Nombre: Juan Manuel Varela

Entrega Métodos de Monte Carlo

Unidad 3 – Sesión 06 - Ejercicio 6.1

Descripción del problema:

Se idealiza una montaña como un cono inscrito en una región cuadrada de lado 1 km. La base de la montaña es circular, con centro en (0.5, 0.5) y radio r = 0.4km, y la altura es H = 8km. La altura de cada punto (x, y) de la montaña está dada por la función:
$f(x,y) \ = H - H/r \times \sqrt{(x \ - \ 0.5)^2 \ + \ (y \ - \ 0.5)^2}$, en la zona definida por el círculo, y 0 fuera del círculo. El volumen total de la montaña (en km cúbicos) puede verse como la integral de la función altura en la región.

• Parte a: escribir un programa para calcular el volumen por Monte Carlo. Realizar 10^6 replicaciones y estimar el valor de ζ y el error cometido (con nivel de confianza 0.95), utilizando como criterio la aproximación normal.

• Parte b: en base al valor estimado en la parte a, calcular el número de replicaciones necesario para obtener un error absoluto menor a 10^{-3} (con nivel de confianza 0.95).

• Parte c: realizar esa cantidad de replicaciones y estimar ζ y su intervalo de confianza.

Solución a aplicar en Python:

1. Importar random #Esta biblioteca incluye funciones para generar números seudoaleatorios
2. Importar math #Esta biblioteca incluye funciones para calcular el entero inmediatamente mayor a un número real y raiz cuadrada
3. Importar time #Esta biblioteca incluye funciones que permiten calcular el tiempo de ejecución
4. Importar norm de scipy.stats #Incluye funciones para calcular valores de la distribución normal
5. Definir semilla
   #Parte a
6. Leer número de replicaciones
7. Leer δ, tal que 1- δ sea el nivel de confianza deseado
8. Definir función: fi(x,y): """Evalúa la función $f(x,y) \ = H - H/r \times \sqrt{(x \ - \ 0.5)^2 \ + \ (y \ - \ 0.5)^2}$ en el punto (x,y)"""
   8.1. centro=(0.5, 0.5)
   8.2. radio=0.4
   8.3. H=8
   8.4. Distancia= distancia de (x,y) al centro
   8.5. If Distancia<=radio:
      8.5.1.Altura=H-H/radio*distancia
   8.6. Else: Altura=0
   8.7. Salida: Altura
9. Definir función: integración_monte_carlo(función, n, nivel): """Estima por monte carlo el valor de la integral la función en la región $[0,1]^2$ utilizando un tamaño de muestra n y halla un intervalo de confianza del nivel deseado"""

9.1. S=0, T=0 #Inicialización
9.2. For i entre 1 y n:
 9.2.1. Sortear $\text{Punto}_{(i)}$ con distribución uniforme en $[0,1]^2$
 9.2.2. If i>1 then $T=T+(1-1/i)*(\text{funcion}(\text{Punto}_{(i)})-S/(i-1))^2$
 9.2.3. S = S + funcion($\text{Punto}_{i(i)}$) #Acumular en S y T*
9.3. Integral= S/n #Estimador puntual de la integral
9.4. Varianza_función= T/(n-1) #Estimador puntual de la varianza de función(Punto)
9.5. Varianza_integral= Varianza_función/n #Estimador puntual de la varianza de la integral
9.6. Cálculo [I_1(S,n,1-nivel), I_2(S,n,1-nivel)] #Intervalo de confianza del nivel ingresado para la integral
9.7. Salida: Integral, error, Varianza_función

10. Inicio = tiempo de inicio
11. Estimación, error_est, varianza_fi= integración_monte_carlo(fi, n, 1- δ)
12. Tiempo_ejecución=Tiempo final-inicio #Cálculo del tiempo de ejecución
13. Escribir: "Estimador: {Estimación}, Error: {error_est}, Tiempo (segundos): {Tiempo_ejecución}
 #Parte b
14. Leer ε
15. Cálculo de n_N #n requerido de acuerdo a la aproximación normal
16. Escribir: "Numero de replicaciones necesarias: {n_N}"
 #Parte c
17. Definir semilla
18. Estimación_2, error_est_2, varianza_fi_2= integración_monte_carlo(fi, n_N, 1- δ)
19. Tiempo_ejecución=Tiempo final-inicio #Cálculo del tiempo de ejecución
20. Escribir: "Estimador: {Estimación_2}, Error: {error_est_2}, Tiempo (segundos): {Tiempo_ejecución}

