Integración numérica basado en el método de Monte Carlo

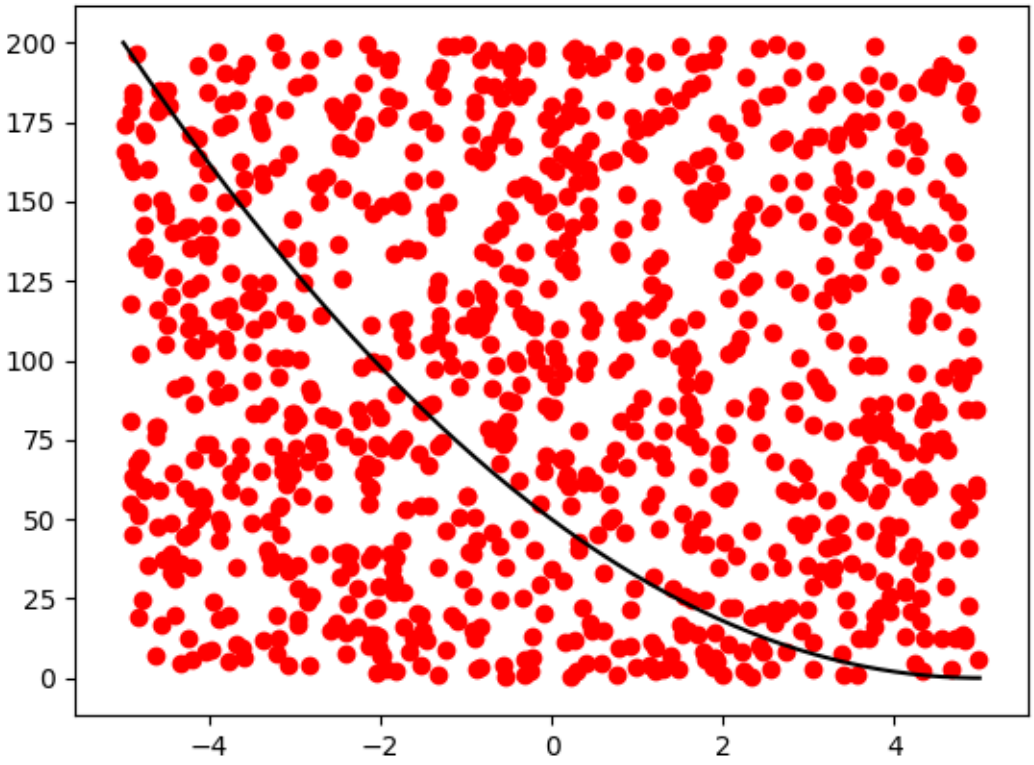
Código

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import time

def integra_mc(fun, a, b, num_puntos=10000):

    start_time = time.time()
    x=np.linspace(a,b,num_puntos)
    M=max((fun(x)))
    rectangle = (b-a)*M
    N_debajo = 0
    xs = []
    ys = []
    for i in range(num_puntos):
        random_x = np.random.uniform(a, b)
        xs.append(random_x)
        random_y = np.random.uniform(0, M)
        ys.append(random_y)
        if random_y < fun(random_x):
            N_debajo = N_debajo + 1
    I = (N_debajo/float(num_puntos))*rectangle
    time_elapsed = time.time() - start_time
    return(I, xs, ys, time_elapsed)

def vector_integra_mc(fun, a, b, num_puntos=10000):

    start_time = time.time()
    x=np.linspace(a,b,num_puntos)
    M=max((fun(x)))
    rectangle = (b-a)*M
    random_x = np.random.uniform(a,b,[1, num_puntos])
    random_y = np.random.uniform(0,M,[1, num_puntos])
    N_debajo = np.sum(random_y < fun(random_x))
    I = (N_debajo/float(num_puntos))*rectangle
    time_elapsed = time.time() - start_time
    return(I, random_x.reshape(-1), random_y.reshape(-1), time_elapsed)óó
    return((2)*((x-5)**2))

