NAG Toolbox for MATLAB

d01ea

1 Purpose

d01ea computes approximations to the integrals of a vector of similar functions, each defined over the same multi-dimensional hyper-rectangular region. The function uses an adaptive subdivision strategy, and also computes absolute error estimates.

2 Syntax

```
[mincls, wrkstr, finest, absest, ifail] = d01ea(a, b, mincls, maxcls,
nfun, funsub, absreq, relreq, wrkstr, 'ndim', ndim, 'lenwrk', lenwrk)
```

3 Description

d01ea uses a globally adaptive method based on the algorithm described by van Dooren and De Ridder 1976 and Genz and Malik 1980. It is implemented for integrals in the form:

$$\int_{a_1}^{b_1}\int_{a_2}^{b_2}\cdots\int_{a_n}^{b_n}(f_1,f_2,\ldots,f_m)\,dx_n\ldots dx_2dx_1,$$

where $f_i = f_i(x_1,x_2,\ldots,x_n)$, for $i = 1,2,\ldots,m$.

Upon entry, unless **mincls** has been set to a value less than or equal to 0, the (sub)program divides the integration region into a number of subregions with randomly selected volumes. Inside each subregion the integrals and their errors are estimated. The initial number of subregions is chosen to be as large as possible without using more than **mincls** calls to user-supplied (sub)program **funsub**. The results are stored in a partially ordered list (a heap). The function then proceeds in stages. At each stage the subregion with the largest error (measured using the maximum norm) is halved along the co-ordinate axis where the integrands have largest absolute fourth differences. The basic rule is applied to each half of this subregion and the results are stored in the list. The results from the two halves are used to update the global integral and error estimates (**finest** and **absest**) and the function continues unless $\|\mathbf{absest}\| \leq \max(\mathbf{absreq}, \|\mathbf{finest}\| \times \mathbf{relreq})$ where the norm $\|.\|$ is the maximum norm, or further subdivision would use more than **maxcls** calls to **funsub**. If at some stage there is insufficient working storage to keep the results for the next subdivision, the function switches to a less efficient mode; only if this mode of operation breaks down is insufficient storage reported.

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**) – **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 user-supplied (sub)program **funsub** calls to be allowed, in which case **mincls** ≥ 0 or to a negative value. In this case, the function continues the calculation started in a previous call with the same integrands and integration limits: no parameters other than **mincls**, **maxcls**, **absreq**, **relreq** or **ifail** must be changed between the calls.

4: **maxcls – int32 scalar**

The maximum number of user-supplied (sub)program **funsub** calls to be allowed. In the continuation case this is the number of new **funsub** calls to be allowed.

Constraints:

maxcls $\geq$ **mincls**;
maxcls $\geq r$;
where $r = 2^n + 2n^2 + 2n + 1$, if $n < 11$, or $r = 1 + n\left(4n^2 - 6n + 14\right)/3$, if $n \geq 11$.

5: **nfun – int32 scalar**

m, the number of integrands.

Constraint: **nfun** ≥ 1.

6: **funsub – string containing name of m-file**

funsub must evaluate the integrands f_i at a given point.

Its specification is:

```
[f] = funsub(ndim, z, nfun)
```

Input Parameters

1: **ndim – int32 scalar**

n, the number of dimensions of the integrals.

2: **z(ndim) – double array**

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

3: **nfun – int32 scalar**

m, the number of integrands.

Output Parameters

1: **f(nfun) – double array**

The value of the ith integrand at the given point.

7: **absreq – double scalar**

The absolute accuracy required by you.

Constraint: **absreq** ≥ 0.0.

8: **relreq – double scalar**

The relative accuracy required by you.

Constraint: **relreq** ≥ 0.0.

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

If **mincls** < 0, **wrkstr** must be unchanged from the previous call of d01ea.

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 integrals.

Constraint: **ndim** ≥ 1.

2: **lenwrk – int32 scalar**

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

Suggested value: **lenwrk** $\geq 6n + 9m + (n + m + 2)(1 + p/r)$, where p is the value of **maxcls** and r is defined under **maxcls**. If **lenwrk** is significantly smaller than this, the function will not work as efficiently and may even fail.

Constraint: **lenwrk** $\geq 8 \times$ **ndim** $+ 11 \times$ **nfun** $+ 3$.

5.3 Input Parameters Omitted from the MATLAB Interface

None.

