# NAG Toolbox for MATLAB

# d01fd

## 1 Purpose

d01fd 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 Syntax

```
[result, ncalls, ifail] = d01fd(ndim, f, sigma, region, limit, 'r0', r0, 'u', u)
```

## 3 Description

d01fd 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 parameterized 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

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

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 8.

# 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 Parameters

## 5.1 Compulsory Input Parameters

1: **ndim – int32 scalar**

$n$, the number of dimensions of the integral.

*Constraint*: $1 \leq$ **ndim** $\leq 30$.

2: **f – string containing name of m-file**

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

Its specification is:

```
[result] = f(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.

3: **sigma – double scalar**

Indicates the region of integration.

**sigma** $\geq 0.0$

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 the (sub)program **region**.

4: **region – string containing name of m-file**

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

Its specification is:

```
[c, d] = region(ndim, x, j)
```

**Input Parameters**

1: **ndim – int32 scalar**

$n$, the number of dimensions of the integral.

2: **x(ndim) – double array**

$\mathbf{x}(1), \ldots, \mathbf{x}(j-1)$ 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 – int32 scalar**

The index $j$ for which the limits of the range of integration are required.

**Output Parameters**

1: **c – double scalar**

The lower limit $c_j$ of the range of $x_j$.

2: **d – double scalar**

The upper limit $d_j$ of the range of $x_j$.

If **sigma** $\geq 0.0$, **region** is not called by d01fd, but a dummy function must be supplied (`'d01fdv'` may be used).

5: **limit – int32 scalar**

The approximate maximum number of integrand evaluations to be used.

*Constraint*: **limit** $\geq 100$.

## 5.2 Optional Input Parameters

1: **r0 – double scalar**

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 8.)

*Default*: 0.8

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

2: **u – double scalar**

Must specify an adjustable parameter of the transformation to the unit $n$-sphere.

*Suggested value*: a typical value is $\mathbf{u} = 1.5$. (See also Section 8.)

*Default*: 1.5

*Constraint*: $\mathbf{u} > 0.0$.

### 5.3 Input Parameters Omitted from the MATLAB Interface

None.

### 5.4 Output Parameters

1: **result – double scalar**

The approximation to the integral $I$.

2: **ncalls – int32 scalar**

The actual number of integrand evaluations used. (See also Section 8.)

3: **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, **ndim** $< 1$,
or **ndim** $> 30$.

**ifail** $= 2$

On entry, **limit** $< 100$.

**ifail** $= 3$

On entry, **r0** $\leq 0.0$,
or **r0** $\geq 1.0$.

**ifail** $= 4$

On entry, **u** $\leq 0.0$.

## 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 Further Comments

The time taken by d01fd will be approximately proportional to the returned value of **ncalls**, which, except in the circumstances outlined in below, will be close to the given value of **limit**.

(a) Choice of **r0** 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 , an unsuitably large combination of **r0** 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:

| | |
|---|---|
| $n = 1$ : | 56 |
| $n = 2$ : | 1252 |
| $n = 3$ : | 23690 |
| $n = 4$ : | 394528 |
| $n = 5$ : | 5956906 |

## 9 Example

d01fd_f.m

```
function result = functn(ndim,x)
  result = 2.25;
  for i = 1:ndim
    result = result - x(i)^2;
  end
  result=1/sqrt(abs(result));
```

d01fd_region.m

```
function [c,d] = region(ndim, x, j)
      c = -1.5;
      d = 1.5;
      if (j > 1)
         sm = 2.25;
         for i = 1:(j-1)
            sm = sm - x(i)*x(i);
         end
         d = sqrt(abs(sm));
         c = -d;
      end
```

```
ndim = int32(3);
sigma = 1.5;
limit = int32(8000);
[result, ncalls, ifail] = ...
    d01fd(ndim, 'd01fd_f', sigma, 'd01fd_region', limit, 'r0', 0.9)
```

```
result =
   22.1679
ncalls =
        8026
ifail =
           0
```