Numeric Calculus of Double Integrals

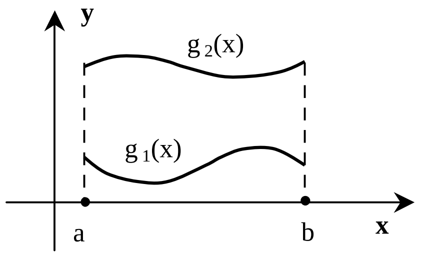
1-June-2003, by Leonardo Volpi

Multidimensional integral is a difficult subject

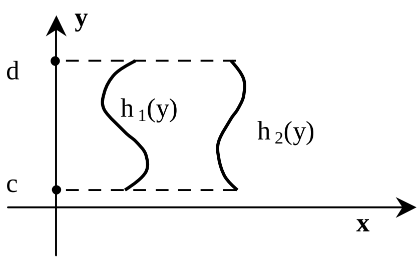
In many book we found that multidimensional integrals are not an easy subject. We agree. Mainly for two reasons: the elaboration effort increases as the nth power of the integral dimension. For double integration, the function f (x, y) must be sampled N^2 times. This increases sharply the elaboration time to get a sufficiently accuracy and increases also the round-off errors. For example, if we need about 100 nodes to get the integral of a one-dimensional function f (x), we will need in theory about 100^2 = 10.000 nodes to reach the same level of accuracy. But the round-off accumulation decreases the max accuracy that we could gain. So we should need much more function samples to reach the final accuracy wanted.
A second reason that greatly complicates the multidimensional integration is the boundary that for a 2D Integral is a region of x-y plane. By contrast, the boundary of one-dimensional integral consists of two constant - upper and lower limits. In our opinion this is the most difficult aspect. The first aspect - the elaboration effort - can be overcame with more power and faster machine, but the second one presents a deeper conceptual difficulty. The fact is that multidimensional integration routines are very rare to find in the public domain. If exist, they are limited to solve the integration in a rectangular domain. But high difficult, there is no a good reason to do nothing.
A 2D-Integration routine for smooth simple functions is better than nothing

2D Integration for Normal Domain

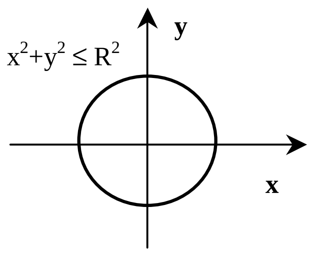
In this article we presents a routine for bi-dimensional integration of a function f (x, y) on a normal domain (both normal to the x-axis and y-axis) or circular domain.



Domain normal to x-axis



Domain normal to y-axis



Circular Domain

For those kinds of 2D-domains the integration formulas can re-write as the following

$$\iint_{D_X} f(x,y)ds = \int_a^b \int_{g_1(x)}^{g_2(x)} f(x,y)dydx$$

$$\iint_{D_Y} f(x,y)ds = \int_c^d \int_{h_1(y)}^{h_2(y)} f(x,y)dxdy$$

$$\iint_C f(x,y)ds = \int_0^{2p}\int_0^{R} f(r\cos(q), r\sin(q))\, r\; dr\; dq$$

Note that the normal domain implies that - at least -one axis must have constant limits. Rectangular domains are a sub case of normal domain in which both axes have constant limits.

Sub Integral_2D_N

The subroutine **Integral_2D_N** just approximate this kinds of integrals
It is written in Visual Basic. The code is open and freeware; it calls the class clsMathParses (you can find both them at the download section of our site)
It uses the 2D-Romberg method

```
Sub Integral_2D_N(Funct, Limit_min(), Limit_max(), Results(), Rank, Accuracy, ErrMsg,
Polar As Boolean)

Input:
Funct = f(x, y) integration function
Limit_min(1) and Limit_max(1) are the boundary for the x-axis
Limit_min(2) and Limit_max(2) are the boundary for the y-axis
(Limits can be constant or functions as well)
Rank = sets the max iterations for Romberg method; points used are about 4^Rank
Accuracy = sets the max relative error wanted
Polar = True/False switches to the polar coordinate for circular domains

Output:
Results(1) = Approximated integral
Results(2) = Estimated error
Results(3) = Counter of points evaluated
ErrMsg = returns any error detected
```

Integration function can be:

- bi-dimensional functions like x^2+y^2-x*y, log(1+x+y), 1/(1+x^2+2*y^2), etc.
- constant numbers like 0 , 2, 1.5, 1E-6 , etc.
- constant expressions like 1/2, √2+1; sin(0.1), etc.

