

## nag\_multid\_quad\_adapt (d01fcc)

### 1. Purpose

**nag\_multid\_quad\_adapt (d01fcc)** attempts to evaluate a multi-dimensional integral (up to 15 dimensions), with constant and finite limits,

$$\int_{a_1}^{b_1} \int_{a_2}^{b_2} \dots \int_{a_n}^{b_n} f(x_1, x_2, \dots, x_n) dx_n \dots dx_2 dx_1$$

to a specified relative accuracy, using an adaptive subdivision strategy.

### 2. Specification

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

```
void nag_multid_quad_adapt(Integer ndim, double (*f)(Integer ndim, double x[]),
                          double a[], double b[], Integer *minpts, Integer maxpts,
                          double eps, double *fival, double *acc, NagError *fail)
```

### 3. Description

The routine evaluates an estimate of a multi-dimensional integral over a hyper-rectangle (i.e., with constant limits), and also an estimate of the relative error. The user sets the relative accuracy required, supplies the integrand as a function **f**, and also sets the minimum and maximum acceptable number of calls to **f** (in **minpts** and **maxpts**).

The routine operates by repeated subdivision of the hyper-rectangular region into smaller hyper-rectangles. In each subregion, the integral is estimated using a seventh-degree rule, and an error estimate is obtained by comparison with a fifth-degree rule which uses a subset of the same points. The fourth differences of the integrand along each co-ordinate axis are evaluated, and the subregion is marked for possible future subdivision in half along that co-ordinate axis which has the largest absolute fourth difference.

If the estimated errors, totalled over the subregions, exceed the requested relative error (or if fewer than **minpts** calls to **f** have been made), further subdivision is necessary, and is performed on the subregion with the largest estimated error, that subregion being halved along the appropriate co-ordinate axis.

The routine will fail if the requested relative error level has not been attained by the time **maxpts** calls to **f** have been made.

**nag\_multid\_quad\_adapt** is based on the HALF subroutine developed by van Dooren and de Ridder (1976). It uses a different basic rule, described by Genz and Malik (1980).

### 4. Parameters

#### **ndim**

Input: the number of dimensions of the integral, *n*.

Constraint:  $2 \leq \mathbf{ndim} \leq 15$ .

#### **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:

```
double f(Integer ndim, double x[])
    ndim
        Input: the number of dimensions of the integral.
    x[ndim]
        Input: the co-ordinates of the point at which the integrand must be evaluated.
```

**a[ndim]**Input: the lower limits of integration,  $a_i$ , for  $i = 1, 2, \dots, n$ .**b[ndim]**Input: the upper limits of integration,  $b_i$ , for  $i = 1, 2, \dots, n$ .**minpts**Input: **minpts** must be set to the minimum number of integrand evaluations to be allowed.Output: **minpts** contains the actual number of integrand evaluations used by the function nag\_multid\_quad\_adapt.**maxpts**

Input: the maximum number of integrand evaluations to be allowed.

Constraints: **maxpts**  $\geq$  **minpts**.

$$\mathbf{maxpts} \geq 2^{\mathbf{ndim}} + 2 \times \mathbf{ndim}^2 + 2 \times \mathbf{ndim} + 1.$$

**eps**Input: the relative error acceptable to the user. When the solution is zero or very small relative accuracy may not be achievable but the user may still set **eps** to a reasonable value and check **fail.code** for **NE\_QUAD\_MAX\_INTEGRAND\_EVAL**.Constraint: **eps**  $>$  0.0.**finval**

Output: the best estimate obtained for the integral.

**acc**Output: the estimated relative error in **finval**.**fail**

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

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

## 5. Error Indications and Warnings

### NE\_INVALID\_INT\_RANGE\_2

Value  $\langle value \rangle$  given to **ndim** not valid. Correct range is  $2 \leq \mathbf{ndim} \leq 15$ .

