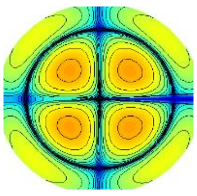


DOING PHYSICS WITH PYTHON

NUMERICAL INTEGRATION [1D] AND [2D] INTEGRALS

Ian Cooper
matlabvisualphysics@gmail.com

DOWNLOAD DIRECTORIES FOR PYTHON CODE

Google drive

GitHub

op001.py   op002.py   op003.py

Computation of the integral of functions of the form *f*(*x*) and *f*(*x*,*y*) using Simpson's 1/3 rule. The code can be changed into a function.

COMPUTATION OF ONE-DIMENSIONAL INTEGRALS

We want to compute a number expressing the definite integral of the function *f*(*x*) between two specific limits *a* and *b*

(1) $$I = \int_a^b f(x)\,\mathrm{d}x$$

The evaluation of such integrals is often called ***quadrature***.

In Python we can consider two approaches to evaluate a definite integral (equation 1). The library **from scipy import integrate** is required. Integration of a function using the functions

1. **quad**
2. **simps** for sampled data using Simpson's rule.

Steps:

- Define the function and its limits
- Use the function quad or simps

To illustrate how to compute the integral a number of examples will be given. Also, it is often a good idea to plot the graph of the *f*(x).

Example 1 op003.py

$$I = \int_0^{\pi/2} \cos(x)\,\mathrm{d}x \qquad f(x) = \cos(x) \quad a = 0 \quad b = \pi/2$$

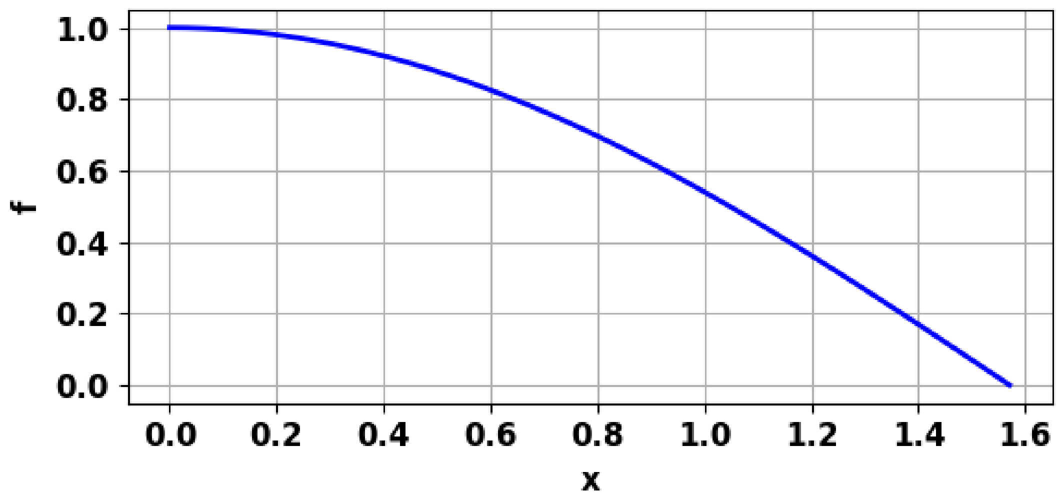
$$I_{exact} = 1.000000\ldots$$

Using quad function returns integral value and error estimate

I_{quad} = 0.9999999999999999 1.1102230246251564e-14

Using simps function

I_{simps} = 1.000000000004289 number of sample N = 299



Plot: cos(x) for x = 0 to x = $\pi / 2$.

Using Simpson's rule: A major problem that arises with non-adaptive methods is that the number *N* of partitions of the function required to provide a given accuracy is initially unknown. One approach to this problem is to successively double the number of partitions, and compare the results as the number of partitions increase.

