TP 2.1 - GENERADORES PSEUDOALEATORIOS

Ramiro Di Giacinti
Ingeniería en Sistemas de Información
Universidad Tecnológica Nacional - FRRO
Zeballos 1341, S2000, Argentina
dia.digiacinti.ramiro@gmail.com

Bruno Mollo
Ingeniería en Sistemas de Información
Universidad Tecnológica Nacional - FRRO
Zeballos 1341, S2000, Argentina
dia.mollo.bruno@gmail.com

Facundo Braida
Ingeniería en Sistemas de Información
Universidad Tecnológica Nacional - FRRO
Zeballos 1341, S2000, Argentina
facundobraida98@gmail.com

Lucía Cappellini
Ingeniería en Sistemas de Información
Universidad Tecnológica Nacional - FRRO
Zeballos 1341, S2000, Argentina
luciacappll@gmail.com

Adriel Gorosito
Ingeniería en Sistemas de Información
Universidad Tecnológica Nacional - FRRO
Zeballos 1341, S2000, Argentina
adrielgorosito14@gmail.com

March 28, 2024

ABSTRACT

Un generador de números pseudoaleatorios es un algoritmo que produce una secuencia de números aparentemente aleatorios a partir de una semilla inicial y una serie de cálculos matemáticos predefinidos. Aunque estos números no son completamente aleatorios, se utilizan en muchas aplicaciones donde se requiere aleatoriedad, como en la simulación de sistemas complejos o en la criptografía. En este informe analizamos tres generadores de números pseudoaleatorios (GLC, Randu y MCM) por medio de cuatro tests (mapa de bits, chi cuadrado, sumas solapadas y Kolmogorov-Smirnov). El objetivo es asegurar que realmente son generadores de números pseudoaleatorios (o refutar, en caso de que los tests den resultados no favorables).

1 Introducción

1.1 Aleatoriedad

Cuando nos referimos a que un hecho es aleatorio o que una cadena de números y letras es aleatoria, nos referimos a la aparente falta de un patrón u orden para poder determinar lo que ocurrirá o que saldrá en el futuro, dicho en otras palabras, es algo impredecible. Sin embargo, si se conoce la distribución de probabilidades de que esos eventos ocurran, entonces los resultados de las distintas salidas es mayormente predecible.

1.2 Número Pseudoaleatorio

Un numero pseudoaleatorio tal como su nombre lo indica, no es completamente aleatorio: es un número generado en un proceso que parece producir números al azar, pero no lo hace realmente. Presentan las mismas características que los números aleatorios, con la gran distinción de que si a los generadores de estos tipos de números se les presentan la mismas condiciones iniciales, el resultado que se obtendrá siempre sera el mismo.

Este tipo de números es ampliamente utilizado en los campos de la estadística, criptografía y la simulación para la generación de grandes secuencias de números aleatorios que permitan el estudio de escenarios o problemáticas de una manera practica cuando el método teórico no puede ser utilizado. Sin embargo, esto resulta ser uno de los mayores inconvenientes con los generadores pseudoaleatorios. Dependiendo de como sean obtenidos estos valores, puede ser que involuntariamente generen patrones y por ende, dejen de ser aleatorios.

1.3 Métodos de Generación

Existen dos grandes grupos en cuanto a la generación de estas secuencias de números:

Métodos Físicos Pueden usarse objetos como dados, ruletas, etc para generar secuencias realmente aleatorias. También pueden utilizarse valores de mediciones de distintas índoles (desintegración radioactiva, ruido térmico, ruido blanco).

Métodos Digitales Utilizan algoritmos generadores de números para determinar una secuencia de números en base a un conjunto de parámetros y un valor inicial llamado "seed". Debido a su naturaleza este tipo de generadores son deterministas, es decir, conociendo la entrada de este algoritmo siempre se obtendrá la misma salida.

