Generadores de números aleatorios

Delmonti Agustín
Universidad Tecnológica Nacional Rosario
Zeballos 1341, S2000
agus.delmonti@gmail.com

Trilla Melody
Universidad Tecnológica Nacional Rosario
Zeballos 1341, S2000
trillamelo@gmail.com

Abril 2020

Resumen

1. Introducción

Los números aleatorios son útiles para una variedad de propósitos, como encriptar información, determinar las cartas a dar en una mano en un juego de solitario, o simular variables desconocidas de sistemas reales complejos como por ejemplo los mercados de valores, el pronóstico del clima, la programación de vuelos en las aerolíneas o la selección de muestras representativas de pacientes al probar nuevos medicamentos. Sin embargo, se establece el interrogante de cómo se pueden generar estos números aleatorios en una computadora con naturaleza determinista, después de todo una computadora sigue sus instrucciones a ciegas y, por lo tanto, es completamente predecible.

Existen dos enfoques principales para generar números aleatorios usando una computadora los cuales se investigarán en el presente trabajo: los **generadores de números pseudoaleatorios** o **pseudo-random number generators (PRNG)** y los **generadores de números aleatorios verdaderos** o **true random number generators (TRNG)**. Los enfoques tienen características bastante diferentes y cada uno tiene sus ventajas y sus desventajas.

2. Marco teórico.

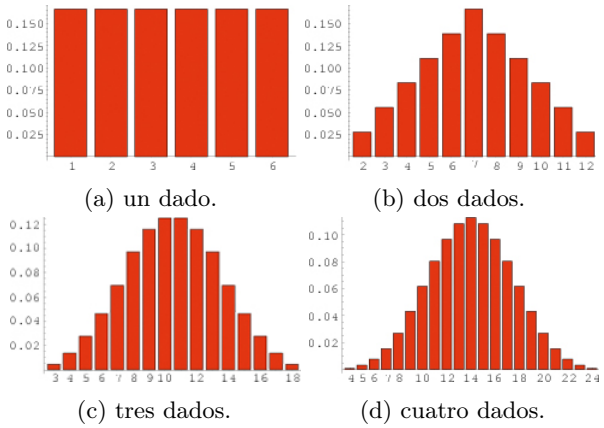
2.1. Aleatoriedad.

Cuando se habla de aleatoriedad naturalmente surgen preguntas: qué es, qué significa que algo sea aleatorio o si verdaderamente puede existir algo aleatorio. Esas son preguntas con importancia práctica, ya que la aleatoriedad es sorprendentemente útil en muchos campos de estudio, entre ellos la simulación de eventos, variables y sistemas.

Siguiendo ninguna ley, los números aleatorios **carecen de previsibilidad**. El resultado de todo suceso aleatorio no puede determinarse en ningún caso antes de que este se produzca.[1]

Una secuencia aleatoria de eventos, símbolos o pasos a menudo no tiene orden y no sigue un patrón o combinación inteligible. Los eventos aleatorios individuales son, por definición, impredecibles, pero la frecuencia de diferentes resultados en numerosos eventos o ensayos es predecible dado que a menudo siguen una distribución de probabilidad. Por ejemplo, cuando se lanzan dos dados, el resultado de cualquier tirada en particular es impredecible: cada dado puede tomar un valor $x \in [1; 6]$ con una probabilidad $p = \frac{1}{6}$ pero la suma de 7 ocurrirá dos veces más que 4 como se puede ver en la Figura 1.

En cierto sentido, no existe un número aleatorio por sí solo. No tiene sentido preguntarse si 2 es un número aleatorio. Más bien hablamos de una secuencia de números aleatorios **independientes** con una



(a) un dado. (b) dos dados.

(c) tres dados. (d) cuatro dados.

Figura 1: suma de tiradas de a n dados, n=1,2,3,4

distribución específica.[2] De estas observaciones se desprende que esta distribución debe ser **uniforme**. Esto es lógico, pues si un valor apareciera más veces (por ejemplo la mitad de las tiradas el dado cae en 6) la distribución estaría sesgada y no sería verdaderamente aleatoria (se espera que caiga en 6 la mitad de las veces).

