# Capítulo 3

# Generación de variables aleatorias

## 3.1. Introducción

Un elemento clave para el desarrollo de un simulador es la generación de valores aleatorios para variables con una determinada distribución de probabilidad (*variables aleatorias*). La introducción de variables aleatorias en los modelos de simulación permiten recrear situaciones que no están completamente controladas, o que dependen de algún factor aleatorio. Por ejemplo, como vimos en temas anteriores, el tiempo entre llegadas de clientes en una estación de servicio suele modelarse como una distribución exponencial.

Un generador de valores para una variable aleatoria se consigue en dos pasos. Primero tenemos que definir un generador de números aleatorios distribuidos uniformemente entre 0 y 1. Después, transformaremos esta secuencia para generar los valores de la variable aleatoria deseada.

Un requerimiento importante que deben cumplir los generadores de números aleatorios es que las series generadas sean *reproducibles*, de tal modo que en distintos experimientos podamos utilizar exactamente la misma secuencia de números aleatorios para reproducir las mismas condiciones.

En la actualidad, prácticamente todos los lenguajes de programación proveen alguna función para generar números aleatorios. Aunque podríamos pensar que estos son suficientes para realizar nuestras simulaciones, es necesario que nos aseguremos de la buena calidad de estos generadores. El empleo de un generador que produzca series con fuertes dependencias o correlacciones puede conducir a simulaciones incorrectas y a conclusiones totalmente falsas.

## 3.2. Generadores de números aleatorios

Un generador de números aleatorios consiste en una función que devuelve los valores de una secuencia de números reales, $(u_1, u_2, \ldots, u_n)$, donde cada $u_i \in [0, 1]$. El primer elemento de la secuencia se le denomina *semilla inicial* de la serie, y a partir de ella debe ser posible generar el resto de la secuencia para que ésta sea *reproducible*.

Entre las propiedades que debe cumplir un buen generador para realizar simulaciones, destacaríamos las siguientes:

- Los valores $u_i$ deben ser independientes y estar idénticamente distribuidos (IID).
- La secuencia generada debe ser bastante larga, con el fin de permitir simulaciones largas y/o con muchas variables aleatorias.

- Debe computarse muy eficientemente, ya que cada simulación podría requerir la generación de una cantidad considerable de números aleatorios, del orden de millones.

La condición de uniformidad debe cumplirse además para todas las subsecuencias de tamaño $k$ de la secuencia generada. Es decir:

- los $u_i$ deben distribuirse uniformemente en $[0,1]$
- los pares $(u_i, u_{i+1})$ deben distribuirse uniformemente en el plano $[0,1] \times [0,1]$
- los tríos $(u_i, u_{i+1}, u_{i+2})$ deben distribuirse uniformemente en el cubo $[0,1]^3$
- etc

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 ($T_{llegadas}$) y los tiempos de servicio ($T_{servicio}$) debemos 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. Fíjate que el valor máximo de $k$ dependerá del modelo de simulación.

En la siguiente sección vamos a describir los generadores de números aleatorios basados en congruencias, los cuales son los más utilizados en simulación.

### 3.2.1. Generadores Congruenciales Lineales (GCL)

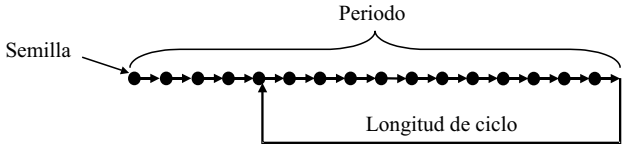
Los generadores congruenciales fueron descubiertos por Lehmer en 1951 cuando observó que los residuos de las potencias sucesivas de un número sugieren un comportamiento aleatorio. La primera propuesta de Lehmer consistió en la siguiente secuencia:

$$x_n = a^n \, mod \, m$$

Es decir, el número $n$-ésimo de la secuencia se obtiene dividiendo la potencia $n$-ésima de un entero $a$ por un entero $m$ y tomando el resto. La expresión anterior es equivalente a la siguiente:

$$x_n = a\, x_{n-1} \, mod \, m$$

De este modo se puede obtener cada elemento de la secuencia a partir del elemento anterior, y en primer lugar de un valor inicial $x_0$ denominado *semilla*.




Muchos de los generadores propuestos actualmente son una generalización del propuesto por Lehmer, donde se incluye un sesgo $b$ en la expresión anterior:

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

La secuencia así obtenida consiste en una serie de números enteros comprendidos entre 0 y $m-1$. Las parámetros del generador $a$, $b$ y $m$ deben de ser números enteros positivos. Por otro lado, cuando $b = 0$ diremos que el generador es *multiplicativo*.

