Métodos de Monte Carlo

Entrega 5 - Grupo 01

(Unidad 4, Sesión 8) Ejercicio 8.1

Integrantes

Nombre	CI	Correo
Nadia Martínez	5.141045-0	nadia.martinez.petrone@fing.edu.uy
Germán Ouviña	4.823.566-1	german.ouvina@fing.edu.uy

Índice

1. Problema

Entrega 5: Ejercicio 8.1 (GRUPAL):

1. Elegir al menos dos fuentes de números aleatorios disponibles en Internet (sitio o tabla con valores). Explicar cómo funcionan, como se accede a los números, y qué características tienen.

2. En base a este análisis, elegir una de las fuentes, fundamentar la selección, y modificar el ejercicio 3.1, parte a (visto en la sesión 3) para que emplee dichos números aleatorios (en lugar de los generados por bibliotecas como hasta el momento). Comparar si la salida obtenida es consistente o no con la obtenida en los experimentos de la parte a del ejercicio 3.1.

2. Solución

2.1. Descripción

A continuación se detallan las dos fuentes de números aleatorios seleccionadas para su estudio, así como el análisis de sus características.

2.1.1. ANU Quantum Random Numbers

ANU Quantum Random Numbers [1] es un sitio web que provee números únicos y verdaderamente aleatorios. Los números son generados en tiempo real en un laboratorio basándose en la detección de fluctuaciones en el vacío cuántico mediante un experimento óptico con divisores de haz y detectores de luz. Estas fluctuaciones no son predecibles ni reproducibles, lo que garantiza la aleatoriedad del resultado. Los números generados luego son publicados en un servidor web y están disponibles para descargar sin costo. Al registrarse se obtiene un API key para poder realizar las consultas. Esto provee 100 consultas por mes. Para consultas ilimitadas es necesario crear una cuenta de AWS para suscribirse.

El acceso a los números se realiza a través de una API pública provista por la Australian National University disponible en https://quantumnumbers.anu.edu.au/.

Un ejemplo de uso simple es

```
GET https://quantumnumbers.anu.edu.au?length=3&type=uint8
```

donde `length` indica la cantidad de números a obtener y `type` indica el tipo de número (por ejemplo uint8, enteros entre 0 y 255, o uint16, enteros entre 0 y 65535). La respuesta es un JSON con un array de números y un indicador de éxito:

```
{
  "type": "uint8",
  "length": 3,
  "data": [158, 42, 77],
  "success": true
}
```

Estos valores pueden normalizarse fácilmente a $[0, 1)$ dividiendo por 256 (o 65536 respectivamente), para usarlos como si fueran generados por una distribución uniforme.

Las características de esta fuente son:

- **Tipo de aletoriedad:** verdadera, basada en física cuántica.
- **Acceso:** Gratuito, vía API REST.
- **Formato disponible:** Enteros (uint8, uint16, hex).
- **Velocidad:** Limitada por el servidor (máximo 1024 números por solicitud).
- **Confiabilidad:** Proyecto respaldado por la Australian National University.

2.1.2. RANDOM.ORG

RANDOM.ORG [2] es un sitio web que ofrece números verdaderamente aleatorios a cualquier persona en internet. Utiliza el ruido atmosférico como fuente aleatoria. El proceso consiste en captar señales de ruido de fondo mediante radios sintonizados entre estaciones, en bandas AM. Estas señales son impredecibles y se consideran un fenómeno físico aleatorio.

Una vez captado el ruido, se digitaliza y procesa mediante técnicas estadísticas para eliminar sesgos y asegurar que los bits generados tengan alta calidad aleatoria. El resultado es una secuencia de bits que no pueden ser replicados ni predecibles, lo que los convierte en una fuente ideal para tareas que requieren verdadera aleatoriedad.

Este servicio existe desde 1998 y fue creado por el Prof. Mads Haahr de la School of Computer Science and Statistics de Trinity College, Dublin en Irlanda. Actualmente es operado por Randomness and Integrity Services Ltd.

El acceso a los números se realiza a través de una API pública basada en el estándar JSON-RPC, disponible en `https://api.random.org/json-rpc/4/`.

Para acceder a la misma se debe registrar una cuenta en RANDOM.ORG y obtener una API Key.
Se pueden realizar solicitudes POST con parámetros como cantidad de números, tipo (entero, decimal, bits), rango, etc. La respuesta incluye los datos en formato JSON, y pueden ser normalizarse a números en $[0, 1)$ dividiendo por 256, para usarlos como si fueran generados por una distribución uniforme.

Las características de esta fuente son:

- **Tipo de aletoriedad:** verdadera, basada en fenómenos naturales (ruido atmosférico).
- **Acceso:** Mediante API key personal (registro gratuito con límites diarios).
- **Método de acceso:** API JSON-RPC compatible con muchos lenguajes.
- **Formato disponible:** Números enteros, decimales, secuencias, bits, etc.
- **Velocidad:** Alta, aunque limitada según el plan contratado.
- **Confiabilidad:** Alta, con respaldo académico (Trinity College Dublin).

2.1.3. Fuente seleccionada

La fuente elegida para trabajar en la siguiente parte fue **ANU Quantum Random Numbers Server**.
Entre las razones que llevaron a elegir dicha fuente se encuentran:

- Facilidad de uso, tiene una API a la que se accede fácilmente a través de una URL y se puede integrar de forma sencilla en el código *Python*.
- API bien documentada y sencilla de comprender.
- API accesible con formatos simples (como enteros de 8 bits) que pueden escalarse fácilmente a distribuciones uniformes en $[0, 1)$.
- La mécanica cuántica garantiza que los números provistos son imparciales, y el experimento está configurado de manera que se ejecuta automáticamente y no requiere intervención humana. Esto la hace particularmente adecuada para contextos académicos y científicos.