Python code **op003.py**

```
import numpy as np
from numpy import pi, sin, cos, exp, linspace
from numpy.linalg import eig
from scipy.integrate import odeint, quad, dblquad, simps
from scipy import pi, sqrt
import matplotlib.pyplot as plt
import time
from mpl_toolkits.mplot3d import axes3d
import sympy as sym
```

```
def func(x):
    f = cos(x)
    return f
# >>>>> Inputs: Grid points, limits
N = 299
a = 0; b = pi/2
# function f(x)
x =linspace(a,b,N)
f = func(x)
# Integrate function uisng quad
Iquad, Ierr = quad(func,a,b)
print(Iquad,Ierr)
# Integrate function using simps
Is = simps(f,x)
print(Is)
# Plot function
plt.rcParams['font.size'] = 12
plt.rcParams["figure.figsize"] = (6,3)

# Figure 1    t vs x
fig1, axes = plt.subplots(nrows=1, ncols=1)
axes.set_xlabel('x',color= 'black',fontsize = 12)
axes.set_ylabel('f',color = 'black',fontsize = 12)
axes.xaxis.grid()
axes.yaxis.grid()
xP = x; yP = f
axes.plot(xP, yP,'b',lw = 2)
fig1.tight_layout()
# fig1.savefig('a1.png')
```

Example 2   op003.py

$$I = \int_0^{\pi/2} e^{2+jx}\,\mathrm{d}x \qquad f(x) = e^{2+jx} \qquad a = 0 \quad b = \pi/2 \qquad j = \sqrt{-1}$$

I_{exact} = 7.38905609893065 + 7.389056098930306495j

Using quad function returns integral value and error estimate

I_{quad} = 7.38905609893065 8.203500211279747e-14

quad function only returns real part

Using simps function (number of sample N = 299)

I_{simps} = 7.389056098962341+7.389056098962339j

SYMBOLIC [1D] INTEGRATION

Integrals can be evaluated symbolically. You need the library **import sympy as sym** to use any of the symbolic tools.

A good reference is

https://scipy-lectures.org/packages/sympy.html#integration

Code from **op003.py**

```
#%%  Symoblic integration: select function by removing #
x = sym.Symbol('x')
y = sym.Symbol('y')

# y = 6 * x ** 5
# y = sym.sin(x)
# y = sym.log(x)
# y = 2 * x + sym.sinh(x)
sym.integrate(y, x)

# Highlight the code and use F9 to execute the code
# It is possible to compute definite integral:
# y = x**4;  a = -1; b = 1; sym.integrate(y, (x, a, b))
# sym.integrate(sym.sin(x), (x, 0, sym.pi))
# sym.integrate(sym.cos(x), (x, -sym.pi / 2, sym.pi / 2))
# Also improper integrals are supported as well:
# sym.integrate(sym.exp(-x), (x, 0, sym.oo))
# sym.integrate(sym.exp(-x ** 2), (x, -sym.oo, sym.oo))
```

[2D], surface, double, integration

$$I = \iint_A f(x,y)\,dA \qquad I = \int_{a_y}^{b_y} \int_{a_x}^{b_x} f(x,y)\,dx\,dy$$

We want to compute the value of a definite integral of the function *f*(*x*,*y*) between two specific limits (a_x, b_x) and (a_y, b_y)

$$I = \int_{a_y}^{b_y} \int_{a_x}^{b_x} f(x,y)\,dx\,dy$$

A very practical and versatile way to compute [2D] integrals is using a [2D] form of Simpson's 1/3 rule.

Simpson's 1/3 rule

This rule is based on using a quadratic polynomial approximation to the function *f*(*x*) over a pair of partitions. *N*-1 is the number of partitions where *N* must be **odd** and $\Delta x \equiv h = (b - a) / (N\text{-}1)$. The integral is expressed below and is known as the ***composite Simpson's 1/3 rule***.

$$I = \frac{h}{3}\left\{\left(f_1 + f_N + 4(f_2 + f_4 + \ldots + f_{N-2}\right) + 2(f_3 + f_5 + \ldots + f_{N-1})\right\}$$

Simpson's rule can be written vector form as

$$I = \frac{h}{3}\mathbf{c}\,\mathbf{f}^{\mathrm{T}}$$

where $\mathbf{c} = [1\;4\;2\;4 \ldots 2\;4\;1]$ and $\mathbf{f} = [\,f_1\;\; f_2\; \ldots\; f_N\,]$.

c and **f** are row vectors and $\mathbf{f}^{\mathbf{T}}$ is a column vector.

