NAG Library Function Document

nag_quad_md_sphere (d01fdc)

1 Purpose

nag_quad_md_sphere (d01fdc) calculates an approximation to a definite integral in up to 30 dimensions, using the method of Sag and Szekeres (see Sag and Szekeres (1964)). The region of integration is an n-sphere, or by built-in transformation via the unit n-cube, any product region.

2 Specification

```
#include <nag.h>
#include <nagd01.h>
void nag_quad_md_sphere (Integer ndim,
     double (*f)(Integer ndim, const double x[], Nag_Comm *comm),
     double sigma,
     void (*region)(Integer ndim, const double x[], Integer j, double *c,
          double *d, Nag_Comm *comm),
     Integer limit, double r0, double u, double *result, Integer *ncalls,
     Nag_Comm *comm, NagError *fail)
```

3 Description

nag_quad_md_sphere (d01fdc) calculates an approximation to

$$\int_{n\text{-sphere of radius } \sigma} f(x_1, x_2, \ldots, x_n) dx_1 dx_2 \cdots dx_n \tag{1}$$

or, more generally,

$$\int_{c_1}^{d_1} dx_1 \cdots \int_{c_n}^{d_n} dx_n f(x_1, \ldots, x_n) \tag{2}$$

where each c_i and d_i may be functions of x_j $(j < i)$.

The function uses the method of Sag and Szekeres (1964), which exploits a property of the shifted p-point trapezoidal rule, namely, that it integrates exactly all polynomials of degree $< p$ (see Krylov (1962)). An attempt is made to induce periodicity in the integrand by making a parameterised transformation to the unit n-sphere. The Jacobian of the transformation and all its direct derivatives vanish rapidly towards the surface of the unit n-sphere, so that, except for functions which have strong singularities on the boundary, the resulting integrand will be pseudo-periodic. In addition, the variation in the integrand can be considerably reduced, causing the trapezoidal rule to perform well.

Integrals of the form (1) are transformed to the unit n-sphere by the change of variables:

$$x_i = y_i \frac{\sigma}{r} \tanh\left(\frac{ur}{1 - r^2}\right)$$

where $r^2 = \sum_{i=1}^{n} y_i^2$ and u is an adjustable parameter.

Integrals of the form (2) are first of all transformed to the n-cube $[-1, 1]^n$ by a linear change of variables

$$x_i = ((d_i + c_i) + (d_i - c_i) y_i)/2$$

and then to the unit sphere by a further change of variables

$$y_i = \tanh\left(\frac{uz_i}{1-r}\right)$$

where $r^2 = \sum_{i=1}^{n} z_i^2$ and u is again an adjustable parameter.

The parameter u in these transformations determines how the transformed integrand is distributed between the origin and the surface of the unit n-sphere. A typical value of u is 1.5. For larger u, the integrand is concentrated toward the centre of the unit n-sphere, while for smaller u it is concentrated toward the perimeter.

In performing the integration over the unit n-sphere by the trapezoidal rule, a displaced equidistant grid of size h is constructed. The points of the mesh lie on concentric layers of radius

$$r_i = \frac{h}{4}\sqrt{n + 8(i-1)}, \quad i = 1, 2, 3, \ldots.$$

The function requires you to specify an approximate maximum number of points to be used, and then computes the largest number of whole layers to be used, subject to an upper limit of 400 layers.

In practice, the rapidly-decreasing Jacobian makes it unnecessary to include the whole unit n-sphere and the integration region is limited by a user-specified cut-off radius $r_0 < 1$. The grid-spacing h is determined by r_0 and the number of layers to be used. A typical value of r_0 is 0.8.

Some experimentation may be required with the choice of r_0 (which determines how much of the unit n-sphere is included) and u (which determines how the transformed integrand is distributed between the origin and surface of the unit n-sphere), to obtain best results for particular families of integrals. This matter is discussed further in Section 9.

4 References

Krylov V I (1962) *Approximate Calculation of Integrals* (trans A H Stroud) Macmillan

Sag T W and Szekeres G (1964) Numerical evaluation of high-dimensional integrals *Math. Comput.* **18** 245–253

5 Arguments

1: **ndim** – Integer *Input*

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

Constraint: $1 \leq$ **ndim** ≤ 30.

2: **f** – function, supplied by the user *External Function*

f must return the value of the integrand f at a given point.

