NAG Library Function Document

nag_quad_md_gauss (d01fbc)

1 Purpose

nag_quad_md_gauss (d01fbc) computes an estimate of a multidimensional integral (from 1 to 20 dimensions), given the analytic form of the integrand and suitable Gaussian weights and abscissae.

2 Specification

```
#include <nag.h>
#include <nagd01.h>
double nag_quad_md_gauss (Integer ndim, const Integer nptvec[], Integer lwa,
       const double weight[], const double abscis[],
       double (*fun)(Integer ndim, const double x[], Nag_Comm *comm),
       Nag_Comm *comm, NagError *fail)
```

3 Description

nag_quad_md_gauss (d01fbc) approximates a multidimensional integral by evaluating the summation

$$\sum_{i_1=1}^{l_1} w_{1,i_1} \sum_{i_2=1}^{l_2} w_{2,i_2} \cdots \sum_{i_n=1}^{l_n} w_{n,i_n} f\left(x_{1,i_1}, x_{2,i_2}, \ldots, x_{n,i_n}\right)$$

given the weights w_{j,i_j} and abscissae x_{j,i_j} for a multidimensional product integration rule (see Davis and Rabinowitz (1975)). The number of dimensions may be anything from 1 to 20.

The weights and abscissae for each dimension must have been placed in successive segments of the arrays **weight** and **abscis**; for example, by calling nag_quad_1d_gauss_wset (d01tbc) or nag_quad_1d_gauss_wgen (d01tcc) once for each dimension using a quadrature formula and number of abscissae appropriate to the range of each x_j and to the functional dependence of f on x_j.

If normal weights are used, the summation will approximate the integral

$$\int w_1(x_1) \int w_2(x_2) \cdots \int w_n(x_n) f(x_1, x_2, \ldots, x_n) dx_n \cdots dx_2 dx_1$$

where $w_j(x)$ is the weight function associated with the quadrature formula chosen for the jth dimension; while if adjusted weights are used, the summation will approximate the integral

$$\int \int \cdots \int f(x_1, x_2, \ldots, x_n) dx_n \cdots dx_2 dx_1.$$

You must supply a function to evaluate

$$f(x_1, x_2, \ldots, x_n)$$

at any values of $x_1, x_2, \ldots, x_n$ within the range of integration.

4 References

Davis P J and Rabinowitz P (1975) *Methods of Numerical Integration* Academic Press

5 Arguments

1: **ndim** – Integer *Input*

On entry: n, the number of dimensions of the integral.

Constraint: $1 \leq \mathbf{ndim} \leq 20$.

2: **nptvec**[**ndim**] – const Integer *Input*

On entry: **nptvec**$[j-1]$ must specify the number of points in the jth dimension of the summation, for $j = 1, 2, \ldots, n$.

3: **lwa** – Integer *Input*

On entry: the dimension of the arrays **weight** and **abscis**.

Constraint: **lwa** $\geq$ **nptvec**$[0]$ + **nptvec**$[1]$ + $\cdots$ + **nptvec**[**ndim** $- 1$].

4: **weight**[**lwa**] – const double *Input*

On entry: must contain in succession the weights for the various dimensions, i.e., **weight**$[k-1]$ contains the ith weight for the jth dimension, with

$$k = \mathbf{nptvec}[0] + \mathbf{nptvec}[1] + \cdots + \mathbf{nptvec}[j-2] + i.$$

5: **abscis**[**lwa**] – const double *Input*

On entry: must contain in succession the abscissae for the various dimensions, i.e., **abscis**$[k-1]$ contains the ith abscissa for the jth dimension, with

$$k = \mathbf{nptvec}[0] + \mathbf{nptvec}[1] + \cdots + \mathbf{nptvec}[j-2] + i.$$

6: **fun** – function, supplied by the user *External Function*

fun must return the value of the integrand f at a specified point.

The specification of **fun** is:

```
double fun (Integer ndim, const double x[], Nag_Comm *comm)
```

1: **ndim** – Integer *Input*

On entry: n, the number of dimensions of the integral.