Simpson's [2D] method

The double integral

$$I = \int_{a_y}^{b_y} \int_{a_x}^{b_x} f(x, y)\, dx\, dy$$

can be approximated by applying Simpson's 1/3 rule twice – once for the *x* integration and once for the *y* integration with *N* partitions for both the *x* and *y* values.

x-values: $x_1\ x_2\ x_3 \cdots x_c \cdots x_N$

y-values: $y_1\ y_2\ y_3 \cdots y_c \cdots y_N$

The lower and upper bounds determine the size of the partitions

$$dx \equiv h_x = \frac{b_x - a_x}{N - 1} \qquad dy \equiv h_y = \frac{b_y - a_y}{N - 1}$$

The *N* *x*-values and *N* *y*-values form a two-dimensional grid of *N* x *N* points. The function *f*(*x*,*y*) and the two-dimensional Simpson's coefficients are calculated at each grid point. Hence, the function *f*(*x*,*y*) and the two-dimensional Simpson's coefficients can be represented by *N* x *N* matrices **F** and **S** respectively.

The Simpson matrix **S** for *N* = 5 is

1 x 1 = 1	4 x 1 = 4	2x1 = 2	4x1 = 4	1x1 = 1
1 x 4 = 4	4 x 4 = 16	2x4 = 4	4x4 = 16	1x4 = 1
1 x 2 = 8	4 x 2 = 8	2x2 = 4	4x2= 8	1x2 = 1
1 x 4 = 4	4 x 4 = 16	2x4 = 4	4x4 = 16	1x4 = 1
1 x 1 = 1	4 x 1 = 4	2x1 = 2	4x1 = 4	1x1 = 1

1	4	2	4	2	4	2	4	1
4	16	8	16	8	16	8	16	4
2	8	4	8	4	8	4	8	2
4	16	8	16	8	16	8	16	4
2	8	4	8	4	8	4	8	2
4	16	8	16	8	16	8	16	4
2	8	4	8	4	8	4	8	2
4	16	8	16	8	16	8	16	4
1	4	2	4	2	4	2	4	1

Therefore, the **two-dimensional Simpson's rule** which is used to estimate the value of the surface integral can be expressed as

$$I = \left(\frac{h_x h_y}{9}\right)\sum_{m=1}^{N}\sum_{n=1}^{N}\left(\mathrm{F}_{mn}\mathrm{S}_{mn}\right)$$

We include a mask matrix **M** (*N*x*N*) which has elements equal to 1 or 0 only. The mask matrix **M** is used where the integration is not over a rectangular area defined by the limits a_x, b_x, a_y and b_y. The integral then becomes

$$I = \left(\frac{h_x h_y}{9}\right) \sum_{m=1}^{N} \sum_{n=1}^{N} \left(\mathrm{M}_{mn}\, \mathrm{F}_{mn}\, \mathrm{S}_{mn}\right)$$

Required steps using the Simpson [2D] method are:

- Define the matrix **S** for the [2D] Simpson coefficients.
- Define the x and y ranges.
- Define the xy mesh for xx and yy using the meshgrid function.
- Calculate the matrix function **F**.
- Calculate the mask matrix **M**.
- Redefine the matrix **F**

  F = M F
- Sum all the elements of the matrix **F S** and multiply by ($h_x\ h_y$ / 9) to give the value of the integral.

We will consider a number of examples which demonstrates how to apply the two-dimensional Simpson's rule using the code **op001.py**

Example 3 op001.py

Integrate $f(x,y) = x^2\, y^3$

x: 0 → 2 and y: 1 → 5

$$I_{xy1} = \int_{a_y}^{b_y} \int_{a_x}^{b_x} f(x,y)\,dx\,dy = \int_1^5 \left[\int_0^2 x^2\, y^3\, dx \right] dy = 416$$

The exact value of the integral can be found analytically and its value is **416**. So, we can compare the numerical estimate with the known exact value.

This is a two-dimensional problem, so we need to specify the values (x,y) at all grid points which are determined from the upper and lower bounds. We can do this using the Python function **meshgrid** to calculate the value of the function $f(x,y)$ at each grid point (x,y).

