Laboratorio de Telemática III

ETS Ingeniería de Telecomunicación de Barcelona

Práctica 2: Generación de variables aleatorias

Israel Martín, Alfonso Rojas
Departamento de Ingeniería Telemática
Universidad Politécnica de Cataluña

Índice

1 Objetivos

1.1 Objetivos generales

Los objetivos generales de la práctica son formar al alumno en la disciplina de generación de valores aleatorios y que adquiera la destreza suficiente como para evaluar la calidad de los generadores empleados a tal efecto.

1.2 Objetivos específicos

Los objetivos específicos planteados en la presente práctica son los siguientes:

- Diferenciar entre generador aleatorio y pseudoaleatorio
- Conocer las propiedades deseables de un generador pseudoaleatorio
- Conocer el mecanismo de generación de valores pseudoaleatorios con mayor difusión: el generador congruencial lineal
- Calcular los valores más apropiados para inicializar un generador congruencial lineal
- Conocer alternativas simples al generador congruencial lineal: Fibonacci y generadores múltiplemente recursivos
- Conocer los principales mecanismos para la combinación de generadores pseudoaleatorios
- Conocer y aplicar los principales mecanismos de generación de variables aleatorias a distribuciones conocidas
- Calcular los principales indicadores sobre la calidad de un generador

1.3 Conocimientos asumidos

Para la realización de la práctica se asume que el alumno tiene una formación mínima en los siguientes apartados.

- Conocimientos estadísticos: variables aleatorias, momentos y nociones simples de estadística.
- Conocimientos del lenguaje de programación C (recomendable C++).

Se recomienda refrescar dichos conocimientos antes de la realización de la práctica.

2 Introducción

La disponibilidad de números aleatorios es una necesidad frecuente en diversas áreas de ingeniería tales como simulación, gestión de claves, verificación, juegos, control de calidad, etc. En el campo de la simulación el interés suele centrarse en la obtención de secuencias de números (p.ej intervalos de sucesos).

Mediante procedimientos físicos pueden generarse secuencias de números aleatorios tanto continuos (p.e. resistencia generadora de ruido) como discretos (p.ej. dado, ruleta). Aunque existen dispositivos hardware capaces de generar número aleatorios reproduciendo alguno de estos fenómenos (por ejemplo, sistemas basados en la generación y medición de ruido térmico), éstos se muestran caros e ineficaces a la hora de integrar la información aleatoria producida con los dispositivos que necesitan de ella. Además, este tipo de generadores presentan una limitación muy a tener en cuenta a la hora de desarrollar sistemas simulados: su aleatoriedad. Como son procesos totalmente aleatorios, no pueden ser reproducidos, y por tanto dificultan el estudio de determinados patrones que pueden ser de interés para el usuario del simulador.

De esta forma en la actualidad los generadores de valores aleatorios empleados son implementaciones software, es decir, algoritmos matemáticos transcritos a un cierto lenguaje de programación con los que se generan valores aleatorios según se requiere de éstos. Dichos algoritmos generan lo que se conoce como valores pseudoaleatorios, es decir, valores aleatorios que vienen determinados por una serie de condiciones iniciales que son deterministas. Para muchos de estos algoritmos, estas condiciones iniciales se centran en la elección de un primer valor “aleatorio” llamado semilla. Los valores obtenidos una vez fijada la semilla son aleatorios y son reproducibles siempre que se emplee la misma semilla. Es decir, dos secuencias serán idénticas si se generan a partir de la misma semilla. Por tanto las condiciones iniciales (o semilla) de un generador pseudoaleatorio determinan la secuencia que se va a generar y por tanto deberá prestarse especial atención a los valores proporcionados en la inicialización de los distintos algoritmos.

3 Propiedades deseables de un generador pseudoaleatorio

La generación de valores aleatorios es de especial relevancia en los procesos de simulación puesto que permite la generación de variables aleatorias uniformes. Dichas variables son la base para la construcción de variables aleatorias más complejas, tanto discretas como continuas. Es por tanto necesario garantizar la calidad de los generadores de valores aleatorios, puesto que los defectos presentes en el generador aleatorio se transmitirá de forma inherente a las variables generadas con el mismo y por ende a los resultados finales de la simulación.

Las características deseables en un generador pseudoaleatorio dependen mucho del ámbito al que se destine dicho generador, si bien hay un grupo de características comúnmente aceptadas y que todo generador pseudoaleatorio debería cumplir:

- **Aleatoriedad.** Deberá producir valores uniformemente distribuidos entre [0,1]
- **Independencia.** Las muestras generadas no deben estar correlacionadas, es decir, cada valor generado debe de ser independiente de las muestras generadas hasta el momento
- **Periodo máximo.** El número máximo de muestras distintas que un generador pseudoaleatorio puede generar es conocido de antemano. De esta forma, todo generador debe tener como propósito generar secuencias con una duración que sea máxima. La duración máxima de la secuencia producida por un generador se conoce como periodo. Cuando un generador obtiene secuencias de duración máxima, se dice que es un *generador de periodo completo*. El periodo se mide como el número de muestras generadas hasta que la secuencia se repite. El valor que se repite puede ser la semilla u otro valor distinto. En éste último caso se distingue entre el ciclo (que es la parte de la secuencia que se repite) y cola (que es la parte de la secuencia que no se repite). Por ejemplo, si se dispusiera de la secuencia: 1, 7, 9, 3, 10, 8, 4, 3, 10, ..., el ciclo de la secuencia sería de 4 (valores 3, 10, 8 y 4) y el periodo de la secuencia sería de 7 (es decir, tendríamos una cola con valores 1, 7 y 9).
- **Secuencia reproducible.** La secuencia debe ser reproducible, es decir, se deben poder establecer una condiciones iniciales a partir de las cuales la secuencia obtenida sea siempre la misma
- **Eficiente.** El algoritmo propuesto deberá consumir los menores recursos posibles tanto en tiempo (ejecución en el menor tiempo posible) como en memoria (utilización de la menor memoria posible)

