Capítulo 2

Números pseudoaleatorios

2.1 Los números pseudoaleatorios

2.2 Generación de números pseudoaleatorios

2.3 Propiedades de los números pseudoaleatorios entre 0 y 1

2.4 Pruebas estadísticas para los números pseudoaleatorios

2.5 Problemas

Referencias

2.1 Los números pseudoaleatorios

Para poder realizar una simulación que incluya variabilidad dentro de sus eventos, es preciso generar una serie de números que sean aleatorios por sí mismos, y que su aleatoriedad se extrapole al modelo de simulación que se está construyendo. Como puede comprender, en la construcción del modelo los números aleatorios juegan un papel relevante.

Así, una de las primeras tareas que es necesario llevar a cabo consiste en determinar si los números que utilizaremos para "correr" o ejecutar la simulación son realmente aleatorios o no. Por desgracia, precisar lo anterior con absoluta certidumbre resulta muy complicado, ya que para ello tendríamos que generar un número infinito de valores que nos permitiera comprobar la inexistencia de correlaciones entre ellos. Esto sería muy costoso y tardado, además volvería impráctico el uso de la simulación aun con las computadoras más avanzadas.

A pesar de lo anterior, podemos asegurar que el conjunto de números que utilizaremos en una simulación se comporta de manera muy similar a un conjunto de números totalmente aleatorios; por ello es que se les denomina *números pseudoaleatorios*. Casi todas las aplicaciones comerciales tienen varios generadores de números pseudoaleatorios que pueden generar un conjunto muy grande de números sin mostrar correlación entre ellos. En el presente capítulo discutiremos algunos de los métodos de generación de números pseudoaleatorios, y precisaremos qué características deben tener para emplearlos como una fuente confiable de variabilidad dentro de los modelos. Asimismo, se mostrarán algunas de las pruebas más comunes para comprobar qué tan aleatorios son los números obtenidos con dichos generadores.

2.2 Generación de números pseudoaleatorios

Para realizar una simulación se requieren números aleatorios en el intervalo (0,1), a los cuales se hará referencia como r_i, es decir, una secuencia $r_i = \{r_1, r_2, r_3, ..., r_n\}$ que contiene "*n*" números, todos ellos diferentes. El valor "*n*" recibe el nombre de *periodo* o *ciclo de vida* del generador que creó la secuencia r_i.

Los r_i constituyen la parte medular de la simulación de procesos estocásticos, y por lo regular, se usan para generar el comportamiento de variables aleatorias, tanto continuas como discretas. Debido a que no es posible generar números realmente aleatorios, consideramos los r_i como números pseudoaleatorios generados por medio de algoritmos determinísticos que requieren parámetros de arranque.

Para simular el comportamiento de una o más variables aleatorias es necesario contar con un conjunto suficientemente grande de r_i que permita, por ejemplo, que la secuencia tenga al menos un periodo de vida de $n = 2^{31} = 2{,}147{,}483{,}648$. De acuerdo con L'Ecuyer[4] una secuencia de r_i con periodo de vida de $n = 2^{31}$ es relativamente pequeña; de hecho, incluso una secuencia de r_i que contenga un ciclo de vida de $n = 2^{64}$ se considera pequeña. En la actualidad, contamos ya con generadores y procesadores capaces de construir una secuencia de r_i con periodo de vida de $n = 2^{200}$.

Tal vez se preguntará; ¿por qué debe interesarnos construir una secuencia de números r_i lo bastante grande? A continuación ilustraremos la razón mediante un ejemplo.

Suponga que queremos simular el tiempo de atención a clientes en un banco que tiene 5 cajeros en paralelo, cada uno de los cuales atiende alrededor de 50 clientes diarios. Para simular el tiempo de atención se requiere un generador de variable aleatoria en función de r_i, como por ejemplo $T_i = 5 + 2r_i$, expresado en minutos para toda $i = 1, 2, 3, \ldots, n$. (El tema de generadores de variables aleatorias se presenta en el capítulo 3). Si simulamos el tiempo de atención de manera aislada, es decir, sin considerar el tiempo transcurrido desde la llegada de éstos, serán necesarios $5 \times 50 = 250$ números r_i para simular un día; si deseáramos simular 5 días se necesitarían $250 \times 5 = 1250$ r_i. Ahora bien, si consideramos el tiempo desde la llegada de los clientes, precisaríamos de 250 r_i para simular el tiempo transcurrido desde la llegada al banco de los 250 clientes por día, y $250 \times 5 = 1250$ r_i para simular el correspondiente al total de clientes atendidos durante 5 días. Por lo tanto, se requerirán 2500 números pseudoaleatorios r_i para simular la operación del banco durante 5 días.