2.2. Generadores reales.

Los generadores de números aleatorios reales o **TRNG** extraen o heredan la aleatoriedad a partir de fenómenos físicos del mundo externo y la introducen en la computadora.[3]

Un fenómeno físico realmente bueno para usar es la radiactividad. Los puntos en el tiempo en que se desintegra una fuente radiactiva son completamente impredecibles, y pueden detectarse fácilmente y alimentarse a una computadora. La radiación del ambiente se puede tabular en un periodo corto de tiempo con un contador Geiger y obtiene una secuencia aleatoria de valores. Otro fenómeno físico adecuado es el ruido atmosférico, que es bastante fácil de detectar con una radio normal.

Las desventajas de obtener números aleatorios con un método físico es que se necesita un dispositivo externo y no es un proceso rápido. Además los TRNG son **no deterministas**, lo que significa que una secuencia dada de números no puede reproducirse, propiedad en muchos casos necesaria para testear software o reproducir experimentos.

2.3. Generadores pseudoaleatorios.

En la mayoría de los casos no necesitamos producir números verdaderamente aleatorios y en tales casos se puede acudir a los generadores de números pseudoaleatorios o **PRNG**. Estos son más eficientes, lo que significa que pueden producir muchos números en poco tiempo en comparación con los TRNG.

Un generador de números aleatorios consiste en una función que devuelve los valores de una secuencia de números reales $\{x_i\}_{i=1}^{N}$, donde cada $x_i \in [0;1]$, los cuales se comportan como números aleatorios aunque no lo sean (de ahí el prefijo *pseudo*). Se aceptan entonces como sustitutos de los números aleatorios para determinadas aplicaciones porque su comportamiento es similar, es decir porque tienen muchas de las propiedades estadísticas más importantes esperadas en los números aleatorios puros: **impredecibilidad, uniformidad e independencia**.

El algoritmo que genera la secuencia de números se denomina **generador** se recursivo y puede ser asociado a la relación de recurrencia H de grado k en la Ecuación 1.

$$x_i = H(x_{i-1}, x_{i-2}, ..., x_{i-k}) \tag{1}$$

Todos los valores de la secuencia $\{x_1, x_2, x_3, ...\}$ son generados por H a partir de un valor inicial x_0 denominado *semilla* o *seed*. A partir de esta definición se puede deducir entonces que:

- los números de la secuencia **no son independientes entre sí** por el simple hecho de responder a una relación de recurrencia matemática.
- los PRNG son **deterministas** (siguen alguna ley), es decir, el generador generará la misma secuencia de números si se introduce la misma semilla al inicio.

El determinismo puede resultar útil si se necesita volver a reproducir la misma secuencia de números en una etapa posterior. La no independencia, por el otro

lado, contradice a la naturaleza de la aleatoriedad y no es beneficioso pues puede crear patrones no deseables en la secuencia. Por ejemplo algo que haría que una secuencia de números parezca menos aleatoria sería que el mismo patrón de números se repita con un periodo chico o que haya una clara monotonía de crecimiento o decrecimiento en los números de la secuencia. Se puede buscar entonces un comportamiento similar a la independencia (*pseudo-independencia*) haciendo que los elementos de la secuencia tengan mínima correlación.

2.3.1. Método de la parte media del cuadrado.

Un método simple de PRNG para generar secuencias de dígitos pseudoaleatorios de 4 dígitos es el método de la parte media del cuadrado o middle square de Jon Von Neuman (1946).

1. Se inicia con una semilla de 4 dígitos.
2. La semilla se eleva al cuadrado, produciendo un número de 8 dígitos. Si el resultado tiene menos de 8 dígitos se añaden ceros no significativos al inicio.
3. Los 4 números del centro serán el siguiente número en la secuencia, y se devuelven como resultado.

En la práctica el método de la parte media del cuadrado no es uno muy bueno. Los números generados son de 4 dígitos y este generador cae rápidamente en ciclos cortos de números en el rango [0;9999] dependiendo de la semilla, o si aparece un cero. En la Figura 2 se muestra una simulación del método en la cual se puede observar claramente como empieza a ciclar los números generados una vez que un numero se repite. La función cuadrática no es muy buena para cubrir el rango de números posibles a generar. Como se extraen los dígitos del medio, se encuentra que solo el 60 % de los diferentes valores realmente se alcanzan efectivamente independientemente de la semilla.[4]

2.3.2. Periodicidad.

Como el método de la parte media del cuadrado, todos los PRNG son **periódicos**, lo que significa que