Pondremos bajo estudio a tres métodos de generación de secuencias de números distintos: GLC de Microsoft Visual Basic, RANDU de IBM y finalmente el método Middle Square propuesto por Jon Von Neuman

1.4 Generadores

1.4.1 GLC

Un generador lineal congruencial (GLC) es un algoritmo que permite obtener una secuencia de números pseudoaleatorios calculados con una función lineal definida a trozos discontinua. Es uno de los métodos más antiguos y conocidos para la generación de números pseudoaleatorios.

La secuencia de números se genera en base al siguiente algoritmo:

$$X_i = (aX_{i-1} + c)\ mod\ m, \quad 0 \leq X_i \leq m - 1$$

Donde:

- a: Multiplicador
- c: Incremento
- m: Módulo
- X_i: Secuencia X de números en posición i

La correcta elección de los parámetros a, c y m determina que tan viable es el algoritmo para proveer secuencias de números que sean realmente aleatorios y no puedan ser descifradas fácilmente.

Se eligió como GLC bajo estudio al utilizado por Microsoft Visual Basic, cuyos parámetros son:

Microsoft Visual Basic $a = 1140671485,\ c = 12820163,\ m = 2^{24}$

1.4.2 RANDU

El generador RANDU es un tipo de GLC, por lo tanto, no hace falta mucha explicación. Es muy similar al GLC de Microsoft Visual Basic, solo varían sus parámetros:

RANDU $a = 2^{16} + 3,\ c = 0,\ m = 2^{31}$

1.4.3 Middle Square

El método "Middle Square" es un algoritmo de generación de números pseudoaleatorios que fue propuesto por primera vez por John von Neumann en 1949.

El método se basa en tomar un número inicial de "n" dígitos, elevarlo al cuadrado, y luego tomar los "n" dígitos centrales del resultado como el siguiente número pseudoaleatorio. Este proceso se repite para generar una secuencia de números pseudoaleatorios.

1.4.4 Python3

Como plus, decidimos utilizar el generador de números pseudoaleatorios de Python3 (proveniente de la función random). El algoritmo que utiliza este generador es muy difícil de explicar (utiliza el algoritmo Mersenne Twister). Sin embargo, a forma de spoiler, resulta ser uno de los mejores generadores. Por lo tanto, al ser de tan buena calidad, decidimos someterlo a los tests para así poder comparar los demás generadores con él.

2 Metodología

2.1 Tests de evaluación

2.1.1 Test de mapa de bits

El test de mapa de bit (bitmap) es una prueba más bien visual que no parece muy rigurosa pero permite detectar patrones dentro de un generador. Consiste en generar una imagen de mapa de bits en la que cada píxel puede ser blanco o negro (su color depende del numero pseudoaleatorio que se genere). Si existe algún ciclo en el generador, este se puede terminar viendo reflejado en la imagen final. En nuestro caso, decidimos realizar bitmaps de 1000 por 1000 píxeles, ya que permite una clara visualización.

2.1.2 Test de Sumas Solapadas

La prueba de sumas superpuestas consiste en generar una secuencia de números aleatorios y sumarlos en grupos superpuestos. Inicialmente, se debe definir el tamaño de cada subsecuencia a sumar (ventana) y el avance dentro de la secuencia en cada suma (paso). Posteriormente se considerarán los resultados de cada suma dento de un nuevo conjunto. La distribución de este conjunto determinará la calidad del generador. Si el resultado de la prueba de sumas superpuestas es una distribución normal, significa que el generador de números aleatorios está funcionando como se esperaba. La distribución normal es la distribución esperada para una secuencia verdaderamente aleatoria, así que si la prueba de sumas superpuestas produce una distribución normal, entonces es probable que el generador de números aleatorios esté produciendo números aleatorios. Sin embargo, la prueba de sumas superpuestas no es una prueba perfecta. Es posible que un generador de números aleatorios produzca una distribución normal aunque no sea verdaderamente aleatorio. Esto puede ocurrir si el generador de números aleatorios se limita a repetir una secuencia de números o si utiliza un algoritmo muy simple para generar números.

