Volume of a sphere using MC

December 2, 2020

1 Estimating the Volume of a Sphere using MC techniques

To estimate the volume of a sphere I employed the fairly standard hit or miss practice using MC integration. I generated 1×10^7 Monte Carlo Samples using $np.random.uniform$ from -1 to 1. In the simulation I created a sphere with radius 1 and a cube with side of length 2. Because the points inside the cube were random they had a chance to be in the sphere or not. To count the number of points inside the sphere I found the sum of the distance from the origin and counted them. This is then compared to the total number of points as a fraction. The fraction inside multiplied by the volume of the cube will equal the volume of the sphere.

To calculate pi is simply done by rearranging the formula for the volume of a sphere and solving for pi

However, to calculate the uncertainties I ran the monte carlo simulation 200 times and found the average. From this I used the known formula of the standard error to find the error in the mean

$$S_E = \frac{\sigma}{\sqrt{n_{sims}}}$$

[18]:
```
import numpy as np
import matplotlib.pyplot as plt
from scipy import random
import scipy.special


N = int(1e7) # Monte Carlo samples
R=1

all = np.random.uniform(-1,1, 3*N).reshape(N,3)
distance = np.sum(all**2,axis=1)**0.5

Ninside = np.sum(distance<=1) # number of points in sphere
fractioninside = Ninside/N # fraction of points in sphere
vol_cube = 2**3 # Volume of cube
vol_sphere = fractioninside * cube # Volume of sphere

solution = 4./3. * np.pi*R**3 #true volume using pi
diff= np.abs(vol_sphere-solution)/solution # Error
```

```
print("N=%i\tN_in=%06d\tV=%.5f" %(N,Ninside,vol_sphere))
print("True value vol_sphere =%.5f" %solution)
print("Difference = %.5f" %(np.abs(solution-vol_sphere)))
print("percentage diff = %f percent" %diff)

pi_est = (vol_sphere*3)/(4*(R**3))

print("-------")
print("True pi val: %f" %(np.pi))
print("Est Pi: %f" %(pi_est))
```

```
N=10000000      N_in=5235829    V=4.18866
True value vol_sphere =4.18879
Difference = 0.00013
percentage diff = 0.000030 percent
-------
True pi val: 3.141593
Est Pi: 3.141497
```

[33]:
```
n_sims=200
import time

times=np.zeros(n_sims)
vol_list = np.zeros(n_sims)
pi_est_list = np.zeros(n_sims)
for i in range(0,n_sims):
    t0=time.time()
    N = int(1e7) # Monte Carlo samples
    R=1

    all = np.random.uniform(-1,1, 3*N).reshape(N,3)
    distance = np.sum(all**2,axis=1)**0.5

    Ninside = np.sum(distance<=1) # number of points in sphere
    fractioninside = Ninside/N # fraction of points in sphere
    vol_cube = 2**3 # Volume of cube
    vol_sphere = fractioninside * cube # Volume of sphere
    pi_est_list[i] = (vol_sphere*3)/(4*(R**3))
    vol_list[i] = vol_sphere
    t=time.time()-t0
    times[i]=t
    if np.mod(i,20)==0: #keep track of sim
        print(i,t)


print("Total time of sim = %f" %(np.sum(times)))
```

```
0 0.634119987487793
```

```
20 0.6301989555358887
40 0.6240849494934082
60 0.6158668994903564
80 0.6180229187011719
100 0.6163930892944336
120 0.6282470226287842
140 0.6265480518341064
160 0.6127808094024658
180 0.6191859245300293
Total time of sim = 124.454432
```

[30]:
```
std_vol = np.std(vol_list)
pf_vol = np.mean(vol_list)
stdError_vol = std_vol/np.sqrt(n_sims)

std_pi = np.std(pi_est_list)
pf_pi = np.mean(pi_est_list)
stdError_pi = std_pi/np.sqrt(n_sims)



print("Volume of 3D Sphere estimate: %.5f +- %.5f" %(pf_vol, stdError_vol))
print("Pi Estimate: %.5f +- %.5f" %(pf_pi, stdError_pi))
```

```
Volume of Sphere estimate: 4.18875 +- 0.00009
Pi Estimate: 3.14156 +- 0.00007
```

[ ]:
```

```