TP 2.1 - GENERADORES PSEUDOALEATORIOS - SIMULACIÓN

Juan Cruz Vazquez
Simulacion
UTN - FRRo
Zeballos 1341, S2000 Rosario, Santa Fe
juancruz.vazquez87@hotmail.com

Alexis Tomás
Simulacion
UTN - FRRo
Zeballos 1341, S2000 Rosario, Santa Fe
alexisjosetomas@gmail.com

May 29, 2023

ABSTRACT

La simulación de un modelo consiste en la construcción de un programa computacional que permite obtener los valores de las varibles de salida para distintos valores de las variables de entrada con el objetivo de obtener conclusiones del sistema que apoyen la toma de decisiones (explicar y/o predecir el comportamiento del sistema). Uno de los principales requerimientos prácticos de un proyecto de simulación consiste en una fuente de números aleatorios (números pseudoaleatorios). En este trabajo nos dedicaremos a estudiar cómo funcionan los generadores de números pseudoaleatorios y realizar una comparación entre ellos.

1 Introduccion

Nuestro trabajo trata del estudio del funcionamiento de los diversos generadores de números pseudoaleatorios. Estos generadores son algoritmos que producen una sucesión de números que es una muy buena aproximación a un conjunto aleatorio de números. Se denominan pseudoaleatorios ya que la sucesión no es exactamente aleatoria en el sentido de que queda completamente determinada por un conjunto relativamente pequeño de valores iniciales llamados "semillas".Para evaluar su aleatoriedad resulta conveniente hacer pruebas estadísticas para estudiar la calidad de estos generadores.
En este trabajo estudiamos los Generadores Congruenciales Lineales y el Generador de Cuadrados Medios, utilizando para esto las pruebas de aleatoriedad llamadas "Prueba de bondad de ajuste χ^2 ", "Prueba de corridas", "Prueba de corridas por arriba y abajo de la media", y la "Prueba de paridad".

2 Metodología

El objetivo es generar sucesiones $U_i \, {}_{i=1}^{n}$ de números independientes que se puedan considerar como observaciones de una distribución uniforme en el intervalo (0,1)

2.1 Verdaderos números aleatorios

Los números completamente aleatorios (no determinísticos) son fáciles de imaginar conceptualmente, por ejemplo podemos imaginar lanzar una moneda, lanzar un dado, etc. En general los números aleatorios se basan en alguna fuente de aleatoreidad física que puede ser teóricamente impredecible (cuántica) o prácticamente impredecible (caótica). Por ejemplo:

- random.org genera aleatoreidad a través de ruido atmosférico.
- EARNIE usa ruido térmico en transistores.
- RAND Corporation publicó en 1955 una table de un millón de números aleatorios que fue ampliamente utilizada. Los números en la tabla se obtuvieron de una ruleta electrónica.

La desventaja de éstos métodos es que son costosos, tardados y no reproducibles.

2.2 Números pseudoaleatorios

Los números pseudoaleatorios se generan de manera secuencial con un algoritmo determinístico, formalmente se definen por:

- **Función de inicialización:** Recibe un número (la semilla) y pone al generador en su estado inicial.
- **Función de transición:** Transforma el estado del generador
- **Función de salidas:** Transforma el estado para producir un número fijo de bits (0 ó 1).

Una sucesión de bits pseudoaleatorios se obtiene definiendo la semilla y llamando repetidamente la función de transición y la función de salidas
Esto implica, entre otras cosas, que una sucesión de números pseudoaleatorios esta completamente determinada por la semilla
Buscamos que una secuencia de números pseudoaleatorios:

- no muestre ningún patrón o regularidad aparente desde un pusto de vista estadístico, y
- dada una semilla inicial, se puedan generar muchos valores antes de repetir el ciclo.

Construir un buen algoritmo de números pseudoaleatorios es complicado.
Una propiedad deseable es que la sucesión de ui parezca una sucesión de observaciones independientes de una Uniforme(0,1).