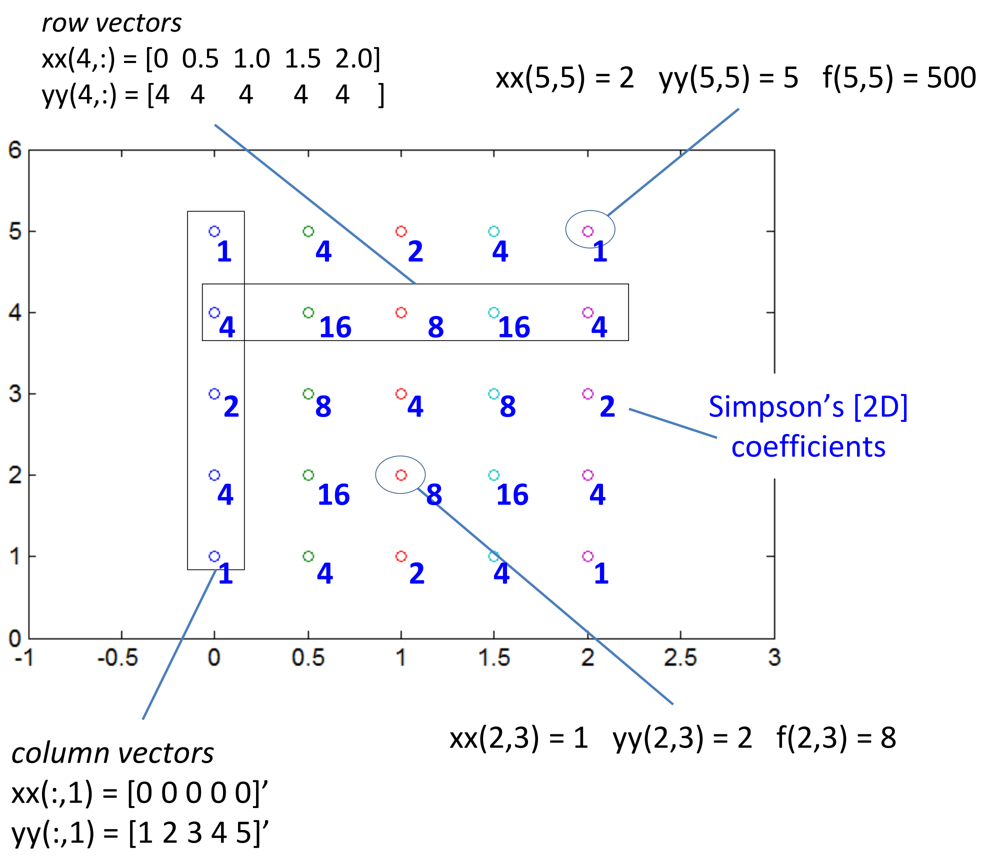
```
ax = 0; bx = 2; ay = 1; by = 5
x = linspace(ax,bx,N)
y = linspace(ay,by,N,N)
xx, yy = np.meshgrid(x,y)
f = xx**2*yy**3
```

[2D] space

To show how the **meshgrid** functions works, see figure (1) and the outputs of the variables *x*, *y*, *xx*, *yy* and *f* that can be displayed in the Console Window.

```
x =   0    0.5000    1.0000    1.5000    2.0000

xx =

        0    0.5000    1.0000    1.5000    2.0000

        0    0.5000    1.0000    1.5000    2.0000

        0    0.5000    1.0000    1.5000    2.0000

        0    0.5000    1.0000    1.5000    2.0000

        0    0.5000    1.0000    1.5000    2.0000


y =   1     2     3     4     5

yy =

      1     1     1     1     1

      2     2     2     2     2

      3     3     3     3     3

      4     4     4     4     4

      5     5     5     5     5


f =

      0     0.2500       1.0000        2.2500       4.0000

      0     2.0000       8.0000      18.0000     32.0000

      0    6.7500     27.0000       60.7500    108.0000

      0   16.0000    64.0000    144.0000    256.0000

      0   31.2500  125.0000    281.2500     500.0000
```

Simpson coefficients



The grid points for $N = 5$ and how these points relate to the Matlab matrices.

