TP 2.1 - GENERADORES PSEUDOALEATORIOS

Kevin Masci Carmody
Cátedra Simulación UTN Frro
41255
masci.carmody.kevin@gmail.com

Celeste Figuera
Cátedra Simulación UTN Frro
48073
figueracele@gmail.com

Florencia Toledo
Cátedra Simulación UTN Frro
47796
flortoledo157@gmail.com

Mijael Nasatsky
Cátedra Simulación UTN Frro
48100
mija191100@gmail.com

27 de mayo de 2023

ABSTRACT

En este informe se detallan los resultados obtenidos al estudiar el comportamiento de distintos generadores de números pseudoaleatorios, los cuales son utilizados en diversas áreas de la computación y la simulación para generar secuencias de números que aparentan ser aleatorias. Cada uno de estos será sometido a cuatro tests de aleatoriedad para evaluar la calidad de sus resultados.

Keywords

Simulación-Números Pseudoaletorios-Generadores

1. Introducción

Una de las características más importantes de la simulación es la capacidad de imitar el comportamiento aleatorio que existe en los sistemas estocásticos. Para simular este comportamiento aleatorio se requiere de un método que provea la generación de estos números aleatorios, así como de rutinas para generar variaciones aleatorias, basadas en distribuciones de probabilidad. En general, la validez de los métodos de transformación depende fuertemente de la hipótesis de que los valores de partida son realizaciones de variables aleatorias, pero esta suposición realmente no se cumple, puesto que los generadores de números aleatorios son simplemente programas determinísticos que intentan reproducir una sucesión de valores que parezca aleatoria.

2. Metodología

2.1. Desarrollo

La simulación se lleva a cabo mediante el lenguaje de programación python, utilizando las siguientes librerías: math, numpy, random y PIL El estudio consiste en realizar tres generadores de números pseudoaleatorios y luego cuatro pruebas de aleatoriedad, las cuales son aplicadas a cada uno de estos para determinar si la secuencia que generan es realmente aleatoria. Además, se realizan gráficos llamados mapas de bits, los cuales consisten en crear una visualización de los números producidos por cada generador. Los parámetros utilizados para cada uno de los generadores fueron:

$$n = 250000$$
$$seed = 9731$$

donde n representa la cantidad de valores a generar y $seed$ la semilla a utilizar.

3. Marco teórico

3.1. Números aleatorios

Cuando hablamos aleatoriedad nos referimos a aquello que es relativo o depende del azar, es decir que no se puede predecir. Si nos referimos a números aleatorios justamente nos referimos a esto último, no podemos predecir qué número será el siguiente.

Un número aleatorio es uno que se extrae de un conjunto de valores posibles, cada uno de los cuales es igualmente

probable. En estadística, esto se denomina distribución uniforme, porque la distribución de probabilidades para cada número es la misma en todo el rango de valores posibles.

Cuando se habla de una secuencia de números aleatorios, estos deben ser independientes unos de otros, y el valor del siguiente elemento no puede ser predecible, a pesar de cuantos elementos ya se hayan producidos. Estos números son usados de diferente maneras para distribuciones particulares.

Sin embargo, es difícil comprobar si una determinada secuencia de números es aleatoria, ya que, si su generador de números aleatorios es bueno, es igualmente probable que aparezca cada secuencia posible de valores. Esto significa que un buen generador de números aleatorios también producirá secuencias que parecen no aleatorias para el ojo humano y que fallan en cualquier prueba estadística a la que podamos exponerlo.

A pesar de esto, a medida que las secuencias pasan más pruebas, aumenta la confianza en la aleatoriedad de los números y también la confianza en el generador. Sin embargo, debido a que esperamos que algunas secuencias no parezcan aleatorias, debemos esperar que algunas de las mismas fallen al menos en algunas de las pruebas. Igualmente, si muchas secuencias fallan en las pruebas, deberíamos sospechar.

3.1.1. Aplicaciones de números aleatorios

Los números aleatorios son cruciales para una gran cantidad de areas, entre ellas criptografía, toma de decisiones, juegos, muestreo y simulación. Nos enfocaremos en esta última, que es la recreación, de manera simplificada de fenómenos complejos, ambientes o experiencias, proveyendo al usuario la oportunidad de otro nivel de entendimiento. Por ejemplo cuando queremos simular fenómenos naturales los números aleatorios son necesarios para lograr una simulación realista.

