



Facultad de Informática
Universidad Politécnica de Madrid



SIMULACIÓN DE SUCESOS DISCRETOS

Capítulo 2. Generación de números aleatorios

ÍNDICE

1. **Introducción**
2. **Contrastes empíricos**
 - **Uniformidad** (Contraste χ^2 y Kolmogorov-Smirnov)
 - **Aleatoriedad** (Rachas, Test de Póker)
3. **Generadores congruenciales**
4. **Otros generadores**
 - **Registro de desplazamiento**
 - **Fibonacci retardados**
 - **No lineales**
 - **Combinación de generadores**
 - **Generadores paralelos**
 - **Generadores comerciales**
5. **Conclusiones**

1. Introducción

Disponibilidad de un buen generador de números aleatorios.

Es un elemento esencial en muchas otras áreas de la **Informática** (algoritmos aleatorizados, verificación de algoritmos, complejidad de algoritmos, criptografía,...), de la **Estadística** (métodos de muestreo y remuestreo, contrastes Montecarlo, Inferencia Bayesiana...),

Además, un elemento importante en aplicaciones como juegos para ordenador, protectores de pantallas,...

La disponibilidad de generadores de números aleatorios en muchos entornos y compiladores haría pensar que para un mero usuario de la Simulación no sería necesario estudiar estas cuestiones.

Sin embargo, una lección que se deriva del estudio de algunos generadores comerciales es que **debemos actuar con sumo cuidado con ellos** (área de investigación bastante activa).

Problema: escoger una fuente de números aleatorios y obtener de esta fuente suficientes números para nuestro experimento de simulación.

Orígenes históricos.

Tablas de números aleatorios:

Tippet (1927): Universidad de Cambridge, 10.000 números aleatorios de 4 dígitos basados en censos.

Royo y Ferrer (1954): 250.000 resultados de la lotería nacional (INE).

Rand Corporation (1954): 1 millón de números aleatorios mediante el uso de mecanismos físicos (ruleta electrónica, medición de ruido electrónico en circuitos...).

Inconveniente del uso de mecanismos físicos → falta de **reproducibilidad**.

Procedimientos **algorítmicos de generación de números** → La idea (von Neumann) es producir números que parezcan aleatorios, empleando las operaciones aritméticas del ordenador: partiendo de una semilla inicial $(u_0, u_1, \dots, u_{-p+1})$, generar una sucesión mediante $u_i = d(u_{i-1}, \dots, u_{i-p})$, para cierta función d .

¿qué se considera como secuencia de números aleatorios?

Definición clásica de Kolmogorov (Kolmogorov y Uspenskii, 1987) (asociada a la idea de complejidad algorítmica). Una sucesión de números es aleatoria si no puede producirse eficientemente mediante un programa más corto que la propia serie.

Definición basada en la Estadística. Una sucesión de números aleatorios (u_i) es una sucesión de números en $(0,1)$ con las propiedades de

- Uniformidad en $(0,1)$ y
- Aleatoriedad o independencia estadística.

Otras propiedades relativas a la **eficiencia computacional**:

- 1) Rapidez;
- 2) Poco consumo de memoria;
- 3) Portabilidad;
- 4) Sencillez en la implementación;
- 5) Reproducibilidad y mutabilidad;
- 6) Periodo suficientemente largo.

2. Contrastes empíricos

Bondad de ajuste o uniformidad

Contraste χ^2

El contraste χ^2 de Pearson es el más antiguo y válido para distribuciones continuas y discretas. Sin embargo, es poco potente, por lo que permite justificar el rechazo de una hipótesis, pero proporciona escaso soporte para su aceptación.

Supongamos que tenemos una muestra x_1, \dots, x_n ($n \geq 25$) de una población con función de distribución $F_n(x)$ desconocida y deseamos contrastar la hipótesis

$$H_0: F_n(x) = F_0(x),$$

para todo $x \in \mathfrak{R}$, donde $F_0(x)$ está completamente especificada (conocemos la distribución y los parámetros de la misma), frente a la alternativa

$$H_1: F_n(x) \neq F_0(x) \text{ para algún } x.$$

En nuestro caso, $F_0(x) = U(0,1)$.

1. Agrupamos los n datos en k clases mutuamente excluyentes, $k \geq 5$, que cubran todo el rango posible de valores, siendo O_i a la frecuencia observada en la clase i .
2. Calculamos la frecuencia esperada, E_i , de la clase i de acuerdo con el modelo $F_\theta(x)$. Conociendo la probabilidad que asigna el modelo a la clase i tenemos

$$E_i = n \times p_i$$

3. Calculamos la discrepancia entre las frecuencias observadas y las esperadas mediante el modelo $F_\theta(x)$:

$$X^2 = \sum_{i=1}^k \frac{(O_i - E_i)^2}{E_i}$$

que se distribuye aproximadamente como una χ^2 cuando el modelo es correcto.

4. Determinamos los grados de libertad:
 - Si el modelo especifica las probabilidades p_i antes de tomar la muestra, entonces el n° de grados de libertad es $k - 1$.
 - Si las p_i se han calculado estimando r parámetros del modelo de máxima verosimilitud, entonces el n° de grados de libertad es $k - r - 1$.
5. Rechazamos el modelo cuando $X^2 \geq \chi^2_\alpha(k - r - 1)$ para un nivel de significación pequeño (0.05, 0.01, 0.001).

En nuestro caso, para contrastar la uniformidad escogeremos k subintervalos de $[0,1]$ de la misma longitud, siendo $p_i = 1/k$ y, por lo tanto, $E_i = n/k$ y $r = 0$, ya que no ha sido necesario estimar ningún parámetro de la distribución para obtener p_i .

Ejemplo. Durante la segunda guerra mundial se dividió el mapa de Londres en cuadrículas de $1/4 \text{ km}^2$ y se contabilizó el número de bombas caídas en cada cuadrícula durante un bombardeo alemán. Los resultados fueron:

Cuadrícula	0	1	2	3	4	5
Frec. Impactos	229	211	93	35	7	1

Contrastar la hipótesis de que los datos siguen una distribución de Poisson.