2.3 Generadores Congruenciales Lineales

Los Generadores Congruenciales Lineales (GCL) tienen la forma

$$X_n - 1 = ({}_aX_n + c) mod(n) \tag{1}$$

Están determinados por los parámetros:

- Módulo: $m > 0$
- Multiplicador: $0 \leq a \leq m$
- Incremento: $c \leq m$
- Semilla: $0 \leq X_0 < m$

Los GCL se introdujeron en 1949 por D.H. Lehemer y son muy populares. La elección de los parámetros determina la calidad del generador: 1. Queremos m grande pues el periodo (longitud del ciclo) del generador no puede tener más de m elementos.
2. Queremos velocidad, en este caso, un valor conveniente para m es el tamaño de palabra (máximo número de bits que puede procesar el CPU en un solo ciclo) de la computadora. Los GCL más eficientes tienen un m igual a una potencia de 2 (es usual 2^{32} o 2^{64}) de esta manera la operación módulo se calcula truncando todos los dígitos excepto los últimos 32 ó 64 bits.
3. ¿Podemos elegir a y c de tal manera que logremos alcanzar el periodo máximo (m)? Un generador congruencial mixto (c > 0) tendrá periodo completo para todas las semillas sí y sólo sí:

- 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.

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 tienden a exhibir defectos, por ejemplo, si se utiliza un GCL para elegir puntos en un espacio de dimensión k los puntos van a caer en a lo más $(k!m)^{\frac{1}{k}}$ hiperplanos paralelos k dimensionales.
Los GCLs continuan siendo utilizados en muchas aplicaciones porque con una elección cuidadosa de los parámetros pueden pasar muchas pruebas de aletoriedad, son rápidos y requieren poca memoria

2.4 Generador Itamaracà

Itamaracà o simplemente "Ita" es una base matematica en Generador de números pseudoaleatorios que genera "infinitas" secuencias de numeros en un rango [0, 1] llevando a cabo una distribucion uniforme.
Como todo Generador de numeros pseudoaleatorios, Ita tiene algunas caracteristicas distintivas. A continuacion se presentan sus condiciones iniciales:

- En primer lugar, elegirN, es decir, un valor maximo dentro de un rango entre 0 y N seleccionado por un criterio elegido por el usuario, donde N pertenece a los numeros naturales
- En este modelo, hay 3 semillas, S_0, S_1 y S_2. Para cada una de estas semillas elija un numero cualquiera pertenenciente a los numeros naturales, que pertenezca al intervalo entre 0 y N.

Tras elegir arbitrariamente los 3 valores de las semillas, el proceso de calculo se divide en dos etapas principales:

- Pn (Proceso de n o Estado Intermedio)
- Calculo Final o Formula General

Pn (Proceso de n) o Estado Intermedio
En esta etapa tenemos que tener en cuenta los valores absolutos que tienen las diferncias entre las 2 semillas que son "moviles" en el tiempo, preferiblemente diciendo, en secuencia.

$$Pn = ABS(S_2 - S_0) \tag{2}$$

Cálculo Final o Formula General
En este paso, debemos multiplicar "x" resultado obtenido en el primer paso (en Pn) por Xrn, es decir, cualquier valor deseado por el usuario, siempre que este valor sea muy cercano a 2 (ej: 1,97 - 1,98 - 1,99789...)

$$FRNS = ABS[N - (Pn * Xrn)] \tag{3}$$

2.5 Pruebas de aleatoriedad

Una forma de evaluar la aleatoriedad de números aleatorios es graficando las secuencias de los mismos. Sin embargo, el ojo humano no es muy bueno discriminando aleatoriedad y las gráficas no escalan. Es por ello que resulta conveniente hacer pruebas estadísticas para evaluar la calidad de los generadores de números pseudoaleatorios.