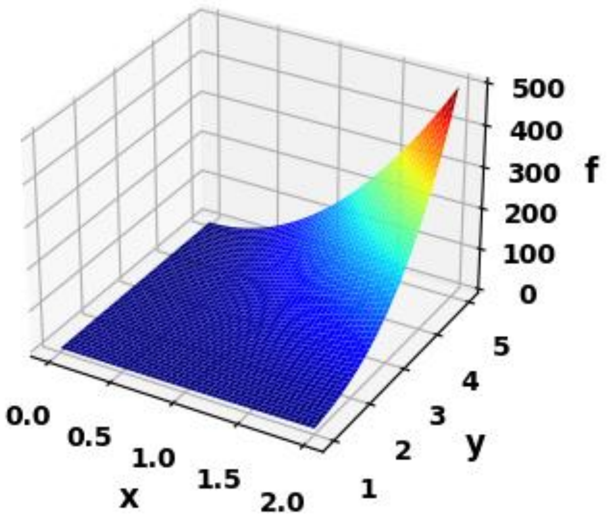
Calculation of [2D] Simpson coefficients

```
sc = np.ones(N)
R = np.arange(1,N,2);   sc[R] = 4;
R = np.arange(2,N-1,2); sc[R] = 2
scx, scy = np.meshgrid(sc,sc)
sc2D = scx*scy
```

Compute the integral

```
hx = (bx-ax)/(N-1); hy = (by-ay)/(N-1)
h = hx * hy / 9
integral = h*sum(sum(sc2D*f))
```

With only 5 partitions and 25 (5x5) grid points, the numerical estimate is **416.0**, the same as the exact value.



Example 3   Double Integrals and Volumes   op001.py

$$Volume = \iint_A f(x,y)\,dx\,dy$$

To gain an intuitive feel for double integrals, the volume of the region enclosed by the area A is equal to the value of the double integral.

Volume V of a rectangular box

$f(x,y) = k$ height of box $k > 0$

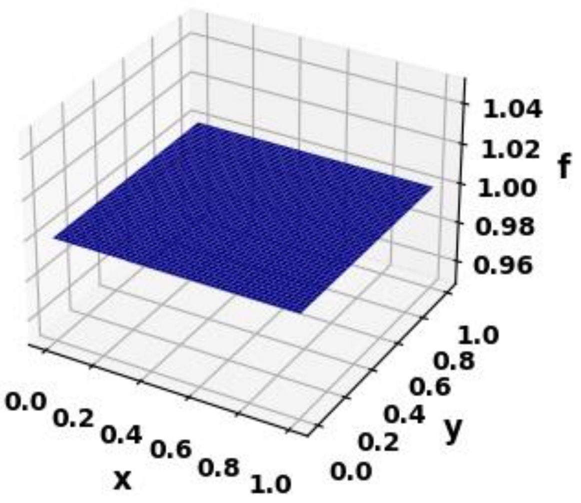
Base of box – the lower bounds (a_x and a_y) and upper bounds (b_x and b_y) determine the area of the rectangular base of the box

$$Volume\ \ V = \int_{a_y}^{b_y}\int_{a_x}^{b_x} k\,dx\,dy$$

Box $k = 1$ $a_x = 0$ $b_x = 1$ $a_y = 0$ $b_y = 1$ $N = 299$

Exact volume (analytical) $V = 1.0000$

Simpson's [2D] rule $V = 1.0000$



Volume *V* of half box

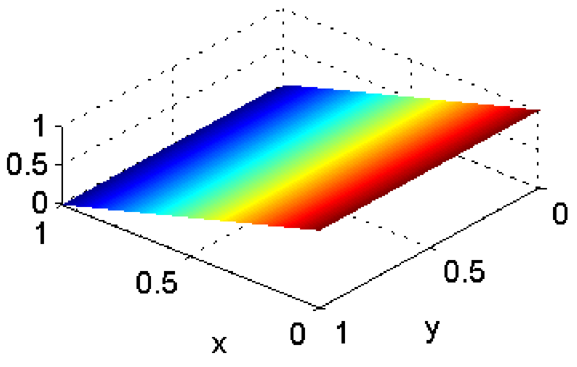
$f(x,y) = 1 - x$

Base of box – the lower bounds (a_x and a_y) and upper bounds (b_x and b_y) determine the area of the rectangular base of the box

