TP 2.1 - GENERADORES PSEUDOALEATORIOS

Lautaro Teta Musa
Universidad Tecnológica Nacional
lautarotetamusa@gmail.com

Ignacio Roca
Universidad Tecnológica Nacional
ignacioroca@gmail.com

Agustín Luzzini
Universidad Tecnológica Nacional
agusluzzini@gmail.com

Nazareno Necchi
Universidad Tecnológica Nacionala
nazanecchi.cer@gmail.com

13 de junio, 2024

ABSTRACT

Este informe presenta el desarrollo y análisis de generadores de números pseudoaleatorios, centrados en dos métodos específicos: el método de los Cuadrados Medios y el Generador Lineal Congruencial (GLC). El estudio incluye la implementación de estos generadores, la aplicación de pruebas estadísticas para evaluar su calidad, y la comparación de sus resultados con el generador de la librería Random de Python. El objetivo es demostrar la eficacia y las limitaciones de cada método en la generación de secuencias numéricas que simulan aleatoriedad.

***K*eywords** Simulación · Trabajo práctico · Generadores Pseudoaleatorios

1 Introducción

Los números aleatorios son secuencias de números que no siguen un patrón predecible y cuya generación está gobernada por la probabilidad. En otras palabras, cada número en la secuencia es independiente de los anteriores y tiene la misma probabilidad de ocurrir dentro de un rango específico. En cambio los números pseudoaleatorios son producidos por algoritmos deterministas que, bajo las mismas condiciones iniciales, generan secuencias idénticas. Los números pseudoaleatorios son fundamentales en diversas aplicaciones científicas y tecnológicas, desde simulaciones y modelado hasta criptografía y análisis de datos. Esta predictibilidad es una ventaja en términos de reproducibilidad de experimentos, pero también plantea desafíos en términos de asegurar la verdadera aleatoriedad de las secuencias generadas.

El propósito de este informe es explorar la generación de números pseudoaleatorios mediante la implementación de dos métodos clásicos: el método de los Cuadrados Medios y el Generador Lineal Congruencial (GLC). Ambos métodos se desarrollarán en Python 3.8 y se someterán a una serie de pruebas para evaluar su desempeño en términos de uniformidad y ausencia de patrones predecibles.

2 Conceptos Teóricos

En esta sección, se presentan los fundamentos teóricos de los métodos de generación de números pseudoaleatorios y las pruebas estadísticas utilizadas para evaluar su calidad. Se describen dos métodos específicos de generación: el método de los Cuadrados Medios y el Generador Lineal Congruencial (GLC). Además, se detallan cuatro pruebas estadísticas que se aplicarán para evaluar el rendimiento de estos generadores.

2.1 Método de los Cuadrados Medios

El método de los Cuadrados Medios es uno de los generadores de números pseudoaleatorios más antiguos y sencillos. Fue propuesto por John von Neumann en 1946. La idea básica es tomar un número semilla, elevarlo al cuadrado y

extraer una porción central de los dígitos resultantes para formar el siguiente número de la secuencia. Este proceso se repite para generar la secuencia de números pseudoaleatorios.

Algoritmo:

1. Seleccionar una semilla inicial x_0 con n dígitos.
2. Calcular $x_{i+1} = (x_i^2)$.
3. Extraer los n dígitos centrales de x_{i+1} para formar el siguiente número.
4. Repetir el proceso a partir del paso 2.

Ventajas:

- Simple de implementar.
- Requiere solo operaciones básicas de multiplicación y extracción de dígitos.

Desventajas:

- Puede entrar rápidamente en ciclos cortos.
- La calidad de los números generados depende críticamente de la semilla inicial.
- No es adecuado para aplicaciones que requieren alta calidad en la aleatoriedad.

2.2 Generador Lineal Congruencial (GLC)

El Generador Lineal Congruencial es uno de los métodos más utilizados para la generación de números pseudoaleatorios. Se basa en una fórmula lineal congruencial para producir una secuencia de números. La fórmula general es:

$$x_{n+1} = (a \cdot x_n + c) \mod m$$

donde:

- x es la secuencia de números pseudoaleatorios.
- a es el multiplicador.
- c es el incremento.
- m es el módulo.
- x_0 es la semilla inicial.

Algoritmo:

1. Seleccionar valores para a, c, m y una semilla inicial x_0.
2. Calcular el siguiente número en la secuencia usando la fórmula $x_{n+1} = (a \cdot x_n + c) \mod m$.
3. Repetir el proceso a partir del paso 2.