Como se mencionó en el capítulo 1, los resultados no pueden basarse en una sola simulación del sistema; por el contrario, es necesario realizar varias *réplicas* de la misma, corriendo cada una de ellas con números pseudoaleatorios diferentes. Si retomamos el ejemplo del banco, simular 5 días otra vez significa que necesitamos otros 2500 números pseudoaleatorios en el intervalo (0,1). En consecuencia, se requieren 5000 r_i para realizar la simulación del sistema de atención a clientes con dos réplicas.

Usted podrá imaginar cuántos números r_i serán necesarios para simular la operación del banco durante un año con 9 réplicas, o cuántos números r_i se requieren para simular un sistema productivo durante un año con varias líneas de producción, y cada una con varias estaciones, y cada estación con uno o más procesos.

Dada la importancia de contar con un conjunto de r_i suficientemente grande, en esta sección se presentan diferentes algoritmos determinísticos para obtenerlo. Por otra parte, es conveniente señalar que el conjunto de r_i debe ser sometido a una variedad de pruebas para verificar si los números que lo conforman son realmente independientes y uniformes. En la sección 2.4 veremos las pruebas estadísticas que determinan si un conjunto r_i tiene las propiedades de independencia y uniformidad. Una vez generado el conjunto r_i mediante un algoritmo determinístico, es necesario someterlo a las pruebas antes mencionadas: si las supera, podrá utilizarse en la simulación; de lo contrario, simplemente deberemos desecharlo.

Un conjunto de r_i debe seguir una distribución uniforme continua, la cual está definida por:

$$f(r) = \begin{cases} 1, & 0 \le r \le 1 \\ 0, & \textit{en cualquier otro valor} \end{cases}$$

Generar un conjunto de r_i es una tarea relativamente sencilla; para ello, el lector sólo tiene que diseñar su propio algoritmo de generación. Lo que resulta difícil es diseñar un algoritmo que genere un conjunto de r_i con periodo de vida lo bastante grande (N), y que además pase sin problema las pruebas de uniformidad e independencia, lo cual implica evitar problemas como éstos:

- Que los números del conjunto r_i no estén uniformemente distribuidos, es decir, que haya demasiados r_i en un subintervalo y en otro muy pocos o ninguno.

- Que los números r_i generados sean discretos en lugar de continuos.
- Que la media del conjunto sea muy alta o muy baja, es decir, que esté por arriba o por debajo de ½.
- Que la varianza del conjunto sea muy alta o muy baja, es decir, que se localice por arriba o por debajo del ½ (la obtención de estos valores se discute en la sección 2.3).

En ocasiones se presentan también anomalías como números r_i seguidos por arriba o por debajo de la media; secuencia de r_i por arriba de la media, seguida de una secuencia por debajo de la media, y viceversa, o varios r_i seguidos en forma ascendente o descendente.

A continuación se presentan diferentes algoritmos determinísticos para generar los r_i, los cuales se clasifican en algoritmos no congruenciales y congruenciales. Los algoritmos no congruenciales que analizaremos en esta obra son: cuadrados medios, productos medios y multiplicador constante. Entre los algoritmos congruenciales se encuentran los algoritmos congruenciales lineales y los no lineales. En este libro abordaremos los algoritmos congruenciales lineales —tales como algoritmo congruencial lineal, multiplicativo y aditivo—, y los algoritmos no lineales, como el algoritmo de Blum, Blum y Shub, y el congruencial cuadrático.