Cada vez mas estudios de simulación requieren más y más números aleatorios y estos resultados son muy sensibles a la calidad del generador de los números aleatorios.Los números aleatorios son la base esencial de la simulación. Usualmente, toda la aleatoriedad involucrada en el modelo se obtiene a partir de un generador de números aleatorios que produce una sucesión de valores que supuestamente son realizaciones de una secuencia de variables aleatorias independientes e idénticamente distribuidas. Posteriormente estos números aleatorios se transforman convenientemente para simular las diferentes distribuciones de probabilidad que se requieran en el modelo. En general, la validez de los métodos de transformación dependen fuertemente de la hipótesis de que los valores de partida son realizaciones de variables aleatorias, pero esta suposición no siempre se cumple puesto que los generadores de números aleatorios son simplementes programas determinísticos que intentan reproducir una sucesión de valores que parezca aleatoria.

3.2. Números pseudo-aleatorios

Los números pseudo-aleatorios no dependen de una fuente de entropía para su genereción, se calculan usando una función matemática, que dados determinados valores de entrada, denominados semilla del generador, arroja una serie de números que a simple vista parecieran ser aleatorios, a pesar de estar completamente determinados por las condiciones iniciales y ser fácilmente reproducibles.

3.3. Generadores de números pseudo-aleatorios utilizados

3.3.1. Generador Lineal Congruencial (GCL)

Uno de los tipos más comunes es el generador lineal congruencial (GCL). El mismo utiliza una relación recursiva que toma una semilla inicial y la transforma en un nuevo número utilizando una ecuación de la forma

$$X_{i+1} = (aX_i + c)modm$$

donde a, c y m son constantes predefinidas y X_i es el número anterior en la secuencia. La calidad de un GCL depende de los valores elegidos para a, c, m y la semilla inicial. Para este generador se utilizan los siguientes parámetros:

$$\begin{aligned} X_0 &= 9731 \\ a &= 1103515245 \\ c &= 12345 \\ m &= 2^{31} \end{aligned}$$

3.3.2. Generador Randu

El generador RANDU produce números pseudoaleatorios utilizando una ecuación lineal congruencial para producir una secuencia de números. Esta ecuación tiene la forma:

$$X_{n+1} = (2^{16} + 3)X_n mod(2^3 1) \quad (1)$$

donde:

X_{n+1} es el número generado en el n-ésimo paso.

mod: operación de módulo.

Este generador es un algoritmo de generación de números pseudoaleatorios que utiliza una ecuación lineal congruencial para producir una secuencia de números. Aunque se utiliza en muchas aplicaciones, se debe tener en cuenta que la ecuación de RANDU puede producir patrones predecibles en la secuencia de números generados.

3.3.3. Generador mid-square

El generador de la parte media del cuadrado (en ingles "middle-square method") es un algoritmo utilizado para producir números pseudoaleatorios.

Este comienza con un número inicial (llamado "semilla") de n dígitos. En cada iteración, se eleva al cuadrado la semilla, se toman los n dígitos centrales del resultado y

se utilizan como la nueva semilla. Los números generados se toman como los n dígitos centrales de la semilla en cada iteración. En este estudio se utilizará una semilla de 4 dígitos, produciendo un número de 8 dígitos cuando se la eleva al cuadrado (si el resultado tiene menos de 8 dígitos se añaden ceros al inicio).

3.3.4. Generador random.uniform (Python)

Este generador es una función de Python que produce números pseudoaleatorios con una distribución uniforme dentro de un rango especificado. La función utiliza dos argumentos: a y b, que representan el inicio y el final del rango, respectivamente.

Al utilizar este generador, se pueden obtener números pseudoaleatorios que están uniformemente distribuidos en el rango especificado. Esto significa que cada número en el rango tiene la misma probabilidad de ser generado.

3.4. Pruebas

Las pruebas de aleatoriedad son un conjunto de técnicas estadísticas que se utilizan para determinar si una secuencia de números aparentemente aleatorios es verdaderamente aleatoria o si hay algún patrón subyacente en ella. Estas pruebas son útiles para evaluar la calidad de los generadores de números pseudoaleatorios, así como para detectar posibles errores en experimentos o simulaciones que requieren de números aleatorios. Estas se basan en la comparación de la secuencia de números en cuestión con un conjunto de criterios de aleatoriedad que se espera que cumpla una secuencia verdaderamente aleatoria. Algunos de estos criterios incluyen la uniformidad de la distribución, la independencia entre los números y la falta de patrones o regularidades en la secuencia.

3.4.1. Análisis visual simple

Una forma de examinar un generador de números es crear una visualización de los números que produce para detectar patrones. Si bien no debe considerar este tipo de enfoque como un análisis exhaustivo o formal, es una forma agradable y rápida de obtener una impresión aproximada del rendimiento de un generador determinado.