2: **x**[**ndim**] – const double *Input*

On entry: the coordinates of the point at which the integrand f must be evaluated.

3: **comm** – Nag_Comm *

Pointer to structure of type Nag_Comm; the following members are relevant to **fun**.

user – double *
iuser – Integer *
p – Pointer

The type Pointer will be `void *`. Before calling nag_quad_md_gauss (d01fbc) you may allocate memory and initialize these pointers with various quantities for use by **fun** when called from nag_quad_md_gauss (d01fbc) (see Section 2.3.1.1 in How to Use the NAG Library and its Documentation).

7: **comm** – Nag_Comm *

The NAG communication argument (see Section 2.3.1.1 in How to Use the NAG Library and its Documentation).

8: **fail** – NagError * *Input/Output*

The NAG error argument (see Section 2.7 in How to Use the NAG Library and its Documentation).

6 Error Indicators and Warnings

NE_ALLOC_FAIL

Dynamic memory allocation failed.
See Section 2.3.1.2 in How to Use the NAG Library and its Documentation for further information.

NE_BAD_PARAM

On entry, argument ⟨*value*⟩ had an illegal value.

NE_INT

On entry, **ndim** = ⟨*value*⟩.
Constraint: **ndim** ≤ 20.

On entry, **ndim** = ⟨*value*⟩.
Constraint: **ndim** ≥ 1.

NE_INT_2

On entry, **lwa** is too small. **lwa** = ⟨*value*⟩. Minimum possible dimension: ⟨*value*⟩.

NE_INTERNAL_ERROR

An internal error has occurred in this function. Check the function call and any array sizes. If the call is correct then please contact NAG for assistance.

An unexpected error has been triggered by this function. Please contact NAG.
See Section 2.7.6 in How to Use the NAG Library and its Documentation for further information.

NE_NO_LICENCE

Your licence key may have expired or may not have been installed correctly.
See Section 2.7.5 in How to Use the NAG Library and its Documentation for further information.

7 Accuracy

The accuracy of the computed multidimensional sum depends on the weights and the integrand values at the abscissae. If these numbers vary significantly in size and sign then considerable accuracy could be lost. If these numbers are all positive, then little accuracy will be lost in computing the sum.

8 Parallelism and Performance

nag_quad_md_gauss (d01fbc) is not threaded in any implementation.

9 Further Comments

The total time taken by nag_quad_md_gauss (d01fbc) will be proportional to

$$T \times \mathbf{nptvec}[0] \times \mathbf{nptvec}[1] \times \cdots \times \mathbf{nptvec}[\mathbf{ndim} - 1],$$

where T is the time taken for one evaluation of **fun**.

10 Example

This example evaluates the integral

$$\int_1^2 \int_0^\infty \int_{-\infty}^\infty \int_1^\infty \frac{(x_1 x_2 x_3)^6}{(x_4+2)^8} e^{-2x_2} e^{-0.5x_3^2} dx_4 dx_3 dx_2 dx_1$$

using adjusted weights. The quadrature formulae chosen are:

x_1: Gauss–Legendre, $a = 1.0$, $b = 2.0$,

x_2: Gauss–Laguerre, $a = 0.0$, $b = 2.0$,

x_3: Gauss–Hermite, $a = 0.0$, $b = 0.5$,

x_4: rational Gauss, $a = 1.0$, $b = 2.0$.

Four points are sufficient in each dimension, as this integral is in fact a product of four one-dimensional integrals, for each of which the chosen four-point formula is exact.