2.1.3 Test de chi cuadrado

El test de chi cuadrado es una técnica estadística que se utiliza para determinar si un conjunto de datos sigue una distribución esperada. Es comúnmente utilizado para verificar la calidad de los generadores de números pseudoaleatorios, aunque también tiene otros usos.

En el contexto de estos generadores, se utiliza para comparar la frecuencia observada de los números generados con la frecuencia esperada de acuerdo con la distribución teórica. Si las frecuencias observadas no difieren significativamente de las frecuencias esperadas, entonces se considera que el generador de números pseudoaleatorios es válido.

Para realizar este test, se divide el rango de valores posibles de los números generados en varios intervalos y se cuenta cuántos números caen en cada intervalo. Luego, se compara esta distribución de frecuencia con la distribución de frecuencia esperada utilizando una prueba de hipótesis de chi cuadrado.

- Si el valor calculado de chi cuadrado es menor que el valor crítico de chi cuadrado para un nivel de significancia dado, entonces se acepta la hipótesis nula de que los números generados tienen una distribución similar a la esperada.
- Si el valor calculado es mayor que el valor crítico, se rechaza la hipótesis nula y se concluye que los números generados no tienen una distribución adecuada.

En resumen, el test de chi cuadrado evalúa si los números generados por un generador de números pseudoaleatorios se distribuyen uniformemente a través del rango de valores posibles, y por lo tanto, se puede utilizar para evaluar la calidad de un generador en términos de uniformidad.

2.1.4 Test de Kolmogorov-Smirnov

El test de Kolmogorov-Smirnov es un método estadístico utilizado para evaluar si una muestra de datos sigue una distribución de probabilidad específica. En el contexto de un generador de números pseudoaleatorios, se utiliza para evaluar la calidad de los números generados.

El test compara la distribución empírica de la muestra generada por el generador de números pseudoaleatorios con la distribución teórica esperada. Para hacer esto, se calcula la función de distribución acumulativa (FDC) para ambos conjuntos de datos y se compara la distancia entre ellas.

Si la distancia es pequeña, significa que los números generados tienen una distribución similar a la esperada, lo que sugiere que el generador es de buena calidad. Sin embargo, si la distancia es grande, significa que los números generados no siguen la distribución esperada y puede indicar que el generador no es adecuado para su uso en aplicaciones que requieren números aleatorios.