4 Generadores pseudoaleatorios

4.1 Generador congruencial lineal (GCL)

Fue introducido por Lehmer en 1951 y su principal cualidad es su rapidez. Son muy populares y sencillos de implementar. Su uso se restringe a tareas de simulación, descartándose su empleo en empresas en las que la seguridad sea un requisito. Las secuencias se generan mediante la siguiente expresión:

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

donde x_0 es el valor inicial o semilla, *a* es multiplicador, *b* es el incremento y *m* el módulo. En particular Lehmer propuso un generador con $b=0$ al que se denominó GCL multiplicativo y no fue hasta el año 58 en que Thomsom utilizara $b\neq0$, el GCL mixto. Se debe tener presente que:

- *a*, *b* y *m* son valores enteros positivos
- El periodo máximo producido por este generador es *m*. Los valores de *m* empleados son de varios ordenes de magnitud
- Los valores generados se encuentran en el conjunto $\mathbb{Z}_m$, es decir, los enteros positivos menores o iguales a *m-1*
- De esta forma, los valores finalmente producidos por el generador deberán ser $U_n = x_n / m$ para obtener valores pertenecientes a los reales en el intervalo [0,1)

4.1.1 GCL multiplicativo

El incremento *b* es cero. Tiene como ventaja con respecto el generador mixto (*a, b* no nulos) que es más rápido puesto que consta de una operación menos. Este tipo de generador puede ser capaz de generar secuencias de periodo máximo *m-1 (e*l valor 0 obviamente no está permitido). Para ello es necesario que:

- existan raíces primitivas módulo *m* (elementos generadores del conjunto reducido de residuos módulo *m, CRR_m*). Generalmente *m* debería ser un número primo
- *a* sea una raíz primitiva módulo *m*

4.1.2 GCL mixto

Es el caso más general, con los parámetros *a* y *b* distintos de cero. Según el teorema de Knuth , un GCL mixto será de periodo completo si:

- *b* y *m* son coprimos ($b < m$)
- *a-1* es múltiplo de los factores primos de *m*
- *a-1* es múltiplo de 4 siempre que *m* sea múltiplo de 4

Una posible combinación de valores, que no la única, en las que las condiciones anteriores se cumplen es $m = 2 \cdot k_1$, $a = 4 \cdot k_2 + 1$, y *b* impar, donde k_1 y k_2 son valores enteros positivos.

Hay que tener presente que un generador de alto periodo no tiene que ser bueno simplemente por el hecho de tener esta cualidad. Hay más cualidades que debe cumplir el generador para ser un buen generador (ver sección 2). Por ejemplo, el valor óptimo para *m* en términos de eficiencia consiste en escoger una potencia de la base empleada. Teniendo en cuenta que un ordenador trabaja en base 2,

el valor óptimo de *m* sería 2^e y *e* es inferior al número de bits de la arquitectura (típicamente 32 o más recientemente 64 bits). Sin embargo, la selección de *m* como una potencia de la base motiva en que los bits menos significativos de los números generados sean menos aleatorios que los bits más significativos. De esta forma, la secuencia que integra las muestras producidas por los bits menos significativos proporcionan periodos más cortos que la secuencia obtenida mediante las muestras resultantes de los bits más significativos. Todo ello resulta en una merma de la uniformidad de la secuencia global obtenida. Existen formas de llegar a un compromiso entre uniformidad y eficiencia. Por ejemplo, algunas recomendaciones serían:

- Hacer que *m* siga la expresión $2^e \pm 1$
- Escoger *m* de tal forma que sea un número primo. Esta tarea no es trivial, en tanto en cuanto *m* debe ser un valor relativamente alto para asegurar un periodo de longitud aceptable

La elección final de *m* vendrá condicionada por la aplicación, puesto que hay múltiples tareas en las que las cifras poco significativas no son importantes y por tanto puede primarse la rapidez en el algoritmo frente a la uniformidad.

4.2 Generadores de Fibonacci o aditivos

El generador de Fibonacci se basa en la secuencia de Fibonacci, es decir,

$$x_n = \left(x_{n-1} + x_{n-2}\right) mod\ m \tag{2}$$

Sin embargo, emplear una cadena de Fibonacci de forma directa, según la expresión (2), no produciría buenos resultados, puesto que las muestras estarían muy correlacionadas (hay una clara dependencia entre las muestras). Para evitar esto, los generadores de Fibonacci se generalizan en la siguiente expresión:

$$x_n = \left(x_{n-k_1} + x_{n-k_2}\right) mod\ m\ ,\quad 1 \leqslant k_1 < k_2 < m \tag{3}$$

donde k_1 *y* k_2 son valores enteros positivos. Este tipo de generadores, denominados de Fibonacci o aditivos, generalmente son buenos para $k_2 - k_1 > 16$ y presentan mejores resultados que los GCL. Sin embargo consumen más recursos, en concreto se debe mantener un *buffer* de $k_2 - k_1$ valores para poder generar la siguiente muestra. La inicialización también consta de k_2 valores, cuya elección tampoco es trivial, puesto que condicionará las características de la secuencia generada.