2.2.1 Algoritmo de cuadrados medios

Este algoritmo no congruencial fue propuesto en la década de los cuarenta del siglo XX por Von Neumann y Metropolis.[1] Requiere un número entero detonador (llamado *semilla*) con *D* dígitos, el cual es elevado al cuadrado para seleccionar del resultado los *D* dígitos del centro; el primer número r_i se determina simplemente anteponiendo el "0." a esos dígitos. Para obtener el segundo r_i se sigue el mismo procedimiento, sólo que ahora se elevan al cuadrado los *D* dígitos del centro que se seleccionaron para obtener el primer r_i. Este método se repite hasta obtener *n* números r_i. A continuación se presentan con más detalle los pasos para generar números con el algoritmo de cuadrados medios.

1. Seleccionar una semilla (X_0) con *D* dígitos ($D > 3$).
2. Sea Y_0 = resultado de elevar X_0 al cuadrado; sea X_1 = los *D* dígitos del centro, y sea $r_i = 0$. *D* dígitos del centro.
3. Sea Y_i = resultado de elevar X_i al cuadrado; sea X_{i+1} = los *D* dígitos del centro, y sea $r_i = 0$. *D* dígitos del centro para toda $i = 1, 2, 3, \ldots n$.
4. Repetir el paso 3 hasta obtener los *n* números r_i deseados.

Nota: Si no es posible obtener los *D* dígitos del centro del número Y_i, agregue ceros a la izquierda del número Y_i.

Para ilustrar la mecánica del algoritmo de cuadrados medios se presenta el siguiente ejemplo.

Ejemplo 2.1

Generar los primeros 5 números r_i a partir de una semilla $X_0 = 5735$, de donde se puede observar que $D = 4$ dígitos.

Solución:

$Y_0 = (5735)^2 = 32890225$ $X_1 = 8902$ $r_1 = 0.8902$
$Y_1 = (8902)^2 = 79245604$ $X_2 = 2456$ $r_2 = 0.2456$
$Y_2 = (2456)^2 = 06031936$ $X_3 = 0319$ $r_3 = 0.0319$
$Y_3 = (0319)^2 = 101761$ $X_4 = 0176$ $r_4 = 0.0176$
$Y_4 = (0176)^2 = 030976$ $X_5 = 3097$ $r_5 = 0.3097$

El algoritmo de cuadrados medios generalmente es incapaz de generar una secuencia de r_i con periodo de vida n grande. Además, en ocasiones sólo es capaz de generar un número, por ejemplo, si $X_0 = 1000$, entonces $X_1 = 0000$; $r_i = 0.0000$ y se dice que el algoritmo se degenera con la semilla de $X_0 = 1000$.

2.2.2 Algoritmo de productos medios

La mecánica de generación de números pseudoaleatorios de este algoritmo no congruencial es similar a la del algoritmo de cuadrados medios. La diferencia entre ambos radica en que el algoritmo de productos medios requiere dos semillas, ambas con D dígitos; además, en lugar de elevarlas al cuadrado, las semillas se multiplican y del producto se seleccionan los D dígitos del centro, los cuales formarán el primer número pseudoaleatorio $r_i = 0.D$ dígitos. Después se elimina una semilla, y la otra se multiplica por el primer número de D dígitos, para luego seleccionar del producto los D dígitos que conformarán un segundo númeror_i. Entonces se elimina la segunda semilla y se multiplican el primer número de D dígitos por el segundo número de D dígitos; del producto se obtiene el tercer númeror_i. Siempre se irá eliminando el número más antiguo, y el procedimiento se repetirá hasta generar los n números pseudoaleatorios. A continuación se presentan con más detalle los pasos del método para generar números con el algoritmo de producto medios.

1. Seleccionar una semilla (X_0) con D dígitos ($D > 3$)
2. Seleccionar una semilla (X_1) con D dígitos ($D > 3$)
3. Sea $Y_0 = X_0 * X_1$; sea $X_2 =$ los D dígitos del centro, y sea $r_i = 0.D$ dígitos del centro.
4. Sea $Y_i = X_i * X_{i+1}$; sea $X_{i+2} =$ los D dígitos del centro, y sea $r_{i+1} = 0.D$ dígitos del centro para toda $i = 1, 2, 3, \ldots n$.
5. Repetir el paso 4 hasta obtener los n números r_i deseados.

Nota: Si no es posible obtener los D dígitos del centro del número Y_i, agregue ceros a la izquierda del número Y_i.