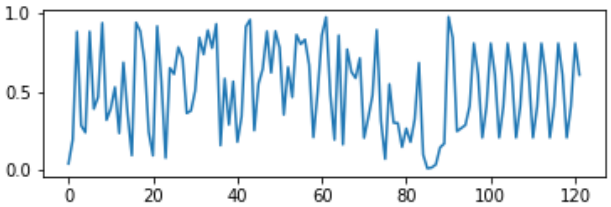


Figura 2: secuencia de números generados por el método de la parte media del cuadrado. Seed = 1022

todas las secuencias de números pseudoaleatorios eventualmente se repiten. Las secuencias son cíclicas y la cantidad de términos en las secuencias antes de que se repita se denomina **periodo**.

Si bien la periodicidad casi nunca es una característica deseable, los PRNG modernos tienen un periodo que es tan largo que puede ignorarse para la mayoría de los propósitos prácticos.[3]. La solución óptima sería hacer el periodo del ciclo lo suficientemente grande (ajustando los parámetros del generador) para que la secuencia se repita mas tarde que temprano.

En teoría, si los números de la secuencia $\{x_i\}_{i=1}^{N}$ son generados sobre un conjunto cíclico $\overline{\mathbb{Z}}_p$ es imposible ir más de p términos de la secuencia sin repetir algún número por el *principio del palomar*. Es decir, m huecos pueden albergar como mucho m objetos si cada uno de los objetos está en un hueco distinto, así que el hecho de añadir otro objeto fuerza a volver a utilizar alguno de los huecos.[5].

2.4. Comparativa.

	TRNG	**PRNG**
Mecanismo	Físico	Matemático
Eficiencia	Pobre	Alta
Independencia	Si	No
Determinismo	No determinista	Determinista
Periodicidad	Aperiódico	Periódico
Distribución	Uniforme	Uniforme

Cuadro 1: comparación de características de los diferentes tipos de generadores de números aleatorios.

Las características de los distintos generadores ha-

cen que los PRNG sean adecuados para aplicaciones donde se requieren muchos números y donde es útil que la misma secuencia se pueda reproducir fácilmente. Ejemplos populares de tales aplicaciones son las aplicaciones de simulación y modelado. Los PRNG no son adecuados para aplicaciones en las que es importante que los números sean realmente impredecibles, como el cifrado de datos y los juegos de azar y en tales casos sería recomendable utilizar algún TRNG.

3. Generadores lineales congruenciales.

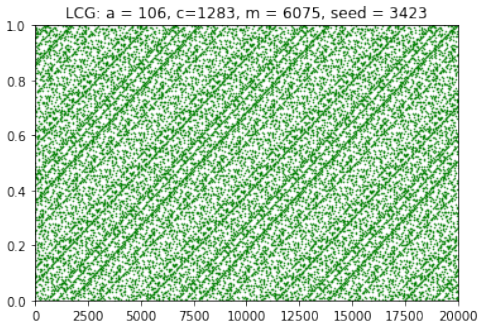
Los generadores lineales congruenciales (GLC) son los generadores de números pseudoaleatorios mas utilizados en simulación. Utilizan una función lineal modular recursiva H para generar la sucesión $\{x_i\}_{i=1}^{N}$ según la ecuación:

$$x_{n+1} = (ax_n + c) \text{ mód } m \tag{2}$$

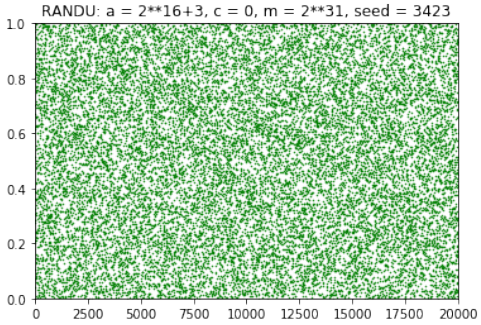
donde a, m y c son enteros positivos que se denominan respectivamente, **multiplicador**, **módulo** e **incremento**. Luego se pueden calcular $\{u_i \in [0;1] \mid u_i = x_i/m \ , \ i = 1, 2, ... \}$ para obtener valores en independientes de m.

Si $c = 0$, el generador es llamado generador **congruencial multiplicativo** (GCM) o generador de números pseudoaleatorios de Lehmer. En caso contrario, el método es llamado generador **congruencial mixto**.

