Generadores de números pseudoaleatorios

Capítulo 1

Números aleatorios

1.1. Clase 5: Generadores de números pseudoaleatorios

1.1.1. Características de un generador de números aleatorios

En simulación estocástica los generadores de números (pseudo)aleatorios con distribución uniforme en el intervalo $[0, 1]$ son empleados de muchas maneras:

- en forma directa, es decir, porque se desea obtener valores uniformemente distribuidos en $[0, 1]$,
- para generar otras variables aleatorias, con distribuciones discretas o continuas,
- para generar conjuntos de variables aleatorias dependientes, como por ejemplo procesos estocásticos y distribuciones multivariadas.

La ejecución de una simulación está fuertemente correlacionada con el generador de v.a. uniformes utilizado, por lo cual es importante garantizar las buenas propiedades del generador. Dado que los números son generados por una sucesión de pasos, sería más preciso referirse a ellos como generadores de **números pseudoaleatorios**.

Definición 1.1.1. Una N-upla de variables aleatorias $(U_1, U_2, \ldots, U_N)$ es una muestra de tamaño N de una variable aleatoria uniforme en $(0, 1)$ si cumple:

1. Para cada $k \geq 2$ y cada $1 \leq i_1 < \ldots < i_k \leq N$ vale que $\mathbb{P}(U_{i_1} \leq u_1, \ldots, U_{i_k} \leq u_k) = \mathbb{P}(U_{i_1} \leq u_1) \ldots \mathbb{P}(U_{i_k} \leq u_k) = u_1 \cdot u_2 \cdots u_k$, cualesquiera sean $u_1, \ldots, u_k$, y
2. para cada $i = 1, \ldots, N$ y cada $u \in \mathbb{R}$ vale

$$\mathbb{P}(U_i \leq u) = \begin{cases} 0 & \text{si } u < 0 \\ u & \text{si } 0 \leq u \leq 1 \\ 1 & \text{si } u > 0. \end{cases}.$$

Esta definición significa que no sólo las secuencias de números deben estar uniformemente distribuidas en $(0, 1)$, sino que además los pares de números generados (u_{i_1}, u_{i_2}) deben distribuirse uniformemente en un cuadrado de lado 1, las ternas $(u_{i_1}, u_{i_2}, u_{i_3})$ en un cubo de lado 1, y así siguiendo.

Por generador de números pseudoaleatorios entenderemos un algoritmo capaz de producir secuencias de números:

$$u_1,\ u_2, \ldots,\ u_N,$$

que sean realizaciones de muestras de tamaño N de variables uniformes.

Además de satisfacer las propiedades (2) y (1), un buen generador debería satisfacer, en la mejor medida posible, las dos siguientes propiedades:

1. **repetibilidad** y **portabilidad**, y
2. **velocidad computacional**.

Por **repetibilidad** se entiende que, si en ocasiones repetidas se dan los mismos parámetros, el generador debe producir la misma sucesión. Esta propiedad garantiza que los resultados de una simulación sean confiables.

Portabilidad significa que, sobre las mismas condiciones de definición, la sucesión sea la misma, independientemente del lenguaje computacional usado para implementar el algoritmo de generación, y de la computadora utilizada. Esta propiedad suele ser difícil de alcanzar, pero aún así es un aspecto deseable.

La **velocidad computacional** está estrictamente ligada a la precisión deseada en los resultados finales de la simulación para la cual el generador es utilizado. Cuanto más rápido sea el generador, más resultados serán obtenidos en el mismo tiempo de uso del computador.

De hecho, toda vez que se realiza una simulación, se acepta una solución de compromiso entre los requerimientos expuestos anteriormente, pero siempre los prioritarios debieran ser (2) y (1).

1.1.2. Breve reseña histórica

Los primeros científicos y personas interesadas en obtener secuencias de números aleatorios usaron procedimientos físicos. Experimentos de simulación basados en tiradas de monedas o dados, ruletas, tienen una larga historia, y aún hoy son usados en juegos tales como bingos y en casinos, loterías, etc.