3 Resultados

3.1 Test de mapa de bits

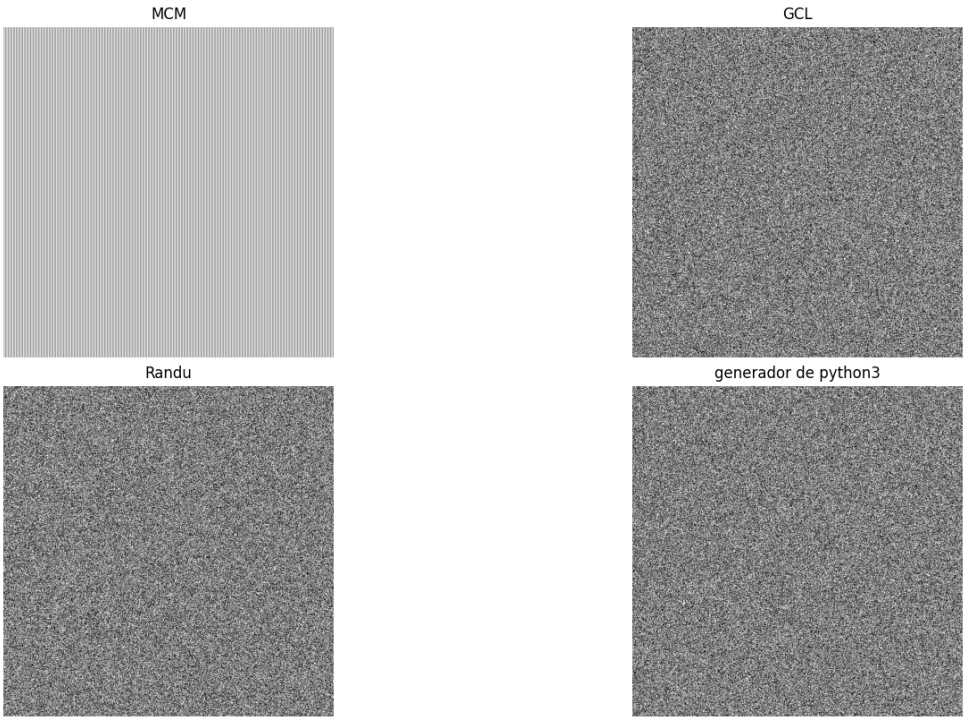

Figure 1: Mapas de bits creados a partir de los cuatro generadores

Podemos observar que GLC, RANDU y el generador de Python3 producen imágenes que parecen ruido aleatorio, es decir, que a simple vista no se puede observar ningún patrón evidente. Por el otro lado MCM genera unas claras lineas verticales, que evidencia los ciclos en este algoritmo.

Con este resultado se tiene un indicio de que, a diferencia de los otros tres, el generador de MCM no es tan bueno como parece. Sin embargo, es muy apresurado sacar una conclusión con un único test, por lo que es necesario seguir probando para poder asegurar con certeza de que el generador no es muy factible.

3.2 Test de Sumas Solapadas

En la siguiente figura se exponen las distribuciones de los conjuntos conformados por los resultados de las sumas solapadas para las secuencias de cada generador.