### NE\_2\_INT\_ARG\_LT

On entry, **maxpts** =  $\langle value \rangle$  while **minpts** =  $\langle value \rangle$ .These parameters must satisfy **maxpts**  $\geq$  **minpts**.

### NE\_QUAD\_MAX\_INTEGRAND\_CONS

**maxpts**  $<$   $\langle value \rangle$ . Constraint: **maxpts**  $\geq 2^{\mathbf{ndim}} + 2 \times \mathbf{ndim}^2 + 2 \times \mathbf{ndim} + 1$ .

### NE\_REAL\_ARG\_LE

On entry, **eps** must not be less than or equal to 0.0: **eps** =  $\langle value \rangle$ .

### NE\_ALLOC\_FAIL

Memory allocation failed.

### NE\_QUAD\_MAX\_INTEGRAND\_EVAL

**maxpts** was too small to obtain the required accuracy.On return, **finval** and **acc** contain estimates of the integral and the relative error, but **acc** will be greater than **eps**.

## 6. Further Comments

Execution time will usually be dominated by the time taken to evaluate the integrand **f**, and hence the maximum time that could be taken will be proportional to **maxpts**.

### 6.1. Accuracy

A relative error estimate is output through the parameter **acc**.

## 6.2. References

- Genz A C and Malik A A (1980) An Adaptive Algorithm for Numerical Integration over an N-dimensional Rectangular Region *J. Comput. Appl. Math.* **6** 295–302.  
 Van Dooren P and De Ridder L (1976) An Adaptive Algorithm for Numerical Integration over an N-dimensional Cube *J. Comput. Appl. Math.* **2** (3) 207–217.

## 7. See Also

nag\_multid\_quad\_monte\_carlo (d01gbc)

## 8. Example

This example program estimates the integral

$$\int_0^1 \int_0^1 \int_0^1 \int_0^1 \frac{4z_1 z_3^2 \exp(2z_1 z_3)}{(1+z_2+z_4)^2} dz_4 dz_3 dz_2 dz_1 = 0.575364.$$

The accuracy requested is one part in 10,000.

### 8.1. Program Text

```

/* nag_multid_quad_adapt(d01fcc) 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>

#ifdef NAG_PROTO
static double f(Integer n, double z[]);
#else
static double f();
#endif

#define NDIM 4
#define MAXPTS 1000*NDIM

main()
{
    Integer ndim = NDIM;
    Integer maxpts = MAXPTS;
    double a[4], b[4];
    Integer k;
    static NagError fail;
    double finval;
    Integer minpts;
    double acc, eps;

    Vprintf("d01fcc Example Program Results\n");
    for (k=0; k < 4; ++k)
    {
        a[k] = 0.0;
        b[k] = 1.0;
    }
    eps = 0.0001;
    minpts = 0;
    d01fcc(ndim, f, a, b, &minpts, maxpts, eps, &finval, &acc, &fail);
    if (fail.code != NE_NOERROR)
        Vprintf("%s\n", fail.message);
    if (fail.code == NE_NOERROR || fail.code == NE_QUAD_MAX_INTEGRAND_EVAL)

```

```
    {
      Vprintf("Requested accuracy =%12.2e\n", eps);
      Vprintf("Estimated value   =%12.4f\n", finval);
      Vprintf("Estimated accuracy =%12.2e\n", acc);
      exit(EXIT_SUCCESS);
    }
  else
    exit(EXIT_FAILURE);
}

#ifdef NAG_PROTO
static double f(Integer n, double z[])
#else
  static double f(n, z)
  Integer n;
  double z[];
#endif
{
  double tmp_pwr;
  tmp_pwr = z[1]+1.0+z[3];
  return z[0]*4.0*z[2]*z[2]*exp(z[0]*2.0*z[2])/(tmp_pwr*tmp_pwr);
}
```

## 8.2. Program Data

None.

## 8.3. Program Results

```
d01fcc Example Program Results
Requested accuracy = 1.00e-04
Estimated value   = 0.5754
Estimated accuracy = 9.89e-05
```

---