Un ejemplo de este tipo de generadores lo encontramos en el trabajo de Mitchell y Moore en el año 58 donde $k_1 = 24$, $k_2 = 55$, $x_0 .. x_{54}$ semillas impares y *m* par.

4.3 Generadores múltiplemente recursivos (GMR)

Desarrollados por l'Ecuyer (1994) y Knuth (1997), son una generalización de los generadores de Fibonacci. Su expresión es

$$x_n = \left(\sum_{i=1}^{k} a_i \cdot x_{n-i}\right) mod\ m \tag{4}$$

Este tipo de generadores pueden obtener un periodo máximo de $m^k - 1$, si se escogen convenientemente los valores de a_i, *k* y *m*.

4.4 Generadores no lineales

En este caso la eficiencia se ve comprometida, dado el aumento de cómputo de las operaciones, a cambio de una menor dependencia entre las muestras.

Un ejemplo sería $x_{n+1} = d \cdot x_n^2 + a \cdot x_n + b \, mod \, m$, conocido como generador cuadrático.

5 Combinación de generadores

Una técnica simple para mejorar generadores pseudoaleatorios consiste en combinar las secuencias producidas por varios de ellos. Existen múltiples posibilidades, de entre las que destacan las siguientes.

5.1 Método de selección aleatoria

Este algoritmo establece un vector *(V)* de *N* posiciones, en las que cada posición guarda un valor pseudoaleatorio. La idea de Mclaren y Marsaglia fue utilizar un generador para seleccionar el valor del índice del vector (y por tanto de la muestra a devolver) y otro generador para generar los valores del vector. El algoritmo sigue entonces la siguiente expresión:

- Se inicializa el vector con *N* valores generados con el generador *x(n)*
- Se genera un valor pseudoaleatorio y_k con otro generador *y(n)* de periodo *m*
- Se devuelve la muestra correspondiente al índice *j* del vector *V*. Lo habitual es calcular este j como $N \cdot y_k$
- Se genera una nueva muestra x_{k+1} con el generador *x(n)* y se sustituye la muestra existente en la posición *j* del vector *V* por ella.

5.2 Suma de valores pseudoaleatorios

5.2.1 XOR sobre valores pseudoaleatorios

Este mecanismo se basa en aplicar la expresión

$$x(n) = x_1(n) + x_2(n) + \ldots + x_k(n) \tag{5}$$

donde el operando suma realiza una operación XOR entre los valores involucrados de *k* generadores distintos. Este mecanismo fortalece la aleatoriedad de la muestra, al tiempo que puede emplearse para aumentar el periodo de la secuencia final.

5.2.2 Generador compuesto aditivo

Este mecanismo fue propuesto por l'Ecuyer en (1996) y consiste en sumar los resultados de k generadores pseudoaleatorios. En cierta medida sigue una expresión similar a la (4), propuesta para los GMR. En concreto la expresión seguida es

$$z(n) = \left(\sum_{i=1}^{k} b_i \cdot x_i(n) \right) mod \, m_1 \tag{6}$$

donde b_i son coeficientes enteros positivos, $x_i(n)$ son los valores producidos por los distintos generadores pseudoaleatorios y m_1 es el módulo del primero de ellos.

Por ejemplo, para el caso de $x_i(n)$ con *k* generadores GMR de periodo máximo *($P_i = m_i^k - 1$)*, el periodo del generador compuesto aditivo puede calcularse como

$$P = \frac{1}{2^{k-1}} \prod_{i=1}^{k} P_k \tag{7}$$

6 Generación de variables aleatorias

6.1 Método de la inversión

Sea F(x) la función de distribución de x. Dicha función cumple dos propiedades:

1) $0 \leq F(X) \leq 1$
2) $Si\ X_1 \leq X_2 \rightarrow F(X_1) \leq F(X_2)$

Bajo esas premisas, se pueden generar valores distribudios según $F(X)$ mediante la inversión de ésta última: $X=F^{-1}(U)$, donde U es una variable uniforme entre 0 y 1. El algoritmo a aplicar es por tanto:

a) Si $y=F(x)$, obtener la función inversa de F, es decir, $x=F^{-1}(y)$

b) Generar un valor uniforme U entre [0,1)

c) Obtener x como $x=F^{-1}(U)$

d) Repetir desde el paso b)

6.2 Método de la aceptación/rechazo (hit and miss)

Consiste en generar una variable aleatoria Y con fdp $f(y)$ a partir de otra variable X con fdp $g(x)$ más fácil de generar. El algoritmo es como sigue:

a) Generar x_i con fdp $g(x)$

b) Generar u_i uniforme en [0,1)

c) Si $u_i =< f(x_i) / t(x_i)$ se entrega x_i, en otro caso volver al principio

NOTA: $t(x) = c \cdot g(x)$ tal que la constante c debe garantizar que $t(x)$ acote superiormente a $f(x)$ en todo punto. Se deber tener presente que la elección de c condiciona completamente la eficiencia del método, puesto que la probabilidad de aceptación es $1/c$.