Solución. El número de clases es 6 y conocemos las frecuencias observadas. Para calcular las frecuencias esperadas necesitamos conocer la probabilidad de que un impacto se produzca en cada una de las clases. Dicha probabilidad toma la forma siguiente para una distribución de Poisson

$$p_i = e^{-\lambda} \frac{\lambda^i}{i!}$$

por lo tanto, necesitamos estimar un parámetro λ mediante el principio de máxima verosimilitud, siendo $r = 1$. Hacemos

$$\lambda = \frac{\sum x_j O_j}{\sum O_j} = \frac{535}{576} = 0.929$$

Obtenemos las frecuencias esperadas de las clases:

$$\begin{array}{lll} E_0 = n p_0 = 227.5 & E_1 = n p_1 = 211 & E_2 = n p_2 = 98 \\ E_3 = n p_3 = 30 & E_4 = n p_4 = 7 & E_5 = n p_5 = 1.5 \end{array}$$

La discrepancia entre frecuencias observadas y esperadas es:

$$X^2 = \frac{(229 - 227.5)^2}{227.5} + \frac{(211 - 211)^2}{211} + \dots + \frac{(1.5 - 1)^2}{1} = 1.27$$

El nº de grados de libertad es $k - r - 1 = 4$ y de la tabla de la χ^2 obtenemos que $\chi^2_{0.25}(4) = 1.92$ y $\chi^2_{0.1}(4) = 1.06$. Por lo tanto, $\chi^2_{0.25}(4) > X^2 > \chi^2_{0.1}(4)$, no siendo α pequeño \rightarrow **Aceptamos la hipótesis.**

Contraste Kolmogorov-Smirnov (K-S)

Es válido para distribuciones continuas, pero muy potente. Deseamos contrastar la hipótesis

$$H_0: F_n(x) = F_0(x)$$

1. Ordenamos los valores muestrales de manera que $x_{(1)} \leq x_{(2)} \leq \dots \leq x_{(n)}$
2. Calculamos la función de distribución empírica de la muestra:

$$F_n(x) = \begin{cases} 0 & \text{si } x < x_{(1)} \\ r/n & \text{si } x_{(r)} \leq x < x_{(r+1)} \\ 1 & \text{si } x > x_{(n)} \end{cases}$$

3. Calculamos la máxima discrepancia entre la función de distribución empírica y la teórica contrastada. Estadístico bilateral de K-S

$$D_n = \sup_{x \in R} |F_n(x) - F_0(x)|.$$

La distribución exacta de D_n está tabulada para valores seleccionados de $n \leq 40$ y del nivel de significación α .

Para muestras grandes, se utiliza la distribución asintótica de D_n , que viene dada, para todo $z \geq 0$, por

$$\lim_{n \rightarrow \infty} P((n)^{1/2} D_n \leq z) = L(z)$$

donde $L(z)$ está tabulada y se comprueba que la aproximación es suficientemente buena para $n \geq 35$. Intuitivamente, esperamos que D_n sea pequeño cuando la hipótesis nula es cierta.

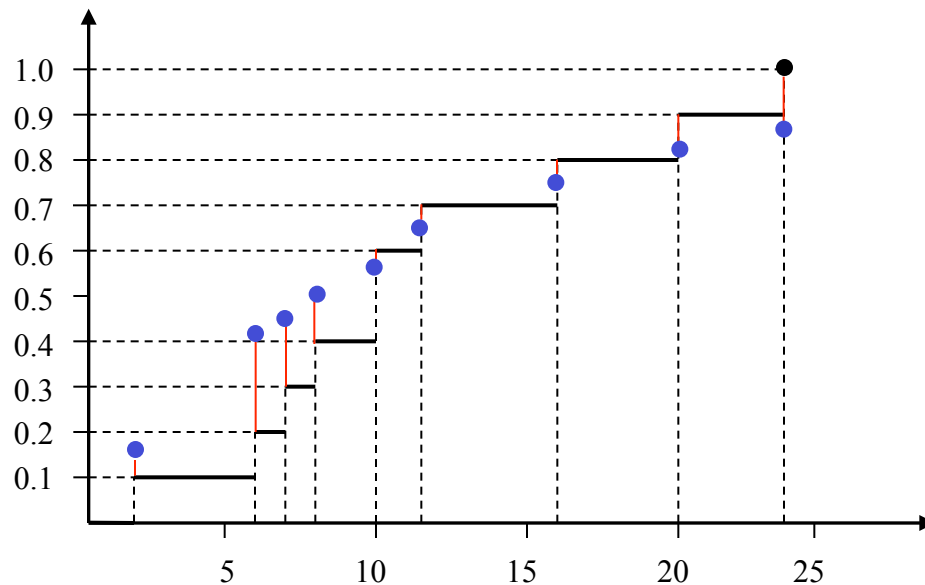
En nuestro caso particular de aleatoriedad, si $x_{(1)} \leq x_{(2)} \leq \dots \leq x_{(n)}$ designa al estadístico de orden, $F_0(x_{(i)}) = x_{(i)}$, y como $F_n(x_{(i)}) = i/n$, resulta

$$D_n = \max_{1 \leq i \leq n} \left\{ \max \left[\left| \frac{i}{n} - x_{(i)} \right|, \left| x_{(i)} - \frac{i-1}{n} \right| \right] \right\}$$

Ejemplo. Contrastar si la siguiente muestra de duraciones de vida puede suponerse exponencial: 16, 8, 10, 12, 6, 10, 20, 7, 2, 24.

Ordenamos la muestra: 2, 6, 7, 8, 10, 10, 12, 16, 20, 24.

Representamos la función empírica:



x	$F_n(x)$	$F_0(x)$	$D_n(x)$
2	0.1	0.16	0.06
6	0.2	0.41	0.21
7	0.3	0.46	0.16
8	0.4	0.5	0.1
10	0.6	0.58	0.02
12	0.7	0.65	0.05
16	0.8	0.75	0.05
20	0.9	0.82	0.08
24	1	0.88	0.12

Función de distribución exponencial

$$F_0(x) = 1 - e^{-\lambda x}$$

Estimamos el parámetro a partir de la media muestral:

$$\frac{16 + 8 + 12 + \dots + 24}{10} = 11.5 = 1/\lambda$$

$D_n = 0.21 < D(\alpha=0.2, n=10) = 0.322$
Aceptamos la hipótesis

Repetición de contrastes

Dada una secuencia de números, para contrastar su **uniformidad** formamos grupos de al menos 40 elementos.

A continuación, para cada uno de los conjuntos formados se calcula el nivel de discrepancia utilizado en el contraste χ^2 .

Finalmente, contrastamos (aplicando, por ejemplo, Kolmogorov-Smirnov) que estos valores provienen de una distribución χ^2 .

$$\underbrace{\underbrace{x_1, x_2, \dots, x_{40}}_{\chi^2_1}, \underbrace{x_{41}, \dots, x_{80}}_{\chi^2_2}, \underbrace{x_{81}, \dots, x_{120}}_{\chi^2_3}, \underbrace{x_{121}, \dots, x_{160}}_{\chi^2_4}, \underbrace{x_{161}, \dots, x_{200}}_{\chi^2_5}, \underbrace{x_{201}, \dots, x_{240}}_{\chi^2_6}, x_{241}, \dots}_{\text{¿proviene de una distribución } \chi^2?}$$

Aleatoriedad o Independencia Estadística

Contraste de Rachas

Dada una secuencia de números x_1, x_2, \dots, x_n .

