TP 2.1 - GENERADORES DE NÚMEROS PSEUDOALEATORIOS

Leilen Avila
Legajo: 41610
Mail: leilenavila@gmail.com
UTN - FRRO
Zeballos 1341, S2000

Natalia Fernandez
Legajo: 44758
Mail: nata.fernandez77@gmail.com
UTN - FRRO
Zeballos 1341, S2000

16 de Mayo, 2020

ABSTRACT

En este trabajo tenemos como objetivo analizar diferentes generadores de números pseudoaleatorios. Analizaremos a través de diferentes muestras, generadores congruenciales y no congruenciales, asi como el propio generador que viene contenido en el lenguaje de programación Python y un generador de números aleatorios físico. Sacaremos nuestras propias conclusiones en base a las muestras obtenidas y luego procederemos a someterlos a diferentes pruebas para validar que tan eficientes son, a través de diferentes métodos matemáticos y estadísticos.

1. Introducción

Un número aleatorio es aquel obtenido al azar, es decir, que todo número tenga la misma probabilidad de ser elegido y que la elección de uno no dependa de la elección del otro. El ejemplo clásico más utilizado para generarlos es el lanzamiento repetitivo de una moneda o dado ideal no trucado. Los números aleatorios permiten a los modelos matemáticos representar la realidad. En general cuando se requiere una impredecibilidad en unos determinados datos, se utilizan números aleatorios.

Por el otro lado, los números pseudoaleatorios son unos números generados por medio de una función (determinista, no aleatoria) y que aparentan ser aleatorios. Estos números pseudoaleatorios se generan a partir de un valor inicial aplicando iterativamente la función. La sucesión de números pseudoaleatorios es sometida a diversos tests para medir hasta qué punto se asemeja a una sucesión aleatoria. Las sucesiones de números pseudoaleatorios son más rápidas de generar que las de números aleatorios. Si las personas tenemos dificultad en generar números aleatorios, mucho más la tiene un ordenador, la dificultad está en que un ordenador es tan “torpe“ que no sabe generarlos. Por eso usan números pseudoaleatorios, que para nuestro fin es lo mismo, pues nadie los puede predecir.

En este trabajo practico resumiremos el diseño y los principios de prueba de distintos generadores. Generaremos números pseudoaleatorios a partir de distintos métodos, analizaremos y testearemos el resultado, y además, lo compararemos con la función random que provee el lenguaje Python y con un generador físico proporcionado por la organización Random.org. Por lo tanto, en este trabajo analizaremos las propiedades de 6 diferentes generadores los cuales se mencionan a continuación:

- GCL RAND
- GCL RANDU
- GCL Mixto
- Metodo de los cuadrados medios
- Random de Python
- Random.org (Generador físico)

2. Generadores de números aleatorios

Un generador de números aleatorios (RNG) es un dispositivo informático o físico diseñado para producir secuencias de números sin orden aparente, es decir, que no se pueden predecir razonablemente mejor que por una posibilidad aleatoria. Los generadores de números aleatorios pueden ser verdaderos generadores de números aleatorios de hardware (HRNG), que generan números genuinamente aleatorios, o generadores de números pseudoaleatorios(PRNG), que generan números que parecen aleatorios, pero en realidad son deterministas, y pueden reproducirse si el estado del PRNG es conocido.

Varias aplicaciones de aleatoriedad han llevado al desarrollo de varios métodos diferentes para generar datos aleatorios, de los cuales algunos han existido desde la antigüedad, entre cuyos rangos se encuentran ejemplos "clásicos" bien conocidos, que incluyen el lanzamiento de dados, el lanzamiento de monedas, el barajado de naipes, el uso de tallos de milenrama(para adivinación) en el IChing, así como innumerables otras técnicas. El método utilizado mide algún fenómeno físico que se espera que sea al azar y luego compensa las posibles sesgos en el proceso de medición. Debido a la naturaleza mecánica de estas técnicas, la generación de grandes números de números suficientemente aleatorios requirió mucho trabajo y tiempo. Por lo tanto, los resultados a veces se recopilan y distribuyen como tablas de números aleatorios.

2.1. Historia de la generación de números aleatorios por computadora

El origen formal de la generación de números aleatorios por computadora comienza en la década de los años 40 con el nacimiento del método de Montecarlo, y von Neumann, Metropolis, Ulam y Lehmer pueden ser nombrados como los pioneros en este campo. John von Neumann conjeturó el potencial de los ordenadores para tratar problemas estocásticos en 1945. Durante los cuarenta, la simulación de procesos estocásticos permaneció restringida al proyecto secreto del Departamento de Defensa de Estados Unidos.

La publicación de "The Monte Carlo method" por N. Metropolis y Stanislaw M. Ulam en 1949 denota el inicio de la historia oficial del método. Dos años más tarde, D.H. Lehmer propuso el generador lineal de congruencia, el cual, con pequeñas modificaciones propuestas por Thomson y Rotenberg, ha llegado a convertirse en el método para la generación de números aleatorios más utilizado en la actualidad. Aunque, originalmente, el método de Montecarlo fue implementado usando ruletas y dados en los problemas de difusión de los neutrones, su auge y creciente uso se debe a que posteriormente se empezaron a emplear números aleatorios generados por computadora. Antes de la llegada de las computadoras, los números aleatorios eran generados por dispositivos físicos.

En 1939, Kendall y Babington-Smith publicaron 100.000 dígitos aleatorios obtenidos con un disco giratorio iluminado con una lámpara flash. En 1955, la Rand Corporation publicó un millón de dígitos producidos controlando una fuente de pulsos de frecuencia aleatoria. Estos se encuentran disponibles en cintas magnéticas de la Rand.

2.2. Generadores físicos de números aleatorios

Los generadores de números verdaderamente aleatorios miden algunos procesos físicos impredecibles o, al menos, difíciles de predecir y utilizar los resultados para crear una secuencia de números aleatorios. O bien se basan en los valores impredecibles a los que se puede acceder desde el software en la computadora o crean la secuencia en un dispositivo de propósito especial que lo alimenta en el sistema operativo.

El proceso de recolección de datos impredecibles generalmente se llama **recolección de entropía**. Algunas de las fuentes de entropía estándar a las que el sistema operativo puede tener acceso incluyen los datos de la tarjeta de sonido, los tiempos de acceso al disco, el tiempo de las interrupciones o los datos de interacción del usuario, como el movimiento del mousse o pulsaciones de teclas.

Hay generadores físicos de números verdaderamente aleatorios basados en principios diferentes, tales como los sistemas caóticos, ruido térmico en los circuitos electrónicos, osciladores de libre funcionamiento, o parámetros biométricos como algunos ejemplos.

Los generadores de números aleatorios cuánticos son un caso particular en las que los datos son el resultado de un suceso cuántico. A diferencia de otros sistemas físicos donde la incertidumbre es el resultado de un conocimiento incompleto del sistema, la aleatoriedad real es una parte esencial de la mecánica cuántica como la conocemos.

2.2.1. Recolección de entropía

El uso de un generador PRNG o RNG depende del sistema y de la velocidad necesaria de generación de los números aleatorios. Los RNG suelen bloquear su generación hasta que se ha generado la suficiente entropía como para que el número generado sea aleatorio. Por otro lado los PRNG tienen mayor capacidad de generación de números aleatorios pero tienden a reducir la entropía del sistema, siendo cada vez más predecibles.

Para evitar que los PRNG sean predecibles, periódicamente se regenera la entropía del sistema cambiando la semilla a un número generado por un RNG. Si logramos que todos los números tengan la misma probabilidad de ocurrencia, su distribución de probabilidad sería uniforme, y estaríamos alcanzando la máxima entropía posible.

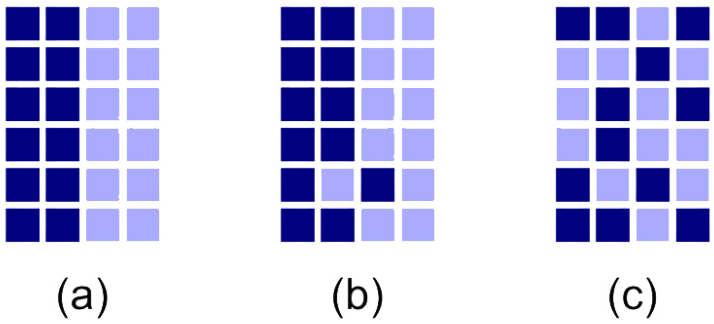

Figura 1: Entropía de Shannon.

La figura (a) presenta una baja entropía, y la figura (c) presenta una entropía alta, estando (b) en medio de ellas. Si se agotara la entropía del sistema, se continuaría con la generación de números aleatorios pero estos serían más susceptibles de ser predichos, perjudicando la seguridad del sistema que los utilice.

Para evitar que esto pase, periódicamente se genera una nueva semilla y se reinicia el proceso de generación de números pseudoaleatorios. Esto es muy útil en la generación de datos aleatorios en un sistema operativo, y además resulta de suma importancia si esos datos se utilizan luego para generar claves de cifrado simétrico o asimétrico.

2.2.2. Inconvenientes

- No puede repetirse una misma secuencia.
- Incorporar una tabla a la computadora implica gran costo de almacenamiento en relación a la cantidad de números.
- Tasa de generación limitada. La formación suele producir números aleatorios a un ritmo mucho menor que los métodos de software.
- En muchos casos, hay una limitación fundamental en la tasa de cambio de la señal muestreada. Algunos de los problemas son parámetros físicos. Si el sistema se muestrea a una velocidad alta, no hay tiempo suficiente para que el sistema cambie y los números aleatorios no serán del todo independientes.
- Es complicado dar argumentos convincentes de que los datos tomados son del todo aleatorios.
- Añadir un dispositivo externo suele presentar un inconveniente.
- Los fallos son difíciles de detectar. Si un generador de números aleatorios de hardware falla, puede ser difícil darse cuenta de ello.