Probablemente uno de los primeros trabajos serios en generación de números aleatorios fue la tabla de Tippett (1927), que consistía de 41.600 dígitos aleatorios. Lamentablemente no satisfacían ni (2) ni (1) de la Definición 1.1.1. La primera máquina fue usada en 1939 por Kendall y Babington-Smith con el fin de producir una tabla de 100.000 dígitos aleatorios y en 1955 la RAND Corporation utilizó extensamente una tabla de 1.000.000 de dígitos aleatorios que fue obtenida a partir de una ruleta electrónica especialmente diseñada. Si bien estos métodos recibieron cierta aceptación, por un lado por satisfacer ciertos tests

y/o porque funcionaban en la práctica, tienen la desventaja de no cumplir la condición de repetibilidad. Por lo tanto sólo pueden ser usados en situaciones donde la repetibilidad no es lo buscado, por ejemplo en loterías; o en ciertas aplicaciones estadísticas, como en muestreos aleatorios de poblaciones no muy grandes.

1.1.3. Principios generales de un buen generador

Al definir un generador, hay ciertos **principios generales** que deben satisfacerse, además de las condiciones que ya hemos marcado:

P1) La secuencia generada debe ser intuitivamente aleatoria.

P2) Esa aleatoriedad debe ser establecida teóricamente o, al menos, debe pasar ciertos tests de aleatoriedad. La aleatoriedad de una secuencia jamás debe ser asumida sin esas verificaciones.

P3) Debe conocerse algo sobre las propiedades teóricas del generador.

El principio **P1** es ciertamente razonable. No debería ser posible poder anticipar cuál es el siguiente número en una secuencia si ésta es aleatoria. El segundo principio (**P2**) se refiere a que deben satisfacerse tests estadísticos en relación a las condiciones (1) y (2) de la Definición 1.1.1. La tercera premisa, (**P3**), garantiza la condición de repetibilidad del generador.

Estos principios generales indican que los generadores ad hoc deben ser evitados. Es decir, no cualquier algoritmo que genere números *arbitrariamente* entre 0 y 1 debe considerarse un buen generador.

Ejemplo 1.1.1. Uno de los primeros trabajos que sugieren un método bien definido de generación de una secuencia determinística intentando imitar una secuencia aleatoria, fue de von Neumann. Este método conocido como "mid square" puede ser escrito de la siguiente manera:

1. Sea $x = 0$ un número entero de 4 dígitos decimales (puede ser que el dígito de más a la izquierda sea 0). Hacer $i = 0$.

2. Calcular ${X_i}^2$. El resultado es un número de 8 dígitos, o si tuviera menos se agregarían tantos ceros a la izquierda como sea necesario. Así ${X_i}^2 = d_8d_7d_6d_5d_4d_3d_2d_1d_0$, con d_j en el conjunto $\{0, 1, \ldots, 9\}$, $0 \leq j \leq 7$.

3. Definir $X_{i+1} = d_5d_4d_5d_3d_2$, esto es, los cuatro dígitos centrales o "middle".

4. Hacer $i = i + 1$ y seguir en 2.

Este método es un ejemplo de lo que llamaríamos un **mal generador de números aleatorios**. Veamos por qué.

Aunque ciertas secuencias obtenidas sean bien aleatorias y se repitan sólo después de un número bien grande de términos, sucede que no se conocen bien las propiedades del generador. En particular, pareciera que las secuencias que se obtienen dependen fuertemente del valor inicial X_0.

Por ejemplo, si tomamos $X_0 = 2100$, la secuencia resulta:

$$2100, \quad 4100 \quad 8100 \quad 4100 \ldots,$$

y para $X_0 = 3792$

$$3792 \qquad 3792 \qquad 3792 \qquad 3792 \qquad \ldots.$$

1.2. Generadores congruenciales

Es claro que con una computadora no es posible generar cualquier número real, en particular si posee infinitas cifras decimales. Los generadores que analizaremos a continuación producen en realidad una secuencia de números enteros

$$y_1, y_2, \ldots, y_N, \qquad y_j \in \{0, 1, \ldots, M-1\},$$

para un cierto entero positivo M "grande", para luego tomar la sucesión de números en $[0, 1)$ como

$$u_1 = \frac{y_1}{M},\ u_2 = \frac{y_2}{M}, \ldots, u_N = \frac{y_N}{M}, \ldots.$$

Nos ocuparemos entonces de generar secuencias de números enteros:

$$y_n = f(y_{n-1}) \mod M,$$

para algún M entero positivo, y comenzando de un valor inicial (semilla) y_0.

