Universidad Nacional San Agustin de Arequipa

Facultad de Ingenierias de Produccion y Servicios

Escuela Profesional de Ingenieria de Sistemas

Fisica Computacional

Alumno:

Fuentes Paredes Nelson Alejandro

Mayo 2020

[ ]:

```

```

[1]:

```
%matplotlib notebook
%matplotlib inline
```

1 Importar Librerias

[2]:

```
import numpy as np
import random
from matplotlib import pyplot as plt
import math
from mpl_toolkits import mplot3d
```

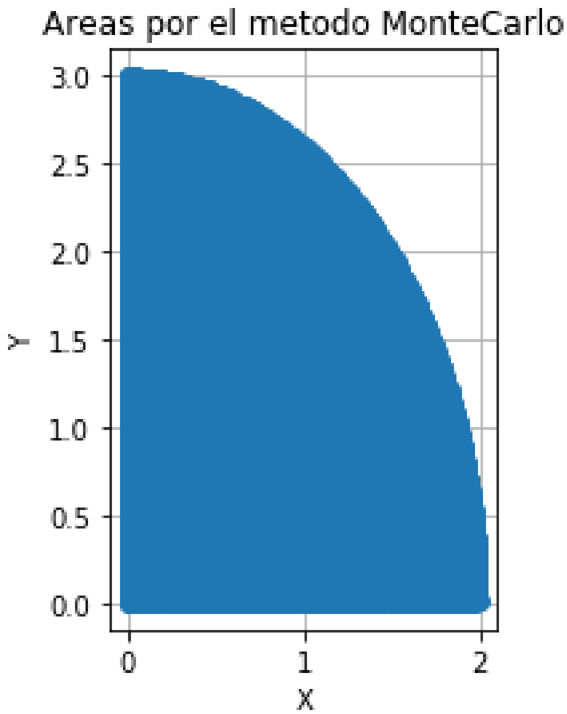
2 Ejercicios

2.1 Evalúe el área de la elipse en el primer cuadrante del programa anterior

[3]:

```
m = 1000
veces = 100
a = np.array([0,0])
b = np.array([2,3])
a_b = b-a
ps = []
sa = 0
saa = 0
for k in range(veces):
    n = 0
    for i in range(m):
        p = np.array([random.random()*a_b[0] + a[0],  random.random()*a_b[1] +␣
 ↪a[1]])
        if p[0]**2/4+p[1]**2/9<1:
            n = n+1
            ps.append(p)
    area = n/m * a_b[0]*a_b[1]
    sa = sa + area
    saa = saa + area**2
prom = sa/veces
desv = math.sqrt(veces*saa-sa**2)/veces
fig, ax = plt.subplots()
ax.set_aspect('equal')
ax.grid()
ax.plot([ p[0] for p in ps], [ p[1] for p in ps], 'o')
ax.set(title='Areas por el metodo MonteCarlo', xlabel = 'X', ylabel='Y')
print('Area\t\t:\t{:.2}\nDesviacion\t:\t{:.2}'.format(prom, desv))
```

```
Area            :      4.7
Desviacion      :      0.078
```