i,x,y,t = integra_mc(function, -5, 5, 1000)
print('Integration using loops')
print("Integral = %.2f" % i)
print("Execution time = %s seconds" % t)
df = pd.DataFrame()
df['x'] = x
df['y'] = y

x = np.linspace(-5,5,1000)
plt.plot(x,function(x),color='black')
plt.scatter(df['x'],df['y'],color='red')
plt.show()


i,x,y,t = vector_integra_mc(function, -5, 5, 1000)
```

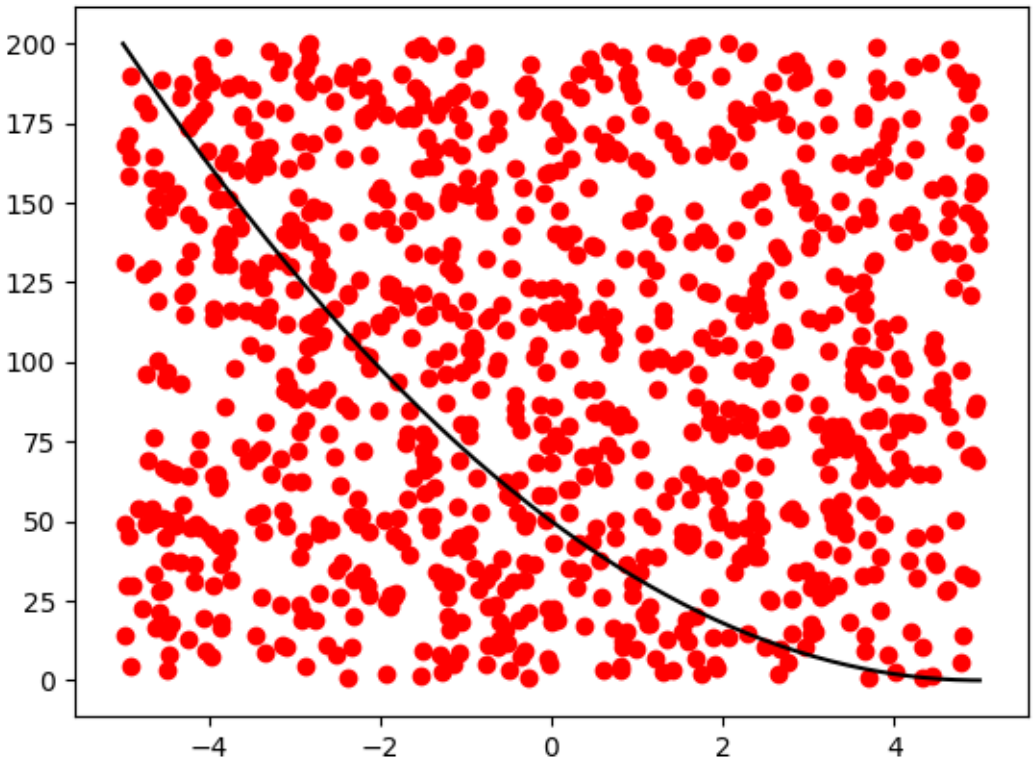
```
print('Integration using vectors')
print("Integral = %.2f" % i)
print("Execution time = %s seconds" % t)
df = pd.DataFrame()
df['x'] = x
df['y'] = y

x = np.linspace(-5,5,1000)
plt.plot(x,function(x),color='black')
plt.scatter(df['x'],df['y'],color='red')
plt.show()

from scipy import integrate as integrate
print("Result from integrate.quad = %.2f" % integrate.quad(function,-5,5)[0])
```

Resultado

```
Integration using loops
   Integral = 712.00
Execution time = 0.00197792053223 seconds
```



```
Integration using vectors
Integral = 678.00
Execution time = 0.000432014465332 seconds
```




```
Result from integrate.quad = 666.67
```

Comentarios

El método basado en vectores es mucho más rápido que lo basado en bucles porque los operaciones entre vectores son implementados en una manera más optimizada. Los resultados obtenidos con ambos métodos son diferentes de los de función integrate.quad. Cada vez los resulatados son diferentes porque método de Monte Carlo no es determinista.