Boundary limits can be:

- constant numbers like 0 , 2 , 10 , 3.141 , etc.
- constant expressions like 1/2 , √2+1, pi, sin(1/2*pi) , exp(1) , etc.
- mono-dimension functions like x/2 , 3y-10 , x^2+x-1 , etc.

For further details about syntax math expressions see the clsMathParser documentation.

Rank limits the max Romberg iterations allowed. The number of nodes Nmax grows exponentially with the Romberg iterations R; that is Nmax = 4^R. Usually Rank is sets to 10, giving about 1.000.000 nodes. Elaboration time depends of course by the numbers of node evaluated and by the complexity of the functions.
Accuracy sets the max relative error. Romberg integration stops when the estimate error goes down under this values.
The subroutine returns the vector *Results()* containing in orders: The approximated integral, the error estimated and the number of true nodes evaluated N < Nmax.

The ErrMsg variable contains any error detected. If no error raises, then ErrMsg = ""
Possible errors are listed in the following tables

Errors	Description
Syntax error of integration function f(x, y)	It happens when the formula contains an errors
Syntax error of bounding functions h(y), g(x)	It happens when any limit expression contains an errors
Variable name must be x, y	It happens when the integration function contains variable different from x and y
too many variables for f(x,y) function	It happens when the integration function contains more than 2 variables
Bounding error	It happens when the limit functions have wrong variables number. The limit functions must have at the most one variable
Normal domain not found for any axes	It happens when no axis has constant limits. At the least, one axes must have constant limits.
Evaluation error	It happens when a function is calculated for incorrect values of its variables. For example: log(0), sqr(-1), etc.

The last error rises when the integration function or one of the limit functions has one or more singularities inside the integration domain. If it happens surely you have written a wrong integral. You must study accurately the integration function and its singularities. This integration routine cannot do this for you.
Now, Let's see how it works

Same testes

In order to examine several integrals it is better to prepare an application for testing.
We use the Excel spreadsheet and the VBA environment.
Open Excel and he VBA editor. Loads both modules Integr_2D_v1.bas and clsMathParse.bas in your project.
Arrange a worksheet like the following example:

	A	B	C	D	E	F	G	H
1	2D Integration	$\iint f(x,y)ds$						Run
2								
3	F(x,y)	Xmin	Xmax	Ymin	Ymax	k max	ErrRel max	Polar
4	ln(1+x+y)	0	1	x^2	x	9	1.00E-14	FALSO
5								
6	integral	time	points	Estim. Error rel.	True Error	True Error rel.	True Integr.	
7	0.102105714	0.1015625	16,641	2.63678E-16	5.13478E-16	5.02889E-15	0.102105714	
8								
9								

Add a new module and copy the following code and assign the macro **Integral_2D_Test** to the "Run" button

```
Sub Integral_2D_Test()
Dim Fxy$, g1$, g2$, h1$, h2$, k_max, ErrorRel, PolarCoor As Boolean
Dim Bound_min(1 To 2), Bound_max(1 To 2), Results(), ErrMsg
'initialization ---------------------------------------------
ErrorRel = [g4]
k_max = [f4]
h1 = [b4]   'left bound function  h1(y)
h2 = [c4]   'right bound function  h2(y)
```

```
g1 = [d4]    'lower bound function  g1(x)
g2 = [e4]    'upper bound function  g2(x)
Fxy = [A4] 'integration function  f(x,y)
PolarCoor = [h4]  'polar coordinates TRUE/FALSE

t0 = Timer
If ErrorRel = "" Then ErrorRel = 10 ^ -7
If k_max = "" Then k_max = 9
If h1 = "" Or h2 = "" Or g1 = "" Or g2 = "" Then
    MsgBox "boundary limits missing", vbCritical
    Exit Sub
End If
If Fxy = "" Then
    MsgBox "Integration function missing", vbCritical
    Exit Sub
End If

Application.Cursor = xlWait
Bound_min(1) = h1
Bound_max(1) = h2
Bound_min(2) = g1
Bound_max(2) = g2

Integral_2D_N Fxy, Bound_min, Bound_max, Results, k_max, ErrorRel, ErrMsg, PolarCoor

t1 = Timer - t0
Application.Cursor = xlDefault

If ErrMsg = "" Or ErrMsg = "Evaluation error" Then
    [a7] = Results(1)
    [b7] = t1
    [c7] = Results(3)  'points evaluated
    [d7] = Results(2)
    If ErrMsg = "Evaluation error" Then
        [a8] = "Singularity: dubious accuracy"
    Else
        [a8] = ""
    End If
Else
    MsgBox ErrMsg
    [a8] = ErrMsg
End If
'
End Sub
```

If you are in hurry, you can use the *Integration 2D v1 test.xls* in the integration package. It has just all these things already implemented.