2.2 Evalúe las siguientes integrales

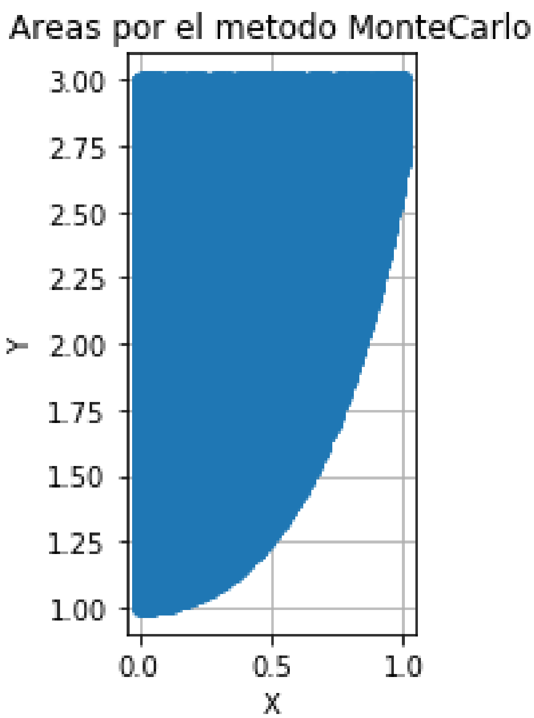
a) $= \int_0^1 e^{x^2} dx$

[4]:

```python
m = 10000
veces = 10
a = np.array([0,0])
b = np.array([1,3])
a_b = b-a
ps = []
sa = 0
saa = 0
for k in range(veces):
    n = 0
    for i in range(m):
        p = np.array([random.random()*a_b[0] + a[0],  random.random()*a_b[1] +
 ↪a[1]])
        if math.exp(p[0]**2)- p[1]<0:
            n = n+1
            ps.append(p)
    area = n/m * a_b[0]*a_b[1]
```

```
    sa = sa + area
    saa = saa + area**2
prom = sa/veces
desv = math.sqrt(veces*saa-sa**2)/veces
fig, ax = plt.subplots()
ax.set_aspect('equal')
ax.grid()
ax.plot([ p[0] for p in ps], [ p[1] for p in ps], 'o')
ax.set(title='Areas por el metodo MonteCarlo', xlabel = 'X', ylabel='Y')
print('Area\t\t:\t{:.2}\nDesviacion\t:\t{:.2}'.format(prom, desv))
```

```
Area            :       1.5
Desviacion      :       0.016
```

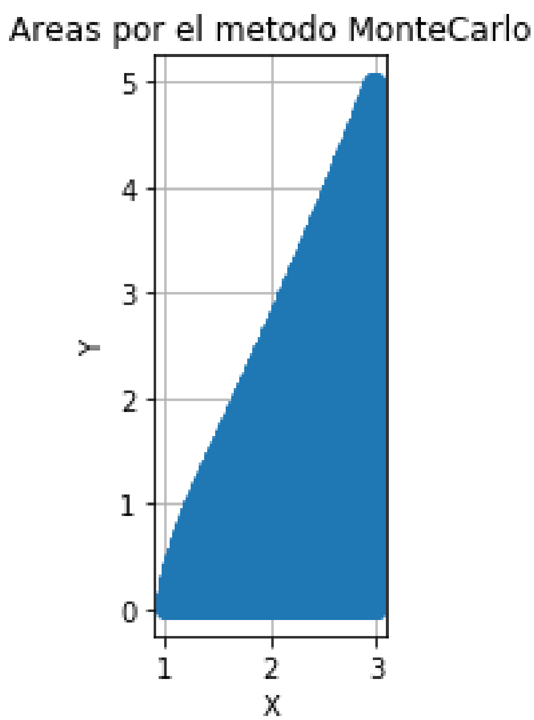


b) $= \int_1^3 \sqrt{x^3-1}dx$

[5]:
```
m = 10000
veces = 10
a = np.array([1,0])
b = np.array([3,5])
a_b = b-a
ps = []
sa = 0
saa = 0
```

```
for k in range(veces):
    n = 0
    for i in range(m):
        p = np.array([random.random()*a_b[0] + a[0],  random.random()*a_b[1] +␣
 ↪a[1]])
        if math.sqrt(p[0]**3-1)- p[1]>0:
            n = n+1
            ps.append(p)
    area = n/m * a_b[0]*a_b[1]
    sa = sa + area
    saa = saa + area**2
prom = sa/veces
desv = math.sqrt(veces*saa-sa**2)/veces
fig, ax = plt.subplots()
ax.set_aspect('equal')
ax.grid()
ax.plot([ p[0] for p in ps], [ p[1] for p in ps], 'o')
ax.set(title='Areas por el metodo MonteCarlo', xlabel = 'X', ylabel='Y')
print('Area\t\t:\t{:.2}\nDesviacion\t:\t{:.2}'.format(prom, desv))
```

```
Area            :       5.4
Desviacion      :       0.041
```