Hay dos tipos de pruebas:

- **empíricas**: evalúan estadísticas de sucesiones de números.
- **teóricas**: se establecen las características de las sucesiones usando métodos de teoría de números con base en la regla de recurrencia que generó la sucesión.

2.6 Pruebas de aletoriedad a realizar dentro de cada caso de estudio

2.6.1 Prueba de bondad de ajuste X^2

La prueba de bondad de ajuste X^2 es una prueba de hipótesis estadística que se usa para averiguar si es probable que una variable provenga o no de una distribución específica. En otras palabras, nos dice si la muestra disponible representa (o ajusta) razonablemente los datos que uno esperaría encontrar en la población.

Hipotesis:

H0 Los datos son muestra de la distribución uniforme.
H1 Los datos no son muestra de la distribución uniforme.

Prueba:

1. Partir la distribución en n celdas que son exhautivas y mutuamente excluyentes.
2. Contar el número de observaciones Oi encontrados en cada celda.
3. Calcular el valor esperado en cada celda

$$e_i = np_i \tag{4}$$

4. Calcular la sumatoria de las estadísticas de prueba

$$X^2 = \sum_{i=1}^{n} \frac{(oi - ei)^2}{ei} \tag{5}$$

5. Si $\chi^2 < \chi^2_{p,q}$, se acepta la hipótesis H0
   p : Representa el intervalo de confianza. q : Representa los grados de libertad.

2.6.2 Prueba 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.

Hipótesis:

H0: Hipótesis nula (criterio de aceptación)

$$|Z| \leq Z_{1-a/2} \tag{6}$$

La secuencia de números es independiente y por lo tanto la secuencia es aleatoria.

H1: Hipótesis alternativa (criterio de rechazo)

$$|Z| > Z_{1-a/2} \tag{7}$$

La secuencia de números no es independiente y por lo tanto la secuencia no es aleatoria.

$$Z = \frac{(a1 - \mu_a)}{\sigma_a} \tag{8}$$

La media υ_a y la varianza σ_a están dadas por:

$$\mu_a = \frac{(2N - 1)}{3} \tag{9}$$

$$\sigma_a^2 = \frac{(16N - 29)}{90} \tag{10}$$

Prueba:

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. 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.
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, por lo que no es posible compararlo con otro número.

Una vez encontrado los signos de cada número de la secuencia dada se procede a calcular el total de corridas que resulta de la suma de la corrida ascendente con la descendente. Una corrida se define como una sucesión de eventos similares, precedidos y seguidos por un evento diferente.

2.6.3 Prueba de corridas media

Este método, también conocido como prueba de corridas por arriba y abajo de la media, consiste en lo siguiente:

- 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.

Sean n_1 y n_2 el número de observaciones individuales por encima y por debajo de la media, respectivamente y sea b el número total de corridas. Note que el número máximo de corridas es N = n_1 + n_2, y el número mínimo de corridas es 1. Dados n_1 y n_2, la media de b (con una corrección de continuidad sugerida por Swed and Eisenhart [1943]) y la varianza de b para una secuencia verdaderamente independiente están dadas por:

$$\mu_b = \frac{2n_1n_2}{n_1+n_2} + 1 \tag{11}$$

$$\sigma_b^2 = \frac{(2n_1n_2(2n_1n_2 - N))}{N^2(N-1)} \tag{12}$$

$$Z = \frac{(b-\mu_b)}{\sigma_b} \tag{13}$$

para n_1 = 0 y n_2 mayor que 20, b se aproxima a una distribución normal y, para este caso, la estadística de la prueba puede ser realizada restando a la media el número de corridas y dividiéndola por la desviación estándar.
Puesto que nos interesa la presencia de un número demasiado grande o demasiado pequeño de corridas, conviene nuevamente aplicar una prueba de dos colas. Si se especifica el nivel de significancia por medio de α, rechazaremos la hipótesis de aleatoriedad sí:

$$|Z| \geq Z_1 - \frac{(\alpha)}{2} \tag{14}$$

2.6.4 Prueba de Paridad

La prueba de paridad se basa en el supuesto de que, al generar n cantidad de numeros pseudialatorios con una distribucion uniforme, deberia suceder que cerca del 50 % de ellos pertenecen a los pares y el otro 50 % restante pertenecen a los impares. Para complementar el estudio, se incluirá el generador de la biblioteca científica de Python (NumPy) para comparar ambos resultados. Para llevar adelante la prueba, se analizaran los porcentajes de paridad mediate un gráfico pastel con sus respectivos porcentajes.

3 Casos de estudio

3.1 Primer caso: Generador Congruencial Lineal

El fin de realizar las pruebas es determinar si nuestro Generador Congruencial Lineal puede ser aceptado como un generador de números aleatorios. Éste debe cumplir con dos propiedades, por lo que las siguientes pruebas serán para determinar la uniformidad de los datos y su aletoriedad. Para la sucesión del GCL se generaron 1000 números y se utilizaron los siguientes parámetros:

- $Semilla$: 2025
- a: 25214903917
- c: 11
- m: 2^{48}

3.1.1 Prueba de bondad de ajuste χ^2

Resultado: Se obtuvieron los siguientes valores:

Intervalo	$\frac{(oi-ei)^2}{ei}$
1	0.16
2	0.01
3	0.04
4	1.96
5	1.0
6	0.16
7	0.01
8	0.04
9	1.96
10	1.0
Total	$\sum_{i=1}^{n} \frac{(oi-ei)^2}{ei} = 6,34$

Para un intervalo de confianza del 95 % y 9 grados de libertad, se obtuvo el valor crítico χ^2 = 16.91897 Dado que

$$6,34 < 16.91897 \tag{15}$$

podemos aceptar la hipótesis H_0, por lo tanto, los datos son muestra de la distribución uniforme.

3.1.2 Prueba de corridas

Resultado: Se realizo la prueba para los numeros generados por el GCL y la secuencia de '+', '-' obtenida nos dio una cantidad de cambios a igual a 676.
Valores obtenidos:

- $\mu_a = 666.33$
- $\sigma_a = 177.45$
- Z = 0.72565
- $Z_{1-a/2} = 1,95996$

Por lo tanto como |Z| < $Z_{1-a/2}$, no rechazamos la hipótesis de aleatoriedad.

3.1.3 Prueba de corridas media

Resultado: Se realizó la prueba para los números generados por el GCL y la cantidad de números por debajo de la media fueron 513 y por arriba 487. Valores obtenidos:

- $\mu_a = 500.662$
- $\sigma_a = 249.41186$
- Z = 0.69899
- $Z_{1-a/2} = 1,95996$

Por lo tanto como |Z| < $Z_{1a/2}$, no rechazamos la hipótesis de aleatoriedad.