2.2.3. Random.org

Desde hace décadas, sitios como Random.org ofrecen en internet números "verdaderamente aleatorios"para todo tipo de propósitos: análisis científicos, juegos, sorteos, etc. Dicen que su calidad es superior a la de los generadores de números pseudoaleatorios porque obtienen los números de procesos puramente físicos, en su caso del ruido atmosférico.

Básicamente se trata de tres radios AM/FM sintonizadas entre emisoras, en la zona donde todo lo que se oye es ruido de fondo. Ese ruido es aleatorio y a partir de él se generan los números que luego se preparan para ofrecerse en diversos formatos.

Además de esta ingeniosa solución que emplea el ruido electrónico hay sistema similares basadas en otros procesos físicos que se consideran impredecibles y son normalmente subatómicos: desintegración radioactiva, ruido térmico o el ruido de disparo(shot) típico de los dispositivos electrónicos y ópticos.

El sitio web fue creado en 1998 por Mads Haahr, un médico y profesor de informática en el Trinity College de Dublín, Irlanda. Se generan números aleatorios basados en el ruido atmosférico capturado por varias radios sintonizadas entre estaciones.

Mediante esto, utilizamos los números generados por el sitio web para poder ver el comportamiento de los mismos mediante la toma de una muestra. Generamos 10.000 números aleatorios que, según el sitio web, son generados por el ruido atmosférico. Gráficamente obtenemos un resultado contundente el cual no parece presentar patrones u otros sinónimos que nos hagan dudar de su aleatoriedad. Demás esta decir que esto es lo que sera evaluado en los apartados siguientes.

El resultado se presenta a continuación:

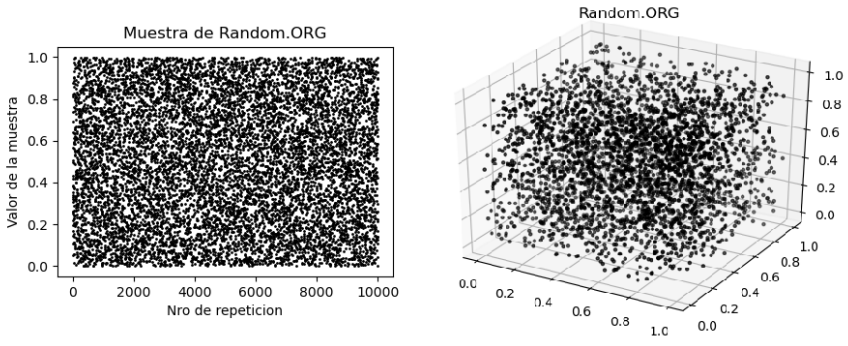

Figura 2: Generador Random.org.

3. Generadores de números pseudoaleatorios

El método más conveniente y más fiable de generar números aleatorios es utilizar algoritmos determinísticos, es decir, aquellos en que se obtiene el mismo resultado bajo las mismas condiciones iniciales, por lo cual requieren parámetros de arranque y deben poseer alguna base matemática sólida. Estos algoritmos producen una sucesión de números que se asemeja a la de una sucesión de realizaciones de variables aleatorias U(0, 1), aunque realmente no lo sea.

El método utiliza algoritmos computacionales que pueden producir largas secuencias de resultados aparentemente aleatorios, que son, de hecho, completamente determinado por un valor inicial más corto, conocida como un valor semilla o clave.

Como resultado, toda la secuencia aparentemente aleatoria puede ser reproducido si se conoce el valor de la semilla. Este tipo de generador de números aleatorios a menudo se llama un generador de números pseudoaleatorios. Este tipo de generador típicamente no se basa en fuentes de entropía de origen natural, aunque se puede sembrar periódicamente por fuentes naturales, es no bloqueante, por lo que no están limitados por una tasa por un evento externo.

3.1. Propiedades deseables

- La sucesión de valores que proporcione debería asemejarse a una sucesión de realizaciones independientes de una variable aleatoria U(0,1). Es decir, que la obtención de cierto valor no esté condicionado por los valores obtenidos anteriormente (no correlacionado).
- Rapidez en la obtención de los números.
- Almacenamiento mínimo. Tanto el propio generador como los números por el generados.
- Los resultados deben ser reproducibles, comenzando con las mismas condiciones iniciales debe ser capaz de reproducir la misma sucesión.
- El generador debe ser rápido y ocupar poca memoria interna
- La sucesión de valores generados debe tener un ciclo no repetitivo tan largo como sea posible
- La secuencia generada debe ser intuitivamente aleatoria. Esa aleatoriedad debe ser establecida teóricamente o, al menos, debe pasar ciertos tests de aleatoriedad.
- Debe conocerse algo sobre las propiedades teóricas del generador.

3.2. Generador congruente lineal(GCL)

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.

Los GCL generan una secuencia de números pseudoaleatorios en la cual el próximo número pseudoaleatorio es determinado a partir del último número generado,es decir, el número pseudoaleatorio $X_{(n+1)}$ es derivado a partir del número pseudoaleatorio X_n. Las ventajas de estos generadores son que utiliza poca memoria y son muy rápidos. Son fáciles para volver a generar la misma secuencia: guardando un solo número, (alcanza con partir desde la misma semilla). Este generador esta determinado por la siguiente formula matemática:

$$X_{n+1} = (a * X_n + c) \text{ mód } (m) \tag{1}$$

Donde:

- m es el módulo, $m > 0$
- a es el multiplicador, $0 \leq a < m$
- c es el incremento, $c \leq m$
- X_0 es la semilla, $0 \leq X_0 < m$

Para transformar los números X_n en el intervalo (0,1) se usa la siguiente ecuación:

$$r_n = \frac{X_n}{m} \tag{2}$$

Las buenas propiedades dependen de una elección apropiada de a, c y m, y en algunos casos X_0. La elección de m se relaciona con: longitud de la secuencia y velocidad computacional. La elección de a y c, en función de m, se relacionan con la aleatoriedad. Vale la pena notar que un periodo grande no determina que el generador congruencial es bueno, debemos verificar que los números que generan se comportan como si fueran aleatorios. Los GCLs continúan siendo utilizados en muchas aplicaciones porque con una elección cuidadosa de los parámetros pueden pasar muchas pruebas de aleatoriedad.

También cabe mencionar que tan pronto como un x_n se repita por primera vez, toda la secuencia se comenzara a repetir de nuevo, es decir, el mismo período de duración m_0, que ya se ha generado completamente, se inicia nuevamente. Un hecho importante es que si $c > 0$, entonces para cada valor inicial, el GCL definido generará una secuencia de números con un período máximo posible si y solo si:

- a debe ser un numero impar, no divisible ni por 3 ni por 5
- m y c son primos relativos.
- a - 1 es divisible por todos los factores primos de m
- a 1 es divisible por 4 si m es divisible por 4

Bajo estas condiciones se obtiene un periodo de vida máximo $N = m = 2^k$ Sin embargo, si no se eligen de forma correcta los diferentes parámetros pueden generarse patrones no deseados que atentan contra la aleatoriedad del método. Visualicemos un ejemplo de esto mediante una simulación de este método variando los valores de a y c:

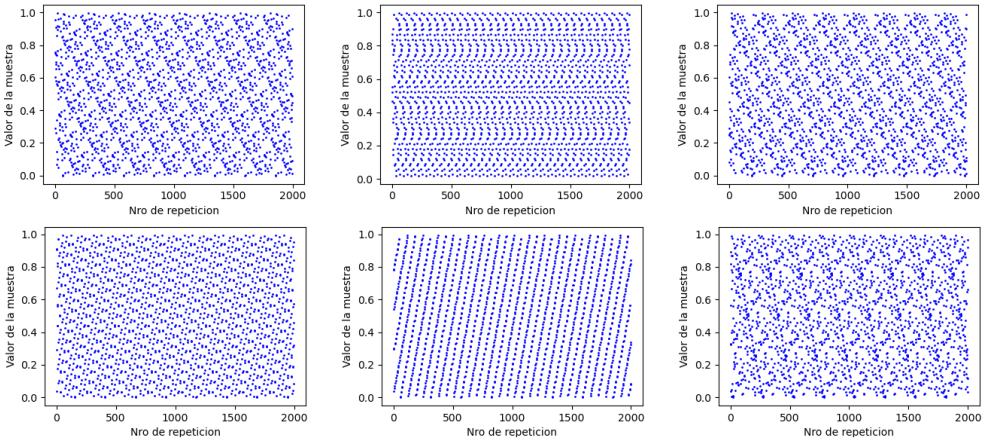

Figura 3: GCL con diferentes parámetros.

Puede verse como se presentar patrones regulares los cuales nos indican directamente que no son buenos generadores de números pseudoaleatorios. Sin embargo, en las siguientes secciones analizaremos diferentes generadores congruentes con diferentes parámetros los cuales parecieran ser mejores a los mostrados recientemente, y posteriormente procederemos a evaluarlos a través de diferentes pruebas.

3.2.1. Generador congruente lineal RAND

Este generador es un Generador congruencial Multiplicativo, surge del visto anteriormente solo que se obtiene haciendo que c = 0. Por muchos años (antes de 1995) el generador de la función rand en Matlab fue este generador congruencial. En comparación con el algoritmo GCL, la ventaja de este es que implica una operación menos a realizar. Entonces la ecuación recursiva resulta:

$$X_{n+1} = (7^5 * X_n) \text{ mód } (2^{31} - 1) \tag{3}$$

Segun Banks, Carson, Nelson y Nicol las condiciones que deben cumplir los parámetros para que el algoritmo congruencial multiplicativo alcance su máximo periodo son:

- $m = 2^g$
- $a = 3 + 8k$ ó $a = 5 + 8k$
- $k = 0, 1, 2, 3, ...$
- X_0 debe ser un numero impar
- g debe ser un entero

A partir de estas condiciones se logra un periodo de vida máximo: $N = m/4 = 2^{g-2}$

Los GLCs tienden a exhibir severos defectos. Por ejemplo, si un GLC es usado para elegir puntos en un espacio n-dimensional, los puntos yacerán, a lo sumo, sobre $(n!m)^{1/n}$ hiper-planos (teorema desarrollado por George Marsaglia). Esto se debe a la correlación serial entre valores sucesivos en la secuencia Xn. La prueba espectral, la cual es una simple evaluación de la calidad de un GLC, está basada en este hecho.