5.4 Output Parameters

1: **mincls – int32 scalar**

Gives the number of user-supplied (sub)program **funsub** calls actually used by d01ea. For the continuation case (**mincls** < 0 on entry) this is the number of new **funsub** calls on the current call to d01ea.

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

Contains information about the current subdivision which could be used in a continuation call.

3: **finest**(**nfun**) **– double array**

finest(i) specifies the best estimate obtained from the ith integral, for $i = 1, 2, \ldots, m$.

4: **absest**(**nfun**) **– double array**

absest(i) specifies the estimated absolute accuracy of **finest**(i), for $i = 1, 2, \ldots, m$.

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 $= 1$

maxcls was too small for d01ea to obtain the required accuracy. The arrays **finest** and **absest** respectively contain current estimates for the integrals and errors.

ifail $= 2$

lenwrk is too small for the function to continue. The arrays **finest** and **absest** respectively contain current estimates for the integrals and errors.

ifail $= 3$

On a continuation call, **maxcls** was set too small to make any progress. Increase **maxcls** before calling d01ea again.

ifail $= 4$

On entry, **ndim** < 1,
or **nfun** < 1,
or **maxcls** $<$ **mincls**,
or **maxcls** $< r$ (see **maxcls**),
or **absreq** < 0.0,
or **relreq** < 0.0,
or **lenwrk** $< 8 \times$ **ndim** $+ 11 \times$ **nfun** $+ 3$.

7 Accuracy

An absolute error estimate for each integrand is output in the array **absest**. The function exits with **ifail** $= 0$ if

$$\max_i(\mathbf{absest}(i)) \leq \max\left(\mathbf{absreq}, \mathbf{relreq} \times \max_i|\mathbf{finest}(i)|\right).$$

8 Further Comments

Usually the running time for d01ea will be dominated by the time in the user-supplied (sub)program **funsub**, so the maximum time that could be used by d01ea will be proportional to **maxcls** multiplied by the cost of a call to **funsub**.

On a normal call, you should set **mincls** $= 0$ on entry.

For some integrands, particularly those that are poorly behaved in a small part of the integration region, d01ea may terminate prematurely with values of **absest** that are significantly smaller than the actual absolute errors. This behaviour should be suspected if the returned value of **mincls** is small relative to the expected difficulty of the integrals. When this occurs d01ea should be called again, but with an entry value of **mincls** $\geq 2r$, (see specification of **maxcls**) and the results compared with those from the previous call.

If the function is called with **mincls** $\geq 2r$, the exact values of **finest** and **absest** on return will depend (within statistical limits) on the sequence of random numbers generated internally within d01ea by calls to g05ka. Separate runs will produce identical answers unless the part of the program executed prior to calling d01ea also calls (directly or indirectly) functions from Chapter G05, and, in addition, the series of such calls differs between runs.

Because of moderate instability in the application of the basic integration rule, approximately the last $1 + \log_{10}(n^3)$ decimal digits may be inaccurate when using d01ea for large values of n.

9 Example

d01ea_funsub.m

```
function f = funsub(ndim, z, nfun)

  f = zeros(nfun,1);
  sm = 0;
  for n=1:ndim
    sm = sm + double(n)*z(n);
  end
  for k=1:nfun
    f(k) = log(sm)*sin(double(k)+sm);
  end
```

```
a = [0; 0; 0; 0];
b = [1; 1; 1; 1];
mincls = int32(0);
maxcls = int32(57);
nfun = int32(10);
absreq = 0;
relreq = 0.001;
wrkstr = zeros(146,1);
warning('off', 'NAG:warning');
[mincls, wrkstr, finest, absest, ifail] = ...
     d01ea(a, b, mincls, maxcls, nfun, 'd01ea_funsub', absreq, relreq,
wrkstr);
while (ifail == 1 || ifail == 3)
  mincls = int32(-1);
  maxcls = maxcls * int32(16);
  [mincls, wrkstr, finest, absest, ifail] = ...
     d01ea(a, b, mincls, maxcls, nfun, 'd01ea_funsub', absreq, relreq,
wrkstr);
end
  finest
   absest
```

```
finest =
    0.0384
    0.4012
    0.3952
    0.0258
   -0.3672
   -0.4227
   -0.0895
    0.3260
    0.4417
    0.1514
absest =
   1.0e-03 *
    0.3536
    0.3164
    0.3062
    0.3490
    0.3284
    0.3136
    0.3487
    0.3408
    0.3148
    0.3414
```