Para ilustrar la mecánica del algoritmo de productos medios se presenta el siguiente ejemplo.

Ejemplo 2.2

Generar los primeros 5 números r_i a partir de las semillas $X_0 = 5015$ y $X_1 = 5734$; observe que ambas semillas tienen $D = 4$ dígitos.

Solución:

$Y_0 = (5015)(5734) = 28756010$ $X_2 = 7560$ $r_1 = 0.7560$
$Y_1 = (5734)(7560) = 43349040$ $X_3 = 3490$ $r_2 = 0.3490$
$Y_2 = (7560)(3490) = 26384400$ $X_4 = 3844$ $r_3 = 0.3844$
$Y_3 = (3490)(3844) = 13415560$ $X_5 = 4155$ $r_4 = 0.4155$
$Y_4 = (3844)(4155) = 15971820$ $X_6 = 9718$ $r_5 = 0.9718$

2.2.3 Algoritmo de multiplicador constante

Este algoritmo no congruencial es similar al algoritmo de productos medios. Los siguientes son los pasos necesarios para generar números pseudoaleatorios con el algoritmo de multiplicador constante.

1. Seleccionar una semilla (X_0) con D dígitos ($D > 3$).
2. Seleccionar una constante (a) con D dígitos ($D > 3$).
3. Sea $Y_0 = a^*X_0$; sea X_1 = los D dígitos del centro, y sea $r_i = 0.D$ dígitos del centro.
4. Sea $Y_i = a^*X_i$; sea X_{i+1} = los D dígitos del centro, y sea $r_{i+1} = 0.D$ dígitos del centro para toda $i = 1, 2, 3, \ldots n$.
5. Repetir el paso 4 hasta obtener los n números r_i deseados.

Nota: Si no es posible obtener los D dígitos del centro del número Y_i, agregue ceros a la izquierda del número Y_i.

Para ilustrar la mecánica del algoritmo de multiplicador constante se presenta el siguiente ejemplo.

Ejemplo 2.3

Generar los primeros 5 números r_i a partir de la semilla $X_0 = 9803$ y con la constante $a = 6965$. Observe que tanto la semilla como la constante tienen $D = 4$ dígitos.

Solución:

$Y_0 = (6965)(9803) = 68277895$ $X_1 = 2778$ $r_1 = 0.2778$
$Y_1 = (6965)(2778) = 19348770$ $X_2 = 3487$ $r_2 = 0.3487$
$Y_2 = (6965)(3487) = 24286955$ $X_3 = 2869$ $r_3 = 0.2869$
$Y_3 = (6965)(2869) = 19982585$ $X_4 = 9825$ $r_4 = 0.9825$
$Y_4 = (6965)(9825) = 68431125$ $X_5 = 4311$ $r_5 = 0.4311$

2.2.4 Algoritmo lineal

Este algoritmo congruencial fue propuesto por D. H. Lehmer[5] en 1951. Según Law y Kelton,[3] no ha sido el más usado. El algoritmo congruencial lineal genera una secuencia de números enteros por medio de la siguiente ecuación recursiva:

$$X_i + 1 = (ax_i + c)\text{mod}(m) \qquad i = 0, 1, 2, 3, \ldots, n$$

donde X_0 es la semilla, a es la constante multiplicativa, c es una constante aditiva, y m es el módulo. $X_0 > 0$, $a > 0$, $c > 0$ y $m > 0$ deben ser números enteros. La operación "mod (m)" significa multiplicar X_i por a, sumar c, y dividir el resultado entre m para obtener el residuo X_{i+1}. Es importante señalar que la ecuación recursiva del algoritmo congruencial lineal genera una secuencia de números enteros $S = \{0, 1, 2, 3, ..., m - 1\}$, y que para obtener números pseudoaleatorios en el intervalo (0,1) se requiere la siguiente ecuación:

$$r_i = \frac{X_i}{m-1} \qquad i = 0, 1, 2, 3, \ldots, n$$

Analice el siguiente ejemplo para comprender mejor la mecánica del algoritmo congruencial lineal:

Ejemplo 2.4

Generar 4 números entre 0 y 1 con los siguientes parámetros: $X_0 = 37$, $a = 19$, $c = 33$ y $m = 100$.