Siguiendo estas indicaciones procedimos a hacer un muestreo de 10.000 números pseudoaleatorios generados con este método, visualizándolo en 2 y 3 dimensiones para poder ver como se comportan los supuestos hiper-planos. El patrón que obtuvimos fue el siguiente:

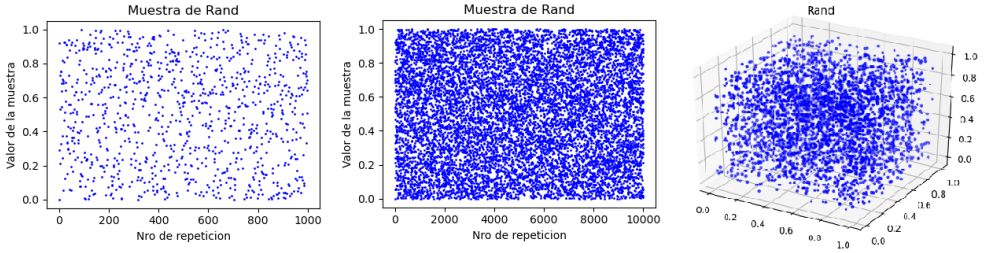

Figura 4: Generador RAND.

En estas imágenes puede verse como visualmente, sin realizar ningún test previo, los números generados parecen comportarse de manera aleatoria y no podría predecirse cual sera el próximo numero que se generara. Obsérvese que este método no contiene los parámetros que generaban determinados patrones vistos en la imagen anterior.

3.2.2. Generador congruente lineal RANDU

La mayoría de los algoritmos de simulación se basaban en generadores de números aleatorios RANDU, este es un generador de números pseudoaleatorios congruentes lineales (LCG) del tipo Park-Miller, que se ha utilizado desde la década de 1960. Se define por la siguiente formula matemática recurrente:

$$X_{n+1} = (2^{16} + 3) * X_n \text{ mód } (2^{31}) \tag{4}$$

Las principales desventajas de este método son que: sólo se puede inicializar con semillas que sean números impares, las probabilidades de aparición de cualquier número par son nulas (sólo genera números impares) y todos los generadores, al final, acaban repitiendo la secuencia. Aunque podría tener un período de hasta 2^{31}, realmente empieza a repetirse bastante antes y otra desventaja es que los números generados tienen una fuerte correlación entre sí.

Procederemos a graficar de manera similar a como hicimos con el generador RAND para visualizar el comportamiento de una muestra con este método:

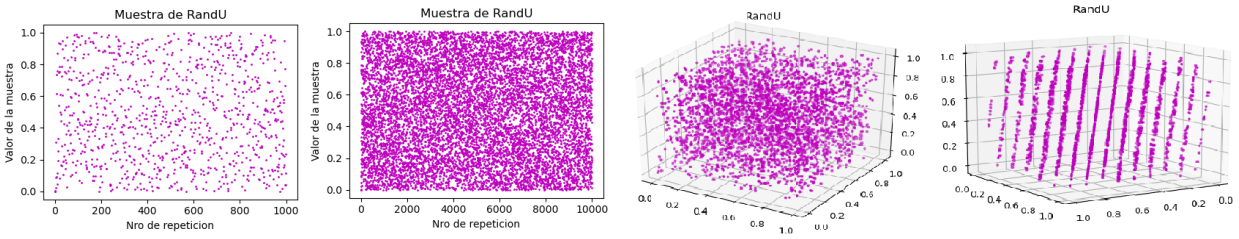

Figura 5: Generador RANDU.

Desde un inicio, las sucesiones parecen ser similares a una uniforme, sin embargo, cuando se grafican ternas surgen patrones no deseados. Si se rota la muestra graficada en la tercer dimensión podrá verse como aparece un patrón de hiper-planos. En general todos los generadores de este tipo van a presentar estructuras reticulares.

Puede verse como cuando los valores uniformes generados U(0,1) se usan de tres en tres en una tupla, el generador está sesgado. Se ve como los números caen en uno de los 15 planos en el espacio tridimensional. Como resultado, U(0,1) están altamente correlacionados y sesgados en cualquier contexto en el que se usen.

Lo crucial es que este generador se utilizó mucho tiempo en la década de 1960. Esto es muy grave porque cualquier experimento científico (p.ej. Monte Carlo) que analice datos usando como fuente RANDU, acabará concluyendo que existe correlación entre las variables del estudio, no porque realmente exista, sino por los números falsamente aleatorios de entrada.

3.2.3. Generador congruente Mixto

Estos generadores generan una secuencia de números pseudoaleatorios en la cual el próximo numero es determinado a partir del ultimo numero generado, es decir, al igual que ocurría anteriormente. Este algoritmo esta caracterizado por darle valor a la constan c mencionada anteriormente. Se mencionó, al inicio del capítulo, la formula de este algoritmo de manera general pero lo cierto es que esta formula es clasificada como un Algoritmos congruente Mixto.

Como hicimos con los generadores antes mencionados, procederemos a obtener una muestra para ver su comportamiento a grandes rasgos. Para esto daremos los siguientes valores a los parámetros de la formula:

$$X_{n+1} = ((106 * X_n) + 1283) \text{ mód } (6075) \tag{5}$$

Entonces gráficamente obtuvimos:

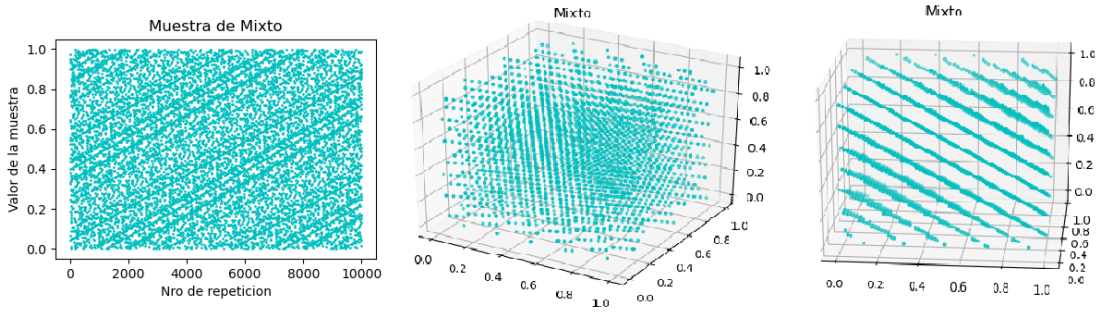

Figura 6: Generador Mixto.

Este método exhibe una estructura reticular regular donde todos los puntos se encuentran en hiper-planos paralelos. Lo que nos indica que no son del todo aleatorios. Podemos ver como, gráficamente, este generador presenta un patrón en el plano bidimensional que luego se traslada al gráfico tridimensional en forma de planos. Entonces, este generador no es bueno precisamente porque al generar este comportamiento se sabe que no se le parece en nada a un buen generador de números pseudoaleatorios.

3.3. Generadores no congruenciales: Método de los cuadrados medios

Este método requiere un numero entero, llamado semilla, con K dígitos, este es elevado al cuadrado para seleccionar del resultado los K dígitos del centro; el primer número se determina simplemente convirtiendo a decimal esos dígitos (anteponiéndole un “0”). Para poder obtener el segundo número se sigue el mismo procedimiento, solo que ahora se elevan al cuadrado los K dígitos del centro que se seleccionaron para obtener el primer número. Este método se repite hasta obtener n números.

Los pasos a seguir para realizar este método entonces quedan determinados por:

1. Seleccionar una semilla con K dígitos.
2. Elevar el valor de la semilla al cuadrado.
3. Extraer K dígitos del centro del resultado
4. Repetir el proceso tantas veces como se desee

3.3.1. ¿Quienes crearon el método?

John von Neumann (Budapest, 28 de diciembre de 1903-Washington, D. C., Estados Unidos, 8 de febrero de 1957) fue un matemático húngaro-estadounidense que realizó contribuciones fundamentales en física cuántica, análisis funcional, teoría de conjuntos, teoría de juegos, ciencias de la computación, economía, análisis numérico, cibernética, hidrodinámica, estadística y muchos otros campos. Se le considera uno de los matemáticos más importantes del siglo XX.

Nicholas Constantine Metropolis (11 de junio de 1915 – 7 de octubre de 1999) fue un matemático, físico y computador científico greco-estadounidense. Metropolis contribuyó con varias ideas originales al campo de las matemáticas y física. Tal vez el aporte más conocido es el Método de Montecarlo.

En 1953 Metropolis fue coautor del primer artículo en una técnica que fue clave para el método que ahora se conoce como "Algoritmo de recocido simulado". También desarrolló un algoritmo (el algoritmo Metropolis o Algoritmo Metropolis-Hastings) para generar muestras de la distribución de Boltzmann, posteriormente generalizado por W.K. Hastings.

3.3.2. Desventaja del método

Este método no parece ser una buena fuente de números aleatorios. Podría suceder que la secuencia caiga en un ciclo corto de repeticiones, siendo el caso extremo el del cero el cual, si aparece en la secuencia, seguirá repitiéndose siempre. A continuación enumeraremos algunas de las desventajas mas importantes que presenta este método:

- Tiene una fuerte tendencia a degenerar a cero rápidamente, donde permanecerá por siempre
- Los números generados pueden repetirse cíclicamente después de una secuencia corta
- La utilización de números primos puede generar ciclos más largos en la generación de números pseudoaleatorios
- Hoy se utilizan generadores congruenciales lineales
- A partir de los años 50 se realizaron diversas experiencias con el método propuesto por von Neumann. Trabajando con números de 4 dígitos, G. E. Forsythe probó con 16 números iniciales. Con 12 de ellos terminó con el ciclo 6100, 2100, 4100, 8100, 6100, etc. Y con otras dos terminó en cero.