1. Construimos una sucesión de símbolos binarios, asignando un 1 si $x_i \leq x_{i+1}$ y 0 si $x_i > x_{i+1}$.
2. Definimos como *racha creciente (decreciente)* de longitud l a un grupo de l unos (0's) consecutivos. Contabilizamos el número de rachas, que denotaremos como n_r .

Puede comprobarse que el número total de rachas en una muestra de n observaciones independientes sigue una distribución aproximadamente normal (si $n \geq 40$) con parámetros

$$\mu = \frac{2n-1}{3} \quad \sigma^2 = \frac{16n-29}{90}$$

3. Comprobamos si el número total de rachas observadas puede provenir con una probabilidad razonable de dicha distribución

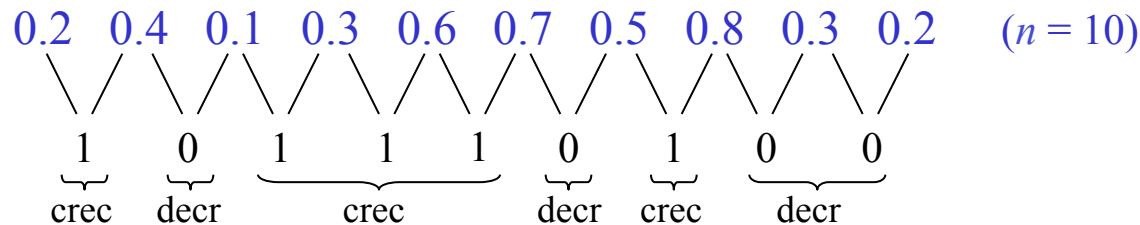
$$P\left(\left|z_{N\left(\frac{2n-1}{3}, \frac{16n-29}{90}\right)}\right| \geq n_r\right)?$$

Los valores críticos vienen tabulados.

Rechazaremos la hipótesis de independencia cuando el número de rachas sea significativamente grande o pequeño (alta dependencia positiva o negativa).

Ejemplo. Contrastar la aleatoriedad de la siguiente secuencia de números: 0.2, 0.4, 0.1, 0.3, 0.6, 0.7, 0.5, 0.8, 0.3, 0.2.

Construimos la secuencia de símbolos binarios para contabilizar las rachas:



El número total de rachas es $n_r = 6$. La distribución del número de rachas en una muestra de tamaño 10 debería ser Normal(6.3, 1.45).

$$\begin{aligned} P(|z_{N(6.3, 1.45)}| \geq 6) &= P(z_{N(6.3, 1.45)} \leq -6) + P(z_{N(6.3, 1.45)} \geq 6) = \\ &P\left(z_{N(0,1)} \leq \frac{-6 - 6.3}{\sqrt{1.45}}\right) + P\left(z_{N(0,1)} \geq \frac{6 - 6.3}{\sqrt{1.45}}\right) = \\ &0 + P(z_{N(0,1)} \geq -0.25) = 1 - P(z_{N(0,1)} \leq -0.25) = \\ &1 - 0.4030 = 0.5987 \end{aligned}$$

La probabilidad teórica de que el número de rachas en la secuencia sea mayor que 6 es 0.5987, por lo que rechazamos la hipótesis de aleatoriedad.

Contraste de rachas por encima y por debajo de la mediana

Otro contraste para calcular rachas se obtiene del recuento de observaciones que se sitúan por encima o por debajo de la mediana (en nuestro caso, la mediana será 0.5). Dada una secuencia de números x_1, x_2, \dots, x_n .

1. Construimos una sucesión de símbolos binarios, asignando un 1 si $x_i \leq \text{mediana}$ y 0 si $x_i > \text{mediana}$.
2. Sea k el número de unos en la sucesión (que será por hipótesis igual al número de ceros) e igual a $(n-1)/2$ si el número de observaciones es impar y no hay observaciones repetidas. Contabilizamos el número de rachas, n_r .

En este caso, el número total de rachas en una muestra de n observaciones independientes sigue una distribución aproximadamente normal (si $n \geq 40$) con parámetros

$$\mu = k + 1 \quad \sigma^2 = \frac{k(k-1)}{2k-1}$$

Test de Póker

Dada una sucesión de números aleatorios en $[0,1]$, x_1, x_2, \dots, x_n :

1. Consideramos la conversión a los números enteros 1, 2, ..., 10 mediante

reemplazar x_j	por	1	si	$0.0 \leq x_j < 0.1$
reemplazar x_j	por	2	si	$0.1 \leq x_j < 0.2$
...
reemplazar x_j	por	10	si	$0.9 \leq x_j < 1.0$

de modo que el nuevo cjto lo será de números enteros aleatorios entre 1 y 10.

2. Tomar los conjuntos formados por grupos de 5 n°s enteros sucesivos $\{x_{5j}, x_{5j+1}, \dots, x_{5j+4}\}$ y determinar para cada uno cuál de los siguientes resultados se cumple:

- Se repite el mismo entero 5 veces (denotado con AAAAA, donde A puede ser cualquier entero del 1 al 10)

- Un entero se repite 4 veces y el otro aparece una sola vez (denotado AAAAB)
- Un entero se repite 3 veces y el otro 2 (denotado AAABB)
- Un entero se repite 3 veces y los otros dos uno cada vez (denotado AAABC)
- Un entero se repite 2 veces, otro entero 2 veces y el tercero 1 vez (denotado AABBC)
- Un entero se repite 2 veces y los otros 3 enteros cada uno una vez (denotado ABCDD)
- Aparecen cinco enteros distintos (ABCDE)

Exactamente una de estas posibilidades ocurrirá para cada grupo de cinco enteros.

Es de esperar que si se da la aleatoriedad, la proporción de veces que puede ocurrir cada posibilidad vendrá dada por la distribución de probabilidad.

$$P(AAAAA) = 0.0001$$

$$P(AAABB) = 0.0090$$

$$P(AABBC) = 0.1080$$

$$P(ABCDE) = 0.3024$$

$$P(AAAAB) = 0.0045$$

$$P(AAABC) = 0.0720$$

$$P(AABCD) = 0.5040$$

El test de póker finaliza con el estudio de la bondad de ajuste a la anterior distribución basándose para ello en el test de la χ^2 , que utilizará el recuento de estas 7 categorías o particiones para el conjunto generado.

Debido a lo baja que es la probabilidad de la categoría AAAAA, se suelen juntar las categorías AAAAA y AAAAB considerándose entonces una partición en 6 conjuntos en vez de 7.

Si no fuera así, ya que el número esperado de ocurrencias de cada categoría debe ser al menos 5, se tendría que para la categoría AAAAA debería ser

$$(0.0001)(n/5) \geq 5,$$

es decir, contrastar $n \geq 250.000$ números aleatorios, lo que significaría un tamaño demasiado grande.

Una versión más simple de este contraste que facilita una programación más sencilla, Knuth (1998), consiste en encontrar para cada conjunto de 5 enteros su número de valores distintos.