1.2.1. Generadores congruenciales lineales

Definición 1.2.1. Sea M un entero positivo, $M \geq 2$. Una sucesión $y_1, y_2, \ldots, y_n, \ldots$ con valores en $\{0, 1, \ldots, M-1\}$ se dice **generada por el generador congruencial lineal con parámetros a, c y M y semilla** y_0 si

$$y_n = (a y_{n-1} + c) \mod M, \qquad n \geq 1,$$

donde a, c e y_0 son enteros del conjunto $\{0, \ldots, M-1\}$.

En la terminología usual a se dice un **multiplicador**, c es el **incremento** y M es el **módulo**. Si $c \neq 0$ el generador se dice **mixto** y si $c = 0$ se dice **multiplicativo**.

Ejemplo 1.2.1. La secuencia

$$0, 1, 6, 15, 12, 13, 2, 11, 8, 9, \ldots$$

fue generada por un generador congruencial lineal. El lector, ¿podría decir cual será el próximo número (entre 0 y 15)?

Ejemplo 1.2.2. La secuencia

$$1, 12, 1, 12, 1, 12, 1, 12, 1, 12, \ldots$$

ha sido generada por otro generador congruencial. ¿Podría decirse que genera una secuencia intuitivamente aleatoria?

Los ejemplos anteriores muestran que no siempre una secuencia generada por un generador congruencial lineal resulta intituitivamente aleatoria. En principio pareciera que cuanto más números distintos aparecen en la secuencia hasta obtener un valor repetido, más impredecible será el próximo número. Aunque esto no es del todo cierto, como veremos un poco más adelante.

Es claro que la secuencia puede tener, a lo sumo, M números diferentes, y que si un número se repite entonces también se repite la secuencia que sigue a ese número. Entonces, si K es el menor número tal que

$$y_n = y_{n+K}, \qquad \text{para todo } n,$$

diremos que K es el **período** de la secuencia $y_0, y_1, \ldots$. Claramente, $K \leq M$.

Pregunta: Ahora, ¿cómo escoger a, c, M y y_0 para obtener las secuencias con mayor período K posible?

Aún consiguiendo una período K grande, es importante elegir a, c y la semilla y_0 de modo que las secuencias tengan un comportamiento lo más aleatorio posible. Por ejemplo, ¿qué ocurre si elegimos M arbitrario, $a = c = 1$ e $y_0 = 0$? Respuesta: obtenemos la secuencia

$$0, 1, 2, 3, \ldots, M - 1,$$

que es de período completo (esto es, $K = M$), pero nadie diría que es aleatoria.

Para el caso de generadores congruenciales lineales existen varios resultados teóricos que permiten conseguir secuencias con períodos grandes. La aleatoriedad debe ser testeada aparte, ninguno de estos resultados garantizan que se satisfagan los tests de aleatoriedad.

Resultados teóricos para generadores congruenciales lineales

Algo a notar es que, si el generador es multiplicativo ($c = 0$), entonces un buen generador no debería alcanzar nunca el valor 0, de lo contrario la secuencia degeneraría en una sucesión infinita de ceros. Por lo tanto, para obtener un período máximo, esto es $K = M$, necesariamente debe ser un generador mixto.

Teorema 1.2.1. Consideremos una secuencia dada por el generador:

$$y_{i+1} = a\, y_i + c \mod M, \qquad c \neq 0.$$

Entonces la secuencia tiene período M si y sólo si se cumplen todas las siguientes condiciones:

- El máximo común divisor entre c y M es 1: $(c, M) = 1$.
- $a \equiv 1 \mod p$, para cualquier factor primo p de M.
- Si 4 divide a M, entonces $a \equiv 1 \mod 4$.