3.3.3. Ejecución que utilizaremos

1. Tomaremos un número al azar, al que llamaremos semilla, que deberá tener una longitud de valor 4.
2. La semilla debe ser elevada al cuadrado, lo que resulta un numero de 8 cifras. En caso de que lo obtenido no tenga 8 cifras, se le agregan ceros a la izquierda hasta completar.
3. Se deben tomar los cuatros números centrales. El valor obtenido se llamará Z_i
4. Para generar el número aleatorio, se ubica un punto decimal antes de la cifra. El valor obtenido será U_i
5. Para generar los próximos números aleatorios se toma como la nueva semilla el Z_{i-1} repitiendo desde el paso 2.

Para comprender mejor el comportamiento de este algoritmo realizamos varias pruebas con diferentes valores de repeticiones representados a continuación:

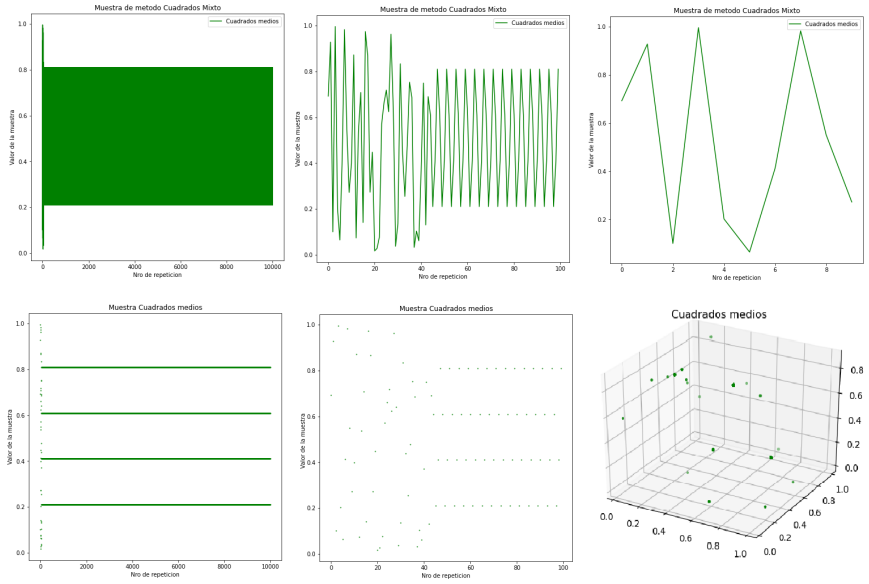

Figura 7: Método de los cuadrados medios.

Se puede ver cómo, cuando el número de repeticiones aumenta, el algoritmo cae en un ciclo corto de cuatro valores y continua oscilando entre los mismos. Estos valores son 0.81, 0.61, 0.41 y 0.21. Las figuras representadas en la parte superior representan los valores se la muestra en términos de una función, mientras que en la parte inferior están representados con puntos. Es por esto que en la primer gráfica de la parte inferior puede verse exactamente que valores de la muestra ocurrieron, siendo el ciclo compuesto por los 4 puntos mencionados anteriormente. En la representación tridimensional, la cantidad de números generados es muy grande, sin embargo, puede verse como aparecen solo unos pocos puntos. Esto es debido a que esta ocurriendo este ciclo y, cuando la cantidad generada crece, los números caen en los mismos 4 lugares. Dada estas circunstancias, podría pensarse que si este generador se utiliza para un pequeño número de repeticiones podría ser útil, pero lo cierto es que debería elegirse una semilla inicial buena para intentar evitar estos inconvenientes.

3.4. Generador de números pseudoaleatorios de Python

Python tiene un módulo incorporado que puede usar para hacer números aleatorios. La función genera un flotante aleatorio uniformemente en el rango semiabierto [0 , 1). Python usa el Mersenne Twister como generador principal. Produce flots de precisión de 53 bits y tiene un período de $2^{19937-1}$. A veces es útil poder reproducir las secuencias dadas por un generador de números pseudoaleatorios. Al reutilizar un valor de inicialización, la misma secuencia debe ser reproducible de ejecución a ejecución siempre que no se ejecuten varios subprocesos. La mayoría de los algoritmos y funciones de inicialización del módulo aleatorio están sujetos a cambios en las versiones de Python, pero garantizan los siguientes dos aspectos que no cambiarán: Si se agrega un nuevo seed, se ofrecerá una compatible con versiones anteriores. Y además, el método random() continuará produciendo la misma secuencia siempre y cuando reciba la misma semilla. Nos basaremos en esta función para tomar una muestra de 10.000 datos para poder evaluar su comportamiento mediante la representación en una gráfica que se muestra a continuación:

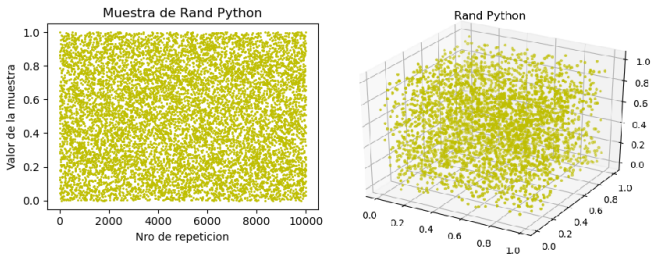

Figura 8: Generador random de Python.

Podemos ver como, junto con el generador anterior, parecen ser buenos generadores ya que a simple vista no parecen producir patrones. Sin embargo no se puede afirmar esto sin previamente realizar diferentes Tests a los diferentes generadores.

4. Comparación entre generadores: Muestreo

Sin la necesidad de realizar ningún test formal, procedimos a visualizar como se comportan los valores de las diferentes muestras de diferentes generadores. Pudimos ver que los generadores congruenciales aparentan ser aleatorios mientras que el generador de los medios cuadrados presenta un ciclo corto(de cuatro valores) que se repite a lo largo del tiempo. Esto era debido a las desventajas que presenta este generador que fueron mencionadas anteriormente. Existen métodos y teorías que se basan en combinar diferentes generadores entre sí, para de esta forma generar secuencias de números pseudoaleatorios mas largas y de una supuesta mejor forma. En este apartado tenemos como objetivo demostrar de una manera sencilla como se pueden combinar diferentes generadores y cuales han sido los metodos mas exitosos a lo largo de la historia.

Para comenzar tomamos todos los generadores que fueron descriptos en los apartados anteriores y los gráficamos en un mismo lugar. De esta forma supondremos que contamos con un generador que estará compuesto por la combinación lineal de los mismos. A continuación se presenta la idea anteriormente planteada:

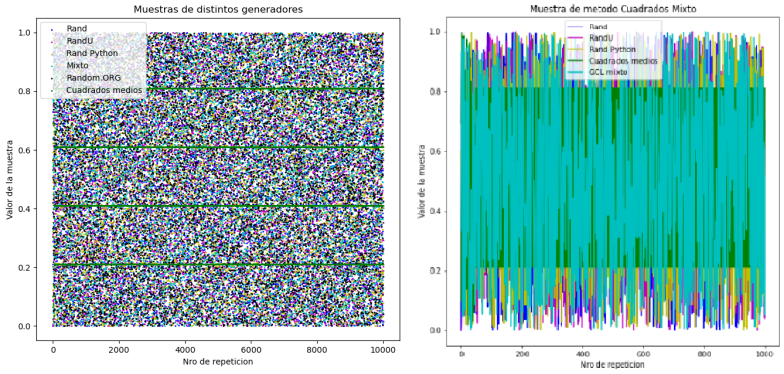

Figura 9: Todas las muestras juntas.

Puede verse como las secuencias de números pseudoaleatorios generados presentará un período mayor (compuesto por de la suma del periodo de todos los generadores). Sin embargo NO quedamos exentos de todas las desventajas que presentaba cada generador. Es decir, la suma de los generadores esta conformada por la suma del ERROR que presenta cada generador. El total estará compuesto por la deficiencia en cuanto a uniformidad del método de los cuadrados medios, la correlación de las variables del generador RANDU, los n hiper-planos del generador congruencial Mixto, etc.

Ahora cabría imaginarnos que esto es debido a una mala elección en cuanto a los generadores que decidimos combinar. Es por esta razón que a continuación presentamos la combinación de generadores que fueron previamente testeados y que fueron utilizados a lo largo de la historia. Si bien están basados en los mismos principios, se demuestra que una buena combinación SI puede generar periodos mas largos, y una mejor eficiencia a la hora de obtenerlos.

Métodos:

- **Generadores congruenciales lineales combinados**
- **OR-exclusivo de números aleatorios de dos o más generadores**: Con esta técnica se suma dos generadores GCL, siendo la suma un or-exclusivo bit por bit. Se ha demostrado que esta técnica aplicada a números ligeramente aleatorios puede ser usada para generar números con mayor aleatoriedad.
- **Barajeo**: Usa una secuencia como un índice para decidir que número generado por otra secuencia será retornado

4.1. Generadores congruenciales lineales combinados

Dado que los generadores congruenciales lineales tienen muy buenas propiedades respecto a su equidistribución en bajas dimensiones, su simplicidad conceptual y su facilidad de implementación, pero también algunas limitaciones respecto a su equidistribución en altas dimensiones, un enfoque posible consiste en plantearse de que manera y bajo que condiciones es posible modificar la salida de un generador congruencial lineal para mejorar sus propiedades, sin alterar significativamente su simplicidad y eficiencia. Hay al menos dos métodos que han sido empleados con éxito en este marco: el método de Shuffling, el de Merging y el de L'Ecuyer.