Ventajas:

- Fácil de implementar y computacionalmente eficiente.
- Con una elección adecuada de los parámetros a, c y m, puede generar secuencias de alta calidad.

Desventajas:

- La elección inadecuada de los parámetros puede resultar en ciclos cortos y patrones predecibles.
- Los números generados tienen una dependencia lineal que puede ser explotada en ciertos contextos (por ejemplo, en criptografía).

2.3 Pruebas Estadísticas para Evaluar Generadores

Para evaluar la calidad de los números pseudoaleatorios generados, se utilizan diversas pruebas estadísticas. A continuación, se describen cuatro pruebas comunes:

2.3.1 Prueba de la Media

Esta prueba verifica si la media de los números generados se aproxima al valor esperado. Para una secuencia de números en el intervalo [0, 1], la media esperada es 0.5.

Fórmula:

$$\text{Media} = \frac{1}{N}\sum_{i=1}^{N} x_i$$

Criterio: Si la media calculada está dentro de un intervalo de confianza alrededor de $0.5 \pm 5\%$, el generador pasa la prueba.

2.3.2 Prueba de la Varianza

Esta prueba verifica si la varianza de los números generados se aproxima a la varianza esperada. Para una secuencia de números en el intervalo [0, 1], la varianza esperada es 1/12.

Fórmula:

$$\text{Varianza} = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \text{Media})^2$$

Criterio: Si la varianza calculada está dentro de un intervalo de confianza alrededor de $1/12 \pm 5\%$, el generador pasa la prueba.

2.3.3 Prueba de Monobit

La prueba de Monobit evalúa la proporción de números por debajo de 0.5 y la proporción de números por encima de 0.5 en la secuencia de números generada. Para una secuencia equilibrada, la cantidad de números por debajo y por encima de 0.5 debería ser aproximadamente igual.

Fórmula:

$$S_n = \frac{1}{N}\sum_{i=1}^{N} x_i$$

donde x_i es el i-ésimo número de la secuencia, con $x_i = 1$ si el número es mayor o igual a 0.5, y $x_i = 0$ si es menor a 0.5.

Criterio: Si la proporción de números por debajo y por encima de 0.5 está dentro de un intervalo de confianza alrededor de $0.5 \pm 5\%$, el generador pasa la prueba.

2.3.4 Prueba de Bitmap

La prueba de Bitmap consiste en representar gráficamente la secuencia de bits generada. Para ello, cada valor obtenido entre 0 y 1 se multiplica por 256 para obtener una escala de grises. Se espera que la imagen resultante no revele patrones visibles y se asemeje a una distribución uniforme de puntos de diferentes tonalidades de gris.

Criterio: Si la imagen resultante muestra una distribución uniforme sin patrones visibles, el generador pasa la prueba.

3 Metodología

3.1 Implementación

Se desarrolló un código en el lenguaje de programación Python que recibe por argumentos los parámetros necesarios para realizar la simulación. Se utiliza la librería matplotlib que nos permite realizar gráficos que nos bridan información sobre la simulación. Para iniciar una simulación se ejecuta un comando de consola como se muestra a continuación. Los parámetros necesarios serán explicados en la sección Parametros

```
$ python cli.py –G <generador> –S <seed> <...args> –N <int> –T <test>
```

3.2 Parametros

Los parámetros utilizados en la simulación son los siguientes:

- **N:** Cantidad de números que vamos a generar.
- **S:** Semilla del generador.
- **T:** Test a realizar
  - **bitmap:** Genera una imagen de los números pseudoaleatorios.
  - **monobit:** Prueba de frecuencia.
  - **media:** Prueba de la media.
  - **varianza:** Prueba de la varianza.
- **G:** Generador utilizado
  - **cm:** Cuadrados Medios
    - **n:** Precisión.
  - **glc:** Generador Lineal Congruencial (GLC)
    - **m:** Módulo.
    - **a:** Multiplicador.
    - **c:** Incremento.
  - **lib:** Librería Random

3.3 Procedimiento

Se realizará la simulación con *N* números y una semilla *S*, para cada uno de los test *Bitmap,Monobit,Media,Varianza* y para cada uno de los generadores *Cuadrados Medios, GLC, Librería Random.* Y comparamos los resultados optenidos.