La demostración puede encontrarse en [3]. Damos un ejemplo y sugerimos al lector pensar en otros casos que respondan a este teorema.

Ejemplo 1.2.3. Consideremos $M = 16$, $c = 3$, $a = 5$. Notemos que si M es una potencia de 2, es relativamente fácil encontrar valores para c y a.

$$y_{n+1} = 5\, y_n + 3 \mod (16).$$

Entonces:

- Se cumple que $(c, M) = (5, 16) = 1$.
- Se cumple que $5 \equiv 1 \mod 2$, donde 2 es el único primo que divide a M.
- 4 divide a 16, y $5 \equiv 1 \mod 4$.

Si se cumplen todas las hipótesis, debería obtenerse una secuencia máxima. Tomamos $y_0 = 0$.

$$y_1 = 5 \cdot 0 + 3 \mod (16), \qquad y_1 = 3,$$

$$y_2 = 5 \cdot 3 + 3 \mod (16), \qquad y_2 = 2,$$

siguiendo el razonamiento la secuencia continúa:

$$\ldots \quad 13 \quad 4 \quad 7 \quad 6 \quad 1 \quad 8 \quad 11 \quad 10 \quad 5 \quad 12 \quad 15 \quad 14 \quad 9 \quad 0 \ldots$$

Notemos en particular que, si $M = 2^n$, entonces a debe ser de la forma $4m + 1$ y c debe ser impar. Una ventaja de tomar M igual a una potencia de 2, digamos 2^n, es que computacionalmente tomar módulo equivale a considerar los últimos n bits de la representación.

Ejemplo 1.2.4. En [5] se presenta el siguiente ejemplo de un modelo de generador provisto por bibliotecas de ANSI C, y que responde a un tipo de generador lineal congruencial mixto. En este caso se toma:

$$a = 1103515245, \qquad c = 12345, \qquad M = 2^{32}.$$

Así en este caso el período de la secuencia es $K = 2^{32} = 4\,294\,967\,296$.

Una desventaja de un generador mixto, es que para cada valor generado se efectúa una suma, operación que no se realiza para un generador multiplicativo. Si bien en este caso el período de la secuencia puede ser a lo sumo $M-1$, es preferible usar este tipo de generadores para ganar en velocidad y costo operacional.

Veamos entonces qué condiciones debe tener un generador multiplicativo para poder obtener un período máximo, $K = M-1$. Antes de presentar un resultado relacionado con estos generadores, definimos el concepto de **raíz primitiva** de un número natural.

Definición 1.2.2. Sea M un número natural. Se dice que a es una **raíz primitiva** de M si

$$a^{(M-1)/p} \not\equiv 1 \mod (M)$$

para cualquier factor primo p de $M-1$.

Con esta definición, enunciamos el siguiente resultado:

Teorema 1.2.2. Para un generador multiplicativo

$$y_{i+1} = a\, y_i \mod M,$$

el período K de la secuencia verifica las siguientes tres propiedades:

- Si $K = M-1$ entonces M es primo.
- Si M es primo, entonces K divide a $M-1$.
- $K = M-1$ si y sólo si a es raíz primitiva de M y M es primo.

Entonces el Teorema 1.2.2 nos da pautas para encontrar generadores multiplicativos con período máximo. Lo óptimo sería determinar un número primo M grande, y una raíz primitiva de a. ¿Tarea simple? Claro que no.

Ejemplo 1.2.5. Si tomamos $M = 19$, entonces $a = 2$ es una raíz primitiva. Para ver esto, notemos que $M-1 = 18$ es divisible por los primos 2 y 3. Como $a^{18/2} = 2^9$, $a^{18/3} = 2^6$, y 19 no divide a $2^9 - 1$ ni a $2^6 - 1$, entonces 2 es raíz primitiva.

Si tomamos como semilla $y_0 = 1$ tenemos la secuencia:

$$1,\quad 2,\quad 4,\quad 8,\quad 16,\quad 13,\quad 7,\quad 14,\quad 9,\quad 18,\quad 17,\quad 15,\quad 11,\quad 3,\quad 6,\quad 12,\quad 5,\quad 10,\quad 1\ldots$$

que tiene período 19.