Solución:

$X_1 = (19*37 + 33) \text{ mod } 100 = 36$ $\qquad r_1 = 36/99 = 0.3636$
$X_2 = (19*36 + 33) \text{ mod } 100 = 17$ $\qquad r_2 = 17/99 = 0.1717$
$X_3 = (19*17 + 33) \text{ mod } 100 = 56$ $\qquad r_3 = 56/99 = 0.5656$
$X_4 = (19*56 + 33) \text{ mod } 100 = 97$ $\qquad r_4 = 97/99 = 0.9797$

En el ejemplo anterior se dieron de manera arbitraria cada uno de los parámetros requeridos: X_0, a, c, m. Sin embargo, para que el algoritmo sea capaz de lograr el máximo periodo de vida N, es preciso que dichos parámetros cumplan ciertas condiciones. Banks, Carson, Nelson y Nicol[1] sugieren lo siguiente:

$m = 2^g$
$a = 1 + 4k$
k debe ser entero
c relativamente primo a m
g debe ser entero

Bajo estas condiciones se obtiene un periodo de vida máximo: $N = m = 2^g$. Veamos un ejemplo más, tomando en cuenta lo anterior.

Ejemplo 2.5

Generar suficientes números entre 0 y 1 con los parámetros $X_0 = 6$, $k = 3$, $g = 3$ y $c = 7$, hasta encontrar el periodo de vida máximo (N).

Como podemos ver, si se cumplen las condiciones que Banks, Carson, Nelson y Nicol sugieren, se logrará el periodo máximo $N = m = 8$. A continuación se presenta el desarrollo de la generación de los números r_i.

$$a = 1 + 4\,(3) = 13 \text{ y } m = 2^3 = 8$$

$X_0 = 6$

$X_1 = (13*6 + 7) \bmod 8 = 5 \qquad r_1 = 5/7 = 0.714$

$X_2 = (13*5 + 7) \bmod 8 = 0 \qquad r_2 = 0/7 = 0.000$

$X_3 = (13*0 + 7) \bmod 8 = 7 \qquad r_3 = 7/7 = 1.000$

$X_4 = (13*7 + 7) \bmod 8 = 2 \qquad r_4 = 2/7 = 0.285$

$X_5 = (13*2 + 7) \bmod 8 = 1 \qquad r_5 = 1/7 = 0.142$

$X_6 = (13*1 + 7) \bmod 8 = 4 \qquad r_6 = 4/7 = 0.571$

$X_7 = (13*4 + 7) \bmod 8 = 3 \qquad r_7 = 3/7 = 0.428$

$X_8 = (13*3 + 7) \bmod 8 = 6 \qquad r_8 = 6/7 = 0.857$

Es importante mencionar que el número generado en $X_8 = 6$ es exactamente igual a la semilla X_0, y si continuáramos generando más números, éstos se repetirían. Además, sabemos que el algoritmo congruencial lineal genera una secuencia de números enteros $S = \{0, 1, 2, 3, ..., m - 1\}$. Observe que en este caso se genera la secuencia $S = \{0, 1, 2, 3, 4, 5, 6, 7\}$.

Ejemplo 2.6

Consideremos de nuevo el ejemplo anterior, pero tratemos de infringir de manera arbitraria alguna de las condiciones. Supongamos que $a = 12$; se sabe que a no es el resultado de $1 + 4k$, donde k es un entero. Veamos el comportamiento del algoritmo congruencial lineal ante tal cambio.

Solución:

$$a = 1 + 4\,(3) = 13 \text{ y } m = 2^3 = 8$$

$X_0 = 6$

$X_1 = (12*6 + 7) \bmod 8 = 7 \qquad r_1 = 7/7 = 1.000$

$X_2 = (12*7 + 7) \bmod 8 = 3 \qquad r_2 = 3/7 = 0.428$

$X_3 = (12*3 + 7) \bmod 8 = 3 \qquad r_3 = 3/7 = 0.428$

El periodo de vida en este caso es $N = 2$, de manera que, como puede ver, el periodo de vida máximo no se logra. Como conclusión tenemos que si no se cumple alguna de las condiciones, el periodo de vida máximo $N = m$ no se garantiza, por lo que el periodo de vida será menor que m.

