TP 2.1: GENERADORES PSEUDOALEATORIOS

Tomás Ponce
Legajo: 44954
Mail: tomasponce@outlook.com.ar
UTN - FRRO
Zeballos 1341, S2000

Sofía Gasparini
Legajo:44762.
Mail: sofigasparini15@gmail.com
UTN - FRRO
Zeballos 1341, S2000

ABSTRACT

Nuestro trabajo tiene como objetivo mostrar como se realiza la generación de números pseudoaleatorios abordados por cinco algoritmos diferentes y 4 test diferentes. La programación del trabajo se realizó con el lenguaje Python en su versión 3.7.

1. Introducción

Se quiere simular y poder observar la generación de números pseudoaleatorios a través de la utilización de cinco algoritmos. Éstos, serán programados por nosotros y uno será propio del lenguaje Python y de la librería Random. Para el análisis de estos generadores utilizaremos determinados tests para evaluar el comportamiento de los resultados obtenidos.

2. Números Aleatorios

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.

El desarrollo de las ciencias de la computación se ha edificado, en gran parte, sobre la base de sólidos conceptos de las matemáticas. los números aleatorios son un buen ejemplo. estos han sido utilizados tradicionalmente en una gran variedad de aplicaciones (juegos, criptografía, experimentos científicos, etc.), y constituyen el fundamento para el estudio y modelaje de sistemas estocásticos.

La calidad de los números aleatorios es un factor crítico de éxito para la solución de problemas en diferentes áreas; ésta es garantizada gracias a la teoría matemática y a las ventajas que ofrecen los computadores para la implementación de los diferentes métodos propuestos para la generación eficiente de dichos números.

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 .

El campo de aplicación de los números pseudoaleatorios es muy amplio, entre ellos se encuentran : simulaciones, muestreos, análisis numéricos, testeo de programas, juegos de azar, toma de decisiones, etc.

3. Números Pseudoaleatorios

Antes de las computadoras, existieron diferentes métodos para generar secuencias de números aleatorios, tales como los procedimientos físicos(monedas, dados, bolilleros, etc), tablas de 40000 dígitos aleatorios por Tipett en 1927(no

resultaron con distribución uniforme),tabla de 100.000 números aleatorios hecha por Kendall en 1939, tabla de 1.000.000 numeros aleatorios a partir de ruido electrónico hecha por Rand Corporation en 1955, etc.

Las principales desventajas de todos estos intentos de realizar una secuencia de números aleatorios de manera física son que: no pueden repetirse una misma secuencia, no hay velocidad computacional e incorporar una tabla a la computadora implica gran costo de almacenamiento en relación a la cantidad de números (si se quiere realizar una tabla infinitamente aleatoria)

En 1946 Jon Von Neuman sugirió usar las operaciones aritméticas de una computadora para generar secuencias de número pseudoaleatorios, mediante el "método del valor medio". Este generador cae rápidamente en ciclos cortos, por ejemplo, si aparece un cero se propagará por siempre.

A inicios de 1950s se exploró el método y se propusieron mejoras, por ejemplo para evitar caer en cero. Metrópolis logró obtener una secuencia de 750,000 números distintos al usar semillas de 38 bits (usaba sistema binario), además la secuencia de Metrópolis mostraba propiedades deseables. No obstante, el método del valor medio no es considerado un método bueno por lo común de los ciclos cortos.

A partir de estos primeros métodos de generación de números pseudoaleatorios empezaron a surgir nuevos y mejores métodos para su utilización. 47

4. Generadores de números pseudoaleatorios

Un generador pseudoaleatorio de números es un algoritmo que produce una sucesión de números que es una muy buena aproximación a un conjunto aleatorio de números.

La mayoría de los algoritmos de generadores pseudoaleatorios producen sucesiones que poseen una distribución uniforme según varios tipos de pruebas. Las clases más comunes de estos algoritmos son generadores lineales congruentes, generadores Fibonacci demorados, desplazamiento de registros con retroalimentación lineal y desplazamientos de registros con retroalimentación generalizada.