Notemos en el Ejemplo 1.2.5, que los primeros elementos de la secuencia no parecen tan aleatorios. Esto se debe en parte que la raíz primitiva elegida, 2, es muy pequeña. Por lo tanto es conveniente tomar una raíz primitiva más grande para superar más rápidamente el valor de M. Afortunadamente, existen resultados que nos permiten encontrar otras raíces primitivas a partir de una conocida. En particular, si a es raíz primitiva y $(m, M-1) = 1$, entonces a^m es raíz primitiva.

Entonces, si en el Ejemplo 1.2.5 tomamos $a = 2^5 = 13 \mod (19)$, la secuencia obtenida sería:

$$1, \quad 13, \quad 17, \quad 12, \quad 4, \quad 14, \quad 11, \quad 10, \quad 16, \quad 18, \quad 6, \quad 2, \quad 7, \quad 15, \quad 5, \quad 8, \quad 9, \quad 3, \quad 1, \quad \ldots$$

Ejemplo 1.2.6. En [5] se menciona el generador congruencial multiplicativo con

$$a = 7^5 = 16807 \qquad M = 2^{31} - 1 = 2\,147\,483\,647.$$

En este caso, M es un **primo de Mersenne**, es decir, de la forma $2^{2^k-1} - 1$. La factorización en primos de $M - 1$ está dada por:

$$M - 1 = 2^{31} - 2 = 2 \cdot 3^2 \cdot \mathbf{7} \cdot 11 \cdot 31 \cdot 151 \cdot 331.$$

Dado que 7 es raíz primitiva de M y $(5, M-1) = 1$, entonces $7^5 = 16807$ es raíz primitiva, y la secuencia generada tiene período:

$$K = M - 1 = 2\,147\,483\,646.$$

Una de las **desventajas** de cualquier generador congruencial lineal, sea mixto o multiplicativo es que no se cumple tan satisfactoriamente la condición (1) de la Definición 1.1.1.

Para el caso de los generadores congruenciales lineales, ocurre lo siguiente. Si la secuencia producida es:

$$y_0,\ y_1,\ y_2, \ldots, y_n\ \ldots,$$

está demostrado que los puntos

$$(y_j, y_{j+1}, \ldots, y_{j+k-1}), \quad j = 0, 1, 2, \ldots$$

están ubicados en no más de

$$(k!M)^{1/k} = (k!)^{1/k}\ \sqrt[k]{M} \quad \text{hiperplanos paralelos.}$$

¿Qué nos dice esto? Para un k suficientemente grande, los puntos quedan ubicados en una cantidad finita de hiperplanos, y por lo tanto existen "franjas" en las que no cae ninguna k-upla. La Figura 1.1 ilustra esta situación para algunos generadores congruenciales lineales.