- Método Shuffling. Este algoritmo consiste en reordenar la salida de un generador cualquiera, en base a la secuencia generada por un segundo generador. Supongamos que tenemos dos generadores congruenciales lineales de secuencias de números pseudoaleatorios. Sean las secuencias $\{X_i, i \geq 0\}$ e $\{X_i, i \geq 0\}$
- Método de L'Ecuyer. Una manera para conseguir generadores con periodos más largos consiste en sumar números aleatorios de dos o más generadores. Utilizando el GCL RAND y teniendo en cuenta el siguiente teorema:
  Teorema: Sean $W_1, W_2, ..., W_n$ variables aleatorias discretas, tales que $W_1 \sim U([0, d-1])$. Entonces

$$W = (\sum_{j=1}^{n} W_j) mod\ d_1 \tag{6}$$

  es una variable aleatoria con distribución uniforme discreta en $[0, d-1]$. Ahora veamos como podemos utilizar este teorema para combinar generadores. Sean $X_{i,1}, X_{i,2}, ..., X_{i,k}$ la i-enesima salida de k generadores congruenciales multiplicativos diferentes y donde el j-esimo generador tiene modulo primo m_j y su multiplicador a ha sido seleccionado de forma tal que tenga periodo máximo $d_1 - 1$. L'Ecuyer sugiere combinar los generadores de la forma:

$$X_i = (\sum_{j=1}^{k} (-1)^{j-1} X_{i,j})\ mod\ (d_1 - 1) \tag{7}$$

  Donde los números pseudoaleatorios se obtendrán haciendo $\frac{X_i}{d_1}$ cuando $X_i > 0$ y para los $X_i = 0$ resultará igual a $\frac{d_1 - 1}{d_1}$.
  Gracias a esto el periodo máximo posible será de

$$P = \frac{(d_1 - 1)(d_2 - 1)...(d_k - 1)}{2^{k-1}} \tag{8}$$

  Un ejemplo de esto podrían ser los siguientes generadores:

$$x_n = 40014 x_{n-1}\ mod\ (2^{31} - 85) \tag{9}$$

$$y_n = 40692 y_{n-1}\ mod\ (2^{31} - 249) \tag{10}$$

  Combinados:

$$W_n = (x_n - y_n)\ mod\ (2^{31} - 86) \tag{11}$$

  Resultando un período de $P = \frac{(2^{31}-86)(2^{31}-250)}{2^{2-1}} = 2{,}3 * 10^{18}$. Esta secuencia tardaría 72,93 años en consumirse. Note que mediante este método se ha conseguido generar un valor muy alto de período comparado con los que previamente presentaban los generadores sin combinar. Esta es una gran ventaja que presentan estos métodos. Cabe mencionar también algo importante y es que la secuencia se repite solo cuando los generadores básicos se sincronizan en $x_n = x_o$ y $y_n = y_o$. No es como en el caso de un solo generador en que al repetirse un número de la secuencia esta también se repita. Ahora los números pueden repetirse sin que implique que la secuencia se repita.

4.2. Comparación de uniformidad

Agrupando todos los generadores en una misma gráfica y variando el número de repeticiones se puede visualizar como todos los generadores congruenciales parecen estar produciendo números pseudoaleatorios. Sin embargo se puede notar un patrón particular en el generador congruencial mixto, con parámetros(m , a , c) = (6075, 106, 1283). Posteriormente los someteremos a diferentes pruebas para poder evaluar la veracidad de los mismos.

Ahora, agrupando las diferentes muestras en intervalos de igual longitud y calculando la frecuencia relativa de cada uno, podremos presenciar como están compuestas las diferentes distribuciones que dicen ser uniformes. Para un tamaño de muestra de 10.000 se obtuvo el siguiente resultado:

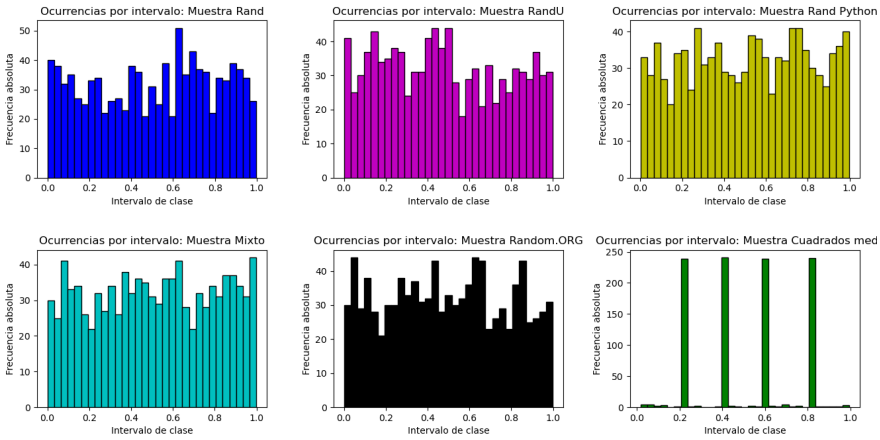

Figura 10: Frecuencia relativa por intervalo

Además, si aumentamos la cantidad de números aleatorios generados a un numero demasiado grande, el cual nos permitirá presuponer que cuando el tamaño de la muestra tiende a infinito podría asemejarse a los resultados representados por las siguientes gráficas:

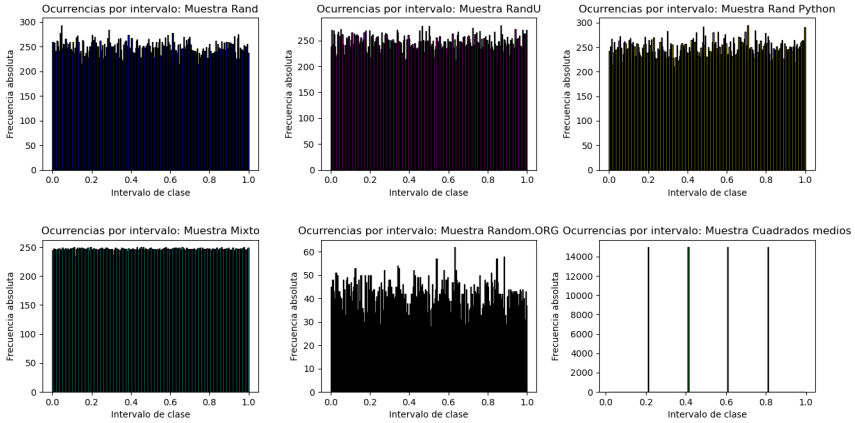

Figura 11: Frecuencia relativa por intervalo

Puede verse como el generador mixto pareciera tener una distribución completamente uniforme y garantizarnos la aleatoriedad del método. Pero como se analizo en los apartados anteriores esto no ocurre ya que posee el problema de los hiper-planos en dimensiones mayores. Además puede verse como el método de los cuadrados medios concentra sus datos en los 4 valores vistos antes. También puede verse como, el resto de los generadores parecer aproximarse a una distribución uniforme. Con esto se concluye las presuposiciones acerca de las muestras generadas por los diferentes métodos y pasaremos a comprobarlo analíticamente mediante la evaluación de los mismos en diferentes pruebas.

5. Sometiendo los generadores a diferentes pruebas

Hay varias pruebas que pueden aplicarse a los generadores de números pseudoaleatorios. Aunque algunas pruebas no se clasifican fácilmente, es conveniente describir las pruebas como empíricas o teóricas Las pruebas empíricas son pruebas estadísticas que se aplican a subsecuencias de los números producidos por un generador. Para una subsecuencia dada, se calcula un estadístico de muestra particular y se evalúa la consistencia del resultado con una muestra U (0,1). Las pruebas empíricas más sofisticadas evalúan la independencia, la equidistribución o los patrones recurrentes, ya sea en los números o en n-tuplas de los números.

Para las pruebas teóricas considere la fórmula que define un generador, a menudo empleando técnicas matemáticas avanzadas para inferir sus propiedades. En este sentido, las pruebas teóricas caracterizan la secuencia completa de números producidos por un generador. Las pruebas teóricas evalúan cosas como el período, las estructuras reticulares, la uniformidad o las correlaciones, a menudo las mismas nociones evaluadas por las pruebas empíricas, pero para todo el período de un generador. Las pruebas teóricas son limitadas, ya que el rendimiento de un generador durante todo el período puede ser diferente del comportamiento de una subsecuencia de números de ese generador utilizado en una aplicación particular.

5.1. Test Chi Cuadrado

La Prueba de Bondad de Ajuste Chi Cuadrado es el test de bondad de ajuste más utilizado. En general un test de bondad de ajuste se utiliza para discriminar si una colección de datos o muestra se ajusta a una distribución teórica de una determinada población. En otras palabras, nos dice si la muestra disponible se ajusta razonablemente los datos que uno esperaría encontrar en la población. Para poder aplicar esta prueba se requiere que los datos estén agrupados en categorías o clases. Una desventaja potencial del test de Chi Cuadrado es que requiere una muestra suficientemente grande de modo que la aproximación de Chi Cuadrado sea válida.

$$X_c^2 = \sum_{i=1}^{k} \frac{(O_i - E_i)^2}{E_i} \tag{12}$$

Donde:

- gl: grados de libertad
- Oi: valor observado para la clase i-enesima
- Ei: valor esperado de la clase i-enesima

5.1.1. Algoritmo basado en una hipótesis

Para comenzar esta prueba tenemos que definir una hipótesis, la cual queda determinada por:

H_0: Los números se ajustan a una distribución uniformes.
H_1: Los números no se ajustan a una distribución uniforme.

La secuencia de pasos a seguir para realizar esta prueba es la siguiente:

1. Dividir el intervalo (0,1) en m subintervalos (se recomienda $m = \sqrt{n}$ intervalos).
2. Clasificar cada numero pseudoaleatorio del conjunto de la muestra en los m intervalos. A la cantidad de números que clasifican en cada intervalo es a lo que se denomina valor observado (O_i)
3. Buscar la cantidad de números que se espera encontrar en cada intervalo, es decir el valor esperado E_i. Teóricamente este numero es igual a $\frac{n}{m}$.
4. Calcular el estadístico con la formula presentada anteriormente para cada uno de los intervalos de clase del histograma.
5. Comparar el estadístico calculado con el máximo permitido (leído en la tabla de Chi-Cuadrado).
    - Si es menor entonces se concluye que no hay evidencia estadística para afirmar que los números aleatorios de la muestra no tienen una distribución uniforme.
    - Si es mayor no se acepta la hipótesis de uniformidad en los números aleatorios generados.