Así, tendríamos 5 categorías:

- 5 valores: todos diferentes
- 4 valores: un par
- 3 valores: dos pares o tres iguales
- 2 valores; cuatro iguales
- 1 valor: los cinco iguales

Esta clasificación es más sencilla de obtener de manera sistemática, siendo el contraste con calidad próxima al de póker.

Otros contrastes que se pueden utilizar para la aleatoriedad son el [contraste de permutaciones](#), el [contraste de huecos](#) y el [contraste de Box-Pierce](#).

Ejemplo. Dada la siguiente muestra de 1500 números, realizar el contraste de Póker de aleatoriedad:

0.12	0.23	0.78	0.84	0.16	0.22	0.29	0.47	0.95	0.13	0.33	0.27	0.42	0.32	0.69	0.72	0.17	0.22	0.09	0.81	...
↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	...
2	3	8	9	2	3	3	5	10	2	4	3	5	4	7	8	2	3	1	9	...
AABCD					AABCD					AABCD					ABCDE					...

Clases	Frec. observadas	Frec. Esperadas
AAAAA	3	$n \times p_{AAAAA} = 300 \times 0.0001 = 0.3$
AAAAB	5	$n \times p_{AAAAB} = 300 \times 0.0045 = 1.35$
AAABB	5	$n \times p_{AAABB} = 300 \times 0.0090 = 2.7$
AAABC	25	$n \times p_{AAABC} = 300 \times 0.0720 = 21.6$
AABBC	27	$n \times p_{AABBC} = 300 \times 0.1080 = 32.4$
AABCD	170	$n \times p_{AABCD} = 300 \times 0.5040 = 151.2$
ABCDE	65	$n \times p_{ABCDE} = 300 \times 0.3024 = 90.72$

$$X^2 = \frac{(8-1.65)^2}{1.65} + \frac{(5-2.7)^2}{2.7} + \frac{(25-21.6)^2}{21.6} + \dots + \frac{(65-90.72)^2}{90.72} = 24.43 + 1.959 + 0.5351 + \dots + 7.314 = 37.46$$

El número de grados de libertad es $k-r-1 = 6-0-1 = 5$ y de la tabla de la distribución de la χ^2 tenemos $\chi^2_{0.005}(6) = 18.5 < X^2 = 37.46$, por lo tanto, **rechazamos la hipótesis**.

ÍNDICE

1. Introducción
2. Contrastes empíricos
 - **Uniformidad** (Contraste χ^2 y Kolmogorov-Smirnov)
 - **Aleatoriedad** (Rachas, Test de Póker)
- 3. **Generadores congruenciales**
4. Otros generadores
 - **Registro de desplazamiento**
 - **Fibonacci retardados**
 - **No lineales**
 - **Combinación de generadores**
 - **Generadores paralelos**
 - **Generadores comerciales**
5. Conclusiones

3. Generadores congruenciales

Los generadores más populares son probablemente los **congruenciales**, debidos a Lehmer (1951). Siguen la fórmula de recursión:

$$x_{n+1} = (a x_n + b) \bmod m,$$

que proporciona números enteros en $[0, m)$, para un **multiplicador** a , **sesgo** b , **módulo** m y **semilla** x_0 , donde suponemos que $a, b \in \{0, 1, \dots, m-1\}$. La utilización de la misma semilla llevará a la misma secuencia de n°s aleatorios (reproducibilidad y mutabilidad).

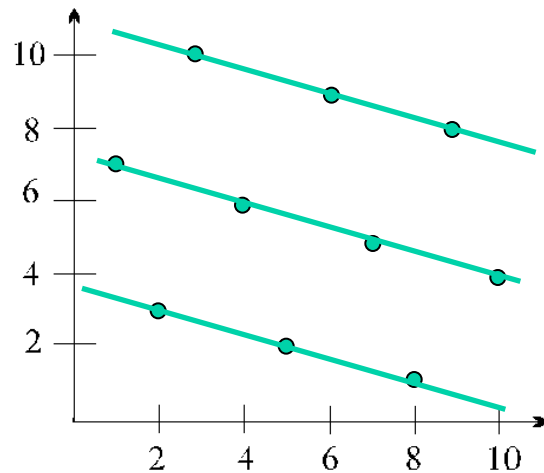
Se convierten en números uniformes en $[0,1)$ dividiendo por m , $u_n = x_n / m$.

Si $b = 0$, se denominan **multiplicativos**: $x_{n+1} = a x_n \bmod m$,

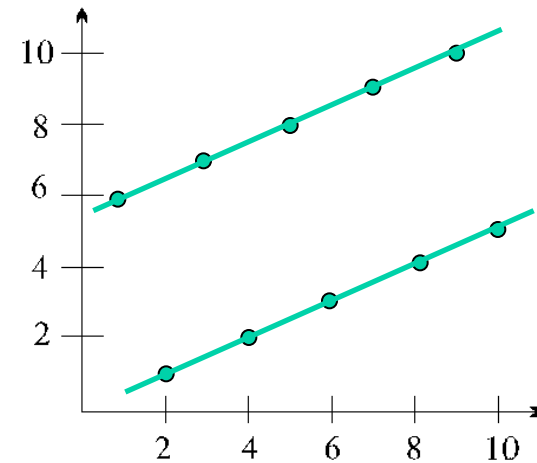
A pesar de su aparente simplicidad y previsibilidad, una selección cuidadosa de los parámetros (a, b, m) permite obtener de manera eficiente sucesiones de números suficientemente largas y aleatorias para muchos propósitos.

Consideremos los generadores multiplicativos ($b = 0$):

Caso	a	m	x_0	Salidas													
1	6	13	1	6	10	8	9	2	12	7	3	5	4	11	1		
2	7	13	10	5	9	11	12	6	3	8	4	2	1	7	10		
3	5	13	5	12	8	1	5	12	8	1	5	12	8	1	5		
4	7	11	5	2	3	10	4	6	9	8	1	7	5	2	3		
5	6	11	3	7	9	10	5	8	4	2	1	6	3	7	9		



caso 4



caso 5

Algunas observaciones son:

- 1) Un generador congruencial tiene ciclos.
- 2) La longitud del ciclo depende de la selección de los parámetros (comparar casos 1 y 3).
- 3) Dentro de selecciones de parámetros que conducen a una misma longitud de ciclo, algunas salidas parecen más aleatorias que otras (comparar casos 1 y 2).
- 4) La representación de los pares (x_i, x_{i+1}) sugiere que éstos se disponen en un número finito de rectas (representación de tuplas \rightarrow hiperplanos).

Trivialmente, el periodo máximo m se alcanza para $a = b = 1$.

Además, cuando $b = 0$ (generador multiplicativo) el máximo periodo es $m-1$.