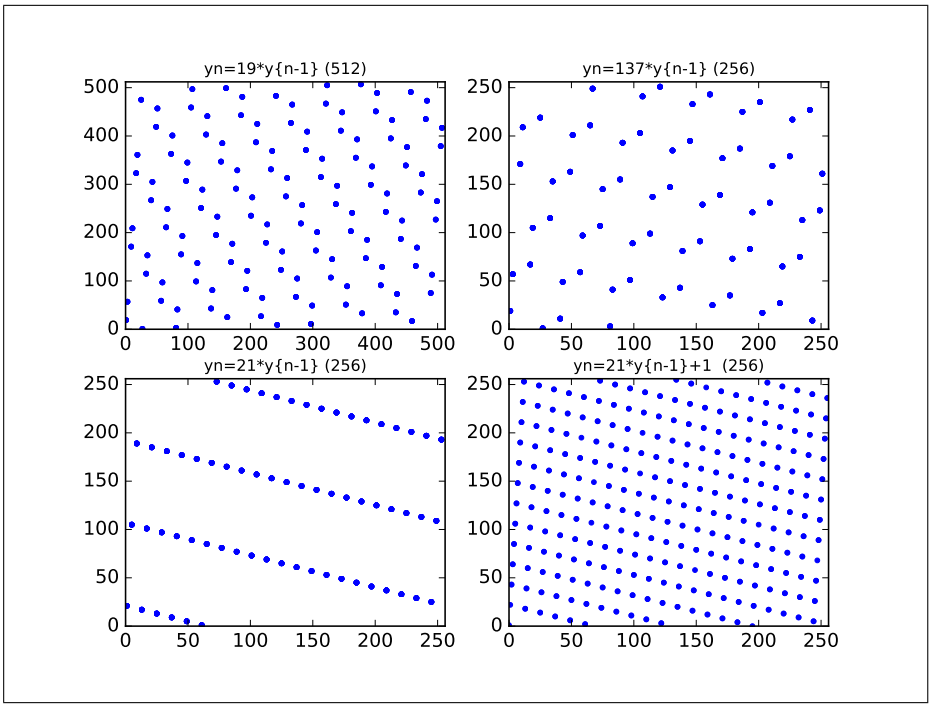

Figura 1.1: Generadores congruenciales. Distribución de pares (y_i, y_{i+1})

Un mal generador y ejemplo de este problema es **RANDU**. Este es un generador del tipo congruencial lineal multiplicativo ($c = 0$), con $M = 2^{31}$ y $a = 2^{16} + 3$:

$$y_n = 65539 \cdot y_{n-1} \quad \text{mód } 2\,147\,483\,648.$$

Las ternas generadas por RANDU se ubican en 15 planos paralelos, dentro del cubo de lado 1. Figura 1.2. Cabe mencionar que este generador fue implementado en computadoras de IBM por mucho tiempo, y difundido a otros sistemas.

1.2.2. Generadores congruenciales lineales combinados

Una forma de producir *mayor aleatoriedad* en una secuencia, y evitar la desventaja de los hiperplanos mencionada en la sección anterior, es utilizar más de un generador congruencial lineal y combinarlos entre ellos. Por combinar entendemos sumarlos o restarlos, y en general se recomienda la resta.

Ahora bien, ¿es cierto que si sumamos dos variables aleatorias uniformes obtenemos otra distribución uniforme? La respuesta es NO, y podemos verlo claramente arrojando dos dados. Supongamos que sus caras son igualmente probables, entonces las sumas no lo son: el 7 es más probable que el 2, por ejemplo:

$$2 = 1 + 1 \qquad 3 = 1 + 2 = 2 + 1 \qquad 4 = 1 + 3 = 2 + 2 = 3 + 1$$

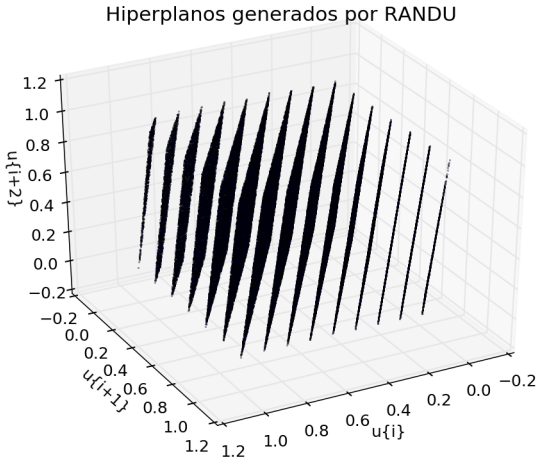

Figura 1.2: Generador RANDU

Sin embargo, si sumamos las caras y consideramos el resultado módulo 6, entonces, vemos que:

$$\begin{aligned}
1 &= 1+6=2+5=3+4=4+3=5+2=6+1 && \text{mód } (6)\\
2 &= 2+6=3+5=4+4=5+3=6+2=1+1 && \text{mód } (6)\\
3 &= 3+6=4+5=5+4=6+3=2+1=1+2 && \text{mód } (6)\\
4 &= 4+6=5+5=1+3=2+2=3+1=6+4 && \text{mód } (6)\\
5 &= 5+6=6+5=1+4=2+3=3+2=4+1 && \text{mód } (6)\\
0 &= 6+6=1+5=2+4=3+3=4+2=5+1 && \text{mód } (6)
\end{aligned}$$

Esto es, ahora cada suma tiene la misma probabilidad de ocurrir, y por lo tanto **la suma módulo 6** tiene distribución uniforme.

