NAG Library Function Document
nag_quad_md_simplex (d01pac)

1 Purpose
nag_quad_md_simplex (d01pac) returns a sequence of approximations to the integral of a function over a multidimensional simplex, together with an error estimate for the last approximation.

2 Specification
#include <nag.h>
#include <nagd01.h>

void nag_quad_md_simplex (Integer ndim, double vert[],
double (*functn)(Integer ndim, const double x[], Nag_Comm *comm),
Integer *minord, Integer maxord, double finvls[], double *esterr,
Nag_Comm *comm, NagError *fail)

3 Description
nag_quad_md_simplex (d01pac) computes a sequence of approximations $\text{finvls}[j-1]$, for $j = \text{minord} + 1, \ldots, \text{maxord}$, to an integral

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

where $S$ is an $n$-dimensional simplex defined in terms of its $n + 1$ vertices. $\text{finvls}[j-1]$ is an approximation which will be exact (except for rounding errors) whenever the integrand is a polynomial of total degree $2j - 1$ or less.

The type of method used has been described in Grundmann and Möller (1978), and is implemented in an extrapolated form using the theory from de Doncker (1979).

4 References

5 Arguments
1: \textbf{ndim} – Integer \hspace{1cm} \textit{Input}

\textit{On entry}: $n$, the number of dimensions of the integral.

\textit{Constraint}: $\text{ndim} \geq 2$.

2: \textbf{vert}[dim] – double \hspace{1cm} \textit{Input/Output}

\textit{Note}: the dimension, \textit{dim}, of the array \textbf{vert} must be at least $(2 \times (\text{ndim} + 1)) \times (\text{ndim} + 1)$.

Where \textbf{VERT}(i,j) appears in this document, it refers to the array element \textbf{vert}[$(j - 1) \times (\text{ndim} + 1) + i - 1$].
On entry: \( \text{VERT}(i,j) \) must be set to the \( j \)th component of the \( i \)th vertex for the simplex integration region, for \( i = 1, 2, \ldots, n + 1 \) and \( j = 1, 2, \ldots, n \). If \( \text{minord} > 0 \), \( \text{vert} \) must be unchanged since the previous call of \text{nag_quad_md_simplex (d01pac)}.  

On exit: these values are unchanged. The rest of the array \( \text{vert} \) is used for workspace and contains information to be used if another call of \text{nag_quad_md_simplex (d01pac)} is made with \( \text{minord} > 0 \). In particular \( \text{VERT}(n + 1, 2n + 2) \) contains the volume of the simplex.

3: \( \text{functn} \) – function, supplied by the user

\( \text{functn} \) must return the value of the integrand \( f \) at a given point.

The specification of \( \text{functn} \) is:

\[
\text{double functn (Integer ndim, const double x[], Nag_Comm *comm)}
\]

1: \( \text{ndim} \) – Integer \hspace{1cm} \text{Input}

\( \text{On entry:} \ n, \text{the number of dimensions of the integral.} \)

2: \( \text{x[ndim]} \) – const double \hspace{1cm} \text{Input}

\( \text{On entry: the coordinates of the point at which the integrand} \ f \text{must be evaluated.} \)

3: \( \text{comm} \) – Nag_Comm *

\( \text{Pointer to structure of type Nag_Comm; the following members are relevant to} \ \text{functn}. \)

\begin{itemize}
  \item \( \text{user} \) – double *
  \item \( \text{iuser} \) – Integer *
  \item \( \text{p} \) – Pointer
\end{itemize}

\( \text{The type Pointer will be} \ void *. \text{Before calling} \ \text{nag_quad_md_simplex (d01pac)} \\)
\( \text{you may allocate memory and initialize these pointers with various quantities for} \)
\( \text{use by} \ \text{functn} \text{when called from} \ \text{nag_quad_md_simplex (d01pac)} \text{(see} \)
\( \text{Section 3.2.1.1 in the Essential Introduction).} \)

4: \( \text{minord} \) – Integer * \hspace{1cm} \text{Input/Output}

\( \text{On entry:} \ \text{must specify the highest order of the approximations currently available in the array} \)
\( \text{finvls}. \ \text{\text{minord} = 0 indicates an initial call;} \ \text{\text{minord} > 0 indicates that} \)
\( \text{finvls[0], finvls[1], \ldots, finvls[\text{minord} − 1]} \text{have already been computed in a previous call of} \)
\( \text{\text{nag_quad_md_simplex (d01pac)}.} \)

\( \text{Constraint:} \ \text{\text{minord} ≥ 0}. \)

\( \text{On exit:} \ \text{\text{minord} = maxord}. \)

5: \( \text{maxord} \) – Integer \hspace{1cm} \text{Input}

\( \text{On entry: the highest order of approximation to the integral to be computed.} \)

\( \text{Constraint:} \ \text{\text{maxord} > \text{minord}}. \)

6: \( \text{finvls[maxord]} \) – double \hspace{1cm} \text{Input/Output}

\( \text{On entry: if} \ \text{\text{minord} > 0, finvls[0], finvls[1], \ldots, finvls[\text{minord} − 1]} \text{must contain approximations} \)
\( \text{to the integral previously computed by} \ \text{nag_quad_md_simplex (d01pac)}. \)

\( \text{On exit: contains these values unchanged, and the newly computed values} \)
\( \text{finvls[\text{minord}], finvls[\text{minord} + 1], \ldots, finvls[\text{maxord} − 1]. finvls[j − 1]} \text{is an approximation to} \)
\( \text{the integral of polynomial degree} \ 2j − 1. \)
7:  **esterr** – double *

*On exit*: an absolute error estimate for **finvls**[maxord – 1].