6.3 Métodos específicos

Son métodos especialmente diseñados para determinadas funciones densidad de probabilidad como por ejemplo la normal. Entre ellos se encuentra el método de Box-Muller a partir del cual se desarrollaron otros métodos polares más eficientes. En particular el algoritmo calcula 2 variables aleatorias θ y r tal que θ está uniformemente distribuida en $(0, 2\pi)$ y r es la raíz cuadrada de $(-2 \cdot lnU)$ con U uniforme en [0,1). Dados θ y r se calcula finalmente $x=r \cdot cos\ \theta$ e $y=r \cdot sen\ \theta$ que son dos variables aleatorias independientes e idénticamente distribuidas ~ N(0,1). De esta forma:

$$x = \sqrt{\left(-2\ln\left(U_1\right)\right)}\cos\left(2\pi U_2\right) \quad \text{e} \quad y = \sqrt{\left(-2\ln\left(U_1\right)\right)}\sin\left(2\pi U_2\right) \quad \text{, con} \quad x, y \sim N(0,1) \tag{8}$$

6.4 Método de la convolución

Este método consiste en generar una variable aleatoria mediante la combinación lineal de varias variables aleatorias (X_i):

$$X = \sum_{1}^{K} a_i X_i \tag{9}$$

El valor de cada una de las variables X_i se puede obtener empleando los métodos ya explicados. Un ejemplo simple de este método se puede encontrar en el método de las 12 uniformes para la

generación de variables normales, basado en el teorema central del límite, por el que combinando 12 variables uniformes independientes e idénticamente distribuidas, se puede obtener una variable normal de media 0 y desviación típica 1.

6.5 Método de la composición

Este mecanismo consiste en la generación de una variable aleatoria *X* mediante una función aditiva que combina linealmente diversas funciones sencillas para conseguir la función de densidad de probabilidad de *X*, es decir *f(x)*:

$$f(x) = \sum_{1}^{K} p_i f_i(x) \tag{10}$$

donde p_i son los pesos aplicados a cada una de las funciones f_i que componen f y se cumple que

$$\sum_{1}^{K} p_i = 1 \tag{11}$$

El algoritmo de generación es el siguiente:

a) Generar un valor U uniforme en (0,1)

b) Escoger la función f_i de acuerdo al valor de U y los pesos proporcionados:

· Si $U < p_1 \rightarrow f_1$

· Si $p_1 \leq U \leq p_2 \rightarrow f_2$

.....

· Si $p_{k-1} \leq U \leq p_k \rightarrow f_k$

c) Obtener *X* a partir de la función f_i seleccionada a partir de un método conocido, como por ejemplo el de la inversión, de la aceptación/rechazo, etc.

7 Calidad de un generador

Para estimar la calidad de un generador existen diferentes pruebas a realizar. Cada una de ellas tiene como propósito verificar las cualidades deseables de un generador, tal y como se indicaron en la sección 2.

7.1 Cálculo del periodo

Una de las cualidades de un generador es que tenga un periodo próximo o igual a *m* y obviamente con un valor altísimo. Por ello es necesario calcular el periodo del mismo, una tarea que puede tener un coste computacional importante, puesto que, a falta de modelos analíticos que simplifiquen el cálculo, se deberían generar tantas muestras como permite la base modular *m* en la que se está operando. En su defecto podría optarse por establecer un cierto límite (por ejemplo 10.000.000 de muestras o más) y verificar si la secuencia generada presenta un periodo inferior a ese valor. De esta forma, se establecería al menos un discriminador entre generadores a partir de su periodo.

7.2 Independencia entre muestras

Esta prueba pretende asegurar que las muestras generadas son independientes, es decir, que no hay una relación aparente entre las muestras generadas sino que éstas son aparentemente aleatorias.

Hay múltiples pruebas que se pueden realizar para verificar la independencia entre muestras.

7.2.1 Gráficas de autocorrelación

Las gráficas de autocorrelación consisten en calcular la correlación estadística entre la muestra y la muestra desplazada *k* muestras. Por ejemplo, la autocorrelación con un desplazamiento de 1 verificaría que no hay dependencia entre valores consecutivos: (x_1, x_2), (x_2, x_3), ..., (x_{n-1}, x_n). Lo deseable es que la muestra desplazada no muestre similitud con la muestra original, es decir, que no haya una dependencia entre la muestra *n* y la *n-k*.

La autocorrelación estadística se calcula mediante

$$R_k = \frac{C_k}{S^2} \quad \text{, donde } S \text{ es la desviación estándar y} \tag{12}$$

$$C_k = \frac{1}{n-k} \sum_{i=1}^{n-k} \left(x_i - E_x\right) \cdot \left(x_{i+k} - E_x\right) \tag{13}$$

$$E_x = \frac{1}{n} \sum_{i=1}^{n} x_i \tag{14}$$

La autocorrelación estadística proporciona valores entre -1 y 1. Si las muestras muestran una clara dependencia para el valor de desplazamiento escogido, la autocorrelación proporcionará valores muy cercanos a los límites (es decir, 1 o -1). Por el contrario, la independencia entre muestras resulta en valores de la autocorrelación próximos a 0.

De esta forma, la gráfica de autocorrelación consiste en representar la autocorrelación estadística de la muestra R_k en función del desplazamiento o retardo *k*. Los valores de *k* que más información aportan sobre la dependencia entre muestras son los comprendidos entre 1 y 10. Desplazamientos mayores de 20 ya no aportan información relevante.