3.1.4 Prueba de Paridad

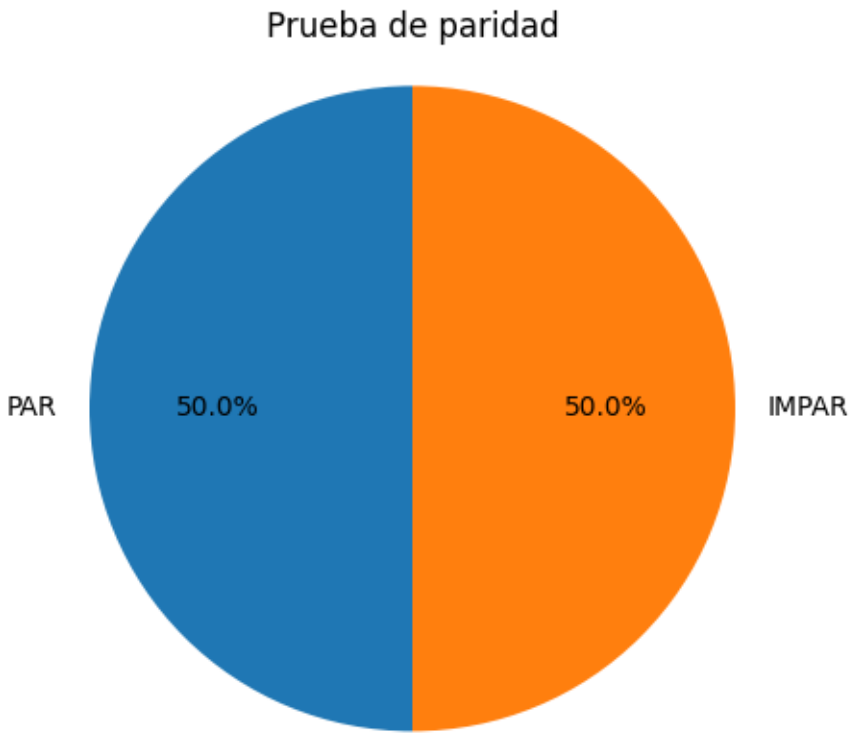

Figure 1: Paridad

Se realizo la prueba para los numeros generados por GCL y la cantidad de numeros pares es la misma que la de los impares por lo tanto se puede concluir que cumple con la hipotesis y se puede afirmar la aleatoriedad de los numeros generados.

3.1.5 Comparación con el generador de Python

A continuación, graficamos los diferentes valores obtenidos en una corrida del experimento en el Generador Congruencial Lineal y en el generador de Python. En la Figura 2 que grafica los valores obtenidos por el GCL, podemos observar claramente un patrón de puntos que dibujan una línea diagonal, mientras que en la Figura 3 no se observa ningún patrón a simple vista (que es lo que debería pasar para que no se pueda descifrar el siguiente número a salir).