5.1.2. Ejecución del Test Chi Cuadrado

Procedimos a realizar esta prueba a las diferentes muestras de los diferentes generadores. Para esto evaluamos cada generador con el algoritmo descripto anteriormente. Con este test obtuvimos diferentes parámetros que permitieron determinar si habían pasado o no la prueba, los cuales fueron mencionados en la sección anterior. Por lo tanto, los parámetros que nos permitieron realizar este test se encuentran en la siguiente tabla:

GENERADORES	CHICUADRADO	¿PASÓ EL TEST?
RAND	Val.Chi(71.56) <Val.table(123.23)	PASÓ
RANDU	Val.Chi(107.82) <Val.table(123.23)	PASÓ
RANDOM PYTHON	Val.Chi(111.74) <Val.table(123.23)	PASÓ
GCL MIXTO	Val.Chi(13.56) <Val.table(123.23)	PASÓ
RANDOM.ORG	Val.Chi(98.26) <Val.table(123.23)	PASÓ
CUADRADOS MEDIOS	Val.Chi(237805.4) >Val.table(123.23)	NO PASÓ

Aclaración: Val.chi(número) hace referencia al valor chi-cuadrado calculado mediante la formula (6) para cada muestra y Val.table(número) hace referencia al valor obtenido directamente de la tabla de chi-cuadrado.

A continuación, para evaluar que estaba ocurriendo realmente y poder entenderlo de una manera mas sencilla decidimos graficarlo. Como el Test Chi Cuadrado trata sobre discriminar si una muestra se ajusta o no a una distribución uniforme, decidimos trazar una linea que representa por el lado del eje X el valor que se esperaría encontrar en un intervalo dado, y por el eje Y el valor real del intervalo de clase. Ambos estarán expresados en términos de la frecuencia absoluta acumulada. Realizamos el experimento para un n = 1.000 y n = 10.000 repeticiones (considerando que esto representa un n suficientemente grande). El resultado fue el siguiente:

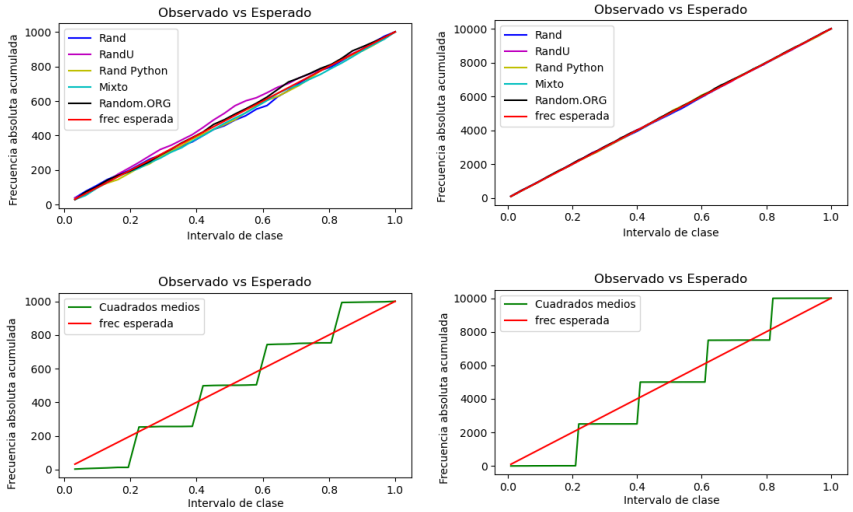

Figura 12: Test Chi Cuadrado

Gráficamente puede verse como los valores de los generadores congruenciales predicen (y el generador aleatorio físico también), cuando n es suficientemente grande, de una manera exacta la recta del valor que se esperaba obtener de una distribución uniforme. Caso contrario, el generador no congruencial no se asemeja a esta recta, representando una curva escalonada que muestra el ciclo corto (de 4 números) en el que cayó este generador. La linea roja representa como debería ser el caso ideal en el que los intervalos de clase esperados coinciden con los observados, puede verse claramente cuan alejado se encuentra el método de los cuadrados medios de lo que se esperaría obtener en el caso ideal. Ahora, repetiremos el mismo procedimiento pero esta vez representaremos las frecuencias absolutas de cada intervalo de clase en vez de las frecuencias acumuladas.

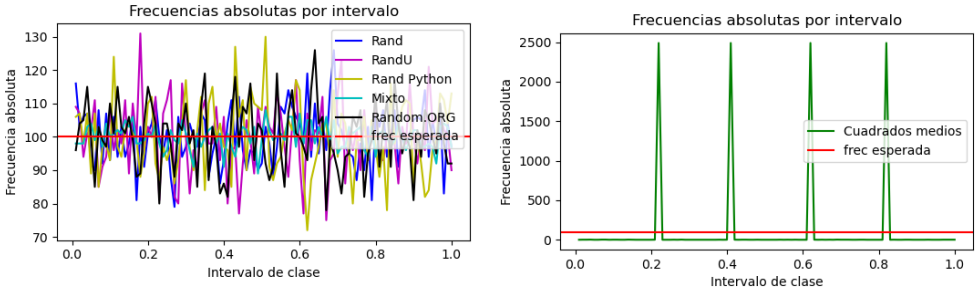

Figura 13: Frecuencias del Test Chi Cuadrado

Se tomo un $n = 10{,}000$ y el valor esperado por lo tanto es de $E_i = 100$. Pueden verse dos comportamientos que permiten notar una diferencia notoria entre ambos tipos de generadores:

- El primer comportamiento esta directamente relacionado con la esencia del test de Chi Cuadrado, quien busca comparar las coincidencias entre la cantidad de números que caen en cada intervalo esperado contra el observado, es lógico pensar que el caso ideal sería, para un numero n suficientemente grande, aquel en el cual se vea representada una constante en el valor y = 100. Esto representaría que la muestra contiene completamente una distribución uniforme. Intentando, las diferentes muestras, aproximarse a esto, los valores obtenidos de las mismas oscilaron en torno al valor $E_i = 100$. Esto es correcto por lo mencionado anteriormente. El error es relativamente pequeño y hemos de suponer que si incrementamos el numero de la cantidad de datos tomados, esto ha de aproximarse al anterior supuesto de distribución uniforme.
- El segundo comportamiento corresponde al generador no congruencial, cuya muestra caía en un ciclo de 4 números cuando incrementábamos la cantidad de números generados. Gráficamente esto se aprecia por el hecho de que estos 4 valores están representados en forma de picos altos. Esta forma irregular nos acerca a entender por qué este generador no pasó el test, ya que el 0 % de los intervalos de clase pasaron la validación de ser parecidos al valor de clase esperado (ninguno se asemeja a $E_i = 100$).

Se puede concluir que todos los generadores congruenciales pasaron esta prueba y que no hay evidencia estadística para afirmar que los números aleatorios de las muestras no tienen una distribución uniforme. Para el caso del generador no congruencial, el método de los cuadrados medios, se rechaza la hipótesis de uniformidad.

5.2. Test de corridas abajo y arriba de la media

Las pruebas de independencia consisten en demostrar que los números generados son estadísticamente independientes entre si, esto es que no dependan uno de otro. Este Test en un método que nos ayuda a evaluar el carácter de aleatoriedad de una secuencia de números estadísticamente independientes y números uniformemente distribuidos. Es decir, dado una serie de números determinar si son o no aleatorios. Este método es uno de los mas sencillos ya que solo implica el diferenciar cuales números están arriba o abajo de la media, pero su sencillez no implica que su importancia sea menor. Para realizar este método:

1. Denotaremos con un numero (1) a aquel numero que se encuentre por debajo de la media.
2. Denotaremos con un numero (0) a aquel numero que se encuentre por arriba de la media

Este procedimiento consiste en determinar una secuencia de 1s y 0s de acuerdo a la comparación de cada numero que cumpla con la condición de ser mayor a 0.5 (en el caso de los 0s) o ser menor a 0.5 (en el caso de los 1s).

Luego se determina el numero de corridas y los valores de n_0 y n_1 Donde:

- n_0: Cantidad de 0s en la secuencia N
- n_1: Cantidad de 1s en la secuencia N
- c_0: Numero de veces que se pasa de 1 a 0 ó de 0 a 1 en N
- N: Cantidad de números $(n_0 + n_1)$

Luego se calcula el valor esperado, la varianza del numero de corridas y el valor estadístico con las siguientes ecuaciones:

$$\mu = \frac{2n_0n_1}{N} + \frac{1}{2} \tag{13}$$

$$\sigma^2 = \frac{2n_0n_1(2n_0n_1 - N))}{N^2(N-1)} \tag{14}$$

$$Z_0 = \frac{C_0 - \mu_{C_0}}{\sigma_{C_0}} \tag{15}$$

Posteriormente, con el valor obtenido de Z_0 obtenido, hay que proceder a evaluar si se encuentra dentro del siguiente intervalo:

$$-Z_{\frac{\alpha}{2}} \leq Z_0 \leq Z_{\frac{\alpha}{2}} \tag{16}$$

Si la condición anterior se cumple, entonces se concluye que los números evaluados son independientes, sino se rechaza el conjunto.

5.2.1. Hipótesis de independencia

En esta ocasión estamos hablando de las pruebas estadísticas que tratan de corroborar si los números en el intervalo(0,1) son independientes o, en otras palabras, si parecen pseudoaleatorios.Para probar la independencia de los números de un conjunto, primero es preciso formular las siguientes hipótesis:

- H_0: Los números del conjunto de la muestra son independientes.
- H_1: Los números del conjunto de la muestra no son independientes.

5.2.2. Ejecución del Test de corridas abajo y arriba de la media

Procedimos a ejecutar el Test para cada muestra y, con las formulas indicadas anteriormente, generamos una tabla donde muestra los diferentes resultados de haber evaluado los diferentes generadores con este test. Esta posee una cantidad de 10.000 números generados que corresponden a una muestra. Puede verse como si el numero es suficientemente grande(se presentan mayores oportunidades de obtener una muestra completamente variable), todos los test pasan la prueba.