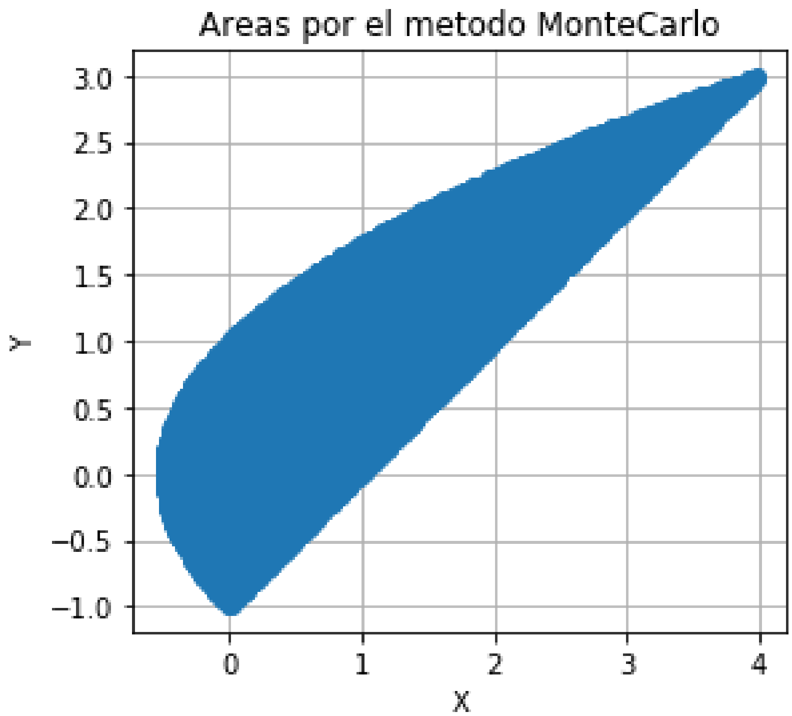
2.3 Calcule el área de la figura limitada por las líneas cuyas ecuaciones son

$y^2 = 2x + 1$

$x - y - 1 = 0$

[6]:
```
m = 10000
veces = 10
a = np.array([-1,-1])
b = np.array([4,3])
a_b = b-a
ps = []
sa = 0
saa = 0
for k in range(veces):
    n = 0
    for i in range(m):
        p = np.array([random.random()*a_b[0] + a[0],  random.random()*a_b[1] +␣
 ↪a[1]])
        if 2*p[0]+1- p[1]**2>0 and p[0]-1- p[1]<0:
            n = n+1
            ps.append(p)
    area = n/m * a_b[0]*a_b[1]
    sa = sa + area
    saa = saa + area**2
prom = sa/veces
desv = math.sqrt(veces*saa-sa**2)/veces
fig, ax = plt.subplots()
ax.set_aspect('equal')
ax.grid()
ax.plot([ p[0] for p in ps], [ p[1] for p in ps], 'o')
ax.set(title='Areas por el metodo MonteCarlo', xlabel = 'X', ylabel='Y')
print('Area\t\t:\t{:.2}\nDesviacion\t:\t{:.2}'.format(prom, desv))
```

```
Area            :       5.3
Desviacion      :       0.088
```



