NAG Toolbox

nag_quad_md_adapt (d01fc)

1 Purpose

nag_quad_md_adapt (d01fc) attempts to evaluate a multidimensional integral (up to 15 dimensions), with constant and finite limits, to a specified relative accuracy, using an adaptive subdivision strategy.

2 Syntax

```
[minpts, acc, finval, ifail] = nag_quad_md_adapt(a, b, minpts, maxpts, functn,
eps, 'ndim', ndim)
```

```
[minpts, acc, finval, ifail] = d01fc(a, b, minpts, maxpts, functn, eps, 'ndim',
ndim)
```

3 Description

nag_quad_md_adapt (d01fc) returns an estimate of a multidimensional integral over a hyper-rectangle (i.e., with constant limits), and also an estimate of the relative error. You set the relative accuracy required, return values for the integrand via a function argument **functn**, and also set the minimum and maximum acceptable number of calls to **functn** (in **minpts** and **maxpts**).

The function 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 coordinate axis are evaluated, and the subregion is marked for possible future subdivision in half along that coordinate 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 **functn** 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 coordinate axis.

The function will fail if the requested relative error level has not been attained by the time **maxpts** calls to **functn** have been made; or, if the amount $lenwrk$ of working storage is insufficient. A formula for the recommended value of $lenwrk$ is given in Section 5. If a smaller value is used, and is exhausted in the course of execution, the function switches to a less efficient mode of operation; only if this mode also breaks down is insufficient storage reported.

nag_quad_md_adapt (d01fc) is based on the HALF function developed by van Dooren and de Ridder (1976). It uses a different basic rule, described in Genz and Malik (1980).

4 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** 207–217

5 Parameters

5.1 Compulsory Input Parameters

1: **a**(**ndim**) – REAL (KIND=nag_wp) array

The lower limits of integration, a_i, for $i = 1, 2, \ldots, n$.

2: **b**(**ndim**) – REAL (KIND=nag_wp) array

The upper limits of integration, b_i, for $i = 1, 2, \ldots, n$.

3: **minpts** – INTEGER

Must be set to the minimum number of integrand evaluations to be allowed.

4: **maxpts** – INTEGER

The maximum number of integrand evaluations to be allowed.

Constraints:

$\mathbf{maxpts} \geq \mathbf{minpts}$;
$\mathbf{maxpts} \geq \alpha$, where $\alpha = 2^{\mathbf{ndim}} + 2 \times \mathbf{ndim}^2 + 2 \times \mathbf{ndim} + 1$.

5: **functn** – REAL (KIND=nag_wp) FUNCTION, supplied by the user.

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

```
[result] = functn(ndim, z)
```

Input Parameters

1: **ndim** – INTEGER

n, the number of dimensions of the integral.

2: **z**(**ndim**) – REAL (KIND=nag_wp) array

The coordinates of the point at which the integrand f must be evaluated.

Output Parameters

1: **result**

The value of the integrand f at the given point.

6: **eps** – REAL (KIND=nag_wp)

The relative error acceptable to you. When the solution is zero or very small relative accuracy may not be achievable but you may still set **eps** to a reasonable value and check for the error exit **ifail** $= 2$.

Constraint: **eps** > 0.0.

5.2 Optional Input Parameters

1: **ndim** – INTEGER

Default: the dimension of the arrays **a**, **b**. (An error is raised if these dimensions are not equal.)

n, the number of dimensions of the integral.

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

5.3 Output Parameters

1: **minpts** – INTEGER

Contains the actual number of integrand evaluations used by nag_quad_md_adapt (d01fc).

2: **acc** – REAL (KIND=nag_wp)

The estimated relative error in **finval**.

3: **finval** – REAL (KIND=nag_wp)

The best estimate obtained for the integral.

4: **ifail** – INTEGER

ifail $= 0$ unless the function detects an error (see Section 5).

6 Error Indicators and Warnings

Note: nag_quad_md_adapt (d01fc) may return useful information for one or more of the following detected errors or warnings.

Errors or warnings detected by the function:

ifail $= 1$

On entry, **ndim** < 2,
or **ndim** > 15,
or **maxpts** is too small,
or $lenwrk < 2 \times$ **ndim** $+ 4$,
or **eps** ≤ 0.0.

ifail $= 2$ (*warning*)

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

ifail $= 3$ (*warning*)

$lenwrk$ was too small. On soft failure, **finval** and **acc** contain estimates of the integral and the relative error, but **acc** will be greater than **eps**.

ifail $= -99$

An unexpected error has been triggered by this routine. Please contact NAG.

ifail $= -399$

Your licence key may have expired or may not have been installed correctly.

ifail $= -999$

Dynamic memory allocation failed.

7 Accuracy

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

8 Further Comments

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

9 Example

This example 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 10000.

9.1 Program Text

```
function d01fc_example

fprintf('d01fc example results\n\n');

a = zeros(4,1);
b = ones(4,1);
minpts  = nag_int(0);
maxpts  = nag_int(8000);
epsilon = 0.0001;

[minpts, acc, finval, ifail] = d01fc(...
                                     a, b, minpts, maxpts, @functn, epsilon);

fprintf('Requested accuracy = %10.2e\n', epsilon);
fprintf('Estimated value    = %8.4f\n',finval);
fprintf('Estimated accuracy = %9.1e\n',acc);

function result = functn(ndim,z)
  result = 4*z(1)*z(3)*z(3)*exp(2*z(1)*z(3))/(1+z(2)+z(4))^2;
```

9.2 Program Results

```
d01fc example results

Requested accuracy =   1.00e-04
Estimated value    =   0.5754
Estimated accuracy =   9.9e-05
```