En el presente trabajo analizaremos 5 generadores con sus respectivos métodos.

4.1. Generador por método de los cuadrados medios

Es un método propuesto en los años 40 por los matemáticos John von Neumann y Nicholas Metropolis, siendo utilizado para la generación de números pseudoaleatorios, Esto para obtener una sucesión de números que básicamente se obtienen a partir de recurrencia, los cuales son relevantes en los procesos de simulación debido a que con estos números se hace posible comprobar el correcto funcionamiento de una prueba mediante la observación del comportamiento de las variables que se puedan encontrar a lo largo de la simulación.

4.2. Generador lineal congruencial

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.1 La teoría que sustenta el proceso es relativamente fácil de entender, el algoritmo en si es de fácil implementación y su ejecución es rápida, especialmente cuando el hardware del ordenador puede soportar aritmética modular al truncar el bit de almacenamiento correspondiente.

En nuestro presente trabajo utilizaremos 3 generador GCL: El generador Rand (generador de la función rand en Matlab) , Randu(fue un generador de números aleatorios ampliamente utilizado en los 60s y 70s), y un GCL genérico.

En nuestro caso, elegimos una buena semilla para que se cumpla la condición de aleatoriedad debido a que si la semilla es demasiado pequeña, esta no se cumple.

4.3. Generador Mersenne twister

El Mersenne twister es un Generador de números pseudoaleatorios desarrollado en 1997 por Makoto Matsumoto y Takuji Nishimura reputado por su calidad.

Su nombre proviene del hecho de que la longitud del periodo corresponde a un Número primo de Mersenne. Existen al menos dos variantes de este algoritmo, distinguiéndose únicamente en el tamaño de primos Mersenne utilizados. El más reciente y más utilizado es el Mersenne Twister MT19937, con un tamaño de palabra de 32-bit. Existe otra variante con palabras de 64 bits, el MT19937-64, la cual genera otra secuencia.

En nuestro trabajo utilizaremos el generador random() de la libreria random de python(python utiliza un generador Mersenne Twister)

5. Tests utilizados

Tests Estadísticos: Los tests estadísticos (o contrastes) son el instrumento para validar o rechazar las hipótesis de modelación probabilistas. Ellos tratan de distinguir lo que es plausible de lo que es muy poco verosímil, en el marco de un modelo dado.

Los test de Chi-cuadrado y de Poker que se mencionarán a continuación serán para poder verificar la uniformidad de los números aleatorios generados. Esto nos va a permitir saber si los números generados son una muestra de la distribución uniforme en el intervalo [0,1]. Los demás test de corridas no demostrarán la uniformidad de la distribución sino que permitirán visualizar la aleatoriedad de los números generados.

5.1. Test de Bondad de Chi-Cuadrado

La prueba chi-cuadrado es una de las más conocidas y utilizadas para analizar variables nominales o cualitativas, es decir, para determinar la existencia o no de independencia entre dos variables. Que dos variables sean independientes significa que no tienen relación, y que por lo tanto una no depende de la otra, ni viceversa. Dividiremos en 10 intervalos a los números y los agruparemos para realizar este test((0;0.1), ...,(0.9;1)]

5.2. Test de Poker

Prueba grupos de números juntos como una mano de poker y compara cada mano con la mano esperada usando la prueba Chi-cuadrada. Esta prueba examina en forma individual los dígitos del número pseudoaleatorio generado. La forma como esta prueba se realiza es tomando 5 dígitos a al vez y clasificándolos como : Par, dos pares, tercia, póker quintilla full y todos diferentes. Las probabilidades para cada una de las manos del póker diferentes se muestran a continuación:

Todos diferentes = 0.3024.

Un par = 0.504.

Dos pares = 0.108.

Tercia = 0.072.

Full = 0.009.

Quintilla = 0.0001.