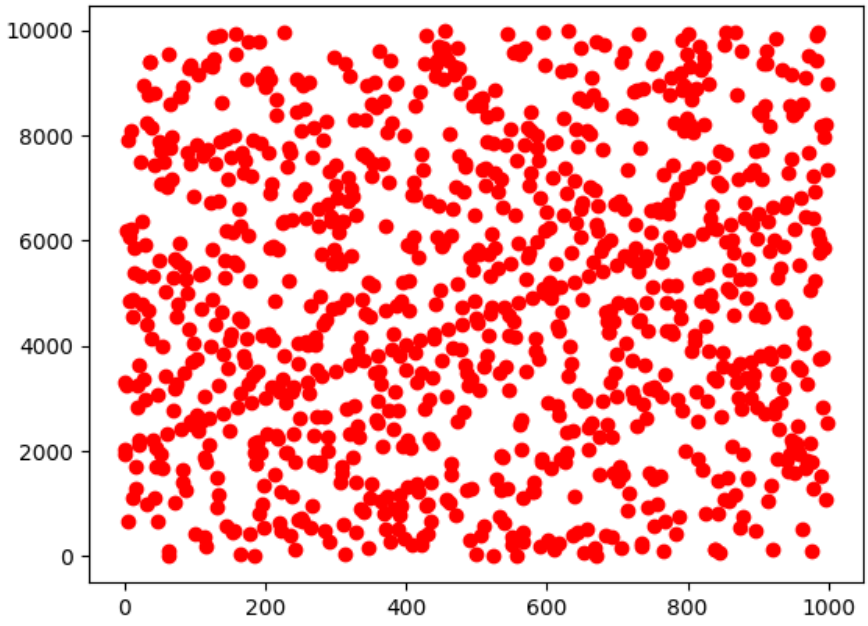

Figure 2: Grafico de números generados por un GCLo

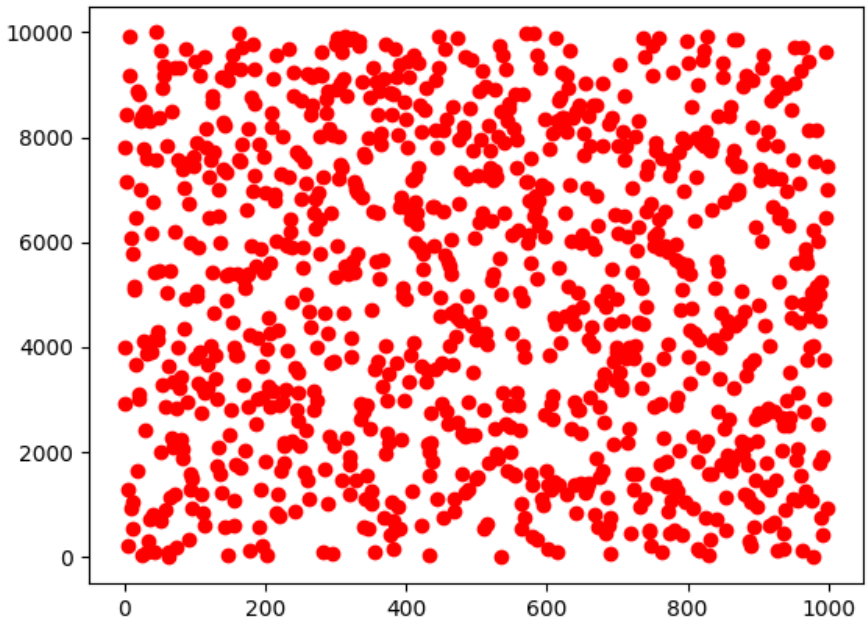

Figure 3: Grafico de números generados por el generador de Python

3.2 Segundo caso de estudio: Generador media de los cuadrados

3.2.1 Prueba de bondad de ajuste χ^2

Resultado:

Se obtuvieron los siguientes valores:

Intervalo	$\frac{(oi-ei)^2}{ei}$
1	16.0
2	19.36
3	96.04
4	96.04
5	18.49
6	17.64
7	100.0
8	338.56
9	16.0
10	96.04
Total	$\sum_{i=1}^{n} \frac{(oi-ei)^2}{ei} = 814.98398$

Para un intervalo de confianza del 95 % y 9 grados de libertad, se obtuvo el valor crítico χ^2 = 16.91897 Dado que

$$814.98398 > 16.91897 \tag{16}$$

no podemos aceptar la hipótesis H_0, por lo tanto, los datos no son muestra de la distribución uniforme.

3.2.2 Prueba de corridas

Resultado: Se realizó la prueba para los números generados por el generador media de cuadrados y la secuencia de '+', '-' obtenida nos dió una cantidad de cambios a igual a 502. Valores obtenidos:

- $\mu_a = 666.33$
- $\sigma_a = 177.45$
- Z = -6,85621
- $Z_{1-a/2} = 1,95996$

Por lo tanto como |Z| > Z1a/2, podemos concluir que la secuencia de números no es independiente y por lo tanto la secuencia no es aleatoria.

3.2.3 Prueba de corridas media

Resultado: Se realizó la prueba para los números generados por el itamaraca y la cantidad de números por debajo de la media fueron 568 y por arriba 432.
Tambien pudimos observar un numero de corridas igual a 574. Valores obtenidos:

- $\mu_a = 491.752$
- $\sigma_a = 240.58736$
- Z = 0.34186
- $Z_{1-a/2} = 1,95996$

Por lo tanto como |Z| < Z1a/2, no rechazamos la hipótesis de aleatoriedad.

3.2.4 Prueba de Paridad

Se realizo la prueba para los numeros generados por el Generador itamaraca para 1000 numeros, esta prueba genero 568 numeros pares y 432 impares.