El periodo depende de la selección de a y b , para lo que se dispone de algunos resultados que indicamos aquí:

Proposición: Un generador congruencial tiene periodo máximo m si y sólo si:

- 1) $\text{mcd}(b, m) = 1$;
- 2) $a \equiv 1 \pmod{p}$ para cada factor primo p de m ;
- 3) $a \equiv 1 \pmod{4}$ si 4 divide a m .

Puesto que b tiene en la práctica el **efecto de una traslación**, estamos interesados en estudiar resultados para generadores multiplicativos.

El valor de m se escoge para que la operación $Y \bmod m$ se realice de forma sencilla. Para un ordenador binario, si $m = 2^\beta$, lo único que debemos hacer para ejecutar tal operación es retener los últimos β bits de Y . Entonces:

Proposición. Un generador multiplicativo con módulo $m = 2^\beta \geq 16$ tiene periodo máximo $m/4$ si y sólo si $a \bmod 8 = 3$ ó $a \bmod 8 = 5$ y x_0 es impar.

También hay procedimientos sencillos para el caso $m = 2^\beta + 1$ (Ripley, 1987).

Proposición. Un generador multiplicativo tiene periodo $m-1$ sólo si m es primo. El periodo divide a $m-1$ y es $m-1$ si y sólo si a es una raíz primitiva de $m-1$, es decir, $a \neq 0$, $a^{(m-1)/p} \not\equiv 1 \pmod{m}$, para todos los factores primos p de $m-1$.

Dos elecciones frecuentes son $m = 2^{31}-1$, en ordenadores de 32 bits, y $m = 2^{16}+1$, en ordenadores de 16 bits.

Mencionemos que se conocen condiciones computables para determinar raíces primitivas.

De hecho, una vez encontrada una, el resto proviene de:

Proposición. Si a es una raíz primitiva de m , $a^k \pmod{m}$ lo es siempre que $\text{mcd}(k, m-1) = 1$.

Entonces, podemos elegir m para que la implementación del generador sea eficiente y, una vez elegido m , podemos elegir a y b en su caso, para que el generador sea de ciclo máximo.

Debemos aún estudiar con más detalle tal elección, pues tenemos muchos grados de libertad. Para motivar la elección, apelamos a nuestra cuarta observación: un resultado de Marsaglia (1968) demuestra que “sucesiones solapantes de n números de un generador multiplicativo caen en, a lo sumo, $(n!m)^{1/n}$ hiperplanos paralelos”. ([Estructura reticular](#) de un generador)

Por ejemplo, tenemos las siguientes cotas en un par de casos representativos

	$n = 3$	$n = 5$	$n = 7$	$n = 9$	$n = 10$
$m = 2^{16}$	73	23	16	14	13
$m = 2^{32}$	2953	220	80	48	41

Los ejemplos que utilizamos como introducción muestran lo pésima que puede ser la estructura reticular de estos generadores.

Un ejemplo famoso es el de la rutina RANDU, que IBM proporcionaba a mediados de los 70, con $m = 2^{31}$, $a = 65539$ y $b = 0$. ¡Para este generador, las tripletas consecutivas de números caen en 15 planos!

La **estructura reticular** de estos generadores sugiere cierta previsibilidad en su salida, lo que va en contra de la idea de aleatoriedad.

Una posibilidad es, entonces, elegir las constantes del generador para mitigar el problema de la estructura reticular. Tal idea conduce al **contraste espectral**, basado en “calcular la distancia máxima entre los hiperplanos adyacentes, tomándose el máximo sobre el conjunto de todos los hiperplanos paralelos que cubran los puntos”. Intuitivamente, cuanto mayor es la distancia, peor es el generador.

La idea se ilustra con nuestros casos 4 y 5. En el ejemplo, la salida del generador con $a = 7$ parece más aleatoria que la del generador con $a = 6$. En el primer caso, la máxima distancia es 3.479; en el segundo, es 4.919.

Para llevar a cabo el contraste, para k -úplas, hemos de resolver el problema de optimización

$$V_k = \min(\alpha_1^2 + \alpha_2^2 + \dots + \alpha_k^2)^{1/2}$$

s.a

$$\alpha_1 + a\alpha_2 + \dots + a^{k-1}\alpha_k = mq$$

$$(\alpha_1, \alpha_2, \dots, \alpha_k, q) \in \mathbf{Z}$$

El valor del contraste espectral es m/V_k , véase Ripley (1987) para una demostración.

Tal problema es de programación entera no lineal, y para su resolución eficiente hemos de apelar a diversos heurísticos, como se expone en Knuth (1981).

Resulta, sin embargo, muy costoso para valores de k mayores que 6.

Generador congruencial estándar

Arquitecturas de 32 bits, Park y Miller (1988) → 3 criterios para elegir un generador multiplicativo → mínimo estándar:

- 1) Ser de periodo máximo;
- 2) Que su salida parezca aleatoria;
- 3) Que se pueda implementar de forma eficiente en aritmética de 32 bits.

Los generadores con $m = 2^a$ presentan el inconveniente de la escasa aleatoriedad de los bits de menor orden.

Elección en boga por la facilidad de implementación y por ser primo es $m = 2^{31} - 1$. Los 3 criterios limitan la elección de multiplicadores.

El multiplicador que se ha convertido en estándar es el asociado a $a = 7^5 = 16807$. 7 es una raíz primitiva de $2^{31} - 1$, por lo que se puede probar que el generador es de periodo máximo.

Generador congruencial estándar en FORTRAN

Para $a = 7^5 = 16807$ y $m = 2^{31} - 1 = 2147483647$ una primera posibilidad es

```
function aleator (isemilla)  
  isemilla = mod (isemilla × 16807.d0, 2147483647.d0)  
  aleator = isemilla / 2147483647  
  return  
end
```

Puede dar problemas de desbordamiento pues $isemilla \times 16807.d0$ puede valer hasta 1.03×2^{45} . Una forma de comprobar el correcto funcionamiento del generador consiste en comenzar con una *isemilla* inicial igual a 1 y comprobar que después de 10.000 llamadas *isemilla* toma el valor 1043618065.

Método de Schrage (Bratley, Fox y Schrage, 1983). Su idea básica es realizar la operación de módulo antes que el producto.