A modo de simplificación, nosotros optamos por tomar los primeros 3 dígitos decimales del número generado y no los primeros 5 como en el caso del test de póker original(Los digitos pueden ser todos diferentes, dos iguales o tres iguales).

5.3. Test de Corridas

Una prueba de Corridas es 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.

Si tenemos una secuencia de números de tal manera que a cada uno de los números siga otro mayor la secuencia dada será ascendente (arriba). Si cada número va seguido por otro menor, la secuencia será descendente (abajo).

5.4. Test de Corridas de Arriba y De Abajo de la Media para Números Uniformemente Distribuidos.

Es generalmente la prueba principal usada para verificar la dependencia. Esta prueba detecta si un patrón inaceptable estadísticamente que se incrementa o decrece existe entre números adyacentes en un flujo de números.

El procedimiento de esta prueba consiste en determinar una secuencia de + y -, de acuerdo con una comparación entre los simbolos del conjunto ri y 0.5. Denotaremos con +, a aquel número que se encuentre por arriba de 0.5.Por lo contrario, si el número se encuentra debajo de 0.5, lo denotaremos con -.Después se determina el número de corridas observadas Co y los valores de n,n0 y n1 (n es el total de números, n0 cantidad de simbolos -", y n1 cantidad de símbolos ").

6. Algoritmos de los tests utilizados

En estas sección presentaremos los algoritmos utilizados por cada test, para que se pueda entender de una forma más clara cómo es el procedimiento de los mismos.

6.1. Test de bondad Chi-cuadrado:

Procedimiento:

1-Partir el soporte de F en c celdas que son exhaustivas y mutuamente excluyentes.

2-Contar el número de observaciones en cada celda Oi.

3-Calcular el valor esperado en cada celda bajo F: ei=npi.

4-Calcular la estadística de prueba.

5-Verificar que el valor obtenido sea menor al valor obtenido en tabla con un porcentaje confianza determinado y acorde a los grados de libertad.

6.2. Test de Poker:

Procedimiento:

1 - Determinar la categoría de cada número del conjunto ri.

2- Contabilizar los números ri de la misma categoría o clase para obtener la frecuencia observada (Oi).

3- Calcular el estadístico de la prueba.

4- Finalmente comparar el estadístico.

6.3. Test Corridas:

Procedimiento:

1- Primeramente le asignaremos un signo a cada número de la secuencia ya sea + ó - , eso dependerá de lo siguiente.

2-Si a un número le sigue otro mayor, se le asigna +. Esto es si Xi <Xi +1 el signo asignado será (+). Siendo Xi un número de la muestra o secuencia de números.

3- Si el número siguiente es menor, se le da un signo -. Esto es si Xi >Xi +1 el signo asignado será (-).

4-Se continuará con la comparación de los números y la asignación de su signo correspondiente hasta N-1. Es decir hasta el penúltimo numero de la secuencia, ya que al último número le sigue un evento nulo(no es posible compararlo con otro número).

6.4. Test Corridas de Arriba y Abajo:

1- Primeramente le asignaremos un signo a cada número de la secuencia ya sea + ó - , eso dependerá de los siguiente

2-Denotaremos con un signo - a aquel número que se encuentre por debajo de la media. Denotaremos con un signo + a aquel número que se encuentre por arriba de la media.

3-Se continuará con la comparación de los números y la asignación de su signo correspondiente hasta N-1. Es decir hasta el penúltimo numero de la secuencia, ya que al último número le sigue un evento nulo(no es posible compararlo con otro número).

7. Metodología

Utilizando el lenguaje de programación python implementamos un programa con el propósito de generar números pseudoaleatorios a través de 5 algoritmos distintos. Utilizaremos las librerías random,numpy y math para estas tareas.