The specification of **f** is:

```
double f (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 **f**.

user – double *
iuser – Integer *
p – Pointer

The type Pointer will be `void *`. Before calling nag_quad_md_sphere (d01fdc) you may allocate memory and initialize these pointers with various quantities for use by **f** when called from nag_quad_md_sphere (d01fdc) (see Section 2.3.1.1 in How to Use the NAG Library and its Documentation).

3: **sigma** – double *Input*

On entry: indicates the region of integration.

sigma ≥ 0.0
The integration is carried out over the n-sphere of radius **sigma**, centred at the origin.

sigma < 0.0
The integration is carried out over the product region described by **region**.

4: **region** – function, supplied by the user *External Function*

If **sigma** < 0.0, **region** must evaluate the limits of integration in any dimension.

The specification of **region** is:

```
void region (Integer ndim, const double x[], Integer j, double *c,
     double *d, 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: $\mathbf{x}[0], \ldots, \mathbf{x}[j-2]$ contain the current values of the first $(j-1)$ variables, which may be used if necessary in calculating c_j and d_j.

3: **j** – Integer *Input*

On entry: the index j for which the limits of the range of integration are required.

4: **c** – double * *Output*

On exit: the lower limit c_j of the range of x_j.

5: **d** – double * *Output*

On exit: the upper limit d_j of the range of x_j.

6: **comm** – Nag_Comm *

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

user – double *
iuser – Integer *
p – Pointer

The type Pointer will be `void *`. Before calling nag_quad_md_sphere (d01fdc) you may allocate memory and initialize these pointers with various quantities for use by **region** when called from nag_quad_md_sphere (d01fdc) (see Section 2.3.1.1 in How to Use the NAG Library and its Documentation).

If **sigma** ≥ 0.0, **region** is not called by nag_quad_md_sphere (d01fdc), but the NAG defined null function pointer NULLFN must be supplied.

5: **limit** – Integer *Input*

On entry: the approximate maximum number of integrand evaluations to be used.

Constraint: **limit** ≥ 100.

6: **r0** – double *Input*

On entry: the cut-off radius on the unit n-sphere, which may be regarded as an adjustable parameter of the method.

Suggested value: a typical value is **r0** $= 0.8$. (See also Section 9.)

Constraint: $0.0 <$ **r0** < 1.0.

7: **u** – double *Input*

On entry: must specify an adjustable parameter of the transformation to the unit n-sphere.

Suggested value: a typical value is **u** $= 1.5$. (See also Section 9.)

Constraint: **u** > 0.0.

8: **result** – double * *Output*

On exit: the approximation to the integral I.

9: **ncalls** – Integer * *Output*

On exit: the actual number of integrand evaluations used. (See also Section 9.)

10: **comm** – Nag_Comm *

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

11: **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, **limit** $=$ ⟨*value*⟩.
Constraint: **limit** ≥ 100.

On entry, **ndim** $=$ ⟨*value*⟩.
Constraint: **ndim** ≤ 30.

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

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.

NE_REAL

On entry, **r0** $= \langle value \rangle$.
Constraint: **r0** < 1.0.

On entry, **r0** $= \langle value \rangle$.
Constraint: **r0** > 0.0.

On entry, **u** $= \langle value \rangle$.
Constraint: **u** > 0.0.

7 Accuracy

No error estimate is returned, but results may be verified by repeating with an increased value of **limit** (provided that this causes an increase in the returned value of **ncalls**).

8 Parallelism and Performance

nag_quad_md_sphere (d01fdc) is not threaded in any implementation.

9 Further Comments

The time taken by nag_quad_md_sphere (d01fdc) will be approximately proportional to the returned value of **ncalls**, which, except in the circumstances outlined in (b) below, will be close to the given value of **limit**.

(a) Choice of r_0 and u

If the chosen combination of r_0 and u is too large in relation to the machine accuracy it is possible that some of the points generated in the original region of integration may transform into points in the unit n-sphere which lie too close to the boundary surface to be distinguished from it to machine accuracy (despite the fact that $r_0 < 1$). To be specific, the combination of r_0 and u is too large if

$$\frac{ur_0}{1 - r_0^2} > 0.3465(t - 1), \qquad \text{if } \textbf{sigma} \geq 0.0,$$

or

$$\frac{ur_0}{1 - r_0} > 0.3465(t - 1), \qquad \text{if } \textbf{sigma} < 0.0,$$

where t is the number of bits in the mantissa of a double number.

The contribution of such points to the integral is neglected. This may be justified by appeal to the fact that the Jacobian of the transformation rapidly approaches zero towards the surface. Neglect of these points avoids the occurrence of overflow with integrands which are infinite on the boundary.

(b) Values of **limit** and **ncalls**

limit is an approximate upper limit to the number of integrand evaluations, and may not be chosen less than 100. There are two circumstances when the returned value of **ncalls** (the actual number of evaluations used) may be significantly less than **limit**.

Firstly, as explained in (a), an unsuitably large combination of r_0 and u may result in some of the points being unusable. Such points are not included in the returned value of **ncalls**.

