NAG Toolbox for MATLAB

d01gb

1 Purpose

d01gb returns an approximation to the integral of a function over a hyper-rectangular region, using a Monte Carlo method. An approximate relative error estimate is also returned. This function is suitable for low accuracy work.

2 Syntax

```
[mincls, acc, wrkstr, finest, ifail] = d01gb(a, b, mincls, maxcls,
functn, eps, wrkstr, finest, 'ndim', ndim, 'lenwrk', lenwrk)
```

3 Description

d01gb uses an adaptive Monte Carlo method based on the algorithm described in Lautrup 1971. It is implemented for integrals of the form:

$$\int_{a_1}^{b_1}\int_{a_2}^{b_2}\cdots\int_{a_n}^{b_n} f(x_1,x_2,\ldots,x_n)\,dx_n\cdots dx_2dx_1.$$

Upon entry, unless **lenwrk** has been set to the minimum value $10 \times$ **ndim**, the function subdivides the integration region into a number of equal volume subregions. Inside each subregion the integral and the variance are estimated by means of pseudo-random sampling. All contributions are added together to produce an estimate for the whole integral and total variance. The variance along each co-ordinate axis is determined and the function uses this information to increase the density and change the widths of the sub-intervals along each axis, so as to reduce the total variance. The total number of subregions is then increased by a factor of two and the program recycles for another iteration. The program stops when a desired accuracy has been reached or too many integral evaluations are needed for the next cycle.

4 References

Lautrup B 1971 An adaptive multi-dimensional integration procedure *Proc. 2nd Coll. Advanced Methods in Theoretical Physics, Marseille*

5 Parameters

5.1 Compulsory Input Parameters

1: **a**(**ndim**) – **double array**

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

2: **b**(**ndim**) – **double array**

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

3: **mincls – int32 scalar**

Must be set

either to the minimum number of integrand evaluations to be allowed, in which case **mincls** ≥ 0;

or to a negative value. In this case, the function assumes that a previous call had been made with the same parameters **ndim**, **a** and **b** and with either the same integrand (in which case d01gb continues calculation) or a similar integrand (in which case d01gb begins the

calculation with the subdivision used in the last iteration of the previous call). See also **wrkstr**.

4: **maxcls – int32 scalar**

The maximum number of integrand evaluations to be allowed. In the continuation case this is the number of new integrand evaluations to be allowed. These counts do not include zero integrand values.

Constraints:

maxcls $>$ **mincls**;
maxcls $\geq 4 \times (\mathbf{ndim} + 1)$.

5: **functn – string containing name of m-file**

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

Its specification is:

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

Input Parameters

1: **ndim – int32 scalar**

n, the number of dimensions of the integral.

2: **x(ndim) – double array**

The co-ordinates of the point at which the integrand f must be evaluated.

Output Parameters

1: **result – double scalar**

The result of the function.

6: **eps – double scalar**

The relative accuracy required.

Constraint: **eps** ≥ 0.0.

7: **wrkstr(lenwrk) – double array**

If **mincls** < 0, **wrkstr** must be unchanged from the previous call of d01gb – except that for a new integrand **wrkstr**(**lenwrk**) must be set to 0.0. See also **mincls**.

8: **finest – double scalar**

Must be unchanged from a previous call to d01gb.

5.2 Optional Input Parameters

1: **ndim – int32 scalar**

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: **ndim** ≥ 1.

2: **lenwrk – int32 scalar**

Default: The dimension of the array **wrkstr**.

For maximum efficiency, **lenwrk** should be about

$$3 \times \mathbf{ndim} \times (\mathbf{maxcls}/4)^{1/\mathbf{ndim}} + 7 \times \mathbf{ndim}.$$

If **lenwrk** is given the value $10 \times \mathbf{ndim}$ then the (sub)program uses only one iteration of a crude Monte Carlo method with **maxcls** sample points.

Constraint: $\mathbf{lenwrk} \geq 10 \times \mathbf{ndim}$.

5.3 Input Parameters Omitted from the MATLAB Interface

None.

5.4 Output Parameters

1: **mincls – int32 scalar**

Contains the number of integrand evaluations actually used by d01gb.

2: **acc – double scalar**

The estimated relative accuracy of **finest**.

3: **wrkstr**(**lenwrk**) **– double array**

Contains information about the current sub-interval structure which could be used in later calls of d01gb. In particular, **wrkstr**(j) gives the number of sub-intervals used along the jth co-ordinate axis.

4: **finest – double scalar**

The best estimate obtained for the integral.

5: **ifail – int32 scalar**

0 unless the function detects an error (see Section 6).

6 Error Indicators and Warnings

Errors or warnings detected by the function:

ifail $= 1$

On entry, $\mathbf{ndim} < 1$,
or $\mathbf{mincls} \geq \mathbf{maxcls}$,
or $\mathbf{lenwrk} < 10 \times \mathbf{ndim}$,
or $\mathbf{maxcls} < 4 \times (\mathbf{ndim} + 1)$,
or $\mathbf{eps} < 0.0$.

ifail $= 2$

maxcls was too small for d01gb to obtain the required relative accuracy **eps**. In this case d01gb returns a value of **finest** with estimated relative error **acc**, but **acc** will be greater than **eps**. This error exit may be taken before **maxcls** nonzero integrand evaluations have actually occurred, if the function calculates that the current estimates could not be improved before **maxcls** was exceeded.

7 Accuracy

A relative error estimate is output through the parameter **acc**. The confidence factor is set so that the actual error should be less than **acc** 90% of the time. If you want a higher confidence level then a smaller value of **eps** should be used.

8 Further Comments

The running time for d01gb will usually be dominated by the time used to evaluate the user-supplied real function **functn**, so the maximum time that could be used is approximately proportional to **maxcls**.

For some integrands, particularly those that are poorly behaved in a small part of the integration region, d01gb may terminate with a value of **acc** which is significantly smaller than the actual relative error. This should be suspected if the returned value of **mincls** is small relative to the expected difficulty of the integral. Where this occurs, d01gb should be called again, but with a higher entry value of **mincls** (e.g., twice the returned value) and the results compared with those from the previous call.

The exact values of **finest** and **acc** on return will depend (within statistical limits) on the sequence of random numbers generated within d01gb by calls to g05ka. Separate runs will produce identical answers unless the part of the program executed prior to calling d01gb also calls (directly or indirectly) functions from Chapter G05, and the series of such calls differs between runs. If desired, you may ensure the identity or difference between runs of the results returned by d01gb, by calling g05kb or g05kc respectively, immediately before calling d01gb.

9 Example

d01gb_functn.m

```
function result = functn(ndim,x)
  result = 4.0*x(1)*x(3)^2*exp(2.0*x(1)*x(3))/(1.0+x(2)+x(4))^2;
```

```
a = [0;
     0;
     0;
     0];
b = [1;
     1;
     1;
     1];
mincls = int32(1000);
maxcls = int32(20000);
eps = 0.01;
wrkstr =zeros(500,1);
finest = 0;
[minclsOut, acc, wrkstrOut, finestOut, ifail] = ...
    d01gb(a, b, mincls, maxcls, 'd01gb_functn', eps, wrkstr, finest)
```

```
minclsOut =
        1728
acc =
    0.0082
wrkstrOut =
     array elided
finestOut =
    0.5755
ifail =
           0
```