8:  **comm** – Nag_Comm *

The NAG communication argument (see Section 3.2.1.1 in the Essential Introduction).

9:  **fail** – NagError *

The NAG error argument (see Section 3.6 in the Essential Introduction).

6  **Error Indicators and Warnings**

**NE_ACCURACY**

The volume of the simplex integration region is too large or too small to be represented on the machine.

**NE_ALLOC_FAIL**

Dynamic memory allocation failed.
See Section 3.2.1.2 in the Essential Introduction for further information.

**NE_BAD_PARAM**

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

**NE_INT**

On entry, *minord* = *value*.
Constraint: *minord* ≥ 0.

On entry, *ndim* = *value*.
Constraint: *ndim* ≥ 2.

**NE_INT_2**

On entry, *maxord* = *value* and *minord* = *value*.
Constraint: *maxord* > *minord*.

**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 3.6.6 in the Essential Introduction for further information.

**NE_NO_LICENCE**

Your licence key may have expired or may not have been installed correctly. See Section 3.6.5 in the Essential Introduction for further information.

7  **Accuracy**

An absolute error estimate is output through the argument **esterr**.

8  **Parallelism and Performance**

**nag_quad_md_simplex** (**d01pac**) is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.
nag_quad_md_simplex (d01pac) makes calls to BLAS and/or LAPACK routines, which may be threaded within the vendor library used by this implementation. Consult the documentation for the vendor library for further information.

Please consult the X06 Chapter Introduction for information on how to control and interrogate the OpenMP environment used within this function. Please also consult the Users’ Note for your implementation for any additional implementation-specific information.

9 Further Comments

The running time for nag_quad_md_simplex (d01pac) will usually be dominated by the time used to evaluate the integrand functn. The maximum time that could be used by nag_quad_md_simplex (d01pac) will be approximately given by

\[ T \times \frac{(\text{maxord} + \text{ndim})!}{(\text{maxord} - 1)!\text{ndim}!} \]

where \( T \) is the time needed for one call of functn.

10 Example

This example demonstrates the use of the function with the integral

\[ \int_0^1 \int_0^{1-x} \int_0^{1-x-y} \exp(x + y + z) \cos(x + y + z) \, dz \, dy \, dx = \frac{1}{4}. \]

10.1 Program Text

/* nag_quad_md_simplex (d01pac) Example Program. */
/* Copyright 2014 Numerical Algorithms Group. */
/* Mark 23, 2011. */

#include <stdio.h>
#include <math.h>
#include <nag.h>
#include <nag_stdlib.h>

#ifdef __cplusplus
extern "C" {
#endif

static double NAG_CALL functn(Integer ndim, const double x[], Nag_Comm *comm);

#ifdef __cplusplus
}
#endif

int main(void)
{
#define VERT(I, J) vert[(J-1) * (ndim+1) + I-1]

/* Scalars */
Integer exit_status = 0;
Integer i, j, maxord, minord, mxord, ndim;
double esterr;
/* Arrays */
Integer iuser[1];
double *finvls = 0, *vert = 0;
/* Nag types */
NagError fail;
Nag_Comm comm;

INIT_FAIL(fail);

printf("nag_quad_md_simplex (d01pac) Example Program Results\n");
/* Skip heading in data file */
#ifdef _WIN32
    scanf_s("%*[\n] ");
#else
    scanf("%*[\n] ");
#endif

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

if (!(finvls = NAG_ALLOC(mxord, double)) ||
    !(vert = NAG_ALLOC(2*(ndim+1)*(ndim+1), double)))
{
    printf("Allocation failure\n");
    exit_status = -1;
    goto END;
}

for (i = 1; i <= ndim+1; i++)
    for (j = 1; j <= ndim; j++) VERT(i, j) = 0.0;
for (j = 2; j <= ndim+1; j++) VERT(j, j - 1) = 1.0;

minord = 0;
iuser[0] = 0; /* Function counter */
comm.iuser = iuser;
for (maxord = 1; maxord <= mxord; maxord++)
{
    /* nag_quad_md_simplex (d01pac).
     * Multidimensional quadrature over an n-simplex.
     */
    nag_quad_md_simplex(ndim, vert, functn, &minord, maxord, finvls,
                         &esterr, &comm, &fail);
    if (fail.code != NE_NOERROR)
    {
        printf("Error from nag_quad_md_simplex (d01pac).
                %s
", fail.message);
        exit_status = 1;
        goto END;
    }

    if (maxord == 1)
        printf("maxord Estimated Estimated Integrand\n";
                value accuracy evaluations\n";
                printf("%4"NAG_IFMT"%13.5f%16.3e"NAG_IFMT"%n", maxord, finvls[maxord-1], esterr, comm.iuser[0]);
}

END:
NAG_FREE(finvls);
NAG_FREE(vert);

return exit_status;
}

static double NAG_CALL functn(Integer ndim, const double x[], Nag_Comm *comm)
{
    comm->iuser[0]++;
    return exp(x[0] + x[1] + x[2]) * cos(x[0] + x[1] + x[2]);
}

10.2 Program Data
None.

Mark 25
## 10.3 Program Results

<table>
<thead>
<tr>
<th>maxord</th>
<th>Estimated value</th>
<th>Estimated accuracy</th>
<th>Integrand evaluations</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.25816</td>
<td>2.582e-01</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>0.25011</td>
<td>8.058e-03</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>0.25000</td>
<td>1.067e-04</td>
<td>15</td>
</tr>
<tr>
<td>4</td>
<td>0.25000</td>
<td>4.098e-07</td>
<td>35</td>
</tr>
<tr>
<td>5</td>
<td>0.25000</td>
<td>1.731e-09</td>
<td>70</td>
</tr>
</tbody>
</table>