Results

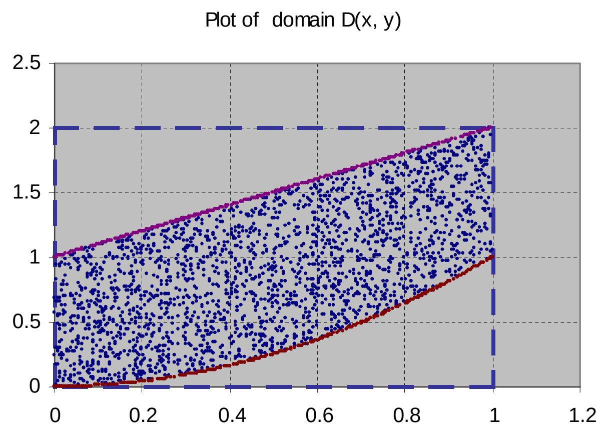
We had tests Integral_2D_N with several examples of double integrals. For each of one we have reported the values returned by the routine: the approximated integral, the estimated relative error, and the number of evaluated nodes. For each integral we have added the exact integral (if known), the true absolute and relative error. Finally we have added the elaboration time, obtained on a PC machine with Pentium III, 1.2 GHz clock

Setting. For all examples *Accuracy* = 1E-14 and *Rank* = 9 , unless different specified.

Example 1:

$$\iint_{D(x,y)} f(x,y)ds = \int_0^1 \int_{x^2}^{x+1} \ln(1+x+y)dydx$$

$$\frac{-9\ln(3)}{4} + 7\ln(2) - \frac{p\sqrt{3}}{12} - \frac{17}{8}$$

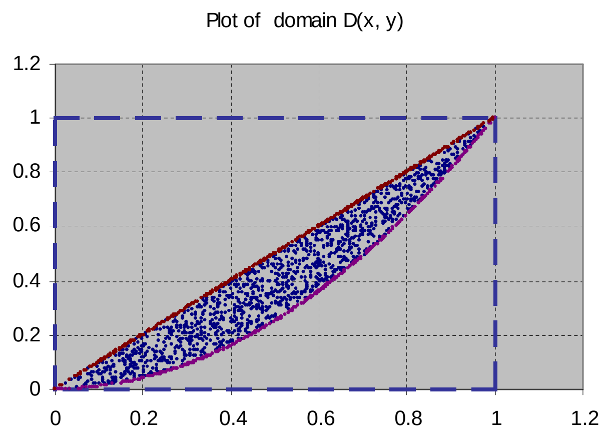


Approx. integral	0.982258328913374
True Integral	0.982258328913371
True Error	2.554E-15
Error relative.	2.600E-15
Estimate. Error rel.	1.110E-16
Time (sec)	0.2617
Points	16641

Example 2:

$$\iint_{D(x,y)} f(x,y)ds = \int_0^1 \int_{x^2}^{x} \ln(1+x+y)dydx$$

$$\frac{5}{9} - \frac{\sqrt{3}}{12}p$$

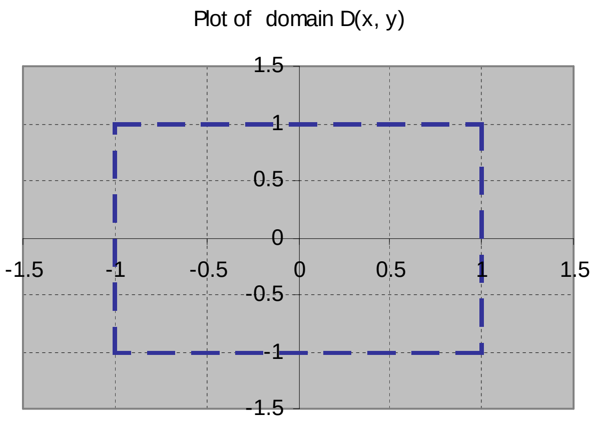


Approx. integral	0.102105714497002
True Integral	0.102105714497001
True Error	5.135E-16
Error relative.	5.029E-15
Estimate. Error rel.	2.637E-16
Time (sec)	0.1016
Points	16641