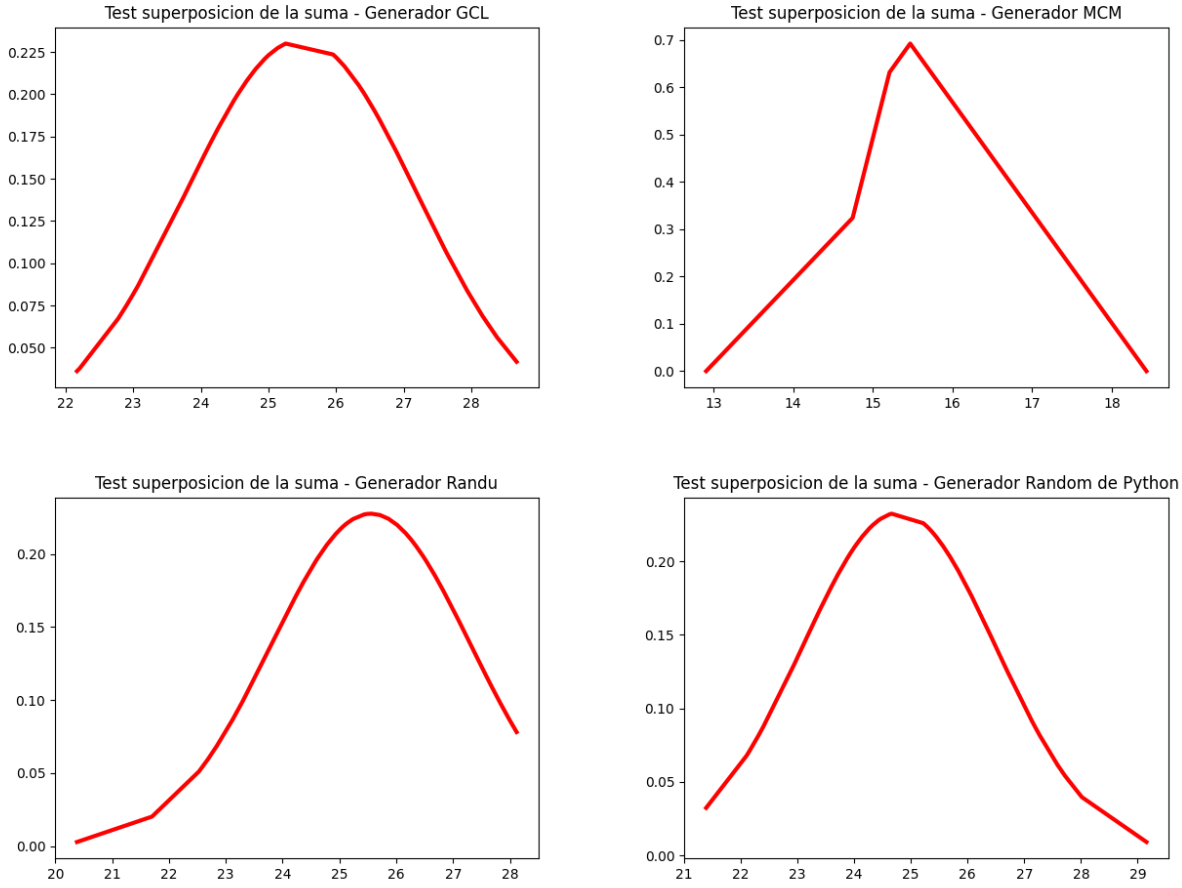

Figure 2: Distribución del conjunto de resultados de sumas solapadas para los cuatro generadores

En este caso se puede observar fácilmente como tres de los cuatro generadores tiene forma de una campana de Gauss, siendo nuevamente el atípico el generador MCM. Esto se debe a que puesto que los números que se generan caen bajo cierto patrón, entonces en las sucesivas sumas se repetirán los valores obtenidos. Gráficamente esto se ve representado por la poca suavidad que posee la función.

El resto de los generadores no presentan este inconveniente, sin embargo, el RANDU no presenta las características de una distribución normal puesto que no tiene la simetría requerida (en este caso presenta una asimetría hacia la izquierda). Esto puede deberse a que este algoritmo beneficia valores mayores y por lo tanto, las sumas tenderán también a ser mayores.

3.3 Test de chi cuadrado

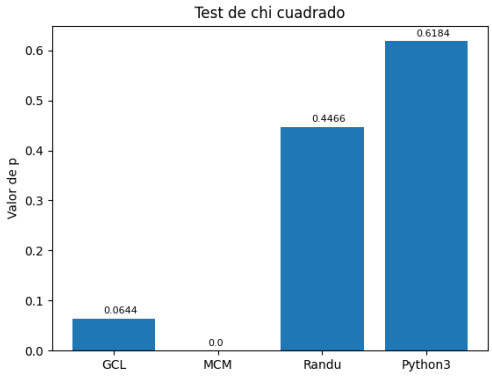

Figure 3: Comparación de los valores de p para cada generador

En la gráfica se pueden ver los valores de p obtenidos para cada secuencia de números pseudoaleatorios generada. Se plantea como nivel de significancia el valor 0.05. Por lo tanto, se busca que el valor de p obtenido sea mucho mayor al valor de significancia. Si esto se cumple, no hay evidencias para decir que el generador no es factible. Además, mientras mayor sea el valor de p, mayor será la calidad del generador.

El valor de p obtenido para la secuencia generada con GLC es de 0.064, lo que indica que la secuencia es aleatoria al nivel de significancia. Sin embargo, es importante tener en cuenta que un valor de p cercano a 0.05 indica que la calidad de la secuencia puede ser marginal, por lo que en general se prefieren valores de p más altos.

Sin embargo, para la secuencia generada con MCM se obtuvo un valor de p de 1.63e-14, lo que indica que la secuencia no es aleatoria al nivel de significancia. En otras palabras, hay suficiente evidencia para rechazar la hipótesis nula de que la secuencia es aleatoria. Este resultado sugiere que el generador MCM no es adecuado para generar números pseudoaleatorios de alta calidad.