3.4.2. Prueba de frecuencia: monobit

La prueba de frecuencia monobit, también conocida como prueba de frecuencia de bits o prueba de frecuencia simple, es utilizada en el análisis de secuencias de bits para evaluar si la proporción de unos y ceros es aproximadamente igual y se ajusta a un patrón aleatorio. El objetivo es verificar si una secuencia binaria dada tiene una distribución uniforme y no muestra ninguna tendencia o sesgo hacia unos o ceros.

El proceso básico de la prueba de frecuencia monobit implica los siguientes pasos:

1. Contar el número de unos y ceros en la secuencia de bits.

2. Se calcula la estadística de prueba utilizando la siguiente fórmula: S = (número de unos - número de ceros) / raíz cuadrada de la longitud de la secuencia
La idea es comparar la diferencia entre el número de unos y ceros con respecto a la esperanza de una distribución aleatoria, que es cero.

3. Se establece un nivel de significancia, generalmente denotado como α, que determina cuán pequeña debe ser la probabilidad de obtener la estadística de prueba por pura casualidad para rechazar la hipótesis nula de que la secuencia es aleatoria.

4. Se compara el valor calculado de la estadística de prueba con el nivel de significancia. Si el valor calculado es menor que el nivel de significancia, entonces se rechaza la hipótesis nula y se concluye que la secuencia no es aleatoria. En caso contrario, no se rechaza la hipótesis nula y se concluye que la secuencia es aleatoria.

Para este test no se establece un nivel de significancia específico, ya que se utiliza el valor predeterminado por una función del lenguaje utilizado, que devuelve la probabilidad de obtener una variable aleatoria normal estándar mayor que un valor absoluto de S, que es la estadística de prueba calculada en el paso 2. Por lo tanto, el nivel de significancia utilizado dependerá del valor calculado de S y la distribución de probabilidad de la variable aleatoria normal estándar.

3.4.3. Prueba de frecuencia: bloque

La prueba de frecuencias en bloques es utilizada para evaluar la aleatoriedad en una secuencia binaria dividiéndola en bloques y analizando las frecuencias de rachas dentro de esos bloques. El objetivo es verificar si la proporción de unos y ceros es uniforme y si no hay patrones repetitivos en la secuencia. El proceso básico de la prueba de frecuencias en bloques implica los siguientes pasos:

1. Dividir la secuencia: Se divide la secuencia binaria en bloques de un tamaño predefinido.

2. Calcular las proporciones: Para cada bloque, se calcula la proporción de unos dividiendo el número de unos en el bloque entre el tamaño del bloque. Esto proporciona la proporción de unos en cada bloque de la secuencia.

3. Establecer la proporción esperada: En una secuencia aleatoria, se espera que la proporción de unos en cada bloque sea de 0.5, lo que significa que debería haber una distribución equilibrada de unos y ceros.

4. Calcular el estadístico chi-cuadrado: Se calcula un estadístico llamado chi-cuadrado para evaluar la discrepancia entre las proporciones observadas y las proporciones esperadas. El estadístico chi-cuadrado mide la diferencia al cuadrado entre cada proporción observada y la proporción esperada, y se suma para todos los bloques.

5. Comparar con un valor crítico: Se establece un valor crítico para el estadístico chi-cuadrado, que depende del nivel de significancia deseado. Si el valor calculado del es-

tadístico chi-cuadrado es menor o igual que el valor crítico, se acepta la hipótesis de que la secuencia es aleatoria. Si el valor calculado es mayor que el valor crítico, se rechaza la hipótesis y se concluye que la secuencia no es aleatoria.

En este estudio se utiliza un valor crítico de 3.841, que corresponde a un nivel de signficancia de 0.05 y 2 grados de libertad.

3.4.4. Prueba de ejecuciones

El runs test, o prueba de ejecuciones, es utilizada para evaluar la aleatoriedad en una secuencia de datos binarios o dicotómicos. El objetivo principal de este es determinar si la secuencia exhibe patrones sistemáticos o si parece ser aleatoria.

En esta prueba, una racha se define como una secuencia consecutiva de unos o de ceros en la secuencia binaria. La idea detrás del runs test es analizar la longitud y cantidad de rachas en la secuencia y comparar estos valores con los esperados en una secuencia verdaderamente aleatoria. Para realizar la prueba de rachas, se siguen estos pasos:

1. Identificar las rachas, es decir, contar el número de veces que aparece una misma secuencia de bits consecutivos en la secuencia completa.