2.4 Calcule el área de la figura limitada por las parábolas

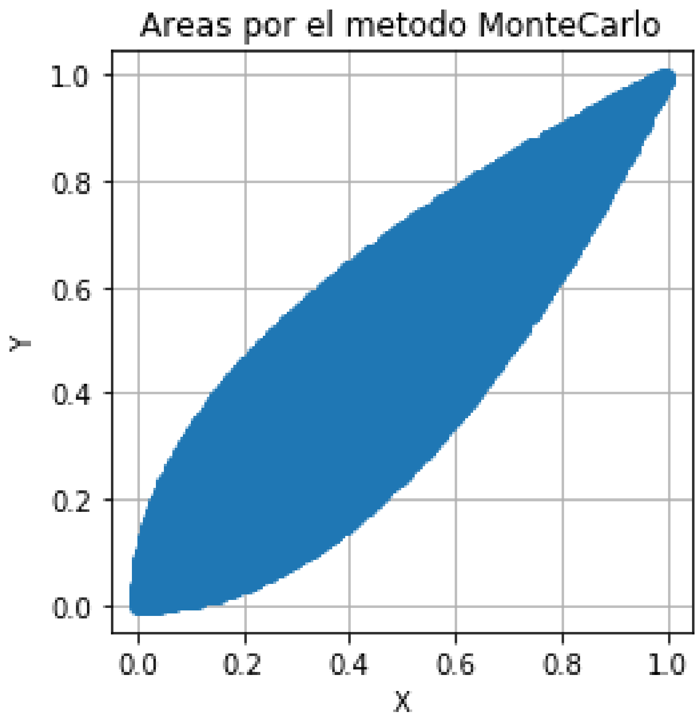
$y = x^2$

$y = \sqrt{x}$

[7]:
```
m = 10000
veces = 10
a = np.array([0,0])
b = np.array([1,1])
a_b = b-a
ps = []
sa = 0
saa = 0
for k in range(veces):
    n = 0
    for i in range(m):
        p = np.array([random.random()*a_b[0] + a[0],  random.random()*a_b[1] +
 ↪a[1]])
        if p[0]**2<p[1] and math.sqrt(p[0])>p[1]:
            n = n+1
            ps.append(p)
    area = n/m * a_b[0]*a_b[1]
    sa = sa + area
```

```
    saa = saa + area**2
prom = sa/veces
desv = math.sqrt(veces*saa-sa**2)/veces
fig, ax = plt.subplots()
ax.set_aspect('equal')
ax.grid()
ax.plot([ p[0] for p in ps], [ p[1] for p in ps], 'o')
ax.set(title='Areas por el metodo MonteCarlo', xlabel = 'X', ylabel='Y')
print('Area\t\t:\t{:.2}\nDesviacion\t:\t{:.2}'.format(prom, desv))
```

```
Area            :       0.33
Desviacion      :       0.0033
```



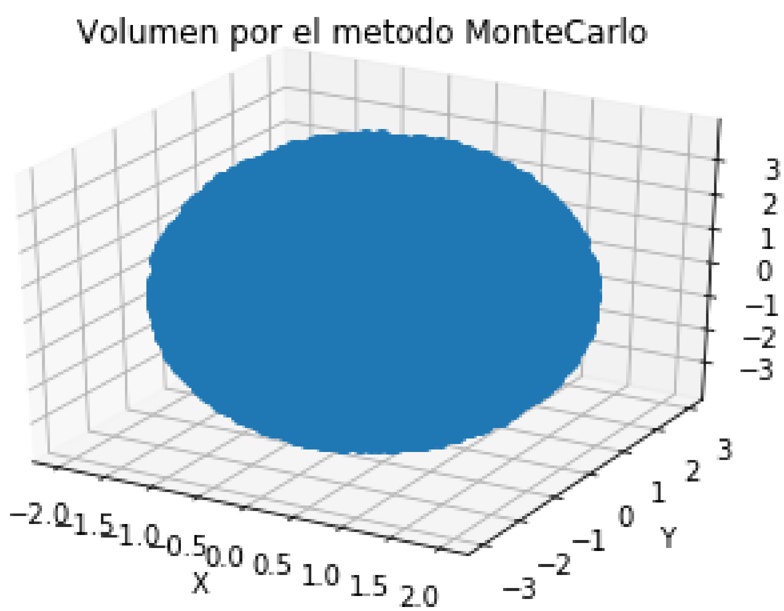
2.5 5 Encuentre el volumen de un elipsoide