GENERADORES	TEST ARRIBA Y ABAJO	¿PASÓ EL TEST?
RAND	$Z_{min}(-1{,}96) \leq Z_0(-1{,}26) \leq Z_{max}(1{,}96)$	PASÓ
RANDU	$Z_{min}(-1{,}96) \leq Z_0(-0{,}07) \leq Z_{max}(1{,}96)$	PASÓ
RANDOM PYTHON	$Z_{min}(-1{,}96) \leq Z_0(-1{,}61) \leq Z_{max}(1{,}96)$	PASÓ
GCL MIXTO	$Z_{min}(-1{,}96) \leq Z_0(0{,}39) \leq Z_{max}(1{,}96)$	PASÓ
RANDOM.ORG	$Z_{min}(-1{,}96) \leq Z_0(1{,}53) \leq Z_{max}(1{,}96)$	PASÓ
CUADRADOS MEDIOS	$Z_{min}(-1{,}96) \leq Z_0(-0{,}03) \leq Z_{max}(1{,}96)$	PASÓ

Puede entenderse que lo dicho anteriormente, sobre que mientras mas grande sea el numero n con mayores probabilidades pasaran todos los generadores la prueba, no es del todo cierto. Ya que para el caso de los cuadrados medios, no influye si la muestra es suficientemente grande. Esto contradice fuertemente lo analizado anteriormente con el test Chi Cuadrado, donde el método de los cuadrados medios no pasaba dichos test.

En este caso cabria preguntarnos por qué este generador CUMPLE con los requerimientos de este test y para esto procedimos a realizar una gráfica representativa de lo que estaba sucediendo:

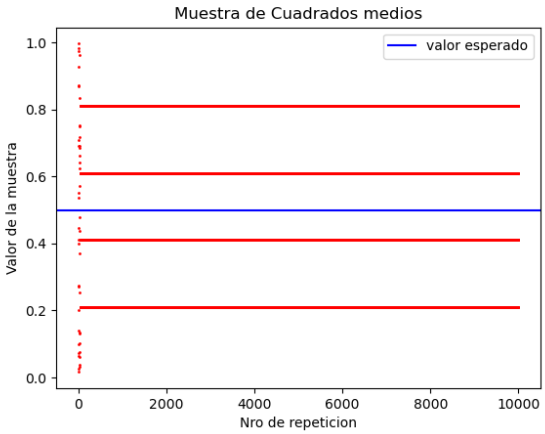

Figura 14: Test corridas para método no congruente

La linea azul representa el valor esperado de 0.5, y los puntos en color rojo representan los valores de la muestra. Puede decirse certeramente que este test pasó la prueba porque la muestra oscilaba entre 4 valores, de los cuales 2 de ellos están por **encima de la media** y 2 de ellos **están por debajo**.

Entonces esto ocasiona que casi con exactitud la mitad de los datos de la muestra correspondan a los 2 valores que están por debajo de la media, y la otra mitad de los datos a los otros 2 valores. Esto es justamente lo que permitió "burlar" a este test logrando que pase la prueba, ya que el mismo lo detectó como si tuviese una distribución uniforme por oscilar entre valores por debajo y por encima la media.

Nuevamente se recalca que los 5 primeros generadores pasaron la prueba de independencia de una forma valida mientras que no ocurrió lo mismo con el último test, quien paso la prueba no cumpliendo con los resultados esperados.

5.3. Test de Medias

Conociendo los parámetros de la distribución uniforme, es posible plantear una prueba de medias, con la cual se trata de verificar que los números pseudoaleatorios generados provienen de un universo uniforme con media de 0.5.

Los pasos a seguir serán

1. Plantear la hipótesis:
   - H_0: Hipótesis nula, $\mu = 0{,}5$.
   - H_1: Hipótesis alternativa, $\mu \neq 0{,}5$.
2. Calcular la media μ con la siguiente formula:

$$r = \frac{1}{n}\sum_{i=0}^{n} r_i \tag{17}$$

   Donde r_i representa el valor de cada numero de la muestra y r el promedio.
3. Calcular los limites superior e inferior de aceptación L_S y L_I respectivamente con las siguientes formulas:

$$L_S = \frac{1}{2} + \frac{Z_{\frac{\alpha}{2}}}{\sqrt{12 * n}} \tag{18}$$

$$L_I = \frac{1}{2} - \frac{Z_{\frac{\alpha}{2}}}{\sqrt{12 * n}} \tag{19}$$

4. Si el valor de la media se encuentra entre los límites L_S y L_I, se acepta que los números tienen una media estadísticamente igual a 0.5 con un nivel de aceptación de $(1 - \alpha)$

Para el calculo de los limites de aceptación se utiliza el estadístico $Z_{\frac{\alpha}{2}}$ el cual se determina por medio de la tabla de distribución normal estándar.

5.3.1. Ejecución del Test de Medias.

Siguiendo los pasos enumerados en la sección anterior generamos una tabla que indica los parámetros de los limites superior e inferior utilizados, así como el valor correspondiente de la media para cada muestra. El resultado fue el siguiente:

GENERADORES	TEST DE MEDIAS ($\mu = 0{,}5$)	¿PASÓ EL TEST?
RAND	$l_i(0{,}4943) \leq \mu = 0{,}5016 \leq l_s(0{,}5057)$	PASÓ
RANDU	$l_i(0{,}4943) \leq \mu = 0{,}4998 \leq l_s(0{,}5057)$	PASÓ
RANDOM PYTHON	$l_i(0{,}4943) \leq \mu = 0{,}497 \leq l_s(0{,}5057)$	PASÓ
GCL MIXTO	$l_i(0{,}4943) \leq \mu = 0{,}4994 \leq l_s(0{,}5057)$	PASÓ
RANDOM.ORG	$l_i(0{,}4943) \leq \mu = 0{,}4976 \leq l_s(0{,}5057)$	PASÓ
CUADRADOS MEDIOS	$\mu = 0{,}5098 > l_s(0{,}5057)$	NO PASÓ

Por lo que se concluye que el conjunto de números pseudoaleatorios, generado mediante métodos congruenciales, pasa la prueba de medias. No siendo el caso con el generador de cuadrados medios, quien presenta una leve desviación del los limites superior e inferior permitidos por el test. Esto esta relacionado con la aleatoriedad de los 4 números que se repiten indefinidamente con este método. Ya que si los 4 números hubiesen estado distribuidos equiprobablemente por encima y por debajo de la media, hubiesen burlado también este test.

En la figura 11 se ve como los numeros que se encuentran en la parte inferior estan mas alejados de la media que los numeros de la parte superior, esto puede calcularse sencillamente mediante:

$$Nros\ de\ la\ parte\ superior\ = (0{,}81 - \mu) + (0{,}61 - \mu) = (0{,}81 - 0{,}5) + (0{,}61 - 0{,}5) = \mathbf{0.42} \tag{20}$$

$$Nros\ de\ la\ parte\ inferior\ = (\mu - 0{,}41) + (\mu - 0{,}21) = (0{,}5 - 0{,}41) + (0{,}5 - 0{,}21) = \mathbf{0.38} \tag{21}$$

Esto nos indica que existirá un desvió hacia la parte superior de la media calculada, y este hecho coincide perfectamente con los resultados que obtuvimos en la tabla, donde el valor de la media se encuentra por encima de 0.5. Por lo tanto las conclusiones observadas son similares a las obtenidas con el test anterior.

5.4. Test Kolmogorov Smirnov

La prueba de Kolmogorov-Smirnov para una muestra es un procedimiento de bondad de ajuste, que permite medir el grado de concordancia existente entre la distribución de un conjunto de datos y una distribución teórica específica. Su objetivo es señalar si los datos provienen de una población que tiene la distribución teórica especificada, es decir, contrasta si las observaciones podrían razonablemente proceder de la distribución especificada. Conviene tener en cuenta que la prueba Kolmogórov-Smirnov es más sensible a los valores cercanos a la mediana que a los extremos de la distribución. Puede mencionarse que este Test:

- Es más poderoso que el Test Chi-Cuadrado.
- Es mas fácil de calcular ya que no requiere agrupación de los datos.
- El estadístico solo depende del tamaño de la muestra.

Sin embargo presenta algunas desventajas:

- Solo se aplica a distribuciones continuas.
- Tiende a ser más sensible cerca del centro de la distribución que en las puntas.
- Se pierde información al agrupar los datos en intervalos.

- La distribución debe especificarse completamente. Es decir, si los parámetros de ubicación, escala y forma se estiman a partir de los datos, la región crítica de la prueba KS ya no es válida. Por lo general, debe determinarse mediante simulación.
- No es sencillo construir los intervalos a partir de las probabilidades.

5.4.1. ¿Quienes crearon el test?

Andrey Nikolaevich Kolmogorov (Tambov, 25 de abril de 1903-Moscú, 20 de octubre de 1987) fue un matemático ruso que realizó aportes de primera línea en los contenidos de teoría de la probabilidad y de topología. Estructuró el sistema axiomático de la teoría de la probabilidad, utilizando el lenguaje teoría de conjuntos, donde los elementos son eventos. Trabajó en lógica constructivista, en las series de Fourier, en turbulencias, en mecánica clásica y fue quien fundó la teoría de la complejidad algorítmica.

Nikolai Vasilyevich Smirnov (17 de octubre de 1900 - 2 de junio de 1966) fue un matemático ruso soviético conocido por su trabajo en diversos campos, incluida la teoría de la probabilidad y las estadísticas. Los principales trabajos de Smirnov en estadística matemática y teoría de la probabilidad se dedicaron a la investigación de distribuciones límite mediante el comportamiento asintomático de integrales múltiples a medida que la multiplicidad aumenta con el límite. Fue uno de los creadores de los métodos no paramétricos en estadística matemática y de la teoría de distribuciones límite de estadísticas de orden.

5.4.2. Definición del algoritmo del Test Kolmogorov Smirno

Esta prueba compara la función de densidad de probabilidad F(x) de la distribución uniforme con la función de densidad de probabilidad empírica, $S_n(x)$, de una muestra de N observaciones. Sabemos que $F(x) = x$ con $0 \leq x \leq 1$. También sabemos por las gráficas vistas en los apartados anteriores que conforme el numero N de la muestra crece, $S_n(x)$ debería aproximarse cada vez mas a $F(x)$.