En el presente trabajo, realizamos 4 tests diferentes con el objetivo de poder corroborar con mayor precisión si los números generados son realmente pseudoaleatorios. En base a diferentes generadores, entre los cuales se encuentran el generador de cuadrado medio, el rand, randu, GCL genérico y Mersenne Twister,implementamos los test que fueron: Test de bondad de Chi Cuadrado, test de poker, test de corridas y el de corridas de arriba y abajo de la media para números uniformemente distribuidos.

Realizamos un total de 30 simulaciones para cada algoritmo de generación de números aleatorios para poder observar los resultados de los tests que nos indicaran si efectivamente los resultados son pseudoaleatorios.A partir de dichas simulaciones sacamos las respectivas conclusiones.

8. Exposición de los resultados y análisis de los mismos

8.1. Tabla comparativa de resultados

En la siguiente tabla se representaran expresaremos los resultados obtenidos por medio de los cuatro test, a partir de los cinco generadores

A través de las simulaciones realizadas pudimos observar que los resultados arrojados por los test dieron los valores esperados para que los generadores resulten generadores de números pseudoaleatorios(todos los test resultaron correctamente) salvo en el caso del generador del cuadrado medio en algunos test.

Generadores	Test Chi-Cuadrado	Pasó
Rand	Valor de Chi-cuadrado obtenido(5.74) <Valor de la tabla(16.92)	Verdadero
Randu	Valor de Chi-cuadrado obtenido(11.559) <Valor de la tabla(16.92)	Verdadero
GCL generico	Valor de Chi-cuadrado obtenido(8.6786) <Valor de la tabla(16.92)	Verdadero
Mersenne Twister	Valor de Chi-cuadrado obtenido(7.150) <Valor de la tabla(16.92)	Verdadero
Cuadrado Medio	Valor de Chi-cuadrado obtenido(134660.22) <Valor de la tabla(16.92)	Falso

Al realizar el test de Chi-cuadrado con los numeros generados, se puede observar que el valor del estadistico Chi-cuadrado obtenido para cada generador es menor al valor que se puede obtener con la tabla con un 95 % de confianza y 9 grados de libertad, excepto para el generador cuadrado medio que se observa un valor excesivamente grande.

Generadores	Test de Poker	Pasó
Rand	Valor de Chi-cuadrado obtenido(2.41) <Valor de la tabla(5.99)	Verdadero
Randu	Valor de Chi-cuadrado obtenido(4.7959) <Valor de la tabla(5.99)	Verdadero
GCL generico	Valor de Chi-cuadrado obtenido(0.669) <Valor de la tabla(5.99)	Verdadero
Mersenne Twister	Valor de Chi-cuadrado obtenido(0.5079) <Valor de la tabla(5.99)	Verdadero
Cuadrado Medio	Valor de Chi-cuadrado obtenido(42629.857) <Valor de la tabla(5.99)	Falso

Al realizar el test de Poker con los números generados, se puede observar que el valor del estadístico Chi-cuadrado obtenido para cada generador es menor al valor que se puede obtener con la tabla con un 95 % de confianza y 2 grados de libertad, excepto para el generador cuadrado medio que se observa un valor excesivamente grande.

Generadores	Test de Corridas	Pasó
Rand	Valor de Zo obtenido(-0.03873) <Zmax(1.96)	Verdadero
Randu	Valor de Zo obtenido(-0.3679) <Zmax(1.96)	Verdadero
GCL generico	Valor de Zo obtenido(0.3873) <Zmax(1.96)	Verdadero
Mersenne Twister	Valor de Zo obtenido(0.7746) <Zmax(1.96)	Verdadero
Cuadrado Medio	Valor de Zo obtenido(-1.9335)<Zmax(1.96)	Verdadero

En este Test lo que se pretende es agrupar las secuencias de mismo signo para poder obtener los valores que permitirán realizar la prueba.Todos los generadores lograron pasar este test

Generadores	Test corridas Arriba y Abajo	Pasó
Rand	Valor de Zo obtenido(-0.03873) <Zmax(1.96)	Verdadero
Randu	Valor de Zo obtenido(-0.6726) <Zmax(1.96)	Verdadero
GCL generico	Valor de Zo obtenido(-1.0639) <Zmax(1.96)	Verdadero
Mersenne Twister	Valor de Zo obtenido(1.16135) <Zmax(1.96)	Verdadero
Cuadrado Medio	Valor de Zo obtenido(-1.2099) <Zmax(1.96)	Verdadero

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.Este test todos los generadores lo lograron pasar.