$\frac{x^2}{4} + \frac{y^2}{9} + \frac{z^2}{16} = 1$

[8]:
```
m = 10000
veces = 10
a = np.array([-2,-3, -4])
b = np.array([2,3, 4])
a_b = b-a
ps = []
sa = 0
```

```
saa = 0
for k in range(veces):
    n = 0
    for i in range(m):
        p = np.array([random.random()*a_b[0] + a[0],  random.random()*a_b[1] +␣
 ↪a[1], random.random()*a_b[2] + a[2]])
        if p[0]**2/4 + p[1]**2/9 + p[2]**2/16<1:
            n = n+1
            ps.append(p)
    area = n/m * a_b[0]*a_b[1]
    sa = sa + area
    saa = saa + area**2
prom = sa/veces
desv = math.sqrt(veces*saa-sa**2)/veces
fig, ax = plt.subplots()
ax = plt.axes(projection='3d')
ax.grid()
ax.plot([ p[0] for p in ps], [ p[1] for p in ps],[ p[2] for p in ps],  'o')
ax.set(title='Volumen por el metodo MonteCarlo', xlabel = 'X', ylabel='Y')
print('Volumen\t\t:\t{:}\nDesviacion\t:\t{:.2}'.format(prom, desv))
```

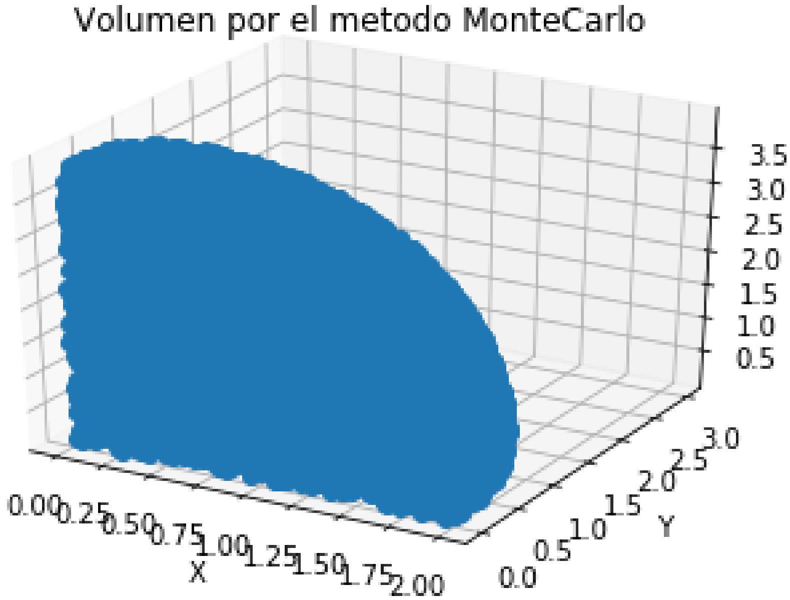
```
Volumen         :       12.58488
Desviacion      :       0.11
```



2.6 Encuentre el volumen del elipsoide en el primer cuadrante del problema 5

[9]:
```python
m = 10000
veces = 10
a = np.array([0,0, 0])
b = np.array([2,3, 4])
a_b = b-a
ps = []
sa = 0
saa = 0
for k in range(veces):
    n = 0
    for i in range(m):
        p = np.array([random.random()*a_b[0] + a[0],  random.random()*a_b[1] +␣
 ↪a[1], random.random()*a_b[2] + a[2]])
        if p[0]**2/4 + p[1]**2/9 + p[2]**2/16<1:
            n = n+1
            ps.append(p)
    area = n/m * a_b[0]*a_b[1]
    sa = sa + area
    saa = saa + area**2
prom = sa/veces
desv = math.sqrt(veces*saa-sa**2)/veces
fig, ax = plt.subplots()
ax = plt.axes(projection='3d')
ax.grid()
ax.plot([ p[0] for p in ps], [ p[1] for p in ps],[ p[2] for p in ps],  'o')
ax.set(title='Volumen por el metodo MonteCarlo', xlabel = 'X', ylabel='Y')
print('Volumen\t\t:\t{:.2}\nDesviacion\t:\t{:.2}'.format(prom, desv))
```

```
Volumen         :       3.1
Desviacion      :       0.022
```