Secondly, no more than 400 layers will ever be used, no matter how high **limit** is set. This places an effective upper limit on **ncalls** as follows:

$$\begin{aligned} n = 1: &\quad 56 \\ n = 2: &\quad 1252 \\ n = 3: &\quad 23690 \\ n = 4: &\quad 394528 \\ n = 5: &\quad 5956906 \end{aligned}$$

10 Example

This example calculates the integral

$$\int\int_s\int \frac{dx_1 dx_2 dx_3}{\sqrt{\sigma^2 - r^2}} = 22.2066$$

where s is the 3-sphere of radius σ, $r^2 = x_1^2 + x_2^2 + x_3^2$ and $\sigma = 1.5$. Both sphere-to-sphere and general product region transformations are used. For the former, we use $r_0 = 0.9$ and $u = 1.5$; for the latter, $r_0 = 0.8$ and $u = 1.5$.

10.1 Program Text

```
/* nag_quad_md_sphere (d01fdc) 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>

#ifdef __cplusplus
extern "C"
{
#endif
  static double NAG_CALL f(Integer ndim, const double x[], Nag_Comm *comm);
  static void NAG_CALL region(Integer ndim, const double x[], Integer j,
                              double *c, double *d, Nag_Comm *comm);
#ifdef __cplusplus
}
#endif

int main(void)
{
  static double ruser[2] = { -1.0, -1.0 };
  Integer exit_status = 0;
  double r0, result, sigma, u;
  Integer i, limit, ncalls, ndim;
  Nag_Comm comm;
  NagError fail;

  INIT_FAIL(fail);

  printf("nag_quad_md_sphere (d01fdc) 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
#ifdef _WIN32
  scanf_s("%" NAG_IFMT "", &ndim);
#else
  scanf("%" NAG_IFMT "", &ndim);
#endif
#ifdef _WIN32
  scanf_s("%" NAG_IFMT "", &limit);
#else
  scanf("%" NAG_IFMT "", &limit);
#endif
#ifdef _WIN32
  scanf_s("%lf %*[^\n] ", &u);
#else
  scanf("%lf %*[^\n] ", &u);
#endif

  for (i = 1; i <= 2; i++) {
    /* nag_quad_md_sphere (d01fdc).
     * Multidimensional quadrature, Sag-Szekeres method,
     * general product region or n-sphere.
     */
    switch (i) {
    case 1:
      printf("\nSphere-to-sphere transformation\n");
      sigma = 1.5;
      r0 = 0.9;
      nag_quad_md_sphere(ndim, f, sigma, NULLFN, limit, r0, u, &result,
                         &ncalls, &comm, &fail);
      break;
    case 2:
      printf("\nProduct region transformation\n");
      sigma = -1.0;
      r0 = 0.8;
      nag_quad_md_sphere(ndim, f, sigma, region, limit, r0, u, &result,
                         &ncalls, &comm, &fail);
      break;
    }
    if (fail.code != NE_NOERROR) {
      printf("Error from nag_quad_md_sphere (d01fdc).\n%s\n", fail.message);
      exit_status = 1;
      goto END;
    }

    printf("\nEstimated value of the integral = %9.3f"
           "\nNumber of integrand evaluations = %4" NAG_IFMT "\n",
           result, ncalls);
  }

END:
  return exit_status;
}

static double NAG_CALL f(Integer ndim, const double x[], Nag_Comm *comm)
{
  Integer i;
  double x_sq = 0.0;

  if (comm->user[0] == -1.0) {
    printf("(User-supplied callback f, first invocation.)\n");
    comm->user[0] = 0.0;
  }

  for (i = 0; i < ndim; i++)
    x_sq += pow(x[i], 2.0);

  return 1.0 / sqrt(fabs(pow(1.5, 2) - x_sq));
```

```
}

static void NAG_CALL region(Integer ndim, const double x[], Integer j,
                            double *c, double *d, Nag_Comm *comm)
{
  Integer i;
  double x_sq = 0.0;

  if (comm->user[1] == -1.0) {
    printf("(User-supplied callback region, first invocation.)\n");
    comm->user[1] = 0.0;
  }
  if (j > 1) {
    for (i = 0; i < (j - 1); i++)
      x_sq += pow(x[i], 2.0);
    *d = sqrt(fabs(pow(1.5, 2) - x_sq));
    *c = -*d;
  }
  else {
    *c = -1.5;
    *d = 1.5;
  }
}
```

10.2 Program Data

None.

10.3 Program Results

```
nag_quad_md_sphere (d01fdc) Example Program Results

Sphere-to-sphere transformation
(User-supplied callback f, first invocation.)

Estimated value of the integral =    22.168
Number of integrand evaluations = 8026

Product region transformation
(User-supplied callback region, first invocation.)

Estimated value of the integral =    22.137
Number of integrand evaluations = 8026
```