Para obtener una secuencia de números entre 0 y 1 a partir de un GCL, bastará con dividir cada $x_n$ por el módulo $m$ de la serie:

$$u_n = x_n/m$$

A pesar de su simplicidad, la selección adecuada de las constantes $(a, b, m)$ permitirá obtener secuencias largas y aleatorias de un modo muy eficiente.

### 3.2.2. Selección de parámetros

Como se muestra en la figura anterior, los GLCs producen secuencias cíclicas. Una de las propiedades que nos interesa en Simulación es que el periodo de repetición de las secuencias sea lo más grande posible. Dicho periodo vendrá determinado por los parámetros del generador. Por ejemplo, se ha demostrado que el máximo periodo se alcanza cuando se dan las siguientes condiciones:

- $b \neq 0$
- $mcd(b, m) = 1$ (primos relativos)
- $a = 1 \; mod \; p$ para cada factor primo $p$ de $m$
- $a = 1 \; mod \; 4$ si 4 divide a $m$

Además de la longitud de los ciclos, en simulación también nos interesa que el generador sea muy eficiente. Dado que los GLCs multiplicativos ($b = 0$) son los más eficientes de computar, a partir de ahora sólo estudiaremos este tipo de generadores.

Es fácil ver que la operación más costosa en un GLC es la operación del resto ($m$ suele ser un número muy grande). Si tomamos $m = 2^\beta$ la operación de resto resulta obvia, ya que basta retener los $\beta$ últimos bits del producto $a \cdot x_i$. Sin embargo, la longitud máxima de este tipo de generadores es $m/4$, y se da cuando la semilla es impar y la constante $a$ tiene la forma $8 \cdot i \pm 3$.

La longitud del periodo de un GLC multiplicativo puede llegar a ser $m - 1$ cuando $m$ es un número primo (en este caso los valores de la serie estarán entre 1 y $m - 1$, nunca valdrán cero). Además, el valor de $a$ tiene que ser cuidadosamente seleccionado. A este respecto, se ha demostrado que si $a$ es una raíz primitiva de $m$, entonces se alcanza el periodo máximo $m - 1$. Esto ocurre cuando $a^n \; mod \; m \neq 1$ para $n = 1, 2, \ldots, m - 2$.

Un ejemplo de GLC multiplicativo con periodo máximo $m - 1$ es el siguiente:

- $a = 7^5 = 16807$
- $b = 0$
- $m = 2^{31} - 1 = 2147483647$

Este ejemplo, de hecho, es el que se ha tomado como estándar para la generación de números aleatorios, debido principalmente a la calidad de las series obtenidas.

Un último factor que influye en la elección de los parámetros de un GLC es la calidad de la serie generada. Como vimos en la introducción, la calidad de una serie se mide por la uniformidad e independencia de los valores de la serie.

A este respecto, una propiedad inherente de los generadores GLC es que producen una estructura reticular cuando se toman subsecuencias de la serie generada. Un ejemplo típico de estructura reticular se muestra en la figura 3.1, donde se consideran los pares de números consecutivos de la serie $(u_i, u_{i+1})$.

En esta estructura reticular pueden identificarse una serie de líneas (unas con pendientes positivas y otras con pendientes negativas) donde se sitúan todos los pares de la serie. Dependiendo de la distancia entre estas líneas, los pares se distribuyen más o menos uniformemente en el plano. Generalmente, cuanto mayor sea la distancia, peor será el generador.

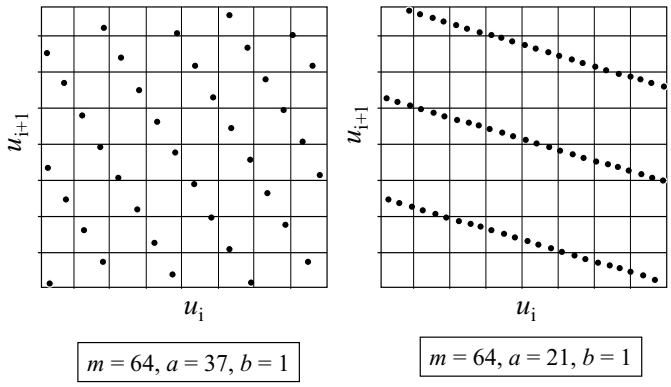



Figura 3.1: Estructura reticular de un GLC

Esta distancia viene determinada por el parámetro $a$. Así pues, el valor de $a$ va a determinar la uniformidad de los valores, y su selección es crítica para la calidad de la serie final.