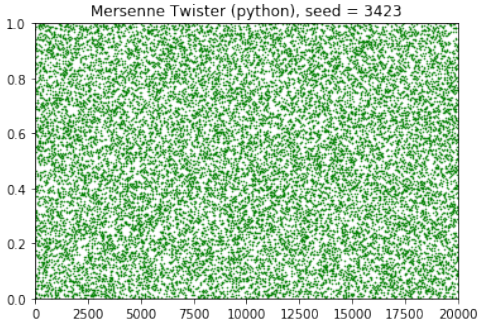
En la Figura 3 se puede ver como las secuencias generadas por 3 generadores diferentes se distribuyen contra el índice de la simulación. En el caso de la Figura (3a), el periodo del generador es muy chico y por lo tanto empieza rápidamente su ciclo. Además existe un patrón notable de crecimiento monótono por lo que no es un buen generador. Los otros dos generadores, (3b) RANDU y (3c) Mersenne-Twister no presentan ningún tipo de anomalía y parecen ser bastante aleatorios.



(a) LGC



(b) RANDU



(c) Mersenne-Twister

Figura 3: gráfica del índice de simulación contra el valor obtenido.

3.1. Calidad del generador.

La calidad del generador dependerá exclusivamente de la selección de las variables a, c y m. Recordemos que la calidad se mide en función de la independencia y uniformidad de los valores de las secuencias

generadas. Para verificar si un generador tiene las propiedades estadísticas deseadas hay disponibles una gran cantidad de test de hipótesis y métodos gráficos empíricos, incluyendo métodos genéricos (de bondad de ajuste y aleatoriedad) y contrastes específicos para generadores aleatorios.

Se trata principalmente de contrastar si las muestras generadas son i.i.d. $U(0;1)$ (análisis univariante), no muestre ningún patrón o regularidad aparente desde un punto de vista estadístico y que dada una semilla inicial, se puedan generar muchos valores antes de repetir el ciclo, con el fin de permitir simulaciones largas y/o con muchas variables aleatorias.

3.2. Teorema de Hull-Dobell

El teorema de Hull-Dobell dice que un generador congruencial mixto tendrá un período completo[6] para todas las semillas si y sólo si:

1. m y el incremento c son primos relativos entre sí.
2. $a-1$ es divisible entre todos los factores primos de m.
3. $a-1$ es divisible entre 4 si m es divisible entre 4.

3.3. Test χ^2

Para evaluar la uniformidad de los números generados por los GLC se realiza una **prueba de bondad de ajuste** o χ^2 de Pearson.[7]

La hipótesis nula H_0 es que una muestra de números generados por el PRNG se distribuye uniformemente en [0;1]. Se establece un grado de significancia de la prueba $\alpha = 0{,}95$. Se dividen las muestras en n intervalos o clases igualmente espaciadas O_i y se calcula χ^2 con $n-1$ grados de libertad (gl) de la siguiente manera:

$$\chi^2 = \sum_{i=1}^{n} \frac{(O_i - E_i)^2}{E_i} \tag{3}$$

donde O_i e E_i son respectivamente la cantidad de valores observados y esperados en el intervalo i. En el caso de los PRNG, como se espera un comportamiento uniforme, $E_i = \frac{N}{n}$.

Dependiendo si el estadístico χ^2 excede el valor critico se puede entonces rechazar o aceptar la hipótesis nula. Si el estadístico de prueba excede el valor crítico de χ^2 , la hipótesis nula (H_0 =no hay diferencia entre las distribuciones) puede rechazarse, y la hipótesis alternativa (H_1 =hay una diferencia entre las distribuciones) puede ser aceptado, ambos con el nivel de confianza α seleccionado. Si el estadístico de prueba cae por debajo del valor de umbral χ^2, entonces no se puede llegar a una conclusión clara, y la hipótesis nula se mantiene (no pudimos rechazar la hipótesis nula), pero no necesariamente se acepta.

Se simularon cuatro generadores GCL y los resultados de las pruebas de se muestran en la Tabla 2.

3.4. Test hiper espectral.

La condición de uniformidad debe cumplirse además para todas las subsecuencias de tamaño k de la secuencia generada. Es decir, sean las d-uplas:

$$(U_{t+1}, U_{t+2}, \ldots, U_{t+d});\ t = (i-1)d,\ i = 1, \ldots, m \tag{4}$$

estas deben ser i.i.d. $\mathcal{U}(0,1)^d$, es decir uniformes independientes en el hipercubo $[0;1]^d$ (análisis multivariante).