Resultados:

Plataforma de cómputo: PC de escritorio, procesador Intel I5-12400, RAM: 32 Gb, Windows 10

Parte a:

Semilla: 6
$n=10^6$
$\delta=0.05$
$\bar{\zeta}$= 1,342396 km^3
ε= 0,003702 km^3

Parte b:

Se aplica la siguiente ecuación:

$$\dot{n}_N(\epsilon,\delta) = \lceil \left(\Phi^{-1}(1-\delta/2)\right)^2 \dot{\sigma}^2_{n'}/\epsilon^2 \rceil$$

Se llega a que para obtener un error absoluto menor a 10^{-3} se necesita realizar 13707408 replicaciones, este valor es mayor al número de replicaciones realizado en la parte a, lo cual es coherente ya que el ε deseado es menor al obtenido en la parte a.

Parte c:

Semilla: 7
n=13707408
δ=0.05
$\bar{\zeta}$= 1,340535km^3
ε= 0,00100 km^3

Entonces el intervalo de confianza es (1,339535km^3, 1,341535km^3)

Se puede ver que se obtuvo exactamente el error deseado.

Anexos:

Log de ejecuciones:

```
runcell(0, 'G:/…/ej_6.1.py')
Parte a:
Ingrese el número de replicaciones, n: 1000000
Ingrese delta, tal que el nivel de confianza sea 1-delta: 0.05
Estimador:  1.342396      Error:  0.003702         Tiempo(segundos):
1.333541
Parte b:
Ingrese epsilon: 0.001
Número de replicaciones necesarias:13707408.000000
Parte c:
Estimador:  1.340535      Error:  0.001000         Tiempo(segundos):
17.826391
```

Código fuente:

```
import random #Esta biblioteca incluye funciones para generar números
seudoaleatorios
import math #Esta biblioteca incluye funciones para calcular el entero
inmediatamente mayor a un número real y raiz cuadrada
import time #Esta biblioteca incluye funciones que permiten calcular el tiempo
de ejecución
from scipy.stats import norm #Incluye funciones para calcular valores de la
distribución normal

random.seed(6)

#Parte a:
print('Parte a:')

n=int(input('Ingrese el número de replicaciones, n: '))
delta=float(input('Ingrese delta, tal que el nivel de confianza sea 1-delta:
'))

def fi(x,y):
    centro=[0.5, 0.5]
    radio=0.4
    H=8
    distancia=math.sqrt((x-centro[0])**2+(y-centro[1])**2)
    if distancia <= radio:
        altura=H-H/radio*distancia
    else: altura=0
    return altura

def integracion_monte_carlo(funcion, n, nivel):
    S=0
    T=0 #Inicialización
    for i in range(1,n+1):
        punto= [random.random() for i in range(2)]
        if i>1:
            T=T+(1-1/i)*(funcion(punto[0],punto[1])-S/(i-1))**2
        S=S+funcion(punto[0],punto[1])#Acumular en S y T*
    integral= S/n #Estimador puntual de la integral
    var_funcion= T/(n-1) #Estimador puntual de la varianza de funcion(punto)
    var_integral= var_funcion/n #Estimador puntual de la varianza de la
integral
```

```
    I1=integral-norm.ppf(nivel+(1-nivel)/2)*math.sqrt(var_integral)
    I2=integral+norm.ppf(nivel+(1-nivel)/2)*math.sqrt(var_integral)
    error=(I2-I1)/2
    return integral, error, var_funcion

inicio= time.time()
est, err, var_fi_n = integracion_monte_carlo(fi, n, 1-delta)
tiempo_ejecucion= time.time()-inicio
print(f'Estimador:{est:10f}\
    \tError:{err:10f}\
    \tTiempo(segundos):{tiempo_ejecucion:10f}')

#Parte b:
print('Parte b:')
epsilon=float(input('Ingrese epsilon: '))

#Se calcula el tamaño de muestra requerido de acuerdo a la aproximación normal
nN=math.ceil(norm.ppf(1-delta/2)**2*var_fi_n/epsilon**2)
print(f'Número de replicaciones necesarias:{nN:10f}')

#Parte c:
print('Parte c:')
random.seed(7)

inicio= time.time()
est2, err2, var_fi_n2 = integracion_monte_carlo(fi, nN, 1-delta)
tiempo_ejecucion= time.time()-inicio

print(f'Estimador:{est2:10f}\
    \tError:{err2:10f}\
    \tTiempo(segundos):{tiempo_ejecucion:10f}')
```