模擬與統計計算 HW2

電機所 碩一 N26120870 林耕澤

第一部分:利用蒙地卡羅模擬方法實作本題積分。

5. $\int_{-2}^{2} e^{x+x^2}\,dx$

程式碼實作:

```python
import matplotlib.pyplot as plt
import random
from scipy import integrate
import numpy as np
plt.style.use("seaborn-poster")

def function(x):
    '''欲積分函式樣式:'''
    return np.exp(x + x**2)

def Monte_Carlo(start,end,n):
    '''蒙地卡羅實作:
        start 積分起始點
        end 積分終點
        n 次數
        '''
    total = 0 #總和
    for i in range(n): #給n次積分範圍內隨機值
        x = random.uniform(-2,2) #uniform random var in -2~2
        total += function(x) #將隨機x所得到的output加入total
    return (total/n)*(end - start),(total/n) #回傳用蒙地卡羅方法所得到的積分面積以及高

def normal_integral(start,end):
    '''一般積分所得到的值:
        start 積分起始點
        end 積分終點
    '''
    area,err = integrate.quad(function,start, end)
    return area


def draw():
    '''製圖'''
    area_mon, height_mon = Monte_Carlo(-2,2,1000000)
    area = normal_integral(-2,2)

    x_diff = np.linspace(-2,2,1000)
    y = function(x_diff)

    plt.plot(x_diff,[height_mon]*1000,color = 'green',label = 'Monte_Carlo: Area =' + str(area_mon))
    plt.fill_between(x_diff,y1=height_mon,y2=0,where=(x_diff>=-2)&(x_diff<=2),facecolor='green',alpha=0.2)
    plt.plot(x_diff,y,color = 'blue',label = 'e^(x^2+x): Aera =' + str(area))
    plt.fill_between(x_diff,y1=y,y2=0,where=(x_diff>=-2)&(x_diff<=2),facecolor='blue',alpha=0.2)
    plt.legend()
    plt.show()

draw()
```

figure 1:程式碼

函式介紹:

function():欲積分函式

Monte_Carlo():蒙地卡羅方法實作

normal_integral():利用 python scipy 中的積分函式所得到的積分值

draw():製圖

結果:

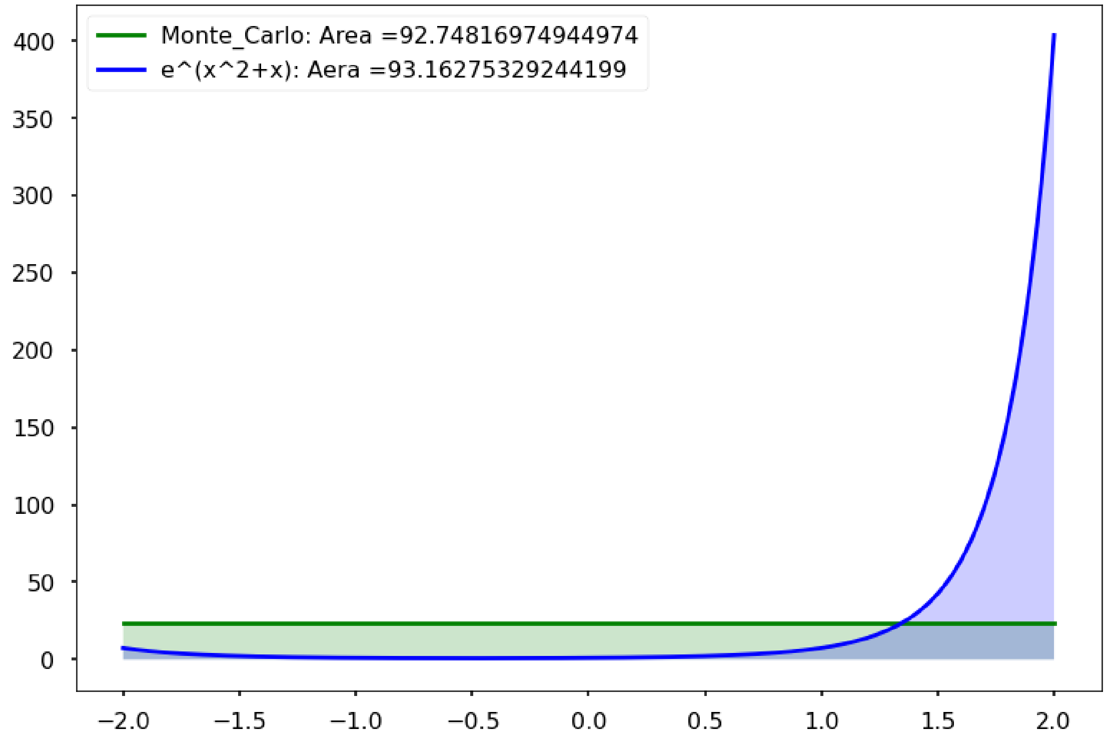

figure 2:X 軸為積分區間，Y 軸為高度(函式輸出值)

由結果可得知使用蒙地卡羅方法(N = 1000000)所得到的積分結果為 92.75 左右，而與使用一般積分方法所得到的值 93.16 相差不遠，此外我也測試了 N=10、100、1000、10000、100000 所得到的結果，結果如下圖:

```
N = 10  251.0881794169133
N = 100  110.95237304295092
N = 1000  103.17949683063529
N = 10000  94.17968468523479
N = 100000  93.09687360625345
```

figure 3: 結果

由此亦可推測 N 越大會越接近理論值，亦符合大數法則。

第二部分:

12. For uniform (0, 1) random variables $U_1, U_2, \ldots$ define

$$N = \text{Minimum}\left\{n: \sum_{i=1}^{n} U_i > 1\right\}$$

That is, N is equal to the number of random numbers that must be summed to exceed 1.