2.2.5 Algoritmo congruencial multiplicativo

El algoritmo congruencial multiplicativo surge del algoritmo congruencial lineal cuando $c = 0$. Entonces la ecuación recursiva es:

$$X_{i+1} = (aX_i) \bmod (m) \qquad i = 0, 1, 2, 3, \ldots, n$$

En comparación con el algoritmo congruencial lineal, la ventaja del algoritmo multiplicativo es que implica una operación menos a realizar. Los parámetros de arranque de este algoritmo son X_0, a y m, los cuales deben ser números enteros y mayores que cero. Para transformar los números X_i en el intervalo (0, 1) se usa la ecuación $r_i = x_i/(m - 1)$. De acuerdo con Banks, Carson, Nelson y Nicol,[1] las condiciones que deben cumplir los parámetros para que el algoritmo congruencial multiplicativo alcance su máximo periodo N, son:

$m = 2^g$
$a = 3 + 8k$ o $a = 5 + 8k$
$k = 0, 1, 2, 3, \ldots$
X_0 = debe ser un número impar
g debe ser entero

A partir de estas condiciones se logra un periodo de vida máximo $N = k/4 = 2^{g-2}$

Ejemplo 2.7

Generar suficientes números entre 0 y 1 con los siguientes parámetros: $X_0 = 17$, $k = 2$ y $g = 5$, hasta encontrar el periodo o ciclo de vida.

Solución:

$$a = 5 + 8\,(2) = 21 \qquad \text{y} \qquad m = 32$$

$X_0 = 17$
$X_1 = (21*17) \bmod 32 = 5$ $\quad r_1 = 5/31 = 0.612$
$X_2 = (21*5) \bmod 32 = 9$ $\quad r_2 = 9/31 = 0.2903$
$X_3 = (21*9) \bmod 32 = 29$ $\quad r_3 = 29/31 = 1.9354$
$X_4 = (21*29) \bmod 32 = 1$ $\quad r_4 = 1/31 = 0.3225$
$X_5 = (21*1) \bmod 32 = 21$ $\quad r_5 = 21/31 = 0.6774$
$X_6 = (21*21) \bmod 32 = 25$ $\quad r_6 = 25/31 = 0.8064$
$X_7 = (21*25) \bmod 32 = 13$ $\quad r_7 = 13/31 = 0.4193$
$X_8 = (21*13) \bmod 32 = 17$ $\quad r_8 = 17/31 = 0.5483$

Si la semilla X_0 se repite, volverán a generarse los mismos números. Por lo tanto, el periodo de vida es $n = 8$, el cual corresponde a $N = m/4 = 32/4 = 8$.

Ejemplo 2.8

Ahora bien, si quebrantamos la condición de que la semilla sea un número impar, digamos con $X_0 = 12$, tenemos:

Solución:

$X_0 = 12$

$X_1 = (21*12) \bmod 32 = 28 \qquad r_1 = 28/31 = 0.9032$

$X_2 = (21*28) \bmod 32 = 12 \qquad r_2 = 12/31 = 0.3870$

En vista de que la semilla X_0 se repite, volverán a generarse los mismos números. Por lo tanto, el periodo de vida es $N = 2$.

2.2.6 Algoritmo congruencial aditivo

Este algoritmo requiere una secuencia previa de n números enteros $X_1, X_2, X_3, X_4, ..., X_n$ para generar una nueva secuencia de números enteros que empieza en $X_{n+1}, X_{n+2}, X_{n+3}, X_{n+4}, ...$ Su ecuación recursiva es:

$$X_i = (x_{i+1} + X_{i-n}) \bmod (m) \qquad i = n + 1, n + 2, n + 3, \ldots N$$

Los números r_i pueden ser generados mediante la ecuación

$$r_i = x_i/m - 1$$

Ejemplo 2.9

Generar 7 números pseudoaleatorios entre cero y uno a partir de la siguiente secuencia de números enteros: 65, 89, 98, 03, 69; $m = 100$.

Sean $X_1 = 65$, $X_2 = 89$, $X_3 = 98$, $X_4 = 03$, $X_5 = 69$. Para generar r_1, r_2, r_3, r_4, r_5, r_6 y r_7 antes es necesario generar $X_6, X_7, X_8, X_9, X_{10}, X_{11}, X_{12}$.