Vertical edges of a normal domain to x-axis can be collapsed to a couple of points, as in this case. The domain is also normal to the y-axis

Example 3:

$\iint_D f(x,y)ds = \int_{-1}^{1}\int_{-1}^{1} e^{-(x^2+y^2)}\, dy\, dx$	$p \cdot (erf(1))^2$

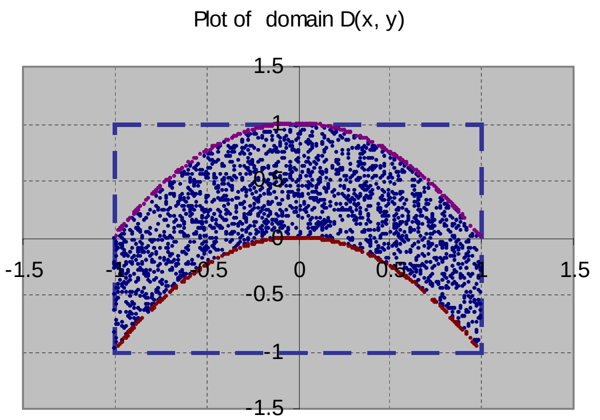


Approx. integral	2.230985141404140
True Integral	2.230985141404130
True Error	1.066E-14
Error relative.	4.777E-15
Estimate. Error rel.	0.000E+00
Time (sec)	0.6680
Points	66049

A rectangular domain is normal to both axes.

Example 4:

$\iint_D f(x,y)ds = \int_{-1}^{1}\int_{-1}^{1} e^{-(x^2+y^2)}\, dy\, dx$	1.237463088347400

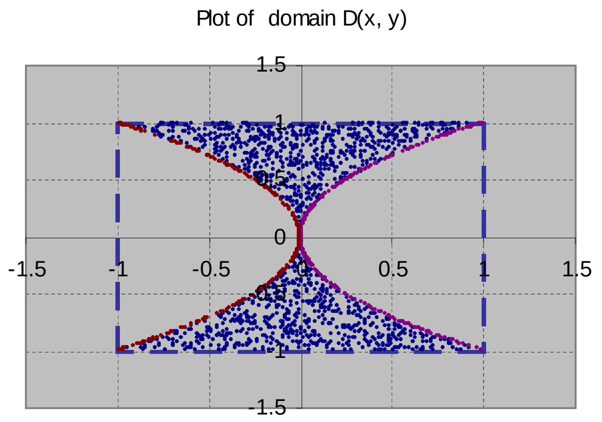


Approx. integral	1.237463088347320
True Integral	1.237463088347400
True Error	-7.927E-14
Error relative.	-6.406E-14
Estimate. Error rel.	4.306E-15
Time (sec)	0.6914
Points	66049

The integration function is the same of example 3, but the boundary limits are different.

Example 5:

$$\iint_D f(x,y)ds = \int_{-1}^{1}\int_{-x^2}^{1-x^2} e^{-(x^2+y^2)}dy\,dx$$	$-\frac{64}{105}$

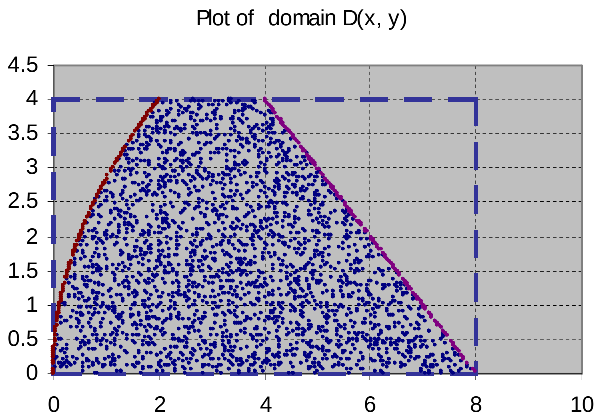


Approx. integral	-0.609523809523816
True Integral	-0.609523809523809
True Error	-7.216E-15
Error relative.	1.184E-14
Estimate. Error rel.	0.000E+00
Time (sec)	0.5508
Points	4225

This domain is normal to y-axis.