Generadores	Tipo de generador	Test Chi-Cuadrado	Test de Poker	Test de Corridas	Test corridas Arriba y Abajo
Rand	Pseudoaleatorio	OK	OK	OK	OK
Randu	Pseudoaleatorio	OK	OK	OK	OK
GCL generico	Pseudoaleatorio	OK	OK	OK	OK
Mersenne Twister	Pseudoaleatorio	OK	OK	OK	OK
Cuadrado Medio	Pseudoaleatorio	FALSO	FALSO	OK	OK

En esta tabla, se pueden observar cuales fueron los generadores que pasaron y fallaron los distintos test que corroboraban que los mismos sean generadores de números pseudoaleatorios.Todos los generadores pasaron todas las pruebas, a excepción del generador cuadrado medio que fallo dos de ellos, específicamente el test Chi-cuadrado y el test de Poker.Esto se debe a que estos test se encargan de comprobar si existe independencia entre las variables, sin embargo el método de los cuadrados muestra claramente la relación que existe entre los números generados.

8.2. Gráficos de dispersión

En los siguientes gráficos se representarán las dispersiones de los valores obtenidos por cada uno de los generadores. La primer imagen representa el generador Rand, la segunda el Randu, la tercera el GCL Genérico, la cuarta el generador Mersenne twister y la quinta hace referencia al generador de Cuadrado Medio.

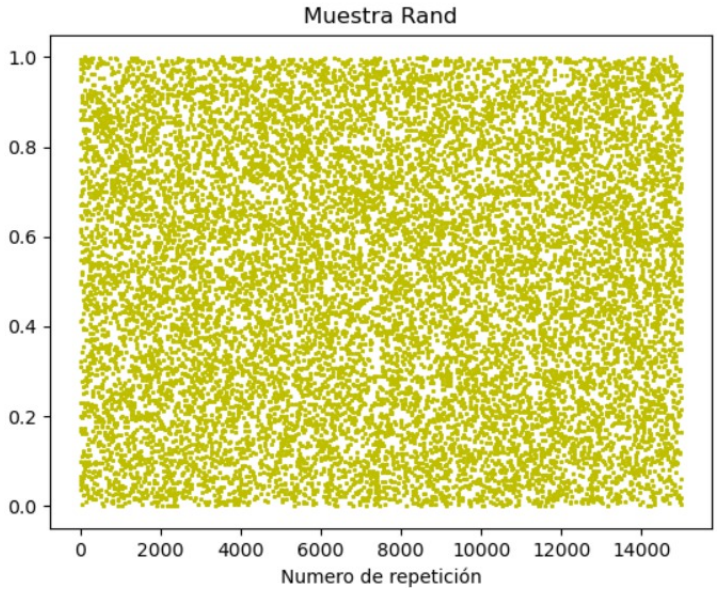

Figura 1: Gráfico de dispersión Rand

En esta gráfica, podemos observar la dispersión que tienen los números generados por el generador Rand. En el eje de las abscisas se encuentran los números de repeticiones del generado y en el eje de las ordenadas, el número generado. Se observa que existe realmente una dispersión.