(a) Estimate $E[N]$ by generating 100 values of N.
(b) Estimate $E[N]$ by generating 1000 values of N.
(c) Estimate $E[N]$ by generating 10,000 values of N.
(d) What do you think is the value of $E[N]$?

程式碼實作:

```python
import matplotlib.pyplot as plt
import random
import numpy as np
plt.style.use("seaborn-poster")

def sum_to_exceed_1(time):
    '''time 次數'''
    N_list = [] #儲存每次需要的N次數
    for i in range(time):
        total = 0
        N = 0
        while(total <=1):#當total超過1則停止
            total += random.uniform(0, 1) #uniform random var in 0~1
            N+=1
        N_list.append(N)
    expect_value = sum(N_list)/time #期望值計算
    return expect_value




def draw():
    '''製圖'''
    x_diff = range(100,100000,100)
    y = []
    for i in x_diff:
        y.append(sum_to_exceed_1(i))
    plt.plot(x_diff,y,label = 'E[N] by generating 100 to 100000 values of N')
    plt.legend()
    plt.show()
    print('E[N] by generating 100000 values of N: '+str(y[-1]))


print(sum_to_exceed_1(100))
print(sum_to_exceed_1(1000))
print(sum_to_exceed_1(10000))
print(sum_to_exceed_1(100000))
draw()
```

figure 4 程式碼

函式介紹:

sum_to_exceed_1():根據題意實作之函式
draw():製圖

根據題目回答問題:

(a)N=100:　<u>2.74</u>
(b)N=1000:　<u>2.723</u>
(c)N=10000: <u>2.7128</u>

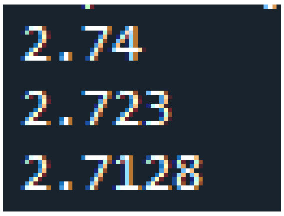

figure 5:1.N=100 2.N=1000 3.N=10000

(d)為了更加準確，我另外製作了一張圖來觀察其收斂情況，由此圖得知其期望值大約為<u>2.718</u>左右。

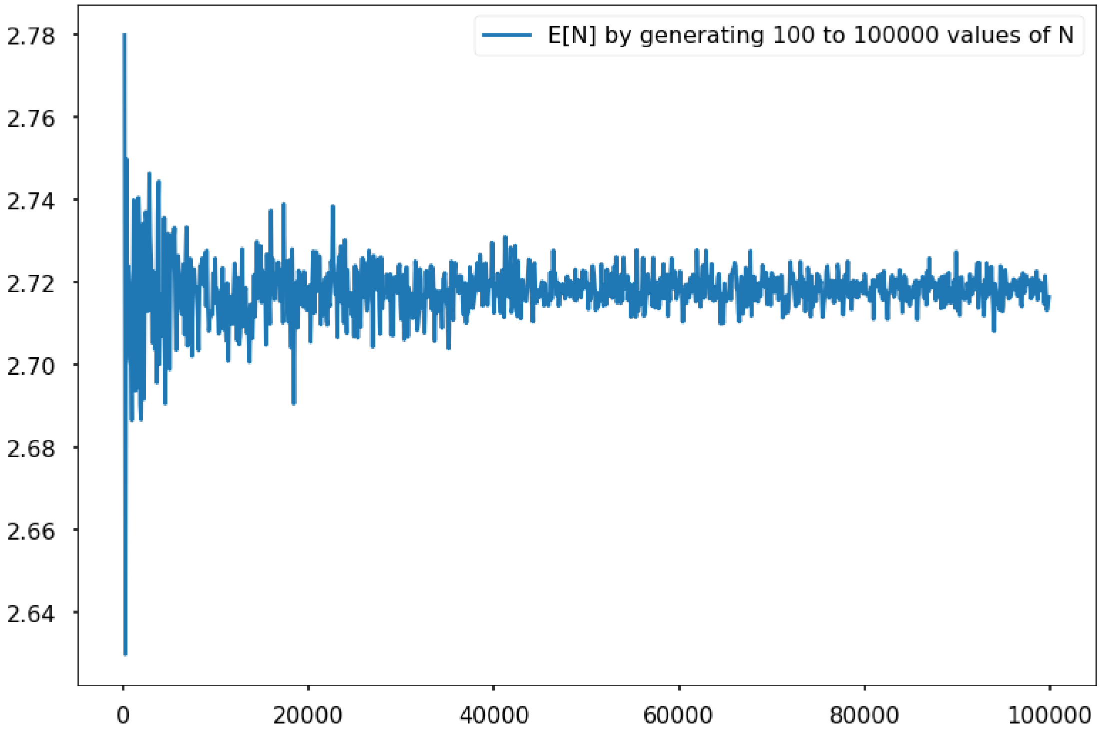

figure 6: X軸為次數，Y軸為期望值