A modo de ejemplo, un primer generador GLC de números aleatorios que fue propuesto por IBM en 1960 tenía los siguiente parámetros: $a = 65539$, $b = 0$ y $m = 2^{31}$. Con este generador todas las tripletas de números consecutivos de las series $(5 \cdot 10^{10})$ caen en tan solo 15 planos.

Marsaglia demostró en 1968 que el número máximo de hiperplanos paralelos que puede producir un GLC multiplicativo es $(n!m)^{1/n}$, donde $n$ es la cantidad de números consecutivos de las subsecuencias consideradas. Observa que el número de hiperplanos cae rápidamente conforme aumenta la dimensión del espacio $n$.

### 3.2.3. Otras consideraciones

La implementación de un GLC debe realizarse con sumo cuidado, ya que los resultados teóricos obtenidos con unos parámetros determinados podrían no ser ciertos si la implementación no realiza los cálculos exactos. En la implementación debe prestarse especial atención a los redondeos y a los desbordamientos en determinadas operaciones.

En el generador GLC anterior, puede verse que el producto $a \cdot x_i$ puede llegar a $1{,}03 \cdot 2^{45}$, lo cual producirá un desbordamiento, a no ser que se use un tipo entero de al menos 46 bits.

Para estos casos, debemos reformular el cálculo del GLC para evitar el desbordamiento. Una posible solución la propone Schrage (ver [Raj Jain, 1991]), y se resume en el siguiente fragmento de código escrito en Python:

```
int random(x):
  a = 16807
  m = 2147483647
  q = 127773      #m div a
  r = 2836        #m mod a

  x_div_q = x / q
  x_mod_q = x % q
  x_new = a*x_mod_q - r*x_div_q
  if x_new>0:
     x = x_new
  else:
     x = x_new + m
  return x
```

Para comprobar que el generador produce el resultado esperado, el valor para $x_{10000}$ debe ser 1043618065 con $x_0 = 1$.

## 3.3. Test empíricos

Hemos visto en la sección anterior que los generadores de números aleatorios pueden tener más o menos calidad dependiendo de los parámetros seleccionados. Recordemos que la calidad se mide en función de la independencia y uniformidad de los valores de las secuencias generadas.

Para comprobar la calidad de un generador se han propuesto varios tests empíricos. Si un generador no pasa alguno de estos tests deberemos descartarlo ya que no cumple algún requisito mínimo para asegurar que los valores son IID. Sin embargo, si el generador pasa todos los tests, no podemos asegurar que el generador está completamente libre de sesgos o dependencias. Podría pasar que un nuevo test descubra alguna deficiencia en dicho generador y lo descarte definitivamente. También puede ocurrir que algunas semillas produzcan secuencias que pasen los tests, pero otras no, con lo que habría que descartar el generador.

Los test empíricos se realizan de una forma similar:

1. Primero se realizan una serie de *observaciones* sobre la serie a estudiar.
2. Se define una *estimación* a partir de la distribución que se supone deben seguir las observaciones.
3. Se evalúa la *proximidad* entre las observaciones y la estimación.

Para la última tarea, debido a que trabajamos con números aleatorios se utilizará una distribución conocida para tener en cuenta las variaciones de las medidas calculadas.

A continuación vamos a describir los test empíricos más utilizados para comprobar la bondad de los generadores.

### 3.3.1. Test $\chi^2$

Este test es el más utilizado para comprobar si una lista de números satisface una determinada distribución. Este test se basa en una partición del espacio muestral en $k$ categorías disjuntas, de modo que cada observación $o_i$ consiste en contar cuántos números de la serie caen en la categoría $i$-ésima. La estimación de este test se calcula a partir de la función de densidad hipotética. Con esta función se calcula el número de valores estimados $e_i$ que caen en cada categoría.

Para comparar las observaciones con la estimación se utiliza el error cuadrático medio:

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

Debido a la aletoriedad de las observaciones, la medida $X$ no va a ser nunca cero, sino que satisface una distribución aleatoria, precisamente denominada chi-cuadrado ($\chi^2$), con $k-1$ grados de libertad. Así, dependiendo del grado de confianza que queramos en el test ($\alpha$) utilizaremos el valor de la distribución $\chi^2_{[1-\alpha;k-1]}$.

En nuestro caso, si queremos comprobar si un generador de números aleatorios es bueno, compararemos la serie generada de longitud $n$ con la distribución uniforme $U(0,1)$:

- Particionamos el espacio muestral $[0,1]$ en $k$ intervalos disjuntos.
- Calculamos cuántos números de la serie caen en cada intervalo ($o_i$).
- La estimación es que $n/k$ números caigan en cada intervalo, es decir, $e_i = n/k$.

- A partir de la ecuación anterior, el valor de $X$ se calcularía como sigue:

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

- Finalmente, si el valor de $X$ es mayor que $\chi^2_{[1-\alpha;k-1]}$, entoces descartaremos el generador.