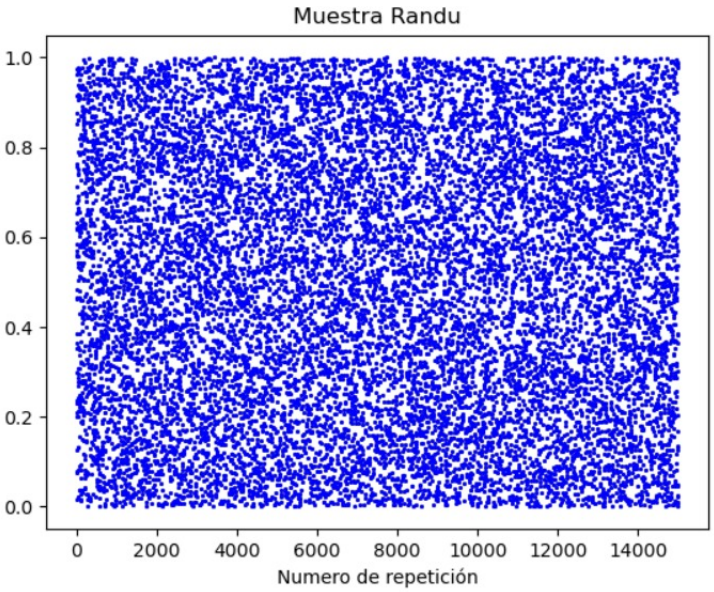

Figura 2: Gráfico de dispersión Randu

En esta gráfica, podemos observar la dispersión que tienen los números generados por el generador Randu. En el eje de las abscisas se encuentran los números de repeticiones del generado y en el eje de las ordenadas, el número generado. Se observa que existe realmente una dispersión.

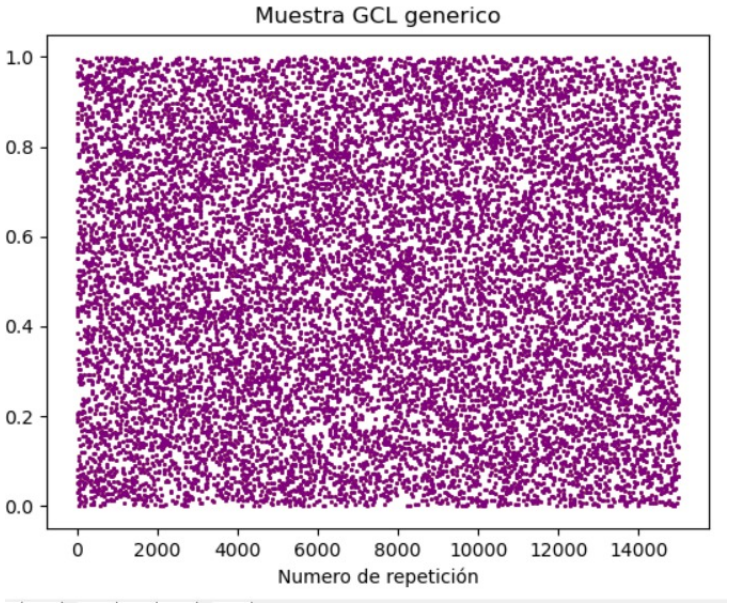

Figura 3: Gráfico de dispersión Gcl Genérico

En esta gráfica, podemos observar la dispersión que tienen los números generados por el generador GCL generico. En el eje de las abscisas se encuentran los números de repeticiones del generado y en el eje de las ordenadas, el número generado. Se observa que existe realmente una dispersión.