2.2. Pseudocódigo

Procedimiento EstimacionMonteCarloVolumen (int n, int b, region R, real δ)
Parámetros de entrada:

- n - tamaño de la muestra
- b - tamaño de batch
- R - región a calcular el volumen
- δ - nivel de confianza

Parámetros de salida:

- $\hat{X}$ - estimador del volumen
- $\hat{V}$ - estimador de la desviación estándar
- ω_1, ω_2 - intervalo de confianza para $(1 - \delta)$

Pseudocódigo:

1. $X, \hat{X}, \hat{V} = 0$
2. For $i = 0, ..., \frac{n}{b} + 1$ do
   a) Sortear coordenadas de b puntos $X_i = X_{bi}, X_{bi+1}, ..., X_{bi+b-1}$ siguiendo la distribución $U(0, 1)$ por coordenada
   b) Si $X_j \in R \rightarrow X = X + 1$ para cada X_j en la matriz X_i
3. $\hat{X} = X/n$
4. $\hat{V} = \hat{X}(1 - \hat{X})/(n - 1)$
5. $\omega_1, \omega_2 = I_1(X, n, \delta), I_2(X, n, \delta)$

El anterior pseudocódigo bosqueja un método de monte carlo para el cálculo de volumen de una región, siguiendo las pautas del pseudocódigo presentado en clase. Extendiendo el código de la entrega anterior, se agrega la noción de la ejecución por batches, definiendo un tamaño de batch b y operando con una matriz de b puntos de manera vectorial, en n/b iteraciones. Esto permite agilizar el proceso sin comprometer el uso de memoria. Por otra parte, se agrega el cálculo del intervalo de confianza según el método indicado y el uso de X como acumulador separado al estimador $\hat{X} = X/n$.

En resumidas cuentas, se reutilizó la versión del método implementado para el laboratorio 3, el cual mejoraba el método empleado en el laboratorio 2 debido a la implementación de batches. El único cambio importante realizado fue al sortear los puntos, leyéndolos de una tabla generada luego de consumir el servicio anteriormente mencionado.

Debido a limitaciones con el uso gratuito de la API, se implementó un script (no adjunto en la entrega) para consumir el servicio (pidiendo numeros uint16 de a batches de 1024), normalizar los valores (dividiendolos entre 65536) y agruparlos en un archivo *.csv* el cual se encuentra adjunto en la entrega. La tabla generada es de dimensiones $33962, 6$, es decir, cuenta con 33962 puntos de 6 coordenadas cada uno. Esta organización se definió con el objetivo de facilitar la lectura para el problema en concreto, al leer el archivo en tandas de largo 1000 hasta llegar al final.

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 de ejercicios anteriores (ya que es una adaptación del código anteriormente generado).

Cabe destacar que es necesario ejecutarlo con la tabla de valores *tabla.csv* ubicada en la misma carpeta, ya que de ahí se extraen los valores aleatorios.

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:** 33962

3.2. Resultados

En la experimentación de este laboratorio, se ejecutó el código del ejercicio 3.1 empleando por un lado la fuente de números pseudoaleatoria provista por la biblioteca *NumPy* en *Python* (la cual es la que se ha estado utilizando desde el comienzo del curso) y por otro, la fuente de números aleatoria mencionada en anteriores secciones.

Debido a la naturaleza del servicio (limita la cantidad de números que pueden obtenerse gratuitamente), el tamaño de muestra utilizado es de $33962,6$, es decir, 33962 puntos de 6 coordenadas. Con el objetivo de comparar la performance de ambas fuentes de datos, se ejecutaron ambas soluciones con dicho valor.

A continuación, se adjunta la tabla comparativa marcando en rojo los valores de interés mencionados anteriormente, concretamente los generados para esta entrega con el tamaño de muestra 33962. Se mantienen en la tabla valores de anteriores entregas, para comparar la precisión de los métodos:

N° de iteraciones	**Estimador ($\hat{X}$)**	**Desviación ($\hat{V}$)**	**IdeC (Chebyshev)**	**Tiempo (s)**	**Aleatoriedad**
10^4	0.0004000000	0.0001999700	0.0000583618 0.0027360494	0.16	Pseudo
33962	0,0003533360	0,0001019828	0,0001048558 0,0011899473	0.17	Pseudo
33962	0,0002650020	0,0000883236	0,0000667478 0,0010514912	0.30	Real
10^5	0.0002600000	0.0000509838	0.0001110070 0.0006088491	1.62	Pseudo
10^6	0.0002790000	0.0000167010	0.0002136403 0.0003643482	14.06	Pseudo
10^7	0.0002757000	0.0000052500	0.0002531995 0.0003001994	159.15	Pseudo
10^8	0.0002818400	0.0000016786	0.0002744325 0.0002894474	1568.21	Pseudo
184443973	0.0002819610	0.0000012362	0,0002764863 0,0002875440	2941.14	Pseudo

Tanto de la tabla como de las gráficas se puede observar el comportamiento esperado: los estimadores de volumen y desviación tienen comportamiento asintótico como de costumbre y los intervalos de confianza acotan el valor del estimador de volumen, acercandose cada vez más para mayor cantidad de iteraciones. El intervalo de Agresti-Coull converge mejor y más rápido que el de Chebyshev como se esperaba.

En resumen: El aumento del intervalo de confianza con ANU QRNG se debe a que estos números introducen variabilidad real, mientras que los pseudoaleatorios como NumPy pueden dar resultados más .^estables"pero artificialmente suaves.

4. Bibliografía

[1] *ANU Quantum Random Numbers*. Disponible en: `https://quantumnumbers.anu.edu.au/`.

[2] *RANDOM.ORG*. Disponible en: `https://www.random.org/`.