7.2.2 Diagrama de dispersión

El diagrama de dispersión tiene por objetivo detectar dependencias entre muestras. Lo más habitual es emplear este diagrama para verificar la dependencia entre muestras consecutivas, si bien puede aplicarse entre muestras con mayor distancia entre ellas. Para el caso de muestras consecutivas, el diagrama de dispersión se obtiene representando los pares de valores (x_i, x_{i+1}) en un plano, es decir, de dibujar un punto en el plano XY para cada uno de los pares de valores. Si la muestra es independiente, se obtiene una nube de puntos sin ninguna forma aparente. Sin embargo, si existe una dependencia entre muestras, el diagrama de dispersión mostrará un cierto patrón. Por ejemplo, si la muestras generadas presentan una alta correlación, el diagrama de dispersión tenderá a una recta. Se recomiendan como mínimo 50 pares de valores para poder extraer conclusiones del diagrama de dispersión.

Téngase presente que el diagrama de dispersión también es generable en 3D. En ese caso las tripletas de valores representados sería (x_i, x_{i+1}, x_{i+2}).

7.3 Distribución de probabilidades

Son muchas la pruebas que pueden hacerse para verificar la distribución de probablidad de la secuencia obtenida y por tanto del buen hacer del generador analizado. Las más relevantes son las siguientes.

7.3.1 Verificación por momentos

Consiste en verificar que valores estadísticos puntuales de la secuencia obtenida, coinciden con los esperados. En este apartado aparece normalmente el cálculo de la media y la varianza, entre otros valores.

7.3.2 Función densidad de probabilidad

Por ejemplo para una variable aleatoria uniforme consiste en verificar que la fdp de la secuencia es ecuánime entre los distintos valores, es decir, que representa una densidad de probabilidad uniforme entre los valores esperados. Debe notarse que las muestras con las que se trabaja son muestras finitas, por lo que no pueden representar distribuciones continuas como la uniforme. Para realizar representaciones continuas a partir de conjuntos finitos de muestras se emplean los histogramas.

La obtención de un histograma a partir de una secuencia es simple:

- Se agrupan los distintos valores de la secuencia en un conjunto reducido de intervalos o categorías, de tal forma que el total de los intervalos cubra el conjunto de valores a reproducir. Por ejemplo, si se desea representar una Uniforme[0,1] y se quiere trabajar con 10 intervalos, éstos serán: { [0, 0.1) , [0.1, 0.2), ... [0.8, 0,9), [0.9, 1] }
- Se cuentan los valores que caen en cada uno de los intervalos *(Y)*. Ese será el valor que se asociará al intervalo en cuestión. Si se desea mostrar la función de densidad de probabilidad, basta con hacer que

$$Y_j = Y_j \cdot \left(\sum_{k=1}^{N_i} Y_k \right)^{-1} \tag{15}$$

- Se representa el número de repeticiones por intervalo (*Y*) en función del intervalo

El número de intervalos del histograma es crítico a la hora de proceder a su representación. Un valor insuficiente de intervalos motiva que el histograma no muestre información suficiente como para discernir la distribución seguida. Por contra, un valor excesivo en el número de intervalos hace que el histograma sea demasiado sensible a la secuencia, con lo que no es posible extrapolar la distribución de probabilidad que subyace en ella. Los valores habituales del número de intervalos se sitúan entre 10 y 20. Existen múltiples propuestas para calcular el número de intervalos óptimo para un número de muestras determinados. Todos ellos son meras aproximaciones y por tanto los valores obtenidos deberán ser ajustados en función de la muestra. Uno de los mecanismos más conocidos es el propuesto por Sturges, que sigue la expresión

$$M = 1 + \log_2(n) \tag{16}$$

donde *M* es el número de intervalos y *n* el número de muestras de la secuencia. Téngase presente que si *M* es real debe considerarse el entero superior.

Debe tenerse muy presente que un histograma intenta representar la distribución de probabilidad de la variable aleatoria que representan los valores de la secuencia obtenida. La alta dependencia del histograma con el número de intervalos y la subjetividad de la interpretación hace que esta herramienta no sea excesivamente fiable por sí sola, si bien puede ayudar a la hora de orientar el estudio de una variable aleatoria.

En los casos en los que se tienen indicios sobre la función de densidad de probabilidad *(fdp)* que genera la variable aleatoria considerada, se puede proceder a realizar una comparación de frencuencias entre el histograma empírico y el histograma obtenido a partir de *fdp*.

7.4 Test de hipótesis

Los test de hipótesis tienen como propósito verificar la certeza de una hipótesis planteada sobre la secuencia a estudiar. Según el propósito de la hipótesis, se distinguen dos tipos de test de hipótesis:

- Paramétrico, donde la hipótesis está referida a los parámetros que se extraen de la secuencia (por ejemplo el valor medio)
- No paramétrico, donde el propósito de la hipótesis es identificar la función de distribución que rige la generación de la secuencia. Este tipo de tests se conocen también bajo el nombre de test de bondad de ajuste, puesto que cuantifican cómo se ajusta una distribución a una secuencia determinada

En la asignatura nos vamos a centrar en los test de hipótesis no paramétricos, puesto que lo que se pretende es verificar si la secuencia obtenida ha podido ser generada mediante una distribución uniforme U[0,1]. Los pasos a seguir para realizar un test de hipótesis son los siguientes:

1. Establecer la hipótesis a verificar (llamada hipótesis nula o H_0). En el caso que estudiaremos, esta hipótesis será la siguiente: "La secuencia ha sido generada mediante una variable aleatoria uniforme entre 0 y 1"
2. Establecer el nivel de error asumible *(α)*. Este error, llamado de primera especie, contempla la posibilidad de rechazar la hipótesis aún cuando ésta sea cierta. De igual forma que en el caso de los intervalos de confianza, *1- α* sería el nivel o grado de confianza con el que se puede rechazar (o aceptar) la hipótesis nula
3. Llevar a cabo un test estadístico sobre la muestra y verificar su resultado de acuerdo al error establecido. Rechazar la hipótesis nula de acuerdo al valor obtenido tras el test

Existen múltiples test estadísticos con los que implementar el punto 3. Uno de los más generalizables (es decir, que puede ser empleado con independencia de la distribución que subyace en la secuencia) es el test de Chi-cuadrado (conocido también como Test de Pearson). Para el cálculo del test estadístico empleado en el test de Chi-cuadrado se siguen los siguientes pasos

3.1 Obtener el histograma de la secuencia. Obtener la frecuencia asociada a cada una de las categorías (es decir, el número de muestras asignadas a cada categoría). Este valor se denominará O_i, donde *i* indica la categoría

3.2 Calcular el número de valores teóricos que deberían esperarse en cada categoría, según la distribución indicada en la hipótesis nula. Este valor se conocerá como E_i. En el caso que nos aplica (distribución uniforme), E_i es igual a L / r, siendo L la longitud de la secuencia y r el número de categorías empleado en el histograma del punto anterior

3.3 Calcular el estadístico del test mediante la expresión

$$K_{\alpha,r} = \sum_{i=1}^{r} \frac{(O_i - E_i)^2}{E_i} \tag{17}$$

3.4 Se puede demostrar que $K_{\alpha,r}$ sigue una distribución de χ^2 con *r-1* grados de libertad (Cramér, 1946). De esta forma, se rechaza la hipótesis nula si el valor de $K_{\alpha,r}$ supera al valor de χ^2 esperado para un nivel de confianza *1-α* y *r-1* grados de libertad. Dicho de otra forma, se puede rechazar la hipótesis nula con un *(1-α)%* de seguridad si:

$K_{\alpha,r} > \chi^2_{\alpha,r-1} \rightarrow$ rechazar la hipótesis nula (H_0)

Los valores de α más habituales son el 5%, 2.5% y 1%

7.5 Test para secuencias binarias

Los tests estadísticos para secuencias bianarias más destacados están especificados por el NIST (National Institute of Standards and Technology) y normalmente quedan recogidos y en documentos llamados FIPS (Federal Information Processing Standard). Los más utilizados son los FIPS 1401, de Enero de 1994 y FIPS 1402, de Noviembre de 1999. Para la ejecución de estos tests, se necesitan 20.000 bits consecutivos producidos por el generador de números aleatorios.

1. *Test monobit.* Se basa en asegurar que el número de unos y ceros de la secuencia binaria correspondiente a toda la muestra es idéntica (realmente se permite una unidad de diferencia entre ambos valores)
2. *Test de poker.* Se divide la secuencia en bloques de n bits (normalmente n es igual a 4). Se realiza un análisis frecuencial sobre los 2^n valores generados.
3. *Test de rachas.* Se cuentan las rachas de 1 o más bits, sean unos o ceros (las secuencias superiores a 6 se considerarán de esa longitud). El test se pasa si las rachas, tanto de unos como de ceros, están dentro de unos intervalos preestablecidos.
4. *Test de rachas largas.* Verifica que en la secuencia no existe una ráfaga de unos o ceros demasiado larga. Una racha larga se define como una racha de longitud 26 o más, ya sea de ceros o de unos. El test se supera si no existen rachas largas.

8 Entorno de simulación

En esta práctica se implementarán una serie de generadores pseudoaleatorios y se procederá a su evaluación. Para generar los valores aleatorios emplearemos *OMNeT++*, un simulador de eventos discretos que se empleará a lo largo de todo el curso. Junto a la documentación de la práctica, hay una carpeta llamada *codigo* que contiene el código de los generadores empleados.

La Figura 1 muestra el modelo a simular en *OMNeT++*. Tal y como puede apreciarse, consta de 3 módulos. Los módulos *startSim* y *endSim* sirven para marcar el inicio y el final de la simulación, mientras que *rngGenerator* es el módulo encargado de generar los valores pseudoaleatorios. El número de valores a generar, así como los parámetros de cada generador utilizado, deberán indicarse en el fichero *omnetpp.ini*, antes de al lanzar la simulación.

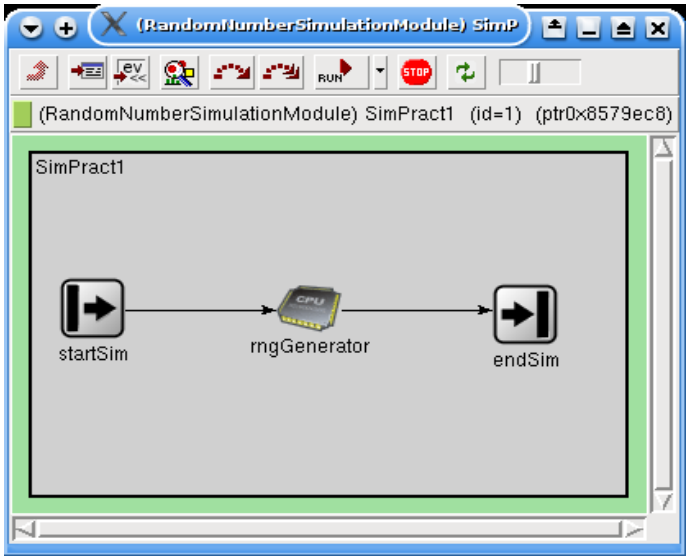

Figura 1: Modelo simulado en Omnet++ para la evaluación de generadores pseudoaleatorios