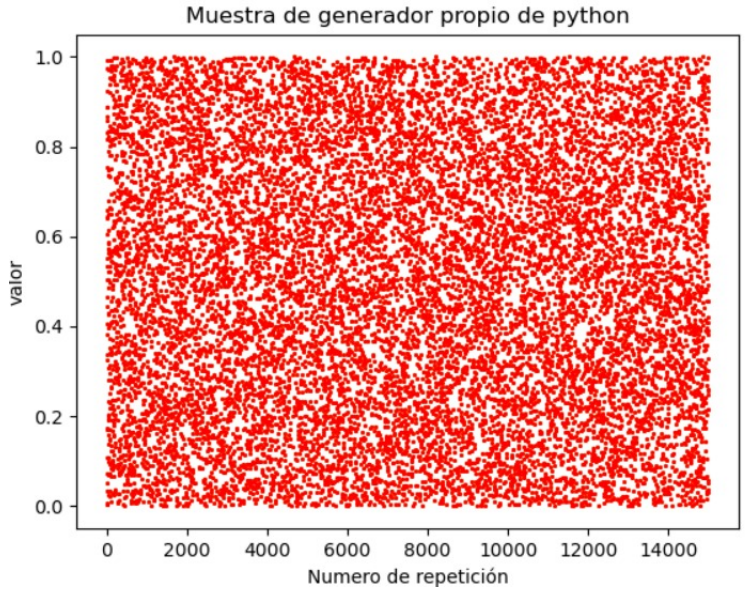

Figura 4: Gráfico de dispersión Mersenne twister

En esta gráfica, podemos observar la dispersión que tienen los números generados por el generador Mersenne twister. En el eje de las abscisas se encuentran los números de repeticiones del generado y en el eje de las ordenadas, el número generado. Se observa que existe realmente una dispersión.

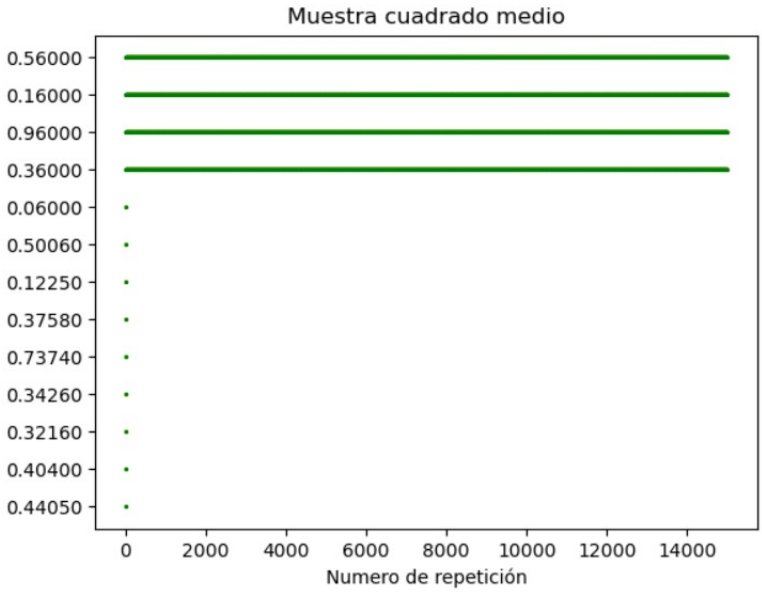

Figura 5: Gráfico de dispersión cuadrado medio

En esta gráfica, podemos observar la dispersión que tienen los números generados por el generador Cuadrado medio. En el eje de las abscisas se encuentran los números de repeticiones del generado y en el eje de las ordenadas, el número generado. Se observa que no existe una dispersión y los mismos números se repiten de manera constante.

9. Fórmulas

9.1. Fórmulas empleadas para los generadores

Generador Rand:

$$x_{n+1} = (7^5)x_n mod(2^{31} - 1)$$

Donde:

- m = 2^{31}
- a = 7^5
- n = 0, 1, 2, 3, ...
- X0= cualquier numero entre 0 y $2^{31} - 1$

Generador Randu:

$$x_{n+1} = (2^{(16)} + 3)x_n mod(2^{31})$$

Donde:

- m = 2^{31}
- a = 7^5
- c = 3
- n = 0, 1, 2, 3, ...
- X0= cualquier numero entre 0 y $2^{31} - 1$

Generador GCL Genérico:

$$x_{n+1} = (ax_n + c)mod(m)$$

Donde:

- m es el "módulo", m >0
- a es el multiplicador, 0 a <m
- c es el incremento, c m
- X0 es la semilla, 0 X0 <m

9.2. Fórmulas empleadas para los test

Test de Bondad de Chi-Cuadrado:

$$x^{(2)} = \sum_{i=1}^{c} \frac{((o_i - e_i)^2}{e_i}$$

Donde:

- e es el valor esperado
- o es el valor observado en cada intervalo

Test de Póker:

$$x_0^{(3)} = \sum_{i=1}^{3} \frac{(FO_i - FE_i)^{(2)}}{FE_i}$$

Donde:

- FE es el valor esperado
- FO es la cantidad observada en cada intervalo

Test de Corridas:

- Media

$$\mu_a = \frac{2N - 1}{3}$$

- Desviación

$$(\sigma_a)^2 = \frac{16N - 29}{90}$$

- Fórmula Z

$$Z = \frac{a_1 - \mu_a}{\sigma_a}$$

Donde:

- N es la cantidad de números generados.
- mu es la media.
- sigma es el desvío.
- a1 es la cantidad de sucesiones.

Test de Corridas de Arriba y De Abajo de la Media para Números uniformemente distribuidos:

- Media

$$\mu_b = \frac{2n_1n_2}{n_1 + n_2} + 1$$

- Desviación

$$\sigma_b = \frac{2n_1n_2(2n_1n_2 - N)}{N^2(N - 1)}$$

- Formula Z

$$Z = \frac{b - \mu_b}{\sigma_b}$$

Donde:

- N es la cantidad de números generados.
- mu es la media.
- sigma es el desvío.
- n1 es la cantidad de números por encima de la media.
- n2 es la cantidad de números por debajo de la media.
- b es la cantidad de sucesiones.

10. Conclusiones

Luego de realizar múltiples simulaciones y al visualizar lo observado, llegamos a la conclusión de que los números generados por el generador Rand, Randu, GCL genérico y Mersenne twister (de la librería random de python) se distribuyen de manera pseudoaleatoria, mientras que la distribución generada por el método de los cuadrados medios no es pseudoaleatoria.

Esto no implica que los números generados sean efectivamente aleatorios , sino que para los tests que nosotros realizamos, se corrobora la aleatoriedad. Sin embargo, no es imposible que si se realizasen mas pruebas con distintos test, ellas podrían abordar resultados fallidos.

El generador que implica mayor ineficiencia al momento de generar números aleatorios es el realizado a partir del método de los cuadrados medios, debido a los números que se repiten de manera constante, y los bucles que este genera , tal como un bucle infinito de ceros. Sin embargo ,su creación para dicha época fue mas que aceptada, debido a la novedad de algoritmo que planteaba este, permitiendo dar paso a los otros algoritmos que surgieron en la posteridad. Este generador logro pasar los test de corridas debido a que estos contemplan la uniformidad de los números generados, y al repetirse una misma secuencia de números cada cierta cantidad de veces, es indudable que pasaría la prueba.

A pesar de que 4 de los generadores que planteamos hayan pasado las 4 pruebas no nos indican que específicamente sean los mejores generadores que se puedan utilizar para fines mas prácticos.Sin embargo, podemos afirmar que dichos generadores pueden servir para los fines expuestos en nuestro trabajo, que efectivamente es corroborar la aleatoriedad de los números generados.

11. Referencias

http://hemaruce.angelfire.com/Unidad_III.pdf

https://psicologiaymente.com/miscelanea/prueba-chi-cuadrado

https://es.wikipedia.org/wiki/N%C3%BAmero_pseudoaleatorio

https://es.wikipedia.org/wiki/N%C3%BAmero_aleatorio

https://es.wikipedia.org/wiki/Generador_de_n%C3%BAmeros_pseudoaleatorios

https://es.wikipedia.org/wiki/Generador_lineal_congruencial

https://es.wikibooks.org/wiki/M%C3%A9todo_de_los_cuadrados_medios_para_la_generaci%C3%B3n_de_n%C3%BAmeros_pseudoaleatorios

http://www2.famaf.unc.edu.ar/~jgimenez/Modelos_y_Simulacion/2013/clase5.pdf

https://www.random.org/analysis/