Para los otros dos generadores (Randu y el utilizado por Python), los valores de p son 0.446 y 0.6184, respectivamente. No hay evidencia para rechazar la hipótesis nula de que la secuencia es aleatoria.

De forma general, estos resultados sugieren que el GLC, Randu y el generador de Python son adecuados para generar números pseudoaleatorios de alta calidad, mientras que el MCM puede no ser muy adecuado.

3.4 Test de Kolmogórov-Smirnov

En la siguiente figura se exponen las gráficas provistas por cada generador sometido al el test de Kolmogórov-Smirnov.

La probabilidad que se usa en el test de Kolmogórov-Smirnov es la probabilidad de que la distribución acumulada empírica (la función de distribución acumulada construida a partir de la secuencia de números aleatorios) difiera significativamente de la distribución acumulada teórica (la función de distribución acumulada que se espera de un generador de números pseudoaleatorios). Esta probabilidad es la probabilidad de observar una estadística D (la distancia máxima entre las dos funciones de distribución acumulada) tan grande como la que se observa en la secuencia de números aleatorios, asumiendo que la distribución teórica es verdadera.

La probabilidad se calcula utilizando la distribución de Kolmogorov-Smirnov, que da la probabilidad acumulada de que la estadística D sea menor o igual que un valor dado. Por lo tanto, es una probabilidad acumulada. Si esta probabilidad es menor que un umbral de significación previamente establecido (en este caso, se estableció el valor de 0.05), entonces se rechaza la hipótesis nula de que la distribución teórica es verdadera, y se concluye que el generador de números pseudoaleatorios no pasa el test de Kolmogorov-Smirnov.

Entonces, en las gráficas, todas (menos la del generador MCM) siguen una distribución muy similar a la esperada. En el caso de la gráfica del test realizado para el generador MCM, se puede ver que difiere muchísimo a lo esperado, concluyéndose así que el generador MCM no pasa el test de Kolmogórov-Smirnov, mientras que las otros tres generadores sí lo hacen.