2. Calcular el número de rachas en la secuencia completa.

3. Calcular el número esperado de rachas en una secuencia aleatoria, utilizando la fórmula:

$$expected_runs = (2 * n - 1)/3$$

donde n es la longitud de la secuencia de bits.

4. Comparar el número de rachas observado con el número esperado de rachas en una secuencia aleatoria. Si la diferencia absoluta entre estos dos valores es mayor que el valor esperado, entonces la secuencia de bits no es aleatoria. De lo contrario, se considera que es aleatoria.

3.4.5. Chi cuadrado

Este test se utiliza para evaluar la uniformidad de una distribución de frecuencia observada en comparación con una distribución esperada. También se puede utilizar para evaluar la calidad de los números pseudoaleatorios producidos por un generador de números aleatorios. Su método consiste en lo siguiente:

1. Dividir el intervalo [0,1] en un número de subintervalos. En este caso, se lo divide en 10 subintervalos.

2. Contar el número de veces que aparece un número en cada subintervalo. Esto se conoce como frecuencia observada.

3. Calcular la frecuencia esperada para cada subintervalo. Si la distribución es uniforme, la frecuencia esperada en cada subintervalo es el mismo número. En este caso, la frecuencia esperada es $n/10$, donde n es el número total de números en la secuencia.

4. Calcular el estadístico de prueba chi-cuadrado, que es la suma de las diferencias al cuadrado entre las frecuencias observadas y esperadas divididas por las frecuencias esperadas.

5. Calcular el valor p asociado al estadístico de prueba. Este valor es la probabilidad de obtener un estadístico de prueba igual o más extremo que el observado, asumiendo que la distribución es uniforme.

6. Comparar el valor p con un nivel de significancia predefinido. Si el valor es menor que el nivel de significancia, se rechaza la hipótesis nula de que la distribución es uniforme. De lo contrario, se acepta la hipótesis nula y se concluye que la distribución es uniforme.

En este caso, el nivel de significancia utilizado para comparar el valor p es 0.05, ya que se compara el valor p con 1 - la función de distribución acumulada de la distribución chi-cuadrado con 9 grados de libertad, lo que corresponde a un nivel de significancia del 5