En la Figura 5 se muestra el test espectral en el plano (2-upla) de muestras de distintos generadores. Se pueden entonces observar si existen patrones de dependencia o correlación entre los números de la secuencia.

Si ahora realizamos un test espectral en el cubo (3-upla) del generador RANDU, podemos observar en la Figura 4 como se forman patrones en otros hiperplanos del espacio: existe una alta correlación en determinadas combinaciones secuencias generadas por ese GCL, razón suficiente para descartarlo. Por otro lado el GCL Mersenne-Twister genera números uniformemente distribuidos en el cubo.

Asegurar la k-uniformidad de las secuencias es crucial para los experimentos de simulación. Con ella se evitan correlaciones entre distintas variables que compartan una misma secuencia. Por ejemplo, si en un simulador utilizamos la misma secuencia de números aleatorios para generar los tiempos entre llegadas ($Tllegadas$) y los tiempos de servicio ($Tservicio$) debe-

GCL	m	a	c	χ^2	H_0
GCL1	2**31 - 1	7**5	3	4.368	ACEPTA
GCL2	2**9-1	7**5	9	251.250	RECHAZA
java.util.Random	2**48	25214903917	11	5.354	ACEPTA
RANDU	2**31	2**16 + 3	0	3.652	ACEPTA

Cuadro 2: test chi-cuadrado en distintos GCL. $\alpha = 0{,}95$ $gl = 9$ $valor\_critico = 16{,}918$

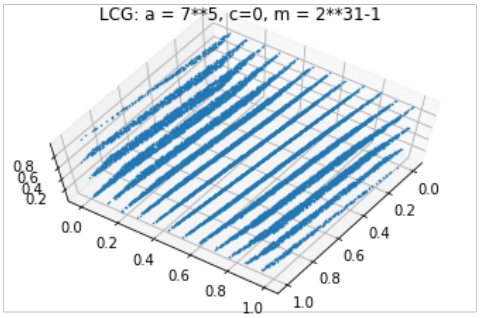


Figura 4: RANDU test de hipercubo. Se observa una dependencia entre las distintos hiperplanos.

mos asegurarnos que los pares consecutivos de valores son independientes, es decir cumplen el criterio de 2-uniformidad. Si esto no se cumple, estaremos introduciendo una correlación no deseada entre ambas variables.

4. Conclusión

Al llevar a cabo un experimento de simulación es necesario asegurar desde un inicio que las distintas variables aleatorias generadas sean uniformes e independientes para no generar efectos secundarios o arrastrar errores y terminar llevando a cabo experimentos sesgados o erróneos. En este contexto es importante la elección de un generador de números pseudoaleatorio rápido y de calidad, el cual pase las diferentes pruebas estadísticas de aleatoriedad. En este trabajo, se presentaron muchos de estos generadores de los cuales, los que mejor se desempeña son los lineales congruenciales. Además se vio como los parámetros del generador pueden influir de forma significativa en las secuencias generadas y porque se deberían elegir cuidadosamente para evitar patrones muy marcados en las secuencias y/o periodos muy cortos. El generador que mejor se desempeña es el Mersenne-Twister que resulta exitoso en la mayoría de los tests estadísticos de aleatoriedad y es el generador defecto de Python y actualmente utilizado por la mayoría de los lenguajes de programación .

Referencias

[1] Wikipedia contributors. Randomness — Wikipedia, the free encyclopedia, 2019. [Online; accessed 11-May-2020].

[2] Donald E. Knuth. *The Art of Computer Programming, Volume 1 (3rd Ed.): Fundamental Algorithms*. Addison Wesley Longman Publishing Co., Inc., USA, 1997.

[3] Mads Haahr. RANDOM.ORG:introduction to randomness and random numbers. `https://www.random.org/randomness/`, 1998.

[4] Wikipedia contributors. Middle-square method — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/w/index.php?title=Middle-square_method&oldid=956138539`, 2020. [Online; accessed 14-May-2020].

[5] Wikipedia contributors. Pigeonhole principle — Wikipedia, the free encyclopedia, 2020. [Online; accessed 10-May-2020].