Solución:

$X_6 = (X_5 + X_1) \bmod 100 = (69 + 65) \bmod 100 = 34 \qquad r_1 = 34/99 = 0.3434$

$X_7 = (X_6 + X_2) \bmod 100 = (34 + 89) \bmod 100 = 23 \qquad r_2 = 23/99 = 0.2323$

$X_8 = (X_7 + X_3) \bmod 100 = (23 + 98) \bmod 100 = 21 \qquad r_3 = 21/99 = 0.2121$

$X_9 = (X_8 + X_4) \bmod 100 = (21 + 03) \bmod 100 = 24 \qquad r_4 = 24/99 = 0.2424$

$X_{10} = (X_9 + X_5) \bmod 100 = (24 + 69) \bmod 100 = 93 \qquad r_5 = 93/99 = 0.9393$

$X_{11} = (X_{10} + X_6) \bmod 100 = (93 + 34) \bmod 100 = 27 \qquad r_6 = 27/99 = 0.2727$

$X_{12} = (X_{11} + X_7) \bmod 100 = (27 + 23) \bmod 100 = 50 \qquad r_7 = 50/99 = 0.5050$

2.2.7 Algoritmos congruenciales no lineales

En esta sección se analizarán dos algoritmos congruenciales no lineales: el congruencial cuadrático y el algoritmo presentado por Blum, Blum y Shub.[2]

2.2.7.1 Algoritmo congruencial cuadrático

Este algoritmo tiene la siguiente ecuación recursiva:

$$X_{i+1} = (aX_i^2 + bX_i + c) \bmod (m) \qquad i = 0,1,2,3,...N$$

En este caso, los números r_i pueden ser generados con la ecuación $r_i = x_i/(m-1)$. De acuerdo con L'Ecuyer,[4] las condiciones que deben cumplir los parámetros m, a, b y c para alcanzar un periodo máximo de $N = m$ son:

$m = 2^g$

a debe ser un número par

c debe ser un número impar

g debe ser entero

$(b - 1) \bmod 4 = 1$

De esta manera se logra un periodo de vida máximo $N = m$.

Ejemplo 2.10

Generar, a partir del algoritmo congruencial cuadrático, suficientes números enteros hasta alcanzar el periodo de vida, para esto considere los parámetros $X_0 = 13$, $m = 8$, $a = 26$, $b = 27$ y $c = 27$. Como todas las condiciones estipuladas para los parámetros se satisfacen, es de esperarse que el periodo de vida del generador sea $N = m = 8$, tal como podrá comprobar al revisar los cálculos correspondientes, que se presentan a continuación.

Solución:

$X_1 = (26*13^2 + 27*13 + 27) \bmod (8) = 4$

$X_2 = (26*4^2 + 27*4 + 27) \bmod (8) = 7$

$X_3 = (26*7^2 + 27*7 + 27) \bmod (8) = 2$

$X_4 = (26*2^2 + 27*2 + 27) \bmod (8) = 1$

$X_5 = (26*1^2 + 27*1 + 27) \bmod (8) = 0$

$X_6 = (26*0^2 + 27*0 + 27) \bmod (8) = 3$

$X_7 = (26*3^2 + 27*3 + 27) \bmod (8) = 6$

$X_8 = (26*6^2 + 27*6 + 27) \bmod (8) = 5$

$X_9 = (26*5^2 + 27*5 + 27) \bmod (8) = 4$

Por otro lado, el algoritmo cuadrático genera una secuencia de números enteros $S = \{0, 1, 2, 3, ..., m - 1\}$, al igual que el algoritmo congruencial lineal.

2.2.7.2 Algoritmo de Blum, Blum y Shub[2]

Si en el algoritmo congruencial cuadrático $a = 1$, $b = 0$ y $c = 0$, entonces se construye una nueva ecuación recursiva:

$$X_{i+1} = (X_i^2) \bmod (m) \qquad i = 0,1,2,3,...n$$

La ecuación anterior fue propuesta por Blum, Blum y Shub[2] como un nuevo método para generar números que no tienen un comportamiento predecible.