Ejercicio 1.2.1. ¿Qué pasaría si uno de los dados tuviera 5 caras, y consideraramos las sumas módulo 6?, ¿y si se consideran módulo 5?

Este resultado se generaliza en el siguiente teorema:

Teorema 1.2.3. Sean $W_1, W_2, \ldots, W_n$ variables aleatorias discretas uniformes, tales que $W_1 \sim U(\{0, d-1\})$ para cierto $d \geq 1$. Entonces

$$W = (\sum_{j=1}^{n} W_j) \quad \text{mód } d$$

es una v.a. uniforme discreta en $\{0, d-1\}$.

La demostración puede encontrarse en [1]. Notemos que sólo se requiere que una de las variables aleatorias sea uniforme en $\{0, d-1\}$.

Bien, esto nos dice que podemos sumar dos o más generadores congruenciales, y tomando módulo uno de ellos obtendríamos un nuevo generador. ¿Cuál es la ventaja de hacer esto? Recordemos que los generadores congruenciales tienen un determinado período. Si sumamos dos generadores con un mismo período K, entonces K será el período de la suma. Pero si los períodos son diferentes, entonces la suma tiene un período mayor.

Este resultado se enuncia de manera general en el siguiente teorema, y su demostración puede verse en [1].

Teorema 1.2.4. Consideremos una familia de N generadores, donde para cada j, $j = 1, 2, \ldots, N$, el generador j tiene período K_j y evoluciona de acuerdo a una ley:

$$f_j(y_{n,j}) = f_j(y_{n-1,j}), \qquad n \geq 1, \quad y_0 = \text{semilla del generador } j.$$

Entonces el período K de la secuencia

$$s_n = (y_{n,1}, y_{n,2}, \ldots, y_{n,N}), \qquad n \geq 1,$$

es igual al mínimo común múltiplo de $K_1, K_2, \ldots, K_N$.

Así, los Teoremas 1.2.3 y 1.2.4 constituyen la base teórica que nos permite obtener nuevos generadores congruenciales de v.a. uniformes a partir de la suma o resta de dos o más generadores. Además, con una buena elección de los períodos de los generadores se podrá garantizar un período mucho más largo para la combinación.

Ejemplo 1.2.7. Un generador de estas características es el propuesto en [1]. Se consideran los generadores:

$$x_n = 40014 x_{n-1} \mod 2^{31} - 85$$
$$y_n = 40692 y_{n-1} \mod 2^{31} - 249$$

Los períodos de estos generadores tienen un solo 2 como factor común:

$$2^{31} - 86 = 2 \cdot 3 \cdot 7 \cdot 631 \cdot 81031, \qquad 2^{31} - 250 = 2 \cdot 19 \cdot 31 \cdot 1019 \cdot 1789,$$

por lo que el período de la secuencia $(x_n - y_n) \mod M$ (para cualquiera de los módulos M) es del orden del producto de los dos períodos dividido 2. En este caso,

$$K \approx 2^{61} = 2\,305\,843\,009\,213\,693\,952 \sim 2.3 \times 10^{18}.$$

La Figura 1.3 ilustra una muestra de aproximadamente 5000 puntos (U_i, U_{i+1}) para cada uno de los tres generadores, con $U_i < 0.001$.