[6] Wikipedia contributors. Linear-congruential-generator — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/wiki/Linear_`

congruential_generator, 2019. [Online; accessed 14-May-2020].

[7] Wikipedia contributors. Pearson's chi-squared test — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Pearson%27s_chi-squared_test&oldid=956308209, 2020. [Online; accessed 16-May-2020].

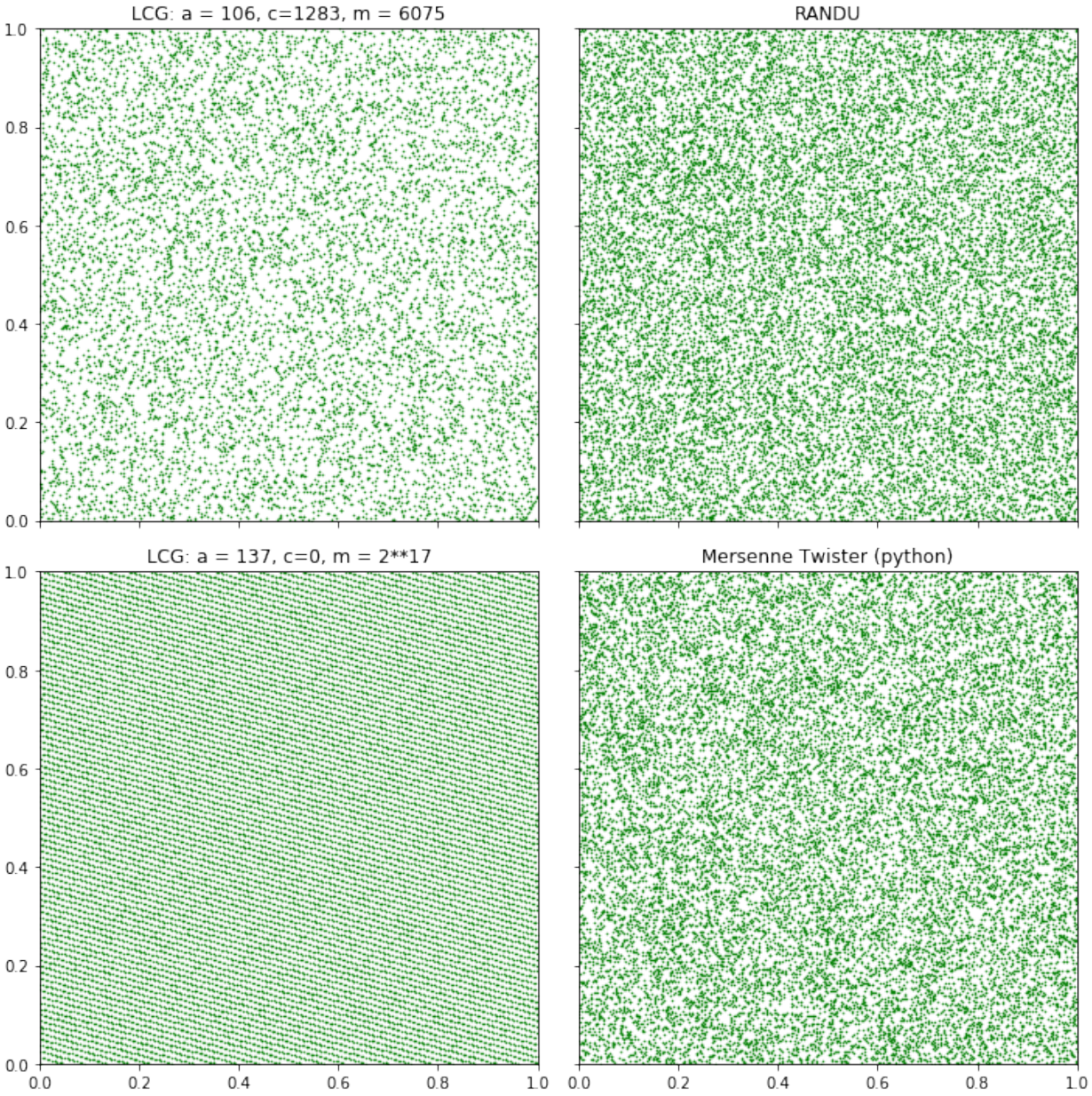


Figura 5: test espectral en el plano de 4 generadores.