10.1 Program Text

```
/* nag_quad_md_gauss (d01fbc) Example Program.
 *
 * NAGPRODCODE Version.
 *
 * Copyright 2016 Numerical Algorithms Group.
 *
 * Mark 26, 2016.
 */
#include <stdio.h>
#include <math.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagd01.h>
#include <nagx04.h>

#ifdef __cplusplus
extern "C"
{
#endif
  static double NAG_CALL fun(Integer ndim, const double x[], Nag_Comm *comm);
#ifdef __cplusplus
}
#endif

int main(void)
{
  static double ruser[1] = { -1.0 };
  Integer exit_status = 0;
  Integer ndim;
  double a, ans, b;
  Integer i, j, lwa;
  double *abscis = 0, *weight = 0;
  Integer *nptvec = 0;
  char nag_enum_arg[40];
  Nag_Comm comm;
  Nag_QuadType quadtype;
  NagError fail;

  INIT_FAIL(fail);

  printf("nag_quad_md_gauss (d01fbc) Example Program Results\n");

  /* For communication with user-supplied functions: */
  comm.user = ruser;

  /* Skip heading in data file */
#ifdef _WIN32
  scanf_s("%*[^\n] ");
```

```
#else
  scanf("%*[^\n] ");
#endif
  /* Input parameters */
#ifdef _WIN32
  scanf_s("%" NAG_IFMT "%*[^\n] ", &ndim);
#else
  scanf("%" NAG_IFMT "%*[^\n] ", &ndim);
#endif

  if (!(nptvec = NAG_ALLOC(ndim, Integer)))
  {
    printf("Allocation failure\n");
    exit_status = -1;
    goto END;
  }
  lwa = 0.0;
  for (i = 0; i < ndim; i++) {
#ifdef _WIN32
    scanf_s("%" NAG_IFMT " ", &nptvec[i]);
#else
    scanf("%" NAG_IFMT " ", &nptvec[i]);
#endif
    lwa = lwa + nptvec[i];
  }
#ifdef _WIN32
  scanf_s("%*[^\n] ");
#else
  scanf("%*[^\n] ");
#endif
  if (!(abscis = NAG_ALLOC(lwa, double)) ||
      !(weight = NAG_ALLOC(lwa, double)))
  {
    printf("Allocation failure\n");
    exit_status = -1;
    goto END;
  }

  j = 0;
  for (i = 0; i < ndim; i++) {
    /* Nag_QuadType */
#ifdef _WIN32
    scanf_s("%39s%*[^\n] ", nag_enum_arg, (unsigned)_countof(nag_enum_arg));
#else
    scanf("%39s%*[^\n] ", nag_enum_arg);
#endif
    quadtype = (Nag_QuadType) nag_enum_name_to_value(nag_enum_arg);
#ifdef _WIN32
    scanf_s("%lf %lf%*[^\n] ", &a, &b);
#else
    scanf("%lf %lf%*[^\n] ", &a, &b);
#endif

    /* nag_quad_1d_gauss_wset (d01tbc).
     * Pre-computed weights and abscissae for
     * Gaussian quadrature rules, restricted choice of rule.
     */
    nag_quad_1d_gauss_wset(quadtype, a, b, nptvec[i], &weight[j], &abscis[j],
                           &fail);
    if (fail.code != NE_NOERROR) {
      printf("Error from nag_quad_1d_gauss_wset (d01tbc).\n%s\n",
              fail.message);
      exit_status = 1;
      goto END;
    }
    j = j + nptvec[i];
  }

  /* nag_quad_md_gauss (d01fbc).
   * Multidimensional Gaussian quadrature over hyper-rectangle.
   */
```

```
  ans = nag_quad_md_gauss(ndim, nptvec, lwa, weight, abscis, fun, &comm,
                          &fail);
  if (fail.code != NE_NOERROR) {
    printf("Error from nag_quad_md_gauss (d01fbc).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
  }

  printf("\nAnswer = %10.5f\n", ans);

END:
  NAG_FREE(nptvec);
  NAG_FREE(abscis);
  NAG_FREE(weight);

  return exit_status;
}

static double NAG_CALL fun(Integer ndim, const double x[], Nag_Comm *comm)
{
  if (comm->user[0] == -1.0) {
    printf("(User-supplied callback fun, first invocation.)\n");
    comm->user[0] = 0.0;
  }
  return pow((x[0] * x[1] * x[2]), 6) / pow((x[3] + 2.0), 8)
         * exp(-2.0 * x[1] - 0.5 * x[2] * x[2]);
}
```

10.2 Program Data

None.

10.3 Program Results

```
nag_quad_md_gauss (d01fbc) Example Program Results
(User-supplied callback fun, first invocation.)

Answer =    0.25065
```