$$\textit{Volume} \quad V = \int_{a_y}^{b_y} \int_{a_x}^{b_x} (1-x)\, dx\, dy$$

$a_x = 0 \quad b_x = 1 \quad a_y = 0 \quad b_y = 1$

$N = 299$



Exact volume (analytical) $V =$ 0.50000

Simpson's [2D] rule $V = 0.50000$

Volume *V* of a part-bowl

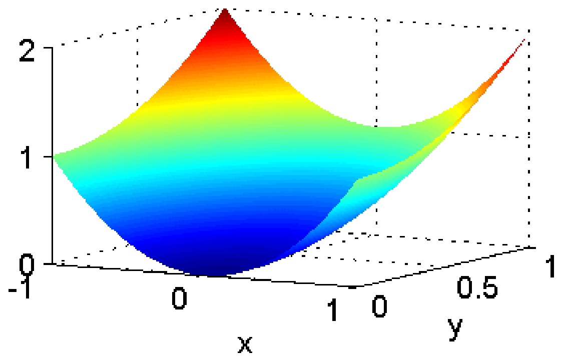
$f(x,y) = x^2 + y^2$

Base of box – the lower bounds (a_x and a_y) and upper bounds (b_x and b_y) determine the area of the rectangular base of the surface

$$Volume \quad V = \int_{a_y}^{b_y} \int_{a_x}^{b_x} \left(x^2 + y^2\right) dx\, dy$$

$a_x = -1$ $b_x = 1$ $a_y = 0$ $b_y = 1$

$N = 299$



Exact volume (analytical) $V =$ 1.3333

Simpson's [2D] rule $V = 1.3333$

Volume *V* over a rectangular base

$$f(x,y) = \cos(x)\ \sin(y)$$

Base of box – the lower bounds (a_x and a_y) and upper bounds (b_x and b_y) determine the area of the rectangular base of the surface

$$Volume \quad V = \int_{a_y}^{b_y} \int_{a_x}^{b_x} \left(\cos(x)\ \sin(y)\right) dx\, dy$$

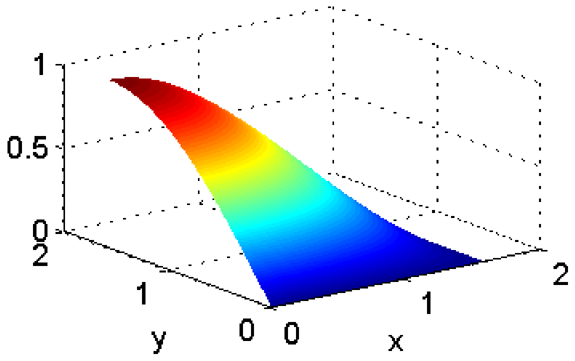
$a_x = 0$ $b_x = \pi/2$ $a_y = 0$ $b_y = \pi/2$ $N = 299$

Exact volume (analytical)

$V = 1.000$

Simpson's [2D] rule

$V = 1.000000000008578$



Volume *of* a hemisphere using Cartesian coordinates

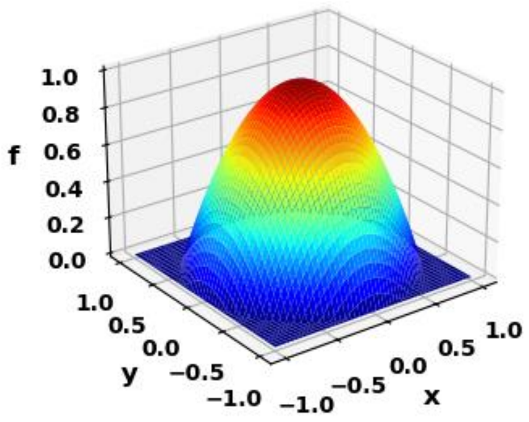
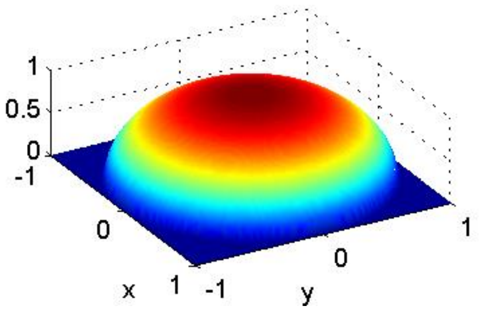
Volume of a hemisphere of radius a $\quad V = \dfrac{2\pi a^3}{3}$