9 Práctica

9.1 Estudio previo

En la presente práctica se requieren ciertos conocimientos de programación y procesado estadístico. Es por ello que se insta al alumno a refrescar dichos aspectos en caso de que lo considere oportuno. La recomendación pasa por recuperar (o adquirir) unos mínimos conocimientos en:

- Contenidos generales de estadística, impartidos en la asignatura de PIPE
- Matlab. Se puede encontrar un sencillo manual introductorio en ATENEA

9.1.1 Estudio previo 1

Realice una función en matlab que tenga por entrada un vector de muestras aleatorias (**v**) y que dé como resultado:

- El periodo, mediana, media y varianza
- El histograma de frecuencias
- La función de autocorrelación
- El diagrama de dispersión (2D/3D)

Para realizar esta función puede utilizar las siguientes funciones disponibles en matlab: mean, median, std, var, hist, xcorr, plot, plot3, sort. Puede hallar más información sobre estas funciones en la ayuda de matlab (help nombre_funcion).

9.2 Ejercicios

Compile el modelo simulado. Para ello abra una consola y sitúese en el directorio donde está el código fuente del modelo a simular. Escriba los siguientes comandos (tenga presente que # indica el prompt de la consola y que por tanto no debe ser escrito):

```
# opp_makemake -f
# make
```

Estas dos instrucciones generan un fichero ejecutable que tendrá el mismo nombre que la carpeta en la que esté situado (si no se ha cambiado se llamará *codigo*) .

Ejecute el simulador (./codigo desde la consola). Al hacerlo aparecerá un entorno gráfico y un combo-box en el que se podrán seleccionar los distintas entornos a simular. En nuestro caso, cada entorno (denominado Run) cargará los parámetros correspondientes a un generador distinto.

Tenga en cuenta que modificar un fichero *.cc supone compilar de nuevo, es decir, ejecutar el comando ./make desde la consola.

9.2.1 Ejercicio 1 (Sesión 1)

Edite el fichero *omnetpp.ini* y descomente la sentencia *rng*-class=*LT3LCG*. Con ello se indica que el generador pseudoaleatorio a utilizar es el *LCG* implementado en la asignatura (fichero *LT3LCG.cc*). Cargue los valores siguientes, en función del generador a analizar:

Parámetro	**Sección**	**LCG_A**	**LCG_B**
a	[LCG 1]	*23445*	*720908*
b	[LCG 1]	*7369*	*8161*
m	[LCG 1]	*19946*	*65537*
seed-1-lcg	[Run 0]	*7247*	*23*

Lance la simulación y escoja el [Run 0]. Establezca el valor de *number_of_random_numbers* en el fichero *omnetpp.ini* para que se generen *10000* valores pseudoaleatorios en el rango [0,1). Proceda a evaluar la calidad del generador mediante las técnicas descritas en el manual de la práctica. En concreto:

- Represente el histograma de frecuencias
- Represente la función de autocorrelación
- Represente el diagrama de dispersión (2D/3D)
- Calcule el periodo y los momentos básicos (media, mediana, varianza, ...)

9.2.2 Ejercicio 2 (Sesión 1)

Edite el fichero *omnetpp.ini* y descomente la sentencia *rng-class=LT3Seleccion* (asegúrese de comentar la línea descomentada en el Ejercicio 1). Con esto se establece como generador pseudoaleatorio una implementación del generador de Selección en la que se dispersan los valores de un *LCG* (A ó B en la sección *[LCG 1]* del fichero *omnetpp.ini*) mediante un segundo generador *LCG* (A ó B en la sección *[LCG 2]*). Establezca el valor del tamaño del buffer (*bufferSize*) a *100*. Emplee los mismos valores de los *LCG* utilizados en el Ejercicio 1 (se puede aleatorizar, por ejemplo, el LCG_A con él mismo). Genere *10000* valores pseudoaleatorios en el rango *[0,1)* con dicho generador y proceda a evaluar su calidad de igual forma a la descrita en el Ejercicio 1.

9.2.3 Ejercicio 3 (Sesión 1)

Comente las ventajas e inconvenientes de cada uno de los generadores utilizados de acuerdo a los resultados obtenidos. ¿Qué generador propondría para utilizar a la hora de simular futuros modelos? Compare los resultados con los obtenidos mediante el generador *Twister-Mersenne*. Para ello asegúrese de que la única línea descomentada del tipo *rng-class* sea la siguiente: *rng-class="cMersenneTwister"*.

9.2.4 Ejercicio 4 (Sesión 1 y 2)

Genere una variable aleatoria exponencial por el método de la inversión con el generador de *Twister-Mersenne* y el peor de los generadores que haya construido. Para ello:

1) Edite el fichero *lt3_funtions.cc* y rellene el código necesario para la generación del valor exponencial en la función *double lt3_exponencial (media)*
2) Edite el fichero *omnetpp.ini*. Descomente la línea adecuada para generar una variable aleatoria exponencial de media igual a *7*

Obtenga una estimación de la función densidad de probabilidad mediante las siguientes técnicas:

- Histograma normalizado y comparación de frecuencias
- Utilice la función *ksdensity* de *Matlab*

Comente y compare las gráficas obtenidas.