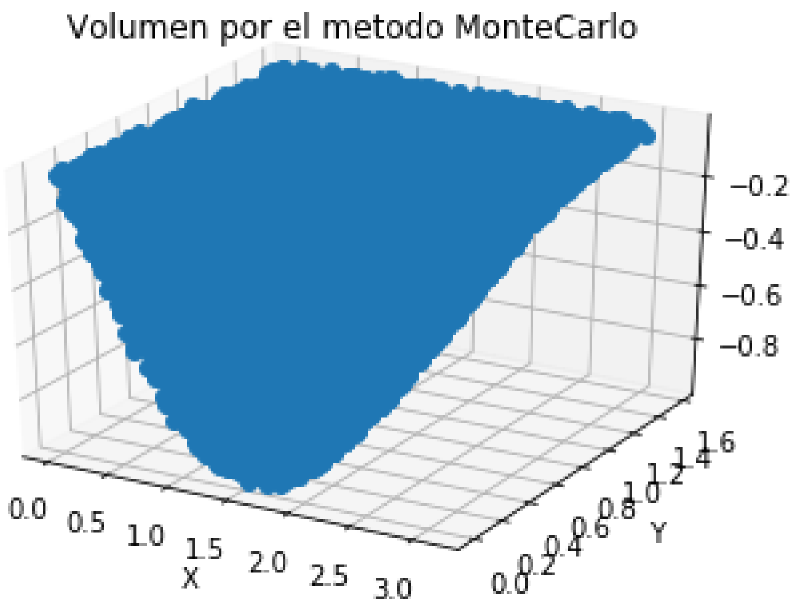
2.7 Evalúe la integral

$\int_0^{\pi/2} \int_0^{\pi} sin(x) * cos(y - \pi) dx dy$

[10]:
```
m = 10000
veces = 10
a = np.array([0,0, -1])
b = np.array([math.pi,math.pi/2, 0])
a_b = b-a
ps = []
sa = 0
saa = 0
for k in range(veces):
    n = 0
    for i in range(m):
        p = np.array([random.random()*a_b[0] + a[0],  random.random()*a_b[1] +␣
 ↪a[1], random.random()*a_b[2] + a[2]])
        if math.sin(p[0])*math.cos(p[1]-math.pi)<p[2]:
            n = n+1
            ps.append(p)
    area = n/m * a_b[0]*a_b[1]
    sa = sa + area
    saa = saa + area**2
prom = sa/veces
desv = math.sqrt(veces*saa-sa**2)/veces
fig, ax = plt.subplots()
ax = plt.axes(projection='3d')
```

```
ax.grid()
ax.plot([ p[0] for p in ps], [ p[1] for p in ps],[ p[2] for p in ps],  'o')
ax.set(title='Volumen por el metodo MonteCarlo', xlabel = 'X', ylabel='Y')
print('Volumen\t\t:\t{:.2}\nDesviacion\t:\t{:.2}'.format(prom, desv))
```

```
Volumen         :       2.0
Desviacion      :       0.025
```