Los pasos a seguir se encuentran enumerados a continuación:

1. Plantear una hipótesis.
2. Ordenar la muestra del mas pequeño al más grande.
3. Calcular el valor de la función de distribución empírica $S_n(x)$.
4. Como la distribución teórica se encuentra especificada en H_0, encontrar la probabilidad esperada $F_0(x)$ para cada valor ordenado de x.

La hipótesis queda definida por:

- H_0: La distribución de frecuencias observadas es consistente con la distribución uniforme(buen ajuste).
- H_1: La distribución de frecuencias observadas no es consistente con la distribución uniforme(mal ajuste).

Para calcular el valor de la función de distribución empírica $S_n(x)$ se procede a calcular:

$$S_n(x) = \frac{1}{i} \qquad i = 1, ..., N \tag{22}$$

La prueba Kolmogorov-Smirnov esta basada en la desviación máxima absoluta entre $F(x)$ y $S_n(x)$. Por lo tanto nos basaremos en el siguiente estadístico de Kolmogorov-Smirnov:

$$D = MAX(|F(x) - S_n(x)|) \tag{23}$$

Una vez ordenada la muestra observada F(x) procedemos a calcular:

$$D^+ = MAX_j(\frac{j}{n} - x_j) \tag{24}$$

$$D^- = MAX_j(x_j - \frac{j-1}{n}) \tag{25}$$

Mediante estas formulas lo que estamos haciendo es calcular la distancia (D) entre el valor observado y el valor esperado. Con D^+ se calcula cuanto se excede (error), el valor observado, por encima del valor esperado. Mientras que D^- calcula cuanto le falta al valor observado para alcanzar el valor esperado (se encuentra por debajo de este ultimo).

Puede entenderse gráficamente de una forma mas sencilla como sigue:

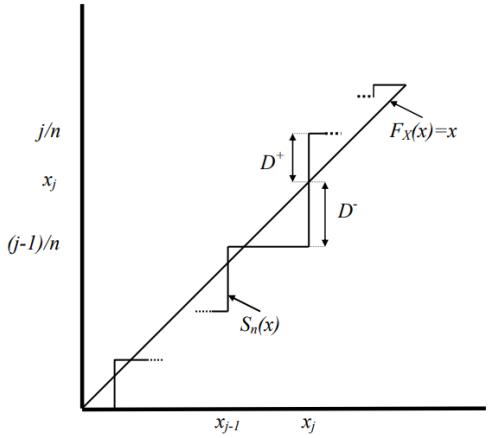

Figura 15: Representación de las formulas. Imagen tomada del sitio web http://webdelprofesor.ula.ve /ingenieria/hhoeger/simulacion/ PARTE5.pdf. Gutierrez, M. (2020)

En esta imagen puede verse como se compara los valores observados con los esperados, calculando las distancias D por encima y por debajo de la función de densidad esperada. Una vez obtenidos D^+ y D^- se procede a calcular el máximo entre ambos de la forma: $D = MAX(D^+, D^-)$

Sabiendo que α = Nivel de significación de la prueba, hay que buscar en la tabla KS para un nivel de significación y tamaño de muestra N. Una vez obtenido este valor hay que comparar si el valor calculado de D es menor o igual al de la tabla, en este caso decimos que no hay evidencia de que los números no sean uniformes.

Cabe mencionar que la prueba Kolmogorov Smirno esta diseñada para distribuciones continuas, muestras pequeñas, y usar todas las observaciones sin hacer intervalos de clase haciendo mejor uso de los datos que la chi-cuadrado. Realizamos esta prueba para todas las muestras de los distintos generadores y los parámetros que calculamos y obtuvimos de D se encuentran representados a continuación:

GENERADORES	TEST KOLMOGOROV SMIRNOV	¿PASÓ EL TEST?
RAND	$D_{calc}(0{,}0893) \leq D_{tabla}(0{,}134)$	PASÓ
RANDU	$D_{calc}(0{,}0582) \leq D_{tabla}(0{,}134)$	PASÓ
RANDOM PYTHON	$D_{calc}(0{,}0818) \leq D_{tabla}(0{,}134)$	PASÓ
GCL MIXTO	$D_{calc}(0{,}0477) \leq D_{tabla}(0{,}134)$	PASÓ
RANDOM.ORG	$D_{calc}(0{,}0623) \leq D_{tabla}(0{,}134)$	PASÓ
CUADRADOS MEDIOS	$D_{calc}(0{,}11) \leq D_{tabla}(0{,}134)$	PASÓ

La desventaja de esto es que si utilizamos, como es sugerido, un numero pequeño de observaciones vamos a obtener(como se ve representado en la tabla anterior) que todos los test pasan la prueba ya que como analizamos, por ejemplo, el ciclo corto de cuatro números del método de los medios cuadrados ocurría luego de un numero sucesivo de observaciones. Al elegir un numero muy pequeño de observaciones se pierden estas consideraciones y los generadores se consideran con distribución uniforme, o por lo menos que no rechazan la hipótesis. Sin embargo si aumentamos el numero de repeticiones a un numero un poco mayor el resultado es el siguiente:

GENERADORES	TEST KOLMOGOROV SMIRNOV	¿PASÓ EL TEST?
RAND	$D_{calc}(0{,}0093) \leq D_{tabla}(0{,}0136)$	PASÓ
RANDU	$D_{calc}(0{,}0067) \leq D_{tabla}(0{,}0136)$	PASÓ
RANDOM PYTHON	$D_{calc}(0{,}0125) \leq D_{tabla}(0{,}0136)$	PASÓ
GCL MIXTO	$D_{calc}(0{,}0028) \leq D_{tabla}(0{,}0136)$	PASÓ
RANDOM.ORG	$D_{calc}(0{,}0084) \leq D_{tabla}(0{,}0136)$	PASÓ
CUADRADOS MEDIOS	$D_{calc}(0{,}2086) > D_{tabla}(0{,}0136)$	NO PASÓ

Vemos como aumentando el numero de repeticiones el generador no congruencial no pasa el test. Esto se debe claramente a que, como se vio con el Test Chi-Cuadrado, existe una forma escalonada de representar este método en términos de la frecuencia acumulada. Estos "escalones" hacen que el calculo de la D se encuentre muy alejado del valor esperado. Si se observa el valor calculado es de 0.2086 mientras que el rango máximo permitido(valor de la tabla) es de 0.0136. Esta disparidad que presenta con respecto a los demás generadores esta dada propiamente por estos escalones.

6. Conclusión

Como conclusión podemos decir que la peor forma de elegir un generador de números pseudoaleatorios es por defecto, simplemente usando uno que esté empaquetado con el software. Muchos generadores estándar, incluido el ampliamente utilizado GCL, tienen períodos que son inadecuados para algunos cálculos de valor en riesgo. La literatura documenta generadores de alta calidad que han sido exhaustivamente probados y recomendados por expertos. Queremos recalcar algunos puntos importantes que se desprendieron a lo largo de este trabajo:

- A pesar de sus limitaciones, es mejor usar generadores lineales como GCL. Estos han sido estudiados durante mucho tiempo y son bien entendidos.
- Un generador no congruencial, en nuestro caso: el método de los cuadrados medios, no es recomendable, quiera o no generar números pequeños. Esto esta altamente relacionado con la gran ineficiencia que presentó a lo largo de todo el análisis.
- Los generadores físicos generan resultados óptimos pero debido a que para realizar una simulación es deseable y exigible que se pueda repetir una misma secuencia, estos no son los adecuados.
- El generador de números pseudoaleatorios de Python, según lo evaluado, presenta características muy cercanas a lo considerado aleatorio y, según lo evaluado, es recomendable utilizarlo para la creación de simulaciones.

Pudimos analizar diferentes aspectos y comportamientos de diferentes generadores y analizarlos mediante diferentes pruebas, si bien a veces no es recomendable que un generador pase todos los Test, ya que como vimos, existen formas en las que un generador puede burlar un Test y pasarlo sin problema alguno.

Referencias

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

[2] Detalle de los pasos para corridas arriba y abajo de la media https://es.slideshare.net/pakitofive/prueba-de-corridas-arriba-y-abajo-de-la-media

[3] Documentación oficial de Scipy. Detalle de las funciones que tiene precargadas. Su funcionamiento y parámetros Web: scipy.org

[4] Phyton https://python-para-impacientes.blogspot.com/2014/08/graficos-en-ipython.html

[5] Probabilidad y estadistica https://relopezbriega.github.io/blog/2015/06/27/probabilidad-y-estadistica-con-python/

[6] Test corridas y bajadas https://es.slideshare.net/pakitofive/prueba-de-corridas-arriba-y-abajo-de-la-media,

[7] Numeros pseudoaleatorios https://es.slideshare.net/iorifoar/numeros-pseudoaleatorios,

[8] Numeros pseudoaleatorios https://es.khanacademy.org/computing/computer-science/cryptography/crypt/v/random-vs-pseudorandom-number-generators

[9] Numeros pseudoaleatorios http://www.sci.csueastbay.edu/ btrumbo/Stat3401/Hand3401/CongGenIntroB.pdf

[10] Numeros pseudoaleatorios https://es.slideshare.net/iorifoar/numeros-pseudoaleatorios,

[11] Numeros pseudoaleatorios https://www.lawebdefisica.com/apuntsmat/num$_{a}leatorios/$,

[12] IBM Historia https://www.ciencia-explicada.com/2012/01/pifia-historica-de-ibm-de-cuando-los.html

[13] GCL combinados https://www.fing.edu.uy/inco/cursos/mmc/unidad04/sesion10/transp.pdf

[14] GCL combinados https://www.monografias.com/trabajos106/numeros-aleatorios/numeros-aleatorios2.shtml

[15] GCL combinados https://www.famaf.unc.edu.ar/ jgimenez/Modelos$_{y}simulacion/2013/clase5.pdf$