Una limitación importante de este test es que si los valores estimados $e_i$ no son los mismos para las distintas categorías (cosa que no ocurre para la distribución uniforme), entonces el error será mayor en las categorías de tamaño menor. Para evitar este efecto, tendríamos que dividir el espacio muestral en categorías equiprobables, las cuales podrían tener tamaños diferentes. Esto implica tomar una decisión sobre el número y forma de las categorías o celdas, lo que a veces no resulta fácil. Por otro lado, para que el test sea fiable, cada categoría debe tener al menos 5 elementos.

En general, el test de chi-cuadrado se ha diseñado para muchas muestras (series muy largas) que sigan una distribución aleatoria discreta.

A modo de ejemplo, a continuación mostramos el resultado de este test aplicado a tres secuencias de 1000 números. La primera es simplemente una serie de números consecutivos, la segunda es una serie con los tiempos de respuesta de una estación M/M/2, y la tercera es una serie de números obtenidos con la función `random()` de Python. Las dos primeras series no pueden ser consideradas aleatorias ya que la primera tiene una fuerte correlacióon entre pares consecutivos de valores, y en la segunda el tiempo de respuesta de un cliente depende de los tiempos de respuesta de los clientes anteriores, es decir también existe una fuerte correlación entre los valores (aunque no tan evidente).

| **Secuencia** | **Resultado** |
|---|---|
| Números consecutivos | 0.02 |
| M/M/2 | 1629.86 |
| Random | 8.72 |

Según este test, una serie se considera buena si el resultado obtenido es menor que 15,987. Observa que el mejor resultado lo obtiene la secuencia de números consecutivos, que por cierto es la menos aleatoria de las tres.

### 3.3.2. Test Kolgomorov-Smirnov

El test chi-cuadrado se basa en observaciones realizadas sobre la función de probabilidad o densidad. El test de Kolgomorov-Smirnov (K-S) se basa en la función de distribución o acumulativa (CDF).

A partir de una secuencia de números $(x_1, \ldots, x_n)$, las observaciones se obtienen a partir de la función CDF discreta producida por esta serie:

$$F_n(x) = \frac{\sharp\{x_i \leq x\}}{n}$$

Es decir, esta función toma en $x$ el número de elementos de la serie que son menores o iguales a $x$.

Esta función se compara con otra función estimada $F_e(x)$. Concretamente, mediremos la desviación máxima entre ambas funciones:

$$K^+ = \sqrt{n}\, max_x[F_n(x) - F_e(x)]$$

$$K^- = \sqrt{n}\, max_x[F_e(x) - F_n(x)]$$

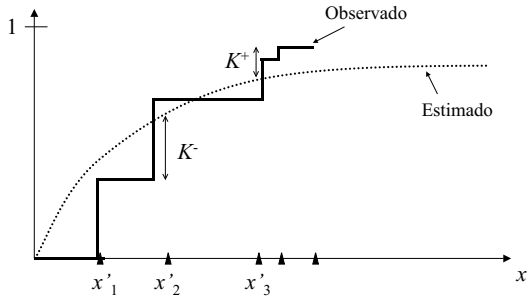



Figura 3.2: Desviaciones máximas en el test K-S

El significado de $K^+$ y $K^-$ se puede ver en la gráfica de la figura 3.2. El valor de $K^+$ expresa la máxima diferencia entre ambas funciones cuando la CDF de las observaciones supera a la CDF estimada. $K^+$ expresa la máxima diferencia entre ambas funciones cuando la CDF estimada supera a la CDF de las observaciones.

Al igual que en el test chi-cuadrado, en este test las desviaciones calculadas siguen una distribución aleatoria, denominada $K$, con $n$ grados de libertad. Así, dependiendo del grado de confianza $\alpha$ deseado, la desviación máxima deberá ser menor que el valor $K_{[1-\alpha;n]}$.

De nuevo, si queremos comprobar si un generador de números aleatorios es bueno, compararemos la serie generada de longitud $n$ con la distribución uniforme $U(0,1)$, del siguiente modo:

- Ordenamos la serie de menor a mayor $(x'_1, \ldots, x'_n)$.
- Calculamos las desviaciones máximas como sigue:

$$K^+ = \sqrt{n}\, max_j[\frac{j}{n} - x'_j]$$

$$K^- = \sqrt{n}\, max_j[x'_j - \frac{j-1}{n}]$$

- Finalmente, si el valor de $K^+$ o $K^-$ es mayor que $K_{[1-\alpha;n]}$, entoces descartaremos el generador.