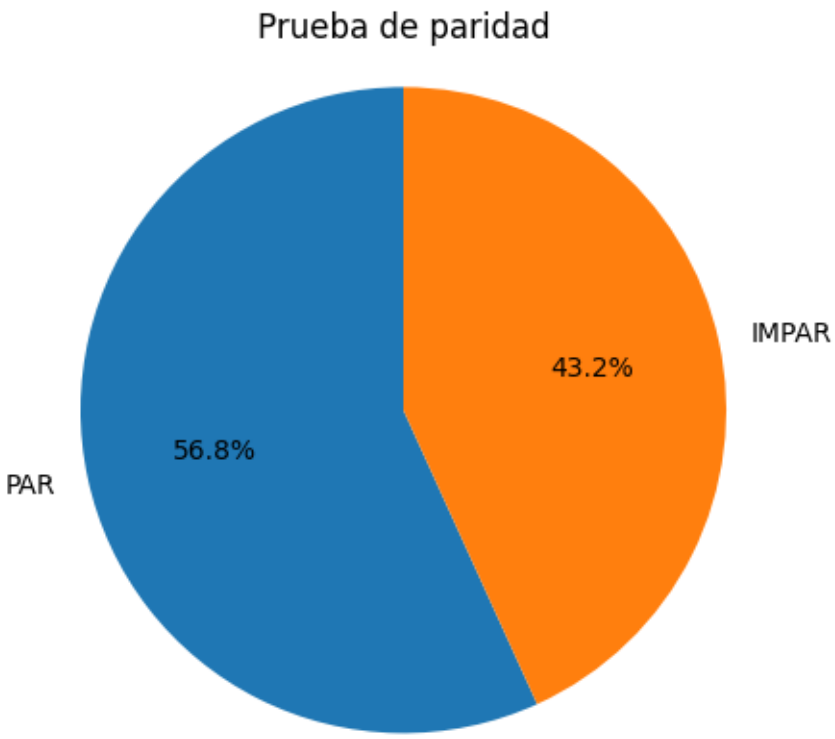

Figure 4: Paridad

Por lo tanto como no cumple con la hipotesis rechazamos la aleatoriedad de los numeros generados.