Supongamos que escribimos $m = aq + r$
con

$$q = m \operatorname{div} a \quad y \quad r = m \bmod a$$

Entonces tenemos

$$\begin{aligned} f(z) &= az \bmod m = az - m (az \operatorname{div} m) + m (z \operatorname{div} q) - m (z \operatorname{div} q) \\ &= \gamma(z) + m \delta(z) \end{aligned}$$

con

$$\begin{aligned} \gamma(z) &= a (z \bmod q) - r (z \operatorname{div} q) \\ \delta(z) &= (z \operatorname{div} q) - (az \operatorname{div} m) \end{aligned}$$

Se puede demostrar que si $r < q$ para todo $z \in \{0, 1, \dots, m-1\}$ resulta

- 1) $\gamma(z) \in \{0, 1\}$
- 2) $a (z \bmod q), r (z \operatorname{div} q) \in \{0, 1, \dots, m-1\}$
- 3) $|\gamma(z)| \leq m-1$

Así, puesto que $1 \leq f(z) \leq m-1$, resulta

$$\begin{aligned} \gamma(z) &= 0 \text{ si y solo si} & 1 \leq \gamma(z) \leq m-1 \\ \gamma(z) &= 1 \text{ si y solo si} & -(m-1) \leq \gamma(z) \leq -1 \end{aligned}$$

Por tanto, para evaluar $f(z)$, evaluamos $\gamma(z)$. Si $\gamma(z) > 0$, hacemos $f(z) = \gamma(z)$; en caso contrario $f(z) = \gamma(z) + m$.

Por ejemplo, para $a = 16807$ y $m = 2147483647$, tenemos $q = 127773$ y $r = 2863$, y la implementación en FORTRAN queda de la siguiente forma:

```
function aleato2 (isemilla)
  k = isemilla/127773
  isemilla = 16807 × (isemilla - k × 127773) - 2836 × k
  if (isemilla.lt0) isemilla = isemilla + 2147483647
  aleato2 = 1./2147483647 × isemilla
  return
end
```

Park y Miller (1988) indican que 23093 multiplicadores permiten la implementación en 32 bits, y entre ellos está 16807, como hemos visto.

Fishman y Moore (1986) utilizaron el **contraste espectral** para seleccionar buenos multiplicadores.

Criterio: si para $2 \leq k \leq 6$ y cada conjunto de hiperplanos paralelos, la distancia euclídea entre hiperplanos adyacentes no supera la mínima distancia alcanzable en más del 25%.

Identificaron así 410 multiplicadores, que no incluyen a los 23093 anteriores.

Fishman realizó una búsqueda exhaustiva de los 23093 anteriores con un criterio del 30%, encontrando varios buenos multiplicadores, entre ellos 16807, aunque, por ejemplo, $a = 48271$ y $a = 69621$ parecen ser mejores. Sin embargo, dada la mayor experiencia con $a = 16807$ aún se suele proponer tal multiplicador como el estándar.

Para una discusión más detallada véase Park y Miller (1988) y Press *et al* (1992).

A pesar de estas buenas propiedades, se han diseñado contrastes muy potentes que permiten rechazar la aleatoriedad de los generadores congruenciales (L'Ecuyer, 1994).

Tales contrastes explotan la estructura reticular de estos generadores.

Por ello y por la necesidad de disponer de generadores de periodo más largo, describimos algunos otros intentos de definir buenos generadores.

ÍNDICE

1. Introducción
2. Contrastes empíricos
 - **Uniformidad** (Contraste χ^2 y Kolmogorov-Smirnov)
 - **Aleatoriedad** (Rachas, Test de Póker)
3. Generadores congruenciales
- 4. Otros generadores
 - **Registro de desplazamiento**
 - **Fibonacci retardados**
 - **No lineales**
 - **Combinación de generadores**
 - **Generadores paralelos**
 - **Generadores comerciales**
5. Conclusiones

4. Otros generadores

4.1. Generadores de Registro de Desplazamiento

Los generadores congruenciales pueden **generalizarse** a recursiones lineales de orden mayor. Para $k \geq 1$, m primo

$$x_n = (a_1 x_{n-1} + \dots + a_k x_{n-k}) \bmod m$$

y el generador se denomina **recursivo múltiple**. (L'ecuyer, 1994)

Su estudio se asocia al estudio del polinomio característico

$$P(z) = z^k - a_1 z^{k-1} - \dots - a_k$$

sobre el álgebra finita F_m con m elementos.

Cuando el polinomio es **primitivo**, Niederreiter (1992), el periodo del generador es $m^k - 1$. El estudio de tales polinomios y la implementación del generador cuando m es grande no es sencillo, por lo que se escoge m pequeño, habitualmente 2, dado que permite una sencilla implementación.

Parece ser que estos generadores tienen buenas propiedades de aleatoriedad, sin embargo, producen estructuras reticulares, como los congruenciales, lo que ha llevado a una cierta polémica y disuasión sobre su calidad.

Ejemplo. Tomando $m = 2$ tenemos el generador $x_n = (a_1x_{n-1} + \dots + a_kx_{n-k}) \bmod 2$, donde $a_1, \dots, a_k \in \{0, 1\}$, que es equivalente a reescribir

$$x_n = x_{n-j_1} \text{ XOR } x_{n-j_2} \text{ XOR } \dots \text{ XOR } x_{n-j_k},$$

cuando $a_{n-j_1} = a_{n-j_2} = \dots = a_{n-j_k} = 1$ y $a_j = 0$ para los otros j .

Una elección habitual es escoger polinomios de la forma $P(z) = z^k - z^{k-r} - 1$ o equivalentemente $x_n = x_{n-r} \text{ XOR } x_{n-k}$

Valores habituales para k y n (garantizan un periodo máximo):

$$\left\{ \begin{array}{l} k = 607 \quad r = 273 \\ x_n = x_{n-273} \text{ XOR } x_{n-607} \end{array} \right\} \text{ o } \left\{ \begin{array}{l} k = 521 \quad r = 32 \\ x_n = x_{n-32} \text{ XOR } x_{n-521} \end{array} \right\}$$

A partir de la sucesión de bits pueden generarse números en $(0,1)$. La propuesta más utilizada son los denominados **generadores de Tausworthe** (1965):

Para un l y t dados, se define

$$u_i = \sum_{s=1}^l 2^{-s} b_{it+s}$$

Cuando $\text{mcd}(2^k-1, t)=1$ se denominan **proprios** y tienen periodo 2^k-1 . Estos generadores tienen buenas propiedades en el sentido de su aleatoriedad aunque producen estructuras reticulares similares a los de los generadores congruenciales.

Uno de los principales generadores actuales perteneciente a esta clase (de registro de desplazamiento) es el **girador de Mersenne** (Mersenne twister method, Matsumoto & Nishimura, 1998), que goza de buenas propiedades estadísticas.

4.2. Generadores de Fibonacci Retardados

Se trata de otra generalización de los generadores congruenciales y parten de la semilla inicial (x_1, x_2, \dots, x_r) y usan la recursión $x_i = x_{i-r} \circ x_{i-s}$, donde r y s son retardos enteros que satisfacen $r > s$ y \circ es una operación binaria que suele ser $+$, $-$, \times o XOR.

Típicamente, los elementos iniciales son enteros y la operación binaria es la suma módulo 2^n .