Function

$x^2 + y^2 < a^2 \quad f(x,y) = \sqrt{a^2 - x^2 - y^2} \qquad x^2 + y^2 \geq a^2 \quad f(x,y) = 0$

$$Volume \quad V = \int_{a_y}^{b_y}\int_{a_x}^{b_x}\left(\sqrt{a^2 - x^2 - y^2}\right)dx\,dy$$

$a_x = -1 \quad b_x = 1 \quad a_y = -1 \quad b_y = 1 \quad a = 1$

 Python plot	 Matlab plot is better than Python plot.

Exact volume (analytical)

$V = 2.094395102393195$

Simpson's [2D] rule

$N = 99$ $V = 2.094$**417986583109**

N = 999 V = 2.094395**646847362**

We must have

$$x^2 + y^2 < a^2 \quad f(x, y) = \sqrt{a^2 - x^2 - y^2} \qquad x^2 + y^2 \geq a^2 \quad f(x, y) = 0$$

The code to define the function and mask matrix:

```
f = np.real(a**2 - xx**2 - yy**2)

# Mask matrix M

M = np.ones([N,N])

M[xx**2+yy**2>1]=0
```

Volume *V* over a triangular base

$$y \le 1 - x \quad f(x,y) = h \qquad y > 1 - x \quad f(x,y) = 0$$

$$Volume \quad V = \int_{a_y}^{b_y} \int_{a_x}^{b_x} h \, dx \, dy$$

$a_x = 0$ $b_x = 1$ $a_y = 0$ $b_y = 1$ height $h = 6$

Exact volume (analytical)

$V = 3.0000$

Simpson's [2D] rule

$N = 299$ $V = 3.00$**6673873549240**

$N = 999$ $V = 3.00$**1944436635462**

$N = 999$ $V = 3.00$**1944436635462**

$N = 2999$ $V = 3.000$**632027607761**

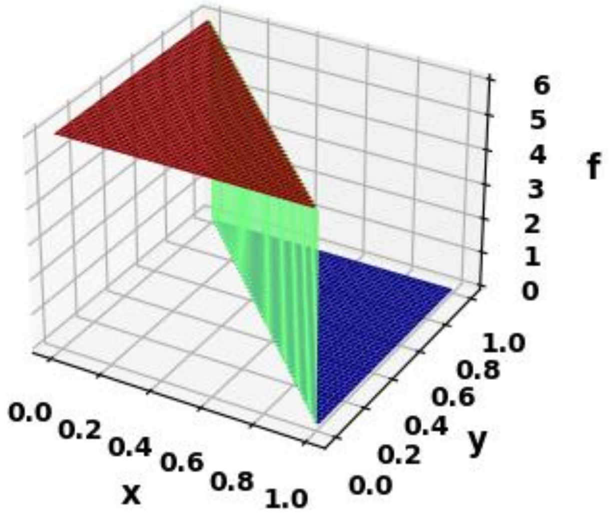
Even with $N = 2999$ the calculation took less than 1.0 s on a fast Windows computer.

The differences between the exact and computed values is due to the rectangular grid and the condition on the function being zero when $y > 1 - x$

($y = 1 - x$ is a diagonal line and the grid is rectangular).

A logical Python function is used to define the function when $y > 1 - x$. The code to define the function is

```
f = 6*xx**0*yy**0
f[yy > 1 - xx] = 0
```



Double Integrals In Polar Coordinates

https://tutorial.math.lamar.edu/classes/calciii/dipolarcoords.aspx

Double integrals in polar coordinates can be expressed as

$$\iint_D f(x,y)\,dx\,dy \equiv \iint_D f(x,y)\,dA \equiv \int_{q_1}^{q_2}\int_{p_1}^{p_2} f\left(p\cos(q), p\sin(q)\right) p\,dp\,dq$$

