel** is negative, the absolute value is used. See Section 6.1.

**max\_num\_subint**

Input: 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 value: a value in the range 200 to 500 is adequate for most problems.

Constraint: **max\_num\_subint**  $\geq 1$ .

**result**

Output: the approximation to the integral  $I$ .

**abserr**

Output: an estimate of the modulus of the absolute error, which should be an upper bound for  $|I - \text{result}|$ .

**qp**

Pointer to structure of type Nag\_QuadProgress with the following members:

**num\_subint** – Integer

Output: the actual number of sub-intervals used.

**fun\_count** – Integer

Output: the number of function evaluations performed by the nag\_1d\_quad\_osc.

**sub\_int\_beg\_pts** – double \*

**sub\_int\_end\_pts** – double \*

**sub\_int\_result** – double \*

**sub\_int\_error** – double \*

Output: 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. If nag\_1d\_quad\_osc is to be called repeatedly, then the user should free the storage allocated by these pointers before any subsequent call is made.

**fail**

The NAG error parameter, see the Essential Introduction to the NAG C Library.

Users are recommended to declare and initialise **fail** and set **fail.print** = TRUE for this function.

## 5. Error Indications and Warnings

**NE\_INT\_ARG\_LT**

On entry, **max\_num\_subint** must not be less than 1: **max\_num\_subint** =  $\langle \text{value} \rangle$ .

**NE\_ALLOC\_FAIL**

Memory allocation failed.

**NE\_QUAD\_MAX\_SUBDIV**

The maximum number of subdivisions has been reached: **max\_num\_subint** =  $\langle \text{value} \rangle$ .

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 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** =  $\langle \text{value} \rangle$ , **epsrel** =  $\langle \text{value} \rangle$ .

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 ( $\langle \text{value} \rangle$ ,  $\langle \text{value} \rangle$ ).

The same advice applies as in the case of **NE\_QUAD\_MAX\_SUBDIV**.

## 6. Further Comments

The time taken by the function depends on the integrand and the accuracy required.

If the function fails with an error exit other than **NE\_INT\_ARG\_LT** 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\_osc** 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,  $\int_{a_i}^{b_i} f(x)dx \simeq r_i$  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 = sub_int_beg_pts[i - 1],
b_i = sub_int_end_pts[i - 1],
r_i = sub_int_result[i - 1]  and
e_i = sub_int_error[i - 1].

```

### 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

- Malcolm M A and Simpson R B (1976) Local Versus Global Strategies for Adaptive Quadrature  
*ACM Trans. Math. Softw.* **1** 129–146.
- Piessens R, De Doncker-Kapenga E, Überhuber C and Kahaner D (1983) *QUADPACK, A Subroutine Package for Automatic Integration* Springer-Verlag.
- Piessens R (1973) An Algorithm for Automatic Integration *Angew. Inf.* **15** 399–401.

## 7. See Also

**nag\_1d\_quad\_gen** (d01ajc)  
**nag\_1d\_quad\_brkpts** (d01alc)

## 8. Example

To compute

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

### 8.1. Program Text

```

/* nag_1d_quad_osc(d01akc) Example Program
 *
 * Copyright 1991 Numerical Algorithms Group.
 *
 * Mark 2, 1991.
 */

#include <nag.h>

```

```

#include <stdio.h>
#include <nag_stdlib.h>
#include <math.h>
#include <nagd01.h>
#include <nagx01.h>

#ifdef NAG_PROTO
static double f(double x);
#else
static double f();
#endif

main()
{
    double a, b;
    double epsabs, abserr, epsrel, result;
    Nag_QuadProgress qp;
    Integer max_num_subint;
    static NagError fail;
    double pi = X01AAC;

    Vprintf("d01akc Example Program Results\n");
    epsabs = 0.0;
    epsrel = 0.001;
    a = 0.0;
    b = pi * 2.0;
    max_num_subint = 200;

    d01akc(f, a, b, epsabs, epsrel, max_num_subint, &result, &abserr, &qp,
           &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);
    if (fail.code != NE_NOERROR)
        Vprintf("%s\n", fail.message);
    if (fail.code != NE_INT_ARG_LT)
    {
        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);
        exit(EXIT_SUCCESS);
    }
    else
        exit(EXIT_FAILURE);
}

#ifdef NAG_PROTO
static double f(double x)
#else
static double f(x)
double x;
#endif
{
    return x*sin(x*30.0)*cos(x);
}

```

## 8.2. Program Data

None.

### 8.3. Program Results

```
d01akc Example Program Results
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 =  4.49e-14
qp.fun_count - number of function evaluations = 427
qp.num_subint - number of subintervals used =   4
```

---