La caracterización del periodo máximo de los generadores de Fibonacci retardados se basa en el [análisis de sucesiones lineales recursivas de enteros](#) de la forma

$$x_i = (a_1 x_{i-1} + a_2 x_{i-2} + \dots + a_r x_{i-r}) \bmod m$$

Marsaglia (1985). Cuando $r = 17$, $s = 5$ y se escoge la operación $+$ ó $-$ sobre los enteros mod 2^n , el periodo es $(2^{17}-1)2^{n-1}$. Cuando es XOR, el periodo es $2^{17}-1$.

La implementación de un generador de este tipo es sencilla mediante una lista circular y dos punteros.

4.3. Generadores no lineales

Dada la estructura reticular de los generadores lineales, se sugiere utilizar generadores no lineales. Se distinguen dos formas de introducir no linealidad:

1. Usar un generador con función de transición lineal, produciendo la salida mediante una transformación no lineal del estado.
2. Usar un generador con función de transición no lineal.

No producen una estructura reticular como la de los lineales. Su estructura es altamente no lineal: típicamente, un hiperplano t -dimensional tendrá a lo sumo t t -uplas solapantes de números.

Niederreiter (1992) proporciona estudios teóricos sobre su estructura que sugiere sus buenas propiedades teóricas. Sin embargo, no existen aún implementaciones rápidas con parámetros bien contrastados, aunque el método que parece más prometedor, véase Eichenauer-Herrmann (1995), es el denominado **método congruencial de inversión explícita**.

Método congruencial de inversión explícita

Sea $m \geq 5$ un número primo y $F_m = \{0, 1, \dots, m-1\}$ el álgebra finita de orden m .

Para un entero z , se define $\bar{z} \in F_m$,

$$\bar{z} \equiv z^{m-2} \pmod{m}$$

que es la inversa de z para la multiplicación en F_m , si $\bar{z} \neq 0 \pmod{m}$.

Existen algunos algoritmos eficientes para su cálculo.

Dados $a, b \in F_m$, $a \neq 0$, la sucesión es

$$y_n = \overline{a n + b} \quad n \geq 0$$

Su periodo máximo es m .

No hay tanta experiencia en los generadores no lineales como con los lineales.

Entre los no lineales ha emergido como principal opción el algoritmo [Advance Encryption Standard](#) (AES), Hellekalek y Wegenkittl (2003).

4.4. Combinación de generadores

Para incrementar el periodo e intentar evitar las regularidades que muestran los generadores lineales congruenciales se ha sugerido combinar diferentes generadores para obtener uno **híbrido**, que sea de mayor calidad que los anteriores. Tales combinaciones pueden considerarse **heurísticas**, algunas con resultado bastante pobre.

El fundamento es esencialmente empírico, aunque también se han desarrollado algunos aspectos teóricos.

1. Se ha observado que el periodo de un generador híbrido es, en general, bastante más largo que el de sus componentes siendo, además, posible su determinación.
2. Hay resultados teóricos que sugieren que algunas formas de combinación de generadores mejoran su comportamiento estadístico. Por ejemplo, consideremos dos sucesiones aleatorias (x_n) , (y_n) y la sucesión combinada elemento a elemento de ambas (z_n) , donde $z_n = x_n \circ y_n$ y \circ representa alguna operación binaria.

Ideas básicas para combinar generadores son el **barajeo** y **composición**.

Un análisis teórico de esta familia se ha llevado a cabo con esquemas bastante simples, por lo que no hay garantía de suficiente calidad, aunque algunos contrastes sugieren potencialidad de estas ideas.

Permutación fija (barajeo): Se generan los números en bloques de longitud L y se aplica una permutación fija a cada bloque antes de su uso.

Permutación aleatoria (barajeo): (MacLaren y Marsaglia, 1965): consiste en aplicar una permutación aleatoria a una sucesión. Si tenemos dos sucesiones aleatorias (u_n) , (v_n) y $T(0), \dots, T(k-1)$ están inicialmente ocupados por los valores u_1, \dots, u_k , en cada etapa utilizamos v_n para seleccionar aleatoriamente un elemento de T : hacemos $J = \text{ent}(kv_{k+1})$, salimos con $T(J)$ y lo reemplazamos con u_{n+1} .

Ejemplo (permutación aleatoria, barajeo)

Partimos de dos sucesiones aleatorias:

$(u_n) \rightarrow 0.7 \ 0.2 \ 0.32 \ 0.84 \ 0.25 \ 0.12 \ 0.33 \ 0.47 \ 0.84 \ 0.72 \ \dots$

$(v_n) \rightarrow 0.1 \ 0.7 \ 0.23 \ 0.42 \ 0.35 \ 0.21 \ 0.47 \ 0.72 \ 0.68 \ 0.12 \ \dots$

Tomamos $k = 5$ y hacemos

$T(0) = u_1 = 0.7 \quad T(1) = u_2 = 0.2 \quad T(3) = 0.32 \quad T(4) = 0.84 \quad T(5) = 0.25$

Tomamos valores de la segunda secuencia aleatoria y construimos los índices que me señalarán los elementos de T que debemos devolver, a la vez que actualizo T :

$v_1 = 0.1 \rightarrow j = \text{ent}(6 \times 0.1) = 0.6 \rightarrow j = 0 \rightarrow$ el valor que debemos devolver es $T(0) = 0.7$. Hago $T(0) = u_6 = 0.12$

$v_2 = 0.7 \rightarrow j = \text{ent}(6 \times 0.7) = 4.2 \rightarrow j = 4 \rightarrow$ el valor que debemos devolver es $T(4) = 0.84$. Hago $T(1) = u_7 = 0.33$

....

L'Ecuyer (1988) describe la **composición** de generadores congruenciales lineales multiplicativos de orden $k = 1$ con módulos primos distintos m_1, \dots, m_I . Si x_{in} denota el estado del generador i -ésimo en la etapa n , se define la combinación $z_n = (\sum_{i=1}^I \alpha_i x_{in}) \bmod m_1$ para algunos enteros α_i , que suelen tomar la forma $(-1)^{i-1}$.

$$\begin{array}{ccc} x_{n+1}^1 = a x_n^1 \bmod m_1 & \dots & x_{n+1}^I = a x_n^I \bmod m_I \\ \Downarrow & & \Downarrow \\ x_{n+1}^1 & \dots & x_{n+1}^I \\ \underbrace{\hspace{10em}} & & \\ z_{n+1} = (\sum_{i=1}^I \alpha_i x_{n+1}^i) \bmod m_1 & & \end{array}$$

Algunos de los mejores generadores disponibles hoy en día corresponden a la composición de generadores múltiples recursivos, L'Ecuyer (2006), uno de los más empleados es el denominado **MRG32k3a**.