4 Experimentos

4.1 Test Bitmap

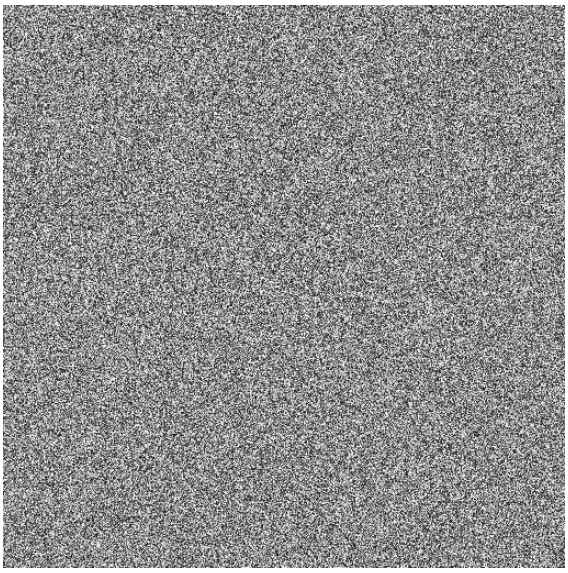

Figure 1: Gráfica de test Bitmap para generador de Cuadrados Medios

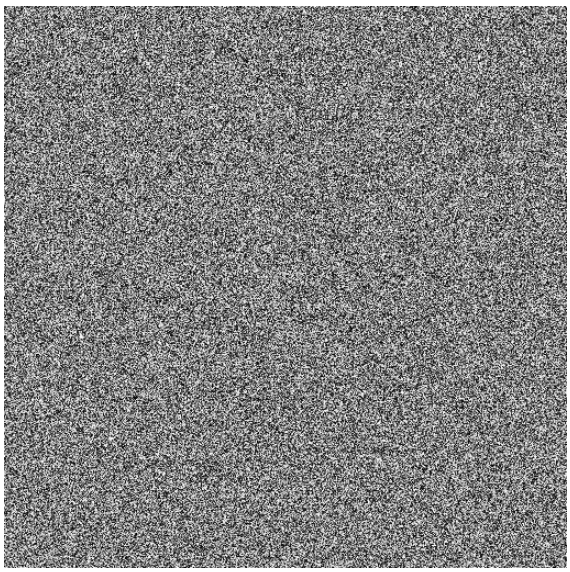

Figure 2: Gráfica de test Bitmap para generador GLC

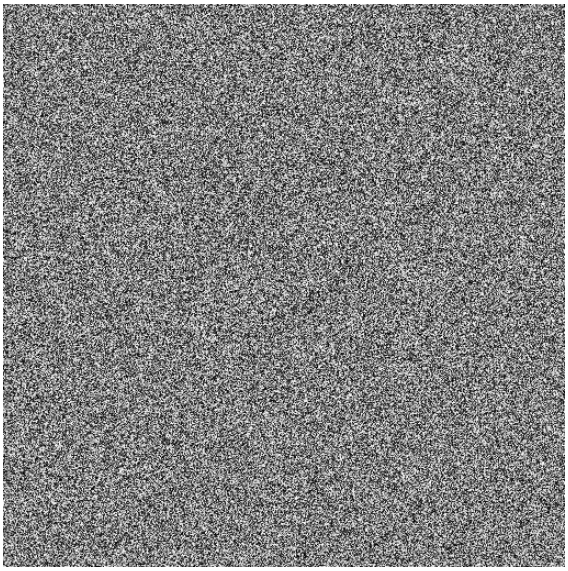

Figure 3: Gráfica de test Bitmap para generador de Librería Random

Observamos que las gráficas del test Bitmap para los tres generadores (Cuadrados Medios, GLC y Librería Random) muestran puntos en blanco , gris y negro distribuidos de manera aparentemente uniforme. No se observan patrones regulares o agrupaciones, lo que sugiere una buena aleatoriedad visual en la secuencia de bits generadas por estos métodos.

4.2 Test Monobit

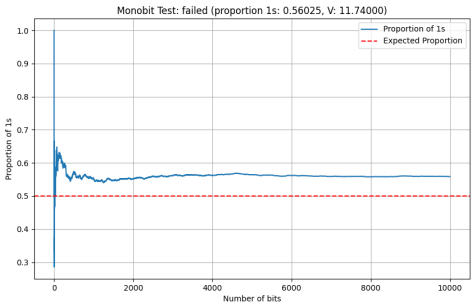

Figure 4: Gráfica de test Monobit para generador de Cuadrados Medios

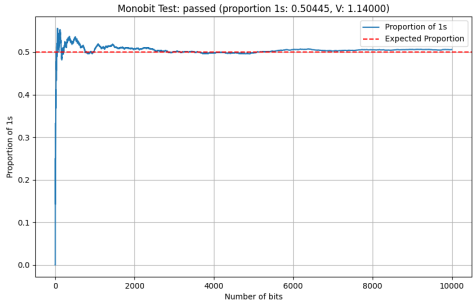

Figure 5: Gráfica de test Monobit para generador GLC

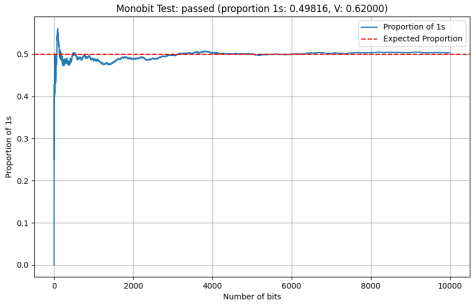

Figure 6: Gráfica de test Monobit para generador de Librería Random

Table 1: Tabla comparativa de generadores de números pseudoaleatorios para test Monobit

Generador	Tipo de Generador	Test Monobit	Pasó?
Cuadrados Medios	Pseudoaleatorio	0.56025	NO
GLC	Pseudoaleatorio	0.50445	SI
Librería Random	Pseudoaleatorio	0.49816	SI

Observamos que el generador de Cuadrados Medios no es adecuado para generar números pseudoaleatorios, ya que presenta una proporción de 1s (0.56025) significativamente mayor a la esperada, lo que indica una desviación importante de la aleatoriedad ideal y hace que falle el test. Por otro lado, tanto el generador GLC como el generador de la Librería Random pasan satisfactoriamente el test Monobit, con proporciones de 1s de 0.50445 y 0.49816, respectivamente, muy cercanas al valor ideal de 0.5. Esto sugiere que estos dos generadores producen secuencias de bits con una distribución adecuada y uniforme de 1s y 0s

4.3 Test Media

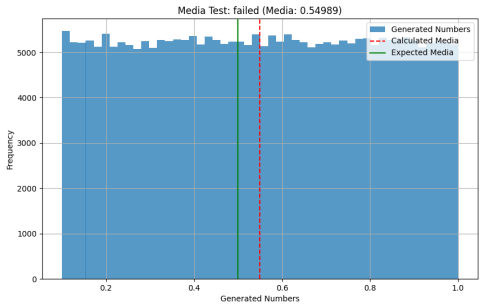

Figure 7: Gráfica de test Media para generador de Cuadrados Medios

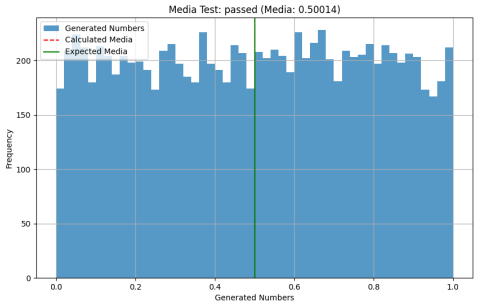

Figure 8: Gráfica de test Media para generador GLC

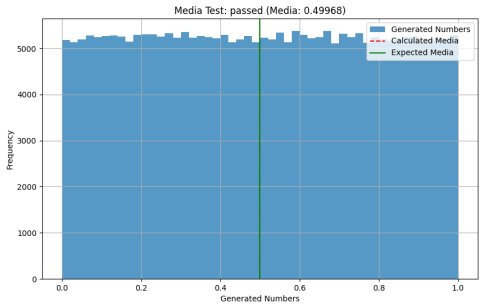

Figure 9: Gráfica de test Media para generador de Librería Random

Table 2: Tabla comparativa de generadores de números pseudoaleatorios para test Media

Generador	Tipo de Generador	Test Media	Pasó?
Cuadrados Medios	Pseudoaleatorio	0.54989	NO
GLC	Pseudoaleatorio	0.50014	SI
Librería Random	Pseudoaleatorio	0.49968	SI

Observamos que el generador de Cuadrados Medios no pasó el test de media, con una media calculada de 0.54689, desviándose significativamente de la media esperada de 0.5. Esto sugiere que su algoritmo no es adecuado para generar números pseudoaleatorios con una distribución uniforme adecuada. Por otro lado, el generador GLC, con una media calculada de 0.50014, y el generador de la Librería Random, con una media de 0.49968, pasaron el test de media, indicando que sus métodos de generación son adecuados para producir una distribución uniforme de números pseudoaleatorios. Los resultados muestran que los generadores GLC y Librería Random son confiables para generar números pseudoaleatorios con una buena distribución uniforme, mientras que el método de Cuadrados Medios no cumple con este criterio. En aplicaciones donde la uniformidad en la distribución de números es crucial, como simulaciones y criptografía, los generadores GLC y Librería Random serían las opciones preferidas.

4.4 Test Varianza

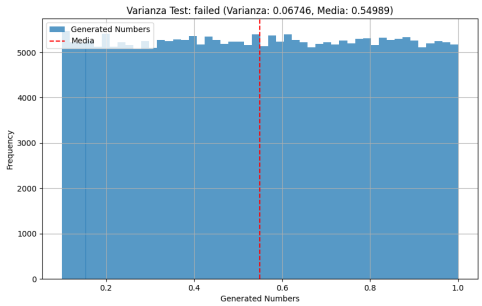

Figure 10: Gráfica de test Varianza para generador de Cuadrados Medios

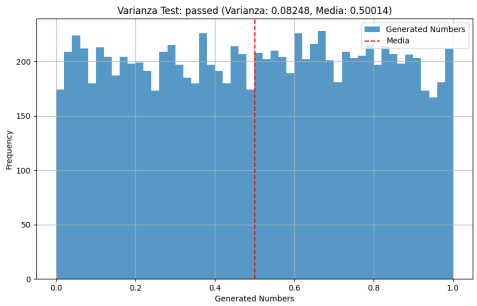

Figure 11: Gráfica de test Varianza para generador GLC

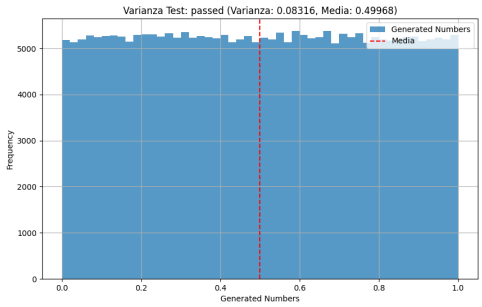

Figure 12: Gráfica de test Varianza para generador de Librería Random

Table 3: Tabla comparativa de generadores de números pseudoaleatorios para test Varianza

Generador	Tipo de Generador	Test Varianza	Pasó?
Cuadrados Medios	Pseudoaleatorio	0.06746	NO
GLC	Pseudoaleatorio	0.08248	SI
Librería Random	Pseudoaleatorio	0.08316	SI

En el análisis del test de varianza, observamos que el generador de Cuadrados Medios no pasó la prueba, presentando una varianza calculada de 0.06746, lo cual dista significativamente de la varianza esperada (0.08333) y, por lo tanto, es considerada inadecuada para aplicaciones que requieren números pseudoaleatorios confiables. En contraste, el generador GLC con una varianza de 0.08248 y el generador de la Librería Random con una varianza de 0.08316, ambos pasaron el test de varianza, demostrando que son capaces de producir una distribución uniforme de números pseudoaleatorios. Estos resultados sugieren que los generadores GLC y Librería Random son efectivos para generar números pseudoaleatorios con una distribución uniforme adecuada, a diferencia del método de Cuadrados Medios.

5 Conclusiones

Los resultados obtenidos revelaron que el método de los Cuadrados Medios presenta serias limitaciones en cuanto a su capacidad para generar números pseudoaleatorios de alta calidad. Su propensión a entrar rápidamente en ciclos cortos y su sensibilidad a la elección de la semilla inicial lo hacen inadecuado para aplicaciones que requieren una aleatoriedad rigurosa. Por otro lado, el Generador Lineal Congruencial mostró ser una opción más robusta y confiable. Este método fue capaz de producir secuencias de números pseudoaleatorios con una distribución uniforme y una buena aleatoriedad, lo que lo hace adecuado para una amplia gama de aplicaciones. Además, se compararon los resultados obtenidos con el generador de números pseudoaleatorios integrado en la librería Random de Python. Tanto el GLC

como la implementación de la Librería Random superaron las pruebas estadísticas aplicadas, lo que sugiere que son opciones viables y confiables para la generación de números pseudoaleatorios en aplicaciones prácticas. En conclusión, para aplicaciones que requieren una alta calidad en la aleatoriedad, como simulaciones y criptografía, se recomienda utilizar el Generador Lineal Congruencial o la Librería Random de Python en lugar del método de los Cuadrados Medios. Estos generadores han demostrado ser más robustos y confiables, proporcionando secuencias de números pseudoaleatorios con una distribución uniforme y una buena aleatoriedad, lo que los hace adecuados para una variedad de escenarios de uso.

References

[1] https://tereom.github.io/est-computacional-2018/numeros-pseudoaleatorios.html

[2] https://www.random.org/analysis/

[3] https://www.random.org/analysis/Analysis2005.pdf