Al contrario que el test chi-cuadrado, el test K-S está diseñado para series con pocas muestras y que siguen una distribución continua. El test K-S hace mejor uso de los datos ya que no necesita tantas muestras como el chi-cuadrado (al menos 5 por cada categoría), y además no necesita diseñar una partición apropiada del espacio muestral. En general, el test K-S es bastante mejor que el chi-cuadrado.

A modo de ejemplo, los resultados obtenidos con las series de la sección anterior serían los siguientes:

| **Secuencia** | **Resultado** |
|---|---|
| Números consecutivos | 0.0316 |
| M/M/2 | 17.296 |
| Random | 0.894 |

Las secuencias que pasan este test no deben superar el valor 1,48 según las tablas de la distribución K-S. Observa que de nuevo sólo la serie M/M/2 es rechazada.

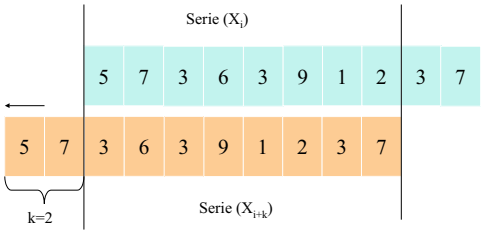



Figura 3.3: Variables $X_i$ y $X_{i+k}$ en un test de autocovarianza

### 3.3.3. Test de correlación serial

Los dos tests anteriores se aplican para medir la uniformidad de las series generadas por los generadores. Otro aspecto que nos interesa medir en una serie es la dependencia de sus valores. Para ello se proponen dos técnicas básicas:

- Realizar un test espectral de la serie, es decir, mostrar en un plano cómo se distribuyen los pares de valores consecutivos de la serie $(u_i, u_{i+1})$ (ver figura 3.1).
- Medir la *autocovarianza* de los valores de la serie.

En esta sección nos vamos a centrar en el segundo método. Éste consiste en calcular la covarianza existente entre la serie original y la misma serie desplazada en $k$ posiciones (ver figura 3.3). Si la covarianza no es cero, podremos asegurar que existe alguna dependencia entre los valores de la serie, y en consecuencia tendremos que descartar el generador. Sin embargo, lo contrario no es cierto: si la covarianza es cero, la serie todavía puede presentar dependencias entre sus valores.

Para medir la autocovarianza de una serie aleatoria, calcularemos la covarianza que existe entre los números de la serie que están separados en $k$ posiciones, esto es, entre $x_i$ y $x_{i+k}$. Esta covarianza se denomina *autocovarianza* de salto $k$, denotada $R_k$:

$$R_k = Cov(X_i, X_{i+k}) = E[(X_i - \mu_i) \cdot (X_{i+k} - \mu_{i+k})]$$

En nuestro caso, como la serie de números debe seguir una distribución uniforme $U(0,1)$, $R_k$ se calcularía como sigue:

$$R_k = \frac{1}{n-k} \sum_{i=1}^{n-k} (x_i - \frac{1}{2})(x_{i+k} - \frac{1}{2})$$

Al igual que en los tests anteriores, la medida $R_k$ es una medida aproximada que debe contrastarse con una distribución aleatoria. Para este caso, cuando $n$ es grande, $R_k$ sigue una distribución normal centrada en 0 y con varianza $1/(144(n-k))$.

Así pues, el generador debe descartarse cuando el intervalo de confianza para $R_k$ no contiene el valor 0, ya que demuestra que existe una dependencia importante entre los valores de la serie. Para $R_k$ el intervalo de confianza se calcula como sigue:

$$R_k \pm \frac{z_{1-\alpha/2}}{12\sqrt{n-k}}$$

A modo de ejemplo, tomaremos de nuevo las tres series anteriores y probaremos este test con diferentes valores de $k$.

| **Secuencia** | $k=1$ | $k=2$ | $k=3$ |
|---|---|---|---|
| Números consecutivos | [0.0797,0.0865] | [0.0796,0.0864] | [0.0794,0.0862] |
| M/M/2 | [0.107,0.114] | [0.107,0.114] | [0.107,0.114] |
| Random | [-0.0015,0.0052] | [-0.0016,0.0051] | [-0.00081,0.0059] |

Según este test, solo pasaría la serie obtenida con `random()`, ya que es la única que incluye el 0 en los intervalos obtenidos.