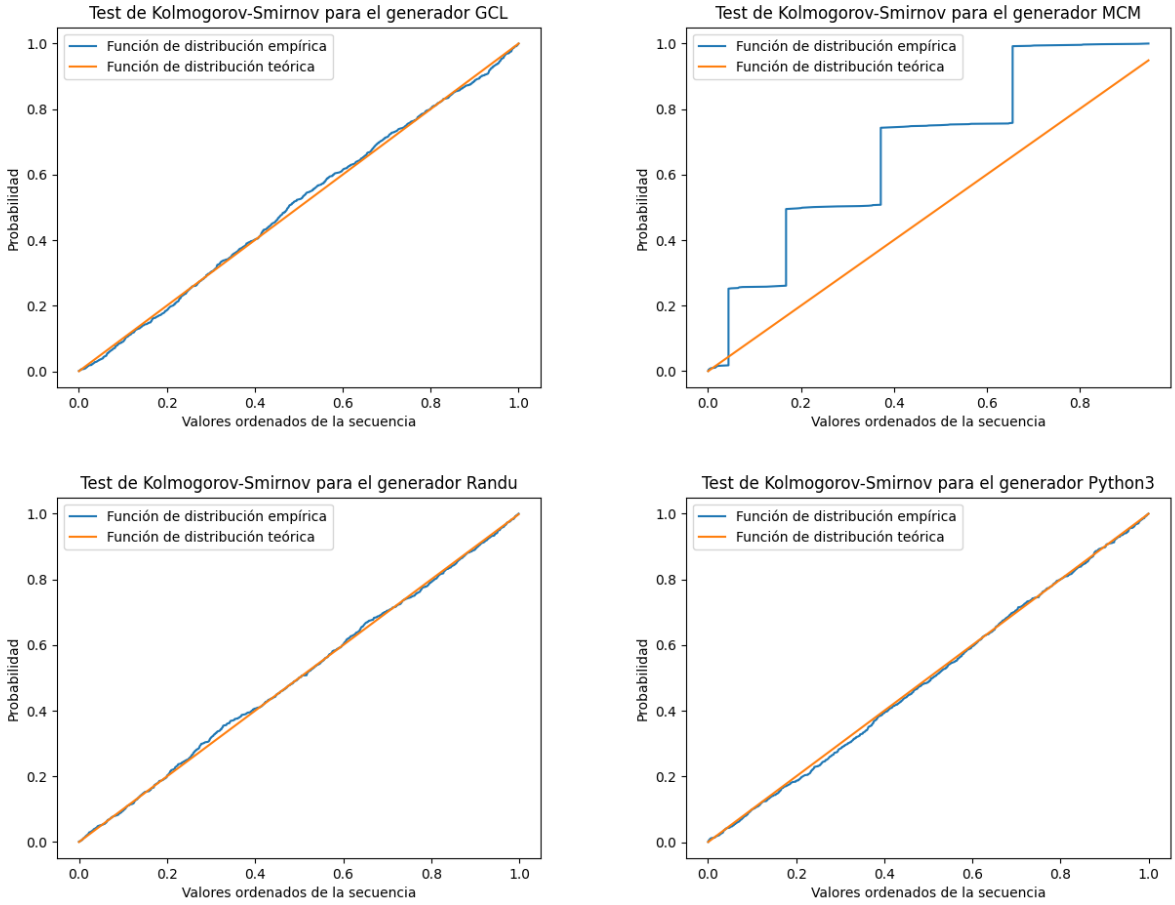

Figure 4: Gráficas del test de Kolmogórov-Smirnov para los cuatro generadores

4 Conclusión

Con los cuatro tests realizados, se concluye que los generadores GLC, Randu y el de Python son de buena calidad. Esto es debido a que los tres mencionados superaron las pruebas realizadas, por ende no hay suficiente evidencia para decir lo contrario.

No es posible decir lo mismo sobre el generador MCM. El susodicho generador no superó con éxitos ninguna prueba: falló todas. Es un generador de muy mala calidad y por consecuencia hay que tener mucho cuidado en donde y en qué contexto se utiliza.

Se inserta una tabla a continuación en forma de resumen del resultado de cada test para cada generador:

Generador	Test de mapa de bits	Test de sumas solapadas	Test chi cuadrado	Test Kolmogorov-Smirnov
GLC	✓	✓	✓	✓
MCM	✗	✗	✗	✗
Randu	✓	✓	✓	✓
Python	✓	✓	✓	✓

Table 1: Tabla de resultados de los tests para cada algoritmo

Volviendo a los generadores que sí superaron las pruebas, de entre los tres, se concluye que el mejor es el de Python3. La secuencia de números pseudoaleatorios utilizada fue generada por la función "random", la cual utiliza un generador de números pseudoaleatorios basado en el algoritmo Mersenne Twister. Este generador es considerado de muy alta calidad y es ampliamente utilizado en la mayoría de los lenguajes de programación y software de simulación. Este

algoritmo es capaz de producir secuencias de números pseudoaleatorios de gran calidad con una buena uniformidad y periodos largos, lo que lo hace adecuado para una amplia gama de aplicaciones. Sin embargo, no extenderemos más sobre este generador debido a su gran complejidad, por lo que se invita al lector a indagar más sobre el tema por su propia cuenta.

Finalizando, se concluye que el mejor generador de números pseudoaleatorios es el utilizado por Python (el Mersenne Twister), seguido por los generadores GLC y Randu (sin un orden en específico). Además, se recomienda evitar el uso del generador MCM.

5 Referencias

- Código Simulación: https://github.com/Ramiro-DG/Simulacion2023/tree/main/RNG
- https://tereom.github.io/est-computacional-2018/numeros-pseudoaleatorios.html
- https://www.random.org/analysis/
- https://en.wikipedia.org/wiki/Diehard_tests