Otros generadores basados en la idea de **barajeo**:

- **Método de Bays-Durham** (1976)

Otros generadores basados en la idea de **composición**:

- **Algoritmo Super-Duper-73** (Marsaglia et al., 1973): Combinación XOR de un generador lineal congruencial ($m=2^{32}$ y $a=69069$) con un pequeño generador de registro de desplazamiento.
- **Algoritmo Super-Duper-64** (Marsaglia 2002): Generador 64 bits mediante una combinación aditiva de un generador lineal congruencial con un LFSR (linear feedback shift-register).
- **Algoritmo KISS** (Marsaglia, 1995): Composición de un generador lineal congruencial y uno de registro de desplazamiento.
- **Generador de Wichmann-Hill** (1982, 1984 corrections)

4.5. Generadores Paralelos

La migración hacia arquitecturas paralelas van haciendo necesaria una adaptación y una revisión de los procedimientos actuales de generación de n°s aleatorios.

La forma más simple y obvia de generar números aleatorios con procesamiento en paralelo es utilizar **un solo generador fuente**.

La situación más sencilla se da cuando se considera un **punto único de comienzo** para ese generador, que proporcionará números a todos los procesadores que los necesiten. Hay dos inconvenientes:

1. A menos que haya una sincronización en la generación de los números aleatorios no será posible garantizar que todos los procesos particulares reciban siempre la misma secuencia exacta de números.
2. Para sistemas paralelos basados en "paso de mensajes" puede haber un aumento considerable de gasto en la transmisión de los números generados a los procesos que los necesitan.

Una forma sencilla de crear generadores separados para conjuntos de procesos consiste en utilizar **puntos de comienzo predeterminados**, es decir, utilizar un único generador común con un conjunto de semillas preseleccionadas para cada proceso.

La ventaja de este procedimiento es que todos los procesos usan el mismo código, aunque las sucesiones generadas y utilizadas por cada uno serán distintas.

4.6. Generadores Comerciales

Artículo de Park y Miller (1988) como advertencia sobre la mala calidad de algunos generadores comerciales.

IMSL implementa generadores multiplicativos de módulo $m = 2^{31}-1$ y multiplicadores $a = 16807, 397204094$ y 950706376 .

El lenguaje de simulación **SIMSCRIPT II.5** implementa el mismo tipo de generador con multiplicador $a = 630360016$, proporcionando semillas suficientemente separadas para producir sucesiones independientes.

El entorno estadístico **S-PLUS** implementa el **algoritmo Super-Duper** de Marsaglia basado en un generador multiplicativo y un generador de Tausworthe.

SPSS implementa el **girador de Mersenne (Mersenne twister method)**.

La biblioteca estándar de [Unix](#) utiliza el generador congruencial:

$$x_{n+1} = (1140671485x_n + 12820163) \bmod 2^{24},$$

$$u_n = x_n / 2^{24}.$$

[Java](#), en su clase `java.util.Random`, utiliza el mismo generador pero sale con

$$u_n = (2^{27} \lfloor x_{2n} / 2^{22} \rfloor + \lfloor x_{2n+1} / 2^{21} \rfloor) / 2^{53}.$$

En [Visual Basic](#) se emplea el generador congruencial

$$x_{n+1} = (1140671485x_n + 12820163) \bmod 2^{24},$$

$$u_n = x_n / 2^{24}.$$

En [Excel-97](#) se emplea el generador

$$u_{n+1} = (9821.0u_n + 0.211327) \bmod 1,$$

mientras que en [Excel-03](#) y [Excel-07](#) se emplea el [generador de Wichman-Hill](#).

Press *et al* (1992) incluyen, entre otros, el [generador mínimo estándar](#), el [generador mínimo estándar barajado según el método de Bays-Durham](#) y el [generador de L'Ecuyer](#).

Press *et al* (2007) se incluyen generadores para las aritméticas de 32 y 64 bits. En ambos casos se utiliza una combinación de los generadores: el [método Xorshift](#) (Marsaglia, 2003); el [método MWC](#) (Multiply With Carry, Couture y L'Ecuyer, 1997) con base $b = 2^{32}$; un generador congruencial lineal módulo 2^{64} y un generador congruencial lineal multiplicativo módulo 2^{64} .

Las combinaciones de estos generadores básicos se realizan mediante operaciones XOR, desplazamiento de bits a izquierda o derecha, la suma de valores obtenidos de diferentes generadores o la introducción de la secuencia de números aleatorios generada por uno de ellos como entrada en otro.

Las revistas [Communications of ACM](#) y [Applied Statistics](#) publican a menudo nuevos algoritmos de generación de números aleatorios.

ÍNDICE

1. Introducción
2. Contrastes empíricos
 - **Uniformidad** (Contraste χ^2 y Kolmogorov-Smirnov)
 - **Aleatoriedad** (Rachas, Test de Póker)
3. Generadores congruenciales
4. Otros generadores
 - **Registro de desplazamiento**
 - **Fibonacci retardados**
 - **No lineales**
 - **Combinación de generadores**
 - **Generadores paralelos**
 - **Generadores comerciales**
- 5. Conclusiones

6. Conclusiones

Se ha hecho una breve descripción de los **contrastes de aleatoriedad**. Los principales contrastes vienen recogidos en sitios como el del **NIST** (National Institute of Standards and Technology) <http://csrc.nist.org/rng>, la librería **TestU01** (L'Ecuyer y Simard, 2007), <http://www.iro.umontreal.ca/~simardr/testu01/tu01.html> y la **DIEHARD** de Marsaglia disponible en <http://www.stat.fsu.edu/pub/diehard/>.

Hemos proporcionado las ideas básicas sobre los principales **generadores** utilizados en la actualidad: lineales congruenciales y de multiplicativos, así como de sus combinaciones.

Tales tipos de generadores destacan porque su teoría está bien desarrollada, se han contrastado en profundidad sus implementaciones y se utilizan en gran medida, siendo los de mayor calidad **MT19937** y **MRG32k3a**.

Algunos generadores no lineales, como **Advance Encryption Standard (AES)**, son también prometedores.

Como **recomendación general**, sería deseable contrastar el generador que se va a usar en una aplicación.

Dos sitios excelentes y actualizados con abundante información sobre números aleatorios son las páginas de Hellekalek <http://random.mat.sbg.ac.at> y de L'Ecuyer <http://www.iro.umontreal.ca/~lecuyer>.

Hay varios sitios que proporcionan generadores verdaderos de números aleatorios.

- <http://www.random.org>, que usa ruido atmosférico;
- <http://random.hd.org/index.html> (Java EntropyPool), que emplea ruido proveniente de entradas en distintos sitios web y otros dispositivos físicos.

Las revistas *Communications of ACM* y *Applied Statistics* publican a menudo nuevos algoritmos de generación de números aleatorios. En particular, los de esta última, entre otra riquísima y abundantísima información, pueden obtenerse de Statlib en <http://lib.stat.cmu.edu/>. La librería científica GNU tiene una sección entera dedicada a números aleatorios [http://www.gnu.org/software/gsl/manual/html\\$\\$_\\$node/Random-Number-Generation.html](http://www.gnu.org/software/gsl/manual/html$$_$node/Random-Number-Generation.html).