Example 6:

$$\iint_D f(x,y)ds = \int_{0}^{4}\int_{\frac{1}{8}y^2}^{8-y} (2x-3y-10)\,dxdy$$	$-\frac{856}{5}$



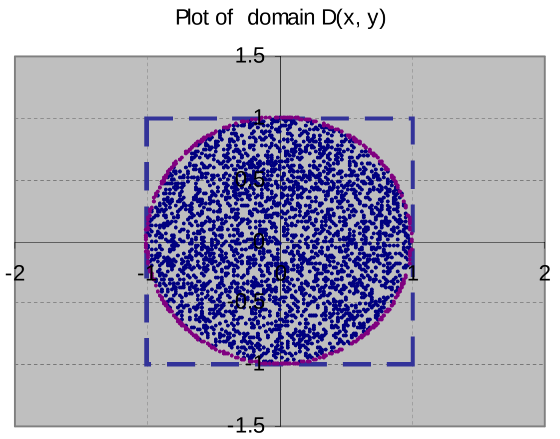
Approx. integral	-171.2
True Integral	-171.2
True Error	0
Error relative.	0
Estimate. Error rel.	0
Time (sec)	0.078125
Points	289

This domain is normal to y-axis.

A circular domain could be also view as a normal domain to both axes. Conceptually speaking there is no difference. On the contrary, there is a great difference for numerical calculus. Let's see this example

Example 7:

$\iint_D f(x,y)ds = \int_{-1}^{1} \int_{-\sqrt{1-x^2}}^{\sqrt{1-x^2}} \left(\frac{5-x^2-2y^2}{p} \right) dydx$	$\frac{17}{4}$



Approx. integral	4.24965898
True Integral	4.25
True Error	-0.00034102
Error relative.	-8.024E-05
Estimate. Error rel.	2.23925E-09
Time (sec)	2.296875
Points	263169

Circular domain
Even with more that 200.000 points, the error is about 3E-4. Note that the limit functions derivatives are discontinue at the point -1 and 1. When this happens, the integration error increases sharply. This can be avoid in polar coordinates

Here is shown the worksheet setting for obtaining the above results:

F(x,y)	Xmin	Xmax	Ymin	Ymax	k max	ErrRel max	Polar
(5-x^2-2y^2)/pi	-1	1	-sqr(1-x^2)	sqr(1-x^2)	9	1.00E-14	FALSO
integral	time	points	Estim. Error rel.	True Error	True Error rel.	True Integr.	
4.24965898	2.296875	263,169	2.23925E-09	-0.00034102	-8.024E-05	4.25	

If we pass in polar coordinates, the circular domain becomes a rectangular domain

$$\iint_D f(x,y)dxdy = \int_0^{2p} \int_0^{1} g(r,q) r \; dr \; dq$$

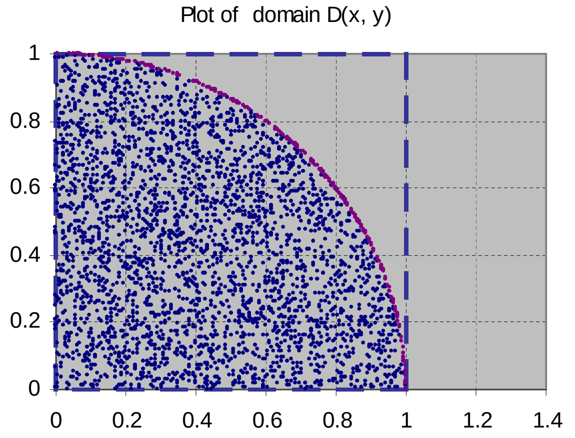
You do not need to transform the integration function f(x,y) into the polar form g(ρ, θ), because the routine **Integral_2D_N** does this for us. We have only to set the correct limits and turn on the *Polar* flag.

F(x,y)	Xmin	Xmax	Ymin	Ymax	k max	ErrRel max	Polar
(5-x^2-2y^2)/pi	0	1	0	2*pi	9	1.00E-14	VERO
integral	time	points	Estim. Error rel.	True Error	True Error rel.	True Integr.	
4.25	0.640625	66,049	8.15034E-15	1.35891E-13	3.19744E-14	4.25	

As we can see the accuracy has increased with an effort reduction: 1E-13 with 66.000 points

Example 8:

$\iint_D f(x,y)ds = \int_0^1 \int_0^{\sqrt{1-x^2}} \frac{1}{p}\left(\frac{9}{2}-x^2-y^2\right)dydx$	1