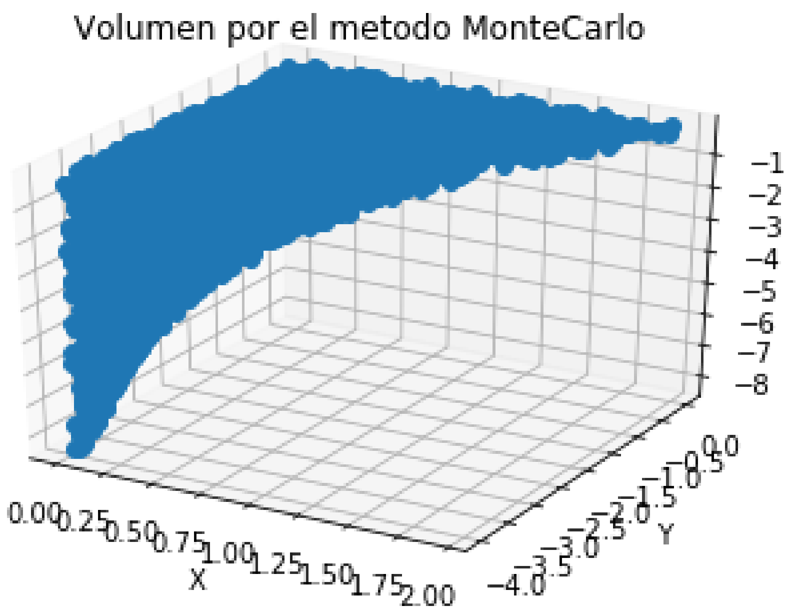
2.8 Evalúe la integral

$\int\int \frac{2y-1}{2x+1} dxdy; x = 0; y = 0; 2x - y = 4$

[11]:
```
m = 10000
veces = 10
a = np.array([0,-4, -10])
b = np.array([2,0, 0])
a_b = b-a
ps = []
sa = 0
saa = 0
for k in range(veces):
    n = 0
    for i in range(m):
        p = np.array([random.random()*a_b[0] + a[0],  random.random()*a_b[1] +
 ↪a[1], random.random()*a_b[2] + a[2]])
        if (2*p[1]-1)/(2*p[0]+1)<p[2] and 2*p[0]-p[1]<4:
            n = n+1
```

```
            ps.append(p)
    area = n/m * a_b[0]*a_b[1]
    sa = sa + area
    saa = saa + area**2
prom = sa/veces
desv = math.sqrt(veces*saa-sa**2)/veces
fig, ax = plt.subplots()
ax = plt.axes(projection='3d')
ax.grid()
ax.plot([ p[0] for p in ps], [ p[1] for p in ps],[ p[2] for p in ps],  'o')
ax.set(title='Volumen por el metodo MonteCarlo', xlabel = 'X', ylabel='Y')
print('Volumen\t\t:\t{:.2}\nDesviacion\t:\t{:.2}'.format(prom, desv))
```

```
Volumen         :       0.82
Desviacion      :       0.029
```



2.9 Evalúe la integral

$\int\int x^2 * y^2 dxdy; y = x^2; x = y^2$

[12]:
```
m = 10000
veces = 100
a = np.array([-2,-2, -10])
b = np.array([2,2, 10])
a_b = b-a
ps = []
sa = 0
```

```
saa = 0
for k in range(veces):
    n = 0
    for i in range(m):
        p = np.array([random.random()*a_b[0] + a[0],  random.random()*a_b[1] +
 ↪a[1], random.random()*a_b[2] + a[2]])
        if p[0]**2*p[1]**2<p[2] and p[0]**2<p[1] and p[1]**2<p[0]:
            n = n+1
            ps.append(p)
    area = n/m * a_b[0]*a_b[1]
    sa = sa + area
    saa = saa + area**2
prom = sa/veces
desv = math.sqrt(veces*saa-sa**2)/veces
fig, ax = plt.subplots()
ax = plt.axes(projection='3d')
ax.grid()
ax.plot([ p[0] for p in ps], [ p[1] for p in ps],[ p[2] for p in ps],  'o')
ax.set(title='Volumen por el metodo MonteCarlo', xlabel = 'X', ylabel='Y')
print('Volumen\t\t:\t{:.2}\nDesviacion\t:\t{:.2}'.format(prom, desv))
```

```
Volumen         :       0.16
Desviacion      :       0.015
```