## 3.4. Variables aleatorias

En muchos experimientos, incluidos los de simulación, los resultados obtenidos presentan cierta incertidumbre. Generalmente, la representación y manejo de esta incertidumbre se realiza mediante *variables aleatorias*. Partiendo del conjunto de todos los posibles resultados en un experimento, denominado *espacio de muestreo* y denotado como $S$, una variable aleatoria define una función que asigna a cada valor de $S$ un valor real que representa de alguna forma su probabilidad de suceso. Las variables aleatorias suelen denotarse con letras mayúsculas (ej. $X$, $Y$, $Z$), mientras los valores que pueden tomar se denotan con letras minúsculas (ej. $x$, $y$, $z$)

Diremos que una variable aleatoria es *discreta* si su espacio de muestreo es discreto, es decir, si existe una correspondencia uno a uno con un subconjunto de los números enteros. Si el espacio de muestreo se define sobre los reales, diremos que la variable aleatoria es *continua*. Vamos a analizar en primer lugar las variables aleatorias discretas.

La función que asigna la probabilidad de cada valor de una variable aleatoria $X$ se denomina *función de densidad*:

$$p(x_i) = P(X = x_i)$$

Una propiedad que debe cumplir esta función es que la suma de todas las probabilidades debe valer 1, es decir:

$$\sum_{i=1}^{\infty} p(x_i) = 1$$

Otra función interesante que caracteriza una variable aleatoria es la *función de distribución* o acumulativa (cdf), la cual define la probabilidad de encontrar un valor menor que otro dado $x$, es decir $P(X \leq x)$. Esta función se define a partir de la de densidad como sigue:

$$F(x) = \sum_{x_i \leq x} p(x_i)$$

Esta función presenta algunas propiedades interesantes:

- Está acotada entre 0 y 1, es decir $0 \leq F(x) \leq 1$
- Es monotona creciente
- $\lim_{x\to\infty} F(x) = 1$ y $\lim_{x\to-\infty} F(x) = 0$

En el caso de una variable aleatoria continua, también asociaremos un función de densidad, denotada $f(x)$, sobre los valores del espacio de muestreo. De forma similar a las variables discretas, la suma sobre todo el espacio de valores debe ser 1, es decir:

$$\int_{-\infty}^{\infty} f(x) \cdot dx = 1$$

Sin embargo, la función $f(x)$ no representa la probabilidad de que $X = x$, ya que ésta valdría siempre 0 ($\int_x^x f(y) \cdot dy = 0$). La nueva interpretación de la función de densidad debe realizarse sobre pequeños intervalos. Así, si tenemos un pequeño incremento $\Delta x \geq 0$, la probabilidad asociada a $x$ será proporcional al área que define la función de densidad sobre el intervalo $[x, x + \Delta x]$, cuanto mayor sea el área mayor será la probabilidad del punto $x$.

$$P(X \in [x, x+\Delta x]) = \int_x^{x+\Delta x} f(y) \cdot dy$$

Sobre las variables aleatorias continuas también podemos definir una función de distribución o acumulativa, la cual representará la probabilidad de encontrar un valor menor que otro dado $x$:

$$F(x) = \int_{-\infty}^{x} f(y) \cdot dy$$

Esta función presenta las mismas propiedades que las que hemos visto anteriormente para las variables discretas.

Para finalizar, definiremos varios parámetros que caracterizan a cualquier variable aleatoria:

| **Parámetro** | **Discreta** | **Continua** |
|---|---|---|
| Valor esperado $E(X)$ | $\sum_{j=1}^{\infty} x_j \cdot p(x_j)$ | $\int_{-\infty}^{\infty} x \cdot f(x) \cdot dx$ |
| Varianza $Var(X)$ | $E(X^2) - E(X)$ | $E(X^2) - E(X)$ |
| Desviación Típica ($\sigma$) | $\sqrt{Var(X)}$ | $\sqrt{Var(X)}$ |

## 3.5. Generación de variables aleatorias

Desde el punto de vista de programación, una variable aleatoria es un generador de números aleatorios que sigue una determinada distribución. En el tema anterior vimos cómo generar secuencias de valores aleatorios que siguen una distribución uniforme $U(0,1)$ mediante los denominados generadores congruenciales lineales. En este tema estudiaremos cómo generar secuencias de valores aleatorios que siguen otras funciones de distribución conocidas y ampliamente utilizadas en simulación.

### 3.5.1. Método de inversion

Este método se basa en la función inversa de la función de distribución de la variable aleatoria $X$. Dado que $F(X)$ está acotada entre 0 y 1, podemos generar valores en $U(0,1)$ y sobre ellos aplicar $F^{-1}(X)$.