9.2.5 Ejercicio 5 (Sesión 2)

A partir de la variable aleatoria exponencial es fácil generar una variable aleatoria discreta geométrica tomando simplemente la parte entera del valor de la variable exponencial. Demuestre tal afirmación obteniendo la relación entre la *media* de la exponencial y el parámetro *p* de la geométrica. Genere el código necesario en la función *double lt3_geometrica (media)* del fichero *lt3_funtions.cc* y obtenga una estimación de su función de probabilidad cuando la media de la exponencial es *7*. Compárela gráficamente con la proporcionada por *Matlab* y por la que genere en *Matlab* a partir de la expresión analítica de la función de probabilidad de una variable geométrica. Proporcione además el coeficiente de variación (desviación típica/media) y comente el resultado en función del generador pseudoaleatorio utilizado (*Twister-Mersenne* o *LCG*).

Un ejemplo donde se utiliza la distribución geométrica es en el modelado del tamaño de un paquete dentro de un determinado intervalo $[v_{min}, v_{max})$. Por ello se pide que modifique la función creada para tener en cuenta este intervalo (*double lt3_geometrica* (v_{min}, v_{max}, *media*)). Para facilitar la tarea se recomienda acotar en primer lugar a un valor máximo v_{max}-v_{min}, validar esta modificación y entonces acotar el valor mínimo trasladando la distribución con un offset igual a v_{min}. Proporcione una estimación de su función de probabilidad. Compárela con las obtenidas anteriormente y con la que genere teóricamente en *Matlab* teniendo en cuenta este intervalo. Comente los resultados considerando también el *Twister-Mersenne* y el peor de los congruenciales del Ejercicio 1.

Utilice los valores siguientes para realizar el ejercicio: $v_{min}=1$, $v_{max}=15$ y $media=7$ (valor medio de la exponencial asociada).

9.2.6 Ejercicio 6 (Sesión 2 y 3)

En este ejercicio se pretende generar una variable aleatoria normal de media nula y desviación estándard 1 por tres métodos conocidos:

1.- Utilice el método de la aceptación/rechazo de muestras a partir de la función densidad de probabilidad $g(x)=e^{-x}$. Puede utilizarse la constante $c=\sqrt{\frac{2e}{\pi}}$ para acotar la función normal a partir de g(x). Téngase presente que la generación de valores normales mediante aceptación/rechazo sólo generará valores positivos. Dada la simetría de la función de densidad normal, bastará con emplear una variable aleatoria uniforme para decidir el signo final.

2.- Utilice el método de Box-Muller.

3.- Utilice el método de la convolución a partir de la suma de n variables aleatorias uniformes en el rango [0,1). Probar con $n=2$ y con $n=12$.

Para ello edite el fichero *lt3_funtions.cc* e implemente el código necesario. Edite el fichero *omnetpp.ini* y seleccione la variable aleatoria correspondiente mediante los parámetros y método adecuados.

Proporcione una representación gráfica de las funciones densidad obtenidas, calcule las medias y desviaciones típicas, y muestre los diagramas de dispersión para los tres métodos anteriores.

Valore la influencia del generador uniforme utilizado. En concreto utilice los generadores *Twister-Mersenne* y LCG_A.

9.2.7 Ejercicio 7 (Sesión 3)

Construye la variable aleatoria *T* como *X+Y+Z*, donde *X, Y, Z* son independientes y están idénticamente distribuidas con una exponencial de media $1/\lambda=7$. Para ello edite el fichero *lt3_funtions.cc* e implemente el código necesario. Fíjese en los ya implementados para ello. Edite el fichero *omnetpp.ini* y seleccione la variable aleatoria correspondiente mediante los parámetros adecuados, de forma similar a como se hizo en el Ejercicio 4 .

Utilice el histograma normalizado para estimar la función densidad de probabilidad obtenida y proporcione el ajuste con 2 momentos utilizando la función *gamma* $G(\alpha,\beta)$. Aproxime α por un entero y proporcione también gráficamente el ajuste. Encuentre asimismo la función densidad de probabilidad de *T* de forma analítica (por ejemplo a partir de la convolución) y compárela con la obtenida por el ajuste.

9.2.8 Ejercicio 8 (Sesión 3 y 4)

La distribución de Chi-cuadrado con *k* grados de libertad no es más que la suma de *k* variables aleatorias normales independientes de media nula y desviación estándard 1, elevadas cada una de ellas al cuadrado. Para ello edite el fichero *lt3_funtions.cc* e implemente el código necesario. Fíjese en los ya implementados para ello. Edite el fichero *omnetpp.ini* y seleccione la variable aleatoria correspondiente mediante los parámetros adecuados.

Compruebe en qué se transforma su función densidad de probabilidad cuando *k* es 2 y encuentre un método de generación eficiente. Verifique la tabla de percentiles de esta distribución para valores de *k* par y *k* grandes. Para ello utilice como ejemplo 4, 6 y 13 grados de libertad y percentiles del 5%, del 10% y del 95%.

9.2.9 Ejercicio 9 (Sesión 4)

Utilice el generador *Twister-Mersenne* y el peor de los generadores construidos para comprobar si éstos superan el *test de Pearson* teniendo en cuenta 6, 24 ó 100 categorías