Approx. integral	0.99997363
True Integral	1
True Error	-2.63704E-05
Error relative.	-2.63704E-05
Estimate. Error rel.	7.35676E-10
Time (sec)	2.171875
Points	263169

Circular domain
Even with more that 200.000 points, the error is about 3E-4.
Note that the limit functions derivatives is discontinue at x = 1.
When this happens, the integration error increases sharply. This can be avoid in polar coordinates

To avoid accuracy decreasing due to discontinue derivative at x =1 of the upper bound function, we can pass to the polar coordinates. In this case the limit are: $0 \le \rho \le 1$ and $0 \le \theta \le \pi/2$

Repeating the calculus, setting the *Polar* flag, we have

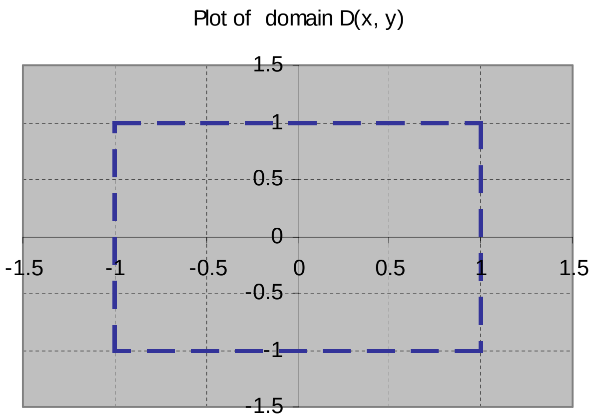
Approx. integral	1.000000000000000
True Integral	1
True Error	0.000E+00
Error relative.	0.000E+00
Estimate. Error rel.	0.000E+00
Time (sec)	0.0703
Points	81

In only 80 points we have the highest accuracy integral (note that in numeric calculus, error = 0 means: less that 1E-16)! Cleary there are good reasons for polar transformation.

Sometimes the derivative of integration function having singular point into integration domain, reduce the accuracy of numerical computing

Example 9:

$\iint_D f(x,y)ds = \int_{-1}^{1}\int_{-1}^{1}\sqrt{\left(2-x^2-y^2\right)}\,dydx$	$\frac{\pi}{3}\left(10-4\sqrt{2}\right)$



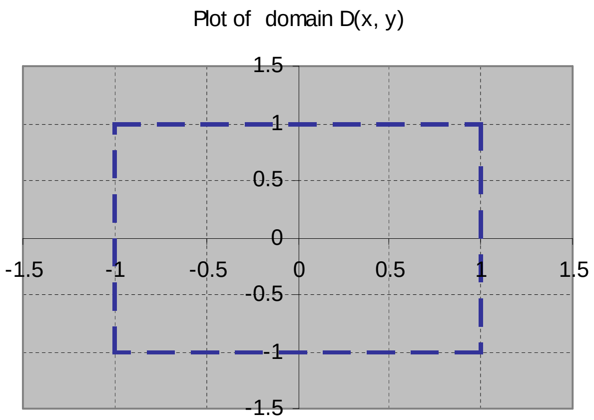
Approx. integral	4.548131859773910
True Integral	4.548131594421480
True Error	2.654E-07
Error relative.	5.834E-08
Estimate. Error rel.	4.102E-12
Time (sec)	2.0781
Points	263169

The approximation is sufficiently accurate (3E-7) but the algorithm has used more than 200,000 points to obtain this result. The difficulty is due to discontinues of the integration function derivative at the four corners (1, 1), (1, -1), (-1 , 1), (-1, -1)

Integration difficulties - from the point of view of the numeric calculus - arise when the integration function derivatives have singular points into the region of integration. In the above example the integration function derivatives have both singularities at the corner of the rectangle.
If we try to integrate a similar function, but with singular points out of the integration region, the result will more accurate with lower effort.

Example 10:

$\iint_D f(x,y)ds = \int_{-1}^{1}\int_{-1}^{1}\sqrt{\left(3-x^2-y^2\right)}\,dydx$	$\frac{4+8\pi-2\pi\sqrt{3}}{3}$



Approx. integral	6.083315014437720
True Integral	6.083315014437670
True Error	5.151E-14
Error relative.	8.468E-15
Estimate. Error rel.	2.920E-16
Time (sec)	0.5781
Points	66049

Derivatives discontinues are at the √3 distance from the origin. So, in the integration region the function and its first derivatives are smoothly. This explain the high accuracy

As expected, the accuracy is 10 billions times higher than the previous example, obtained with a quarter of the effort.