Por ejemplo, tomemos la distribución exponencial con media en $\lambda$, y cuyas funciones de densidad y distribución son las siguientes:

$$\begin{aligned} f(x) &= \lambda \cdot e^{-\lambda \cdot x} \\ F(x) &= 1 - e^{-\lambda \cdot x} \end{aligned}$$

La función inversa de $F(X)$ sería:

$$F^{-1} = -\frac{1}{\lambda} ln(1-u)$$

Y el algoritmo para generar valores aleatorios para esta distribución sería :

```
def exponential(media):
    u = random(0,1)
    return -log(1-u)/media
```

El método de la inversa es especialmente útil para generar valores de variables aleatorias discretas. Veamos un ejemplo.

Supongamos que tenemos una variable aleatoria discreta con espacio de muestreo $S = \{1, 2, 3\}$, y cuya función de densidad es la siguiente:

$$p(1) = 0{,}2 \qquad p(2) = 0{,}3 \qquad p(3) = 0{,}5$$

Entonces su función de distribución sería:

$$F(1) = 0{,}2 \qquad F(2) = 0{,}2 + 0{,}3 = 0{,}5 \qquad F(3) = 0{,}5 + 0{,}5 = 1$$

Así, en el eje de F(x) podemos identificar tres intervalos: $(0, 0{,}2]$, $(0{,}2, 0{,}5]$ y $(0{,}5, 1]$. Para obtener valores para esta variable, generaremos un valor para $U(0, 1)$, y determinaremos su inversa a partir del intervalo en el que caiga. Si éste cae en el primer invervalo, el valor de $F^{-1}$ será 1, si en el segundo 2, y si en el tercero 3. Así, el algoritmo para generar valores para esta variable aleatoria sería el siguiente:

```
def genera_discreta(p1, p2, p3):
    u= random(0,1)
    if u<=p1:
       finv = 1
    elif u<=p1+p2:
       finv = 2
    else:
       finv = 3
    return finv
```

Recuerda que este tipo de distribuciones suele aparecer en los diagramas de sucesos, concretamente cuando un suceso planifica la ocurrencia de otros sucesos con una determinada probabilidad.

Como ejercicio se plantea realizar una función para obtener valores aleatorios según una distribución discreta representada con una lista de cualquier longitud (por ejemplo [0.1, 0.3, 0.4, 0.2] o [0.5, 0.3,0.2], etc.)

### 3.5.2. Método del rechazo

Cuando no podamos calcular la inversa de la función de distribución, lo que sucede en la mayor parte de las distribuciones continuas, entonces tendremos que basarnos en la función de densidad $f(x)$.

El método del rechazo utiliza una función de densidad $g(x)$ para la cual sabemos generar números aleatorios (ej. una uniforme), y que además recubre por completo la función de densidad que queremos obtener, es decir:

$$a \cdot g(x) \geq f(x)$$

El algoritmo para generar valores a partir de $g(x)$ sería el siguiente:

```
repetir
    genera un valor x con densidad g
    genera un valor u con U(0, 1)
hasta que u·a·g(x) <= f(x)
devuelve x
```

Para que funcione eficientemente, el algoritmo debe utilizar una función $g(x)$ sencilla, y que se ajuste lo más posible a la función $f(x)$. De ella dependerá el número de iteraciones del algoritmo.

Veamos un ejemplo presentado en [Raj Jain, 1991]. Supongamos que queremos generar una secuencia de valores para la distribución $beta(2,4)$, cuya función de densidad es:

$$f(x) = 20 \cdot x(1-x)^3$$

Esta función puede ser recubierta por un rectángulo de altura 2,11, que es el máximo de la función. Por lo tanto, tomando $g(x) = U(0,1)$ y $a = 2{,}11$, obtendríamos el siguiente algoritmo:

```
def beta2_4():
    a = 2.11
    x = random(0,1)
    u = random(0,1)
    while (u*a <= (20*x*(1-x)**3)):
        x = random(0,1)
        u = random(0,1)
    return x
```

### 3.5.3. Método de Convolución

Si $X$ es la suma de dos variables aleatorias $Y_1$ y $Y_2$, entonces la función de distribución de $X$ puede obtenerse analíticamente mediante la convolución de las funciones de distribución de $Y_1$ y $Y_2$. Este es el principio del método de convolución: para obtener los valores de $X$ bastará con generar 2 valores para $Y_1$ y $Y_2$, y devolver su suma.

Es importante destacar que la suma de un conjunto de variables aleatorias (convolución) es un concepto diferente al de la suma de sus funciones de distribución (composición).

Existen varias distribuciones que pueden generarse por convolución:

- Una variable con distribución normal puede obternerse sumando un número suficiente de variables aleatorias con cualquier distribución.
- Una variable con distribución $\chi^2$ con $k$ grados de libertad puede obtenerse sumando los cuadrados de $k$ variables con distribución $N(0,1)$.
- Una variable con distribución Erlang-$k$ puede obtenerse sumando $k$ variables con distribución exponencial.
- La suma de dos variables con distribución $U(0,1)$ forma una variable con distribución triangular.