where the cartesian and polar coordinates are

$$x = p\cos(q) \quad y = p\sin(q) \quad p^2 = x^2 + y^2$$

Example 4 area of circle of radius *a* op002.py

$p_1 = 0$ $p_2 = 2$ $q_1 = 0$ $q_2 = 2\pi$ a = 1 $N = 999$

$$A = \int_0^{2\pi}\int_0^1 p\,dp\,dq$$

Exact area A = π = 3.141592653589793

Simpson’s [2D] rule A = 3.141592653589793

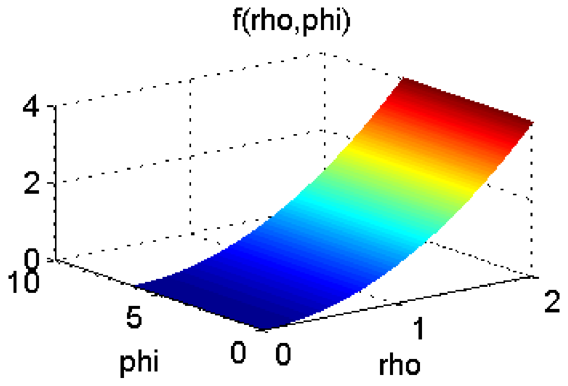
Volume of a hemisphere of radius *a*   op002.py

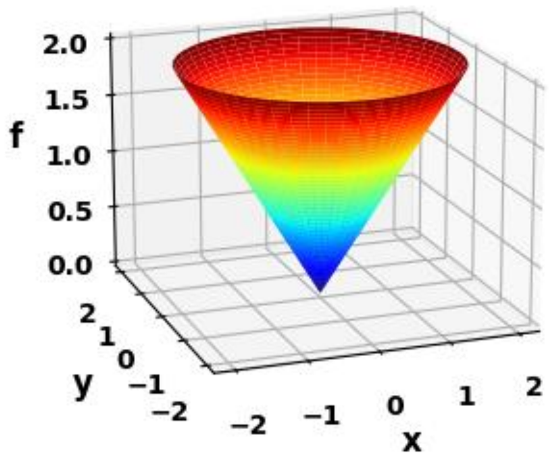
$$f(x,y) = \rho^2 \qquad V = \int_0^{2\pi}\int_0^{a} \rho^2 \, d\rho \, d\phi$$

$a_x = 0 \;\; b_x = 2 \;\; a_y = 0 \;\; b_y = 2\pi \;\; N = 299$

Exact volume (analytical) $V = 16.7551608191455$**62**

Simpson's [2D] rule $V = 16.7551608191455$55





Python code op002.py

```
# Libraries
import numpy as np
from numpy import pi, sin, cos, linspace
from numpy.linalg import eig
from scipy.integrate import odeint
import matplotlib.pyplot as plt
import time
from mpl_toolkits.mplot3d import axes3d

tStart = time.time()

# >>>>> Input number of grid points: N must be odd
N = 999
# p rho /  q phi
# >>>>> Setup RP meshgrid
p1 = 0;  p2 = 2; q1 = 0; q2 = 2*pi
p = linspace(p1,p2,N)
q = linspace(q1,q2,N)
pp, qq = np.meshgrid(p,q)

# Mask matrix M
M = np.ones([N,N])

# Function
f = pp

# Simpson [2D] coefficients
S = np.ones(N)
R = np.arange(1,N,2);   S[R] = 4;
```

```
R = np.arange(2,N-1,2); S[R] = 2
scx, scy = np.meshgrid(S,S)
S = scx*scy

# Calculate integral
hp = (p2-p1)/(N-1); hq = (q2-q1)/(N-1)
h = hp * hq / 9
integral = h*sum(sum(pp*f*S))

print('\n Integral  =  ',integral)

xx = pp*cos(qq); yy = pp*sin(qq)

#%% GRAPHICS
plt.rcParams['font.size'] = 10
fig = plt.figure(figsize=(4,3))
ax = plt.axes(projection='3d')
ax.plot_surface(xx, yy, f, cmap='jet',
    edgecolor='none', alpha=1,antialiased=True)
ax.set_xlabel('x', fontsize=12)
ax.set_ylabel('y', fontsize=12)
ax.set_zlabel('f', fontsize=12)
fig.tight_layout()
ax.view_init(17,-110,0)
# fig.savefig('a1.png')

#%%
tExe = time.time() - tStart
print('  ')
print('Execution time')
print(tExe)
```