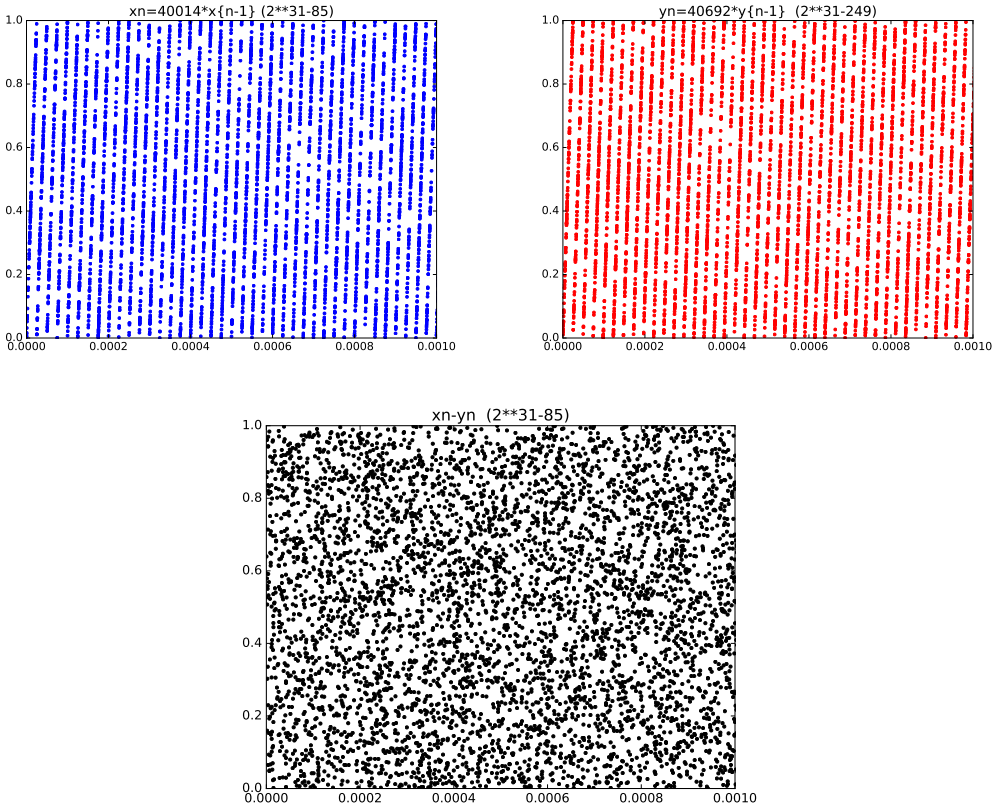

Figura 1.3: Generador del Ejemplo 1.2.7.

1.2.3. Otros generadores eficientes y portables

Existen otras formas de mejorar la eficiencia de los generadores construidos a partir de un generador congruencial, como son del tipo Fibonacci, resta con préstamo, suma con acarreo, y otros tantos. En [4] se presenta una extensa lista de generadores candidatos a ser combinados para obtener aún mejores generadores. Algunos son los siguientes:

Módulo	Secuencia	Período
2^{32}	$x_n = 69069\, x_{n-1} + \text{impar}$	2^{32}
2^{32}	$x_n = x_{n-1} * x_{n-2}$	2^{31}
2^{32}	$x_n = x_{n-1} + x_{n-2} + C$	2^{58}
$2^{31} - 69$	$x_n = x_{n-3} - x_{n-1}$	2^{62}
$2^{32} - 18$	$x_n = x_{n-2} - x_{n-3} - C$	2^{95}

La constante C indica un 0 o un 1 según corresponda, para acarreos o préstamos.

Por último, señalamos que en la biblioteca Python se implementa la rutina **Mersenne-twister**, bastante más compleja que los generadores que hemos visto. Invitamos al lector a investigar sobre este generador en particular.

Bibliografía

[1] PIERRE L'ECUYER, *Efficient and Portable Combined Random Number Generators*, Communications of the ACM, (1988) 31(6), pp.742–774.

[2] OSCAR BUSTOS Y ALEJANDRO FRERY, *Simulacao estocastica : teoría e algoritmos (versao completa)*. Series Monografías en Matemática. Vol 42. Edit. IMPA. 1992.

[3] DONALD E. KNUTH, *Seminumerical Algorithms. The Art of Computer Programming.* Vol 2. Edit. Addison-Wesley. 1998.

[4] GEORGE MARSAGLIA AND ARIF ZAMAN, *Some portable very-long-period random number generators*, Computers in Physics, (1994) 8 (6).

[5] *Numerical recipes in C: The Art of Scientific Computing*. Cambridge University Press. (1988-1992)