## 3.6. Algunas distribuciones útiles

A continuación mostramos brevemente las características de algunas distribuciones continuas que son habitualmente utilizadas en simulación.

**Uniforme:** denotada como $U(a,b)$ con $b > a$, su función de densidad es:

$$f(x) = \frac{1}{b-a}$$

su media es $\frac{a+b}{2}$, y su varianza es $\frac{(b-a)^2}{12}$.

Podemos generar valores para $U(a,b)$, generando un valor $u$ en $U(0,1)$, y devolviendo $a+(b-a)\cdot u$.

Recuerda que para obtener valores de una distribución uniforme se suelen utilizar los generadores congruenciales que vimos al principio de este tema. Además, a patir de esta distribución pueden obtenerse el resto según los métodos vistos en las secciones anteriores.

**Exponencial:** denotada como $Exp(a)$, su función de densidad es:

$$f(x) = \frac{1}{a}e^{-x/a}$$

su media es $a$ y su varianza $a^2$.

Sus valores se generan mediante el método de inversión (sección 3.1).

Dentro de la familia de la distribuciones exponenciales encontramos también la $Erlang(a,m)$, que se obtiene mediante la convolución de $m$ variables exponenciales (de hecho se utiliza para simular la suma de los tiempos de servicio de $m$ servidores en serie); la $Gamma(a,b)$, que es una generalización de las anteriores (donde $a$ juega el papel de $m$ y $b$ es la media de la distribución); y la $Weibull(a,b)$, que generaliza la exponencial en $a=1$, siendo $b$ la media de la distribución.

**Normal:** denotada como $N(\mu,\sigma)$, su función de densidad es:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}}e^{-(x-\mu)^2/2\sigma^2}$$

Su media es $\mu$ y su varianza $\sigma^2$.

Pueden obtenerse valores para esta distribución por convolución de $n$ ($n=12$) variables en $U(0,1)$:

$$N(\mu,\sigma) \sim \mu + \sigma\frac{(\sum_{i=1}^{n} u_i) - n/2}{\sqrt{n/12}}$$

Otro método utilizado para generar esta distribución es el de Box-Muller, el cual genera dos valores independientes para $N(\mu,\sigma)$ a partir de dos valores $u_1$ y $u_2$ de una uniforme $U(0,1)$:

$$\begin{aligned} x_1 &= \mu + \sigma cos(2\pi u_1)\sqrt{-2ln(u_2)} \\ x_2 &= \mu + \sigma sin(2\pi u_1)\sqrt{-2ln(u_2)} \end{aligned}$$

A partir de la distribución normal se pueden obtener diversas distribuciones similares, como por ejemplo la $\chi^2$, $LogNormal(\mu,\sigma)$, la $t-student$, y la distribución $Cauchy(a,b)$.

**Zipf:** En el ámbito de los Sistemas de Información es muy frecuente encontrar distribuciones que siguen una función potencia (*Power Law*). Estas distribuciones son discretas, y expresan una relación entre el orden $i$ de la frecuencia de uso (o popularidad) de un objeto con su probabilidad de ocurrencia $P_i$, a saber: $P_i = 1/i^a$, donde $a$ es aproximadamente 1. Este tipo de distribuciones se observan por ejemplo en la frecuencia de uso de las palabras de un diccionario, en el grado de popularidad de las páginas de la Web, o en la frecuencia de uso en una caché. En los dos últimos casos, el exponente $a$ suele ser menor que 1.

La versión continua de esta distribución es la *Pareto*, que se ha utilizado tradicionalmente en Economía para modelar el reparto de las riquezas (el 20 % de la población dispone del 80 % de las riquezas). En el ámbito de la computación, esta distribución se ha utilizado con éxito para modelar el tamaño de los ficheros de un servidor en Internet (con $a = 1{,}06$). Las funciones de densidad y acumulada de esta distribución son las siguientes:

$$f(x;a) = \frac{a \cdot m^a}{x^{a+1}} \; con \; x > m$$
$$F(x;a) = 1 - (m/x)^a \; con x \geq m$$

En prácticamente todos los libros de simulación podréis encontrar una revisión de las distribuciones más utilizadas en simulación, tanto sus características como el método más apropiado de generar valores.

## 3.7. Bibliografía

Raj Jain "The Art of Computer Systems Performance Analysis". Ed. Wiley (1991).

David Ríos Insua et al. "Simulación métodos y aplicaciones". Ed. Ra-Ma, Madrid (1997)

A. M. Law, W. D. Kelton "Simulation Modeling & Analysis". Ed. McGraw-Hill (1984).