4. Resultados

4.1. Mapas de bits

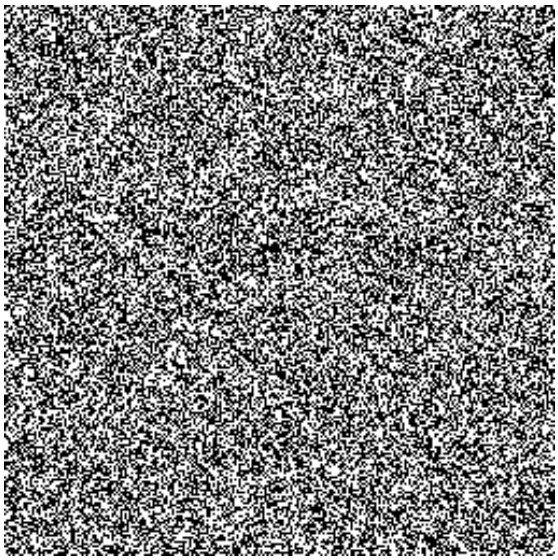

Figura 1: Mapa de bits del generador GCL

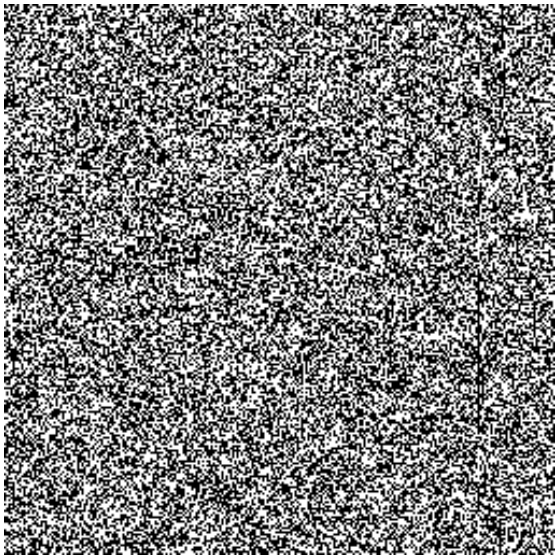

Figura 2: Mapa de bits del generador Randu

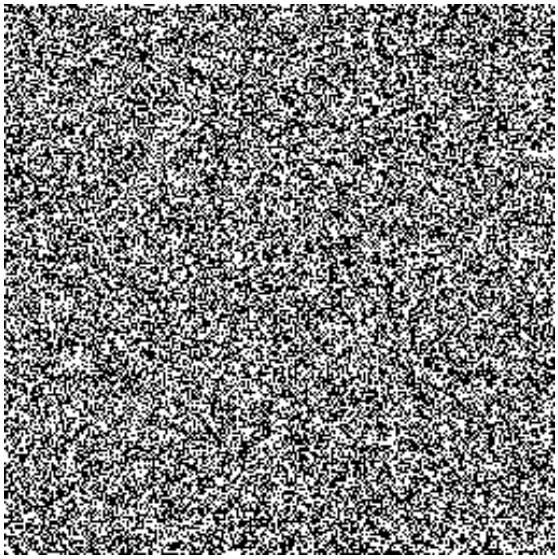

Figura 4: Mapa de bits del generador de Python

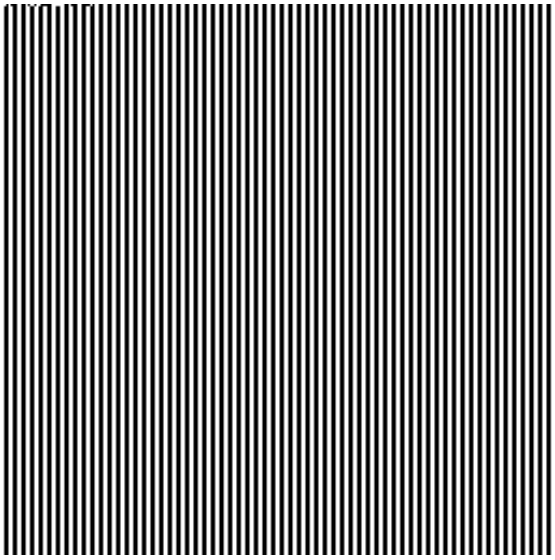

Figura 3: Mapa de bits del generador Mid Square

4.2. Tabla de resultados

Generador	Monobit	Bloque	Runs test	Chi cuadrado	Visual
GCL	p = 0.7520025081334136	False	True	p = 0.5979266964656029	SI
Randu	p = 0.5485062355001472	False	True	p = 0.84360575540906	SI
Mid Square	p = 0.9968084702675367	True	True	p = 0.0	NO
Python	p = 0.261013025945121	False	True	p = 0.684129634935208	SI

El valor p obtenido en las pruebas de Monobit y Chi-Cuadrado es una medida de la evidencia en contra de la hipótesis nula de que la secuencia binaria es aleatoria. La interpretación típica del valor p es la siguiente:

- Si el valor p es alto, no hay suficiente evidencia para rechazar la hipótesis nula. Esto sugiere que la secuencia binaria podría ser considerada como aleatoria.
- Si el valor p es bajo, hay suficiente evidencia para rechazar la hipótesis nula. Esto sugiere que la secuencia binaria no es aleatoria y puede haber algún tipo de patrón o estructura subyacente en ella.

Es importante tener en cuenta que el umbral para considerar un valor p como "bajo"puede variar dependiendo del contexto y de las convenciones estadísticas establecidas. Un umbral comúnmente utilizado es 0.05.

Sin embargo, la interpretación precisa también debe considerar otros factores, como el tamaño de la muestra y el diseño de la prueba estadística utilizada. Además, es recomendable complementar el análisis con otras pruebas y técnicas para evaluar la aleatoriedad de la secuencia de manera más completa.

En el caso de la prueba de frecuencia en bloques, se utiliza un valor crítico de 3.841, que corresponde a un nivel de signficancia de 0.05 y 2 grados de libertad. Si el estadístico obtenido es menor que ese valor, se devuelve True, lo que indica que la secuencia se puede considerar aleatoria. En caso contrario, se devuelve False.

La prueba de ejecuciones o runs test, devuelve True o False según si la secuencia es aleatoria o no lo es. Como se explicó anteriormente, para determinar si la secuencia es aleatoria se calcula la diferencia absoluta entre el valor observado de rachas y el esperado, y se la compara con el valor esperado.

5. Conclusión

Luego de llevar a cabo las pruebas visuales y estadísticas planteadas, podemos ver que ninguno de los generadores utilizados logra superar la totalidad de las pruebas. Por lo que podemos concluir que la utilización de estos es recomendada según las necesidades específicas de aplicación.

Dado que los generadores de números pseudoaleatorios son deterministas, si se utiliza la misma semilla inicial, generarán la misma secuencia de números. Esto permite la reproducción de experimentos o simulaciones, lo cual puede ser útil para fines de investigación y desarrollo.

Referencias

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

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