3.2.5 Comparación con el generador de Python

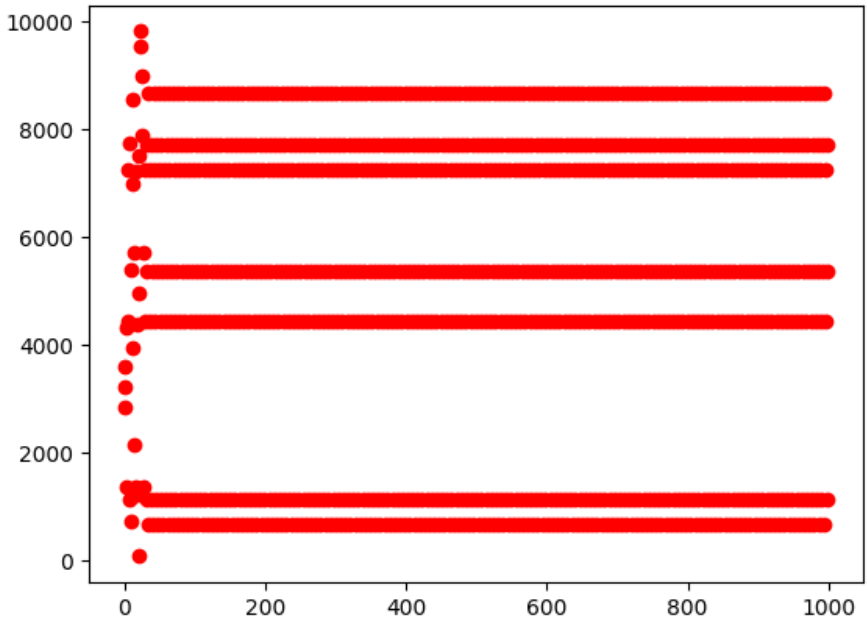

Figure 5: Grafico de números generados por un Itamaraca

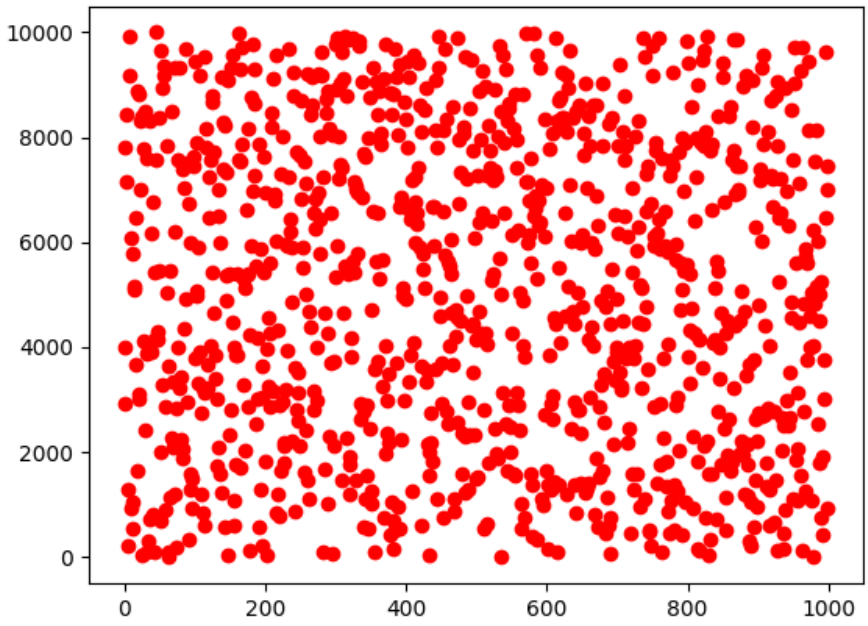

Figure 6: Grafico de números generados por el generador de Python

A continuación, graficamos los diferentes valores obtenidos en una corrida del experimento en el generador media de los cuadrados y en el generador de Python. En la Figura 5 que grafica los valores obtenidos por el generador Itamaraca, podemos observar claramente que , mientras que en la Figura 6 no se observa ningún patrón a simple vista (que es lo que debería pasar para que no se pueda descifrar el siguiente número a salir).

4 Conclusion

Tras realizar severas pruebas a nuestros generadores, a su vez que los fuimos comparando con un generador aceptado como lo es el de Python, podemos decir que el único generador que genera números bastante cercanos a los aleatorios es el Generador Congruencial Lineal pero el defecto que le encontramos es que al momento de analizar el mapa de puntos se puede ver un patrón de una línea diagonal (figura 2). Analizando el otro generador, podemos ver que tiene el defecto que una vez que se genera el número cero, los próximos números generados hasta terminar la repetición seran determinador valores constantes (figura 5), por lo tanto no es confiable.

5 Referencias

[1]Numeros pseudoaleatorios, GCL y pruebas: https://tereom.github.io/est-computacional-2018/ numeros-pseudoaleatorios.html/pruebas-de-aleatoriedad Generador Itamaraca: https://es.slideshare.net/DHPereira1/itamarac-nueva-manera-sencilla-de-generar-nmeros-pseudoaleatorios Prueba de las corridas: https://hemaruce.angelfire.com/Unidad$_III.pdfElcdigoutilizadopararealizarlainvestigacinseencuentraen$:

6 Referencias

1. La documentacion sobre generador GCL:
   `https://tereom.github.io/est-computacional-2018/numeros-pseudoaleatorios.html`
2. La documentacion sobre Generador Itamaracà:
   `https://acortar.link/ANox6g`
   `https://acortar.link/gDi1NA`
3. Prueba de las corridas:
   `https://hemaruce.angelfire.com/Unidad_III.pdf`
4. GitHub :
   `https://github.com/AlexisTomas2000/TP-Simulacion/tree/main/Tp%202.1`