Métodos de Monte Carlo

Entrega 6

(Unidad 4, Sesiones 10 y 11) Ejercicios 10.1 y 11.1

Integrantes

Nombre	CI	Correo
Germán Ouviña	4.823.566-1	german.ouvina@fing.edu.uy

Índice

1. Problema

Entrega 6: Ejercicio 10.1 (INDIVIDUAL):

Resumir en un texto (de entre media y una carilla) los principales contenidos del paper de P. L'Ecuyer, "Software for Uniform Random Number Generation: Distinguishing the Good and the Bad", Proceedings of the 2001 Winter Simulation Conference, IEEE Press, Dec. 2001, 95-105. ¿Cuál es el objetivo del trabajo? ¿Qué generadores de números seudo-aleatorios discute? ¿Cuáles son los hallazgos de esta investigación, y cuáles las conclusiones y recomendaciones presentadas por el autor?

Entrega 6: Ejercicio 11.1 (INDIVIDUAL):

Para generar un punto aleatorio (X_1, X_2) en un círculo de centro $(0, 0)$ y radio 1, es posible hacerlo de la forma siguiente (derivación disponible en las páginas 234 y 235 del libro de referencia del curso, "Monte Carlo: concepts, algorithms and applications", Fishman 1996):

- Se genera un valor aleatorio r, de distribución $F_r(x) = x^2$ para $0 \leq x \leq 1$, y 0 para cualquier otro x
- Se generan dos variables aleatorias independientes Z_1 y Z_2 de distribución normal $(0, 1)$
- Se calcula $X_1 = rZ_1/\sqrt{Z_1^2 + Z_2^2}$ y $X_2 = rZ_2/\sqrt{Z_1^2 + Z_2^2}$

Utilizar esta propiedad para volver a resolver el Ejercicio 6.1 parte a, pero generando únicamente valores de puntos dentro del círculo de base de la montaña:

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.

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.

Comparar la precisión obtenida con la alcanzada en el ejercicio 6.1.

Sugerencia: tener en cuenta que al estar generando puntos dentro del círculo, estamos calculando una integral de Lebesgue-Stieltjes, por lo que es necesario ajustar el integrando de manera que quede explícita la integral en la forma $\int k(z)dF(z)$, con z un vector. En particular, por ser un sorteo uniforme dentro del círculo, la densidad de probabilidad en el círculo es 1/ (area del círculo), y 0 afuera del mismo.

2. Solución

2.1. Descripción

2.1.1. Ejercicio 10

El paper se enfoca en la herramienta denominada **TestU01**, una biblioteca desarrollada en *C* para el testing de generadores de números pseudoaleatorios, denominados *Random Number Generators (RNG)* por el texto. Concretamente, la herramienta permite realizar tests estadísticos sobre RNGs para el intervalo $(0, 1)$ y para generadores de secuencias de bits. La lista de tests posibles es amplia, implementando diversos tipos de test estadísticos conocidos en el campo, así como creando algunos nuevos.

Antes de enfocarse en las tecnicalidades de la herramienta, el paper define ciertos conceptos base, como el de test estadístico o como medir la calidad de de un RNGs. Luego, se enfoca en desarrollar distintos tipos de tests estadísticos para ambos casos de uso (generadores en el intervalo $(0, 1)$ y generadores de secuencias de bits), entrando en detalle sobre el funcionamiento de distintos métodos. Finalmente, explica la diagramación de la herramienta y sus distintos componentes, como las familias de RNGs que emplea y los distintos grupos de tests que ofrece.

Para cerrar el estudio, utilizando la herramienta se realizó experimentación sobre RNGs utilizados en la práctica y se estudió su calidad en base a distintas métricas definidas en el paper, como por ejemplo el llamado *p-value*. El estudio concluye que muchos RNG populares pasan la mayoría de tests, pero que a su vez varios los fallas estrepitosamente, particularmente RNGs genéricos definidos en bibliotecas de lenguajes famosos como Java, Matlab, R, etc. Finalmente, recomienda emplear RNGs que permitan descomponer las cadenas de números generados para evitar el problema de alcanzar el período.

2.1.2. Ejercicio 11

Para resolver este ejercicio se plantearon las dimensiones de la montaña como una función, concretamente la función de altura, y se trató el calculo de su volúmen como el problema de calcular la integral de dicha función.

Comparando con el laboratorio 3, ejercicio 6.1, el cambio introducido está relacionado únicamente con la manera de muestrear los puntos de la base. En el ejercicio original, se sortean puntos del cuadrado $[0, 1]^2$ con probabilidad uniforme $U(0, 1)$ y se comprueba que pertenezcan al círculo base de centro $(0{,}5, 0{,}5)$ y radio 0,4. Si asi lo hacen, se calcula su altura y se acumula en el estimador, luego dividiéndolo por el tamaño de muestra n.

El cambio introducido permite realizar el sorteo de otra manera: se genera un valor aleatorio r de distribución $F_r(x) = x^2$, se generan dos variables aleatorias Z_1 y Z_2 de distribución $N(0, 1)$ y se calculan las coordenadas del punto (X_1, X_2) utilizando dichos valores aleatorios. Este algoritmo permite generar aleatoriamente un punto en un círculo de centro $(0, 0)$ y radio 1. Para traducirlo al círculo del problema, solo hace falta transformar el punto obtenido sumando 0,5 a ambas coordenadas (el centro del círculo) multiplicadas por 0,4 (el radio del círculo).

Este cambio permite generar puntos exactamente en el área necesaria, pero para mantener la coherencia del estimador hace falta multiplicarlo por el valor del área, que en este caso es $\pi * r^2 = \pi * 0{,}4^2 \approx 0{,}502$.

Habiendo realizado estos cambios, el algoritmo implementado funciona de la manera esperada. En la sección 2.2 se entra en mayor detalle sobre las características del código. En la sección 3.2 se detallan los resultados obtenidos para las partes a, b y c.

2.2. Pseudocódigo

Procedimiento EstimacionMonteCarloIntegral (int n, funcion f, real δ, real ϵ)
Parámetros de entrada:

- n - tamaño de la muestra
- f - función a calcular su integral
- δ - nivel de confianza
- ϵ - error máximo

Parámetros de salida:

- $\hat{X}$ - estimador de la integral
- $\hat{V_f}$ - estimador de la varianza de la función f
- $\hat{V_X}$ - estimador de la varianza del estimador puntual $\hat{X}$
- ω_1, ω_2 - intervalo de confianza para $(1 - \delta)$

Pseudocódigo:

1. $X, \hat{X}, \hat{V_f}, \hat{V_X} = 0$
2. For $i = 1, ..., n + 1$ do
   a) Generar $u = U(0, 1)$ y calcular $r = F_r^{-1}(u) = \sqrt{u}$
   b) Generar $Z_1, Z_2 = N(0, 1)$
   c) Generar coordenadas de un punto $p = (X_1, X_2) = (rZ_1\sqrt{Z_1^2 + Z_2^2}, rZ_2\sqrt{Z_1^2 + Z_2^2})$
   d) Evaluar $f(p)$
   e) Acumular $X = X + f(p)$
   f) Si $i > 1 \rightarrow \hat{V_f} = \hat{V_f} + (1 - \frac{1}{i})(f(p) - \frac{X}{i-1})^2$
3. $\hat{X} = AX/n$ con $A = \pi * 0{,}4^2$
4. $\hat{V_f} = \hat{V_f}/(n - 1)$
5. $\sqrt{\hat{V_X}} = \sqrt{\hat{V_f}/n}$
6. $\omega_1, \omega_2 = I_1(\hat{X}, \sqrt{\hat{V_X}}, \delta), I_2(\hat{X}, \hat{V_X}, \delta)$
7. $n_N = n_N(\epsilon, \delta, \sqrt{\hat{V_f}})$

El anterior pseudocódigo bosqueja un método de monte carlo para el cálculo de la integral de una función, siguiendo las pautas del pseudocódigo presentado en clase. Los cambios empleados están relacionados al muestreo de puntos explicado en la sección anterior, siguiendo el método empleado en las diapositivas.

2.3. Código

El código para solucionar el problema se encuentra adjunto en un archivo *.zip* junto al informe, y se encuentra disponible en el repositorio https://github.com/gouvina-fing/fing-mmc2025.

El mismo fue desarrollado en *Python*, cuenta con instrucciones de como instalarlo y correrlo, y ofrece la posibilidad de generar tablas y gráficas comparativas siguiendo la consigna del ejercicio, automáticamente ejecutando el método para los distintos valores de n mencionados.

3. Experimentación

3.1. Especificaciones

A continuación, se desglosan las especificaciones técnicas utilizadas durante la ejecución:

- **Procesador:** 11th Gen Intel(R) Core(TM) i5-11400F @ 2.60GHz
- **Memoria RAM:** 16.0 GB
- **Sistema Operativo:** Windows 10
- **Semilla:** 42
- **Tamaños de muestra:** 10^4, 10^5, 10^6

3.2. Resultados

La consigna de este ejercicio, en lo que refiere a la experimentación, pide ejecutar el algoritmo para $n = 10^6$ y calcular un intervalo de confianza utilizando la aproximación normal, comparando con los resultados obtenidos al ejecutar el algoritmo original en la entrega 3, ejercicio 6.

A continuación, se adjunta la tabla comparativa marcando en rojo los valores de interés según la consigna. Se destaca que se ejecutó el algoritmo para tamaños de muestra n menores al pedido, así como la comparación lado a lado de los valores originales obtenidos en la entrega 3, ejercicio 6, y los valores nuevos obtenidos empleado el nuevo método de muestreo.

N° de iteraciones	**Estimador original**	**Estimador nuevo**	**Varianza original**	**Varianza nueva**	**IdeC original**	**IdeC nuevo**
10^4	1.3488419564	1.3373964346	3.6198489995	3.5037113366	1.3115518761 1.3861320367	1.3007094303 1.3740834389
10^5	1.3429892360	1.3370614528	3.5626099041	3.5338255780	1.3312906809 1.3546877912	1.3254102531 1.3487126525
10^6	1.3436475743	1.3408297318	3.5728849520	3.5610720063	1.3399428354 1.3473523132	1.3371311224 1.3445283412

Se pueden observar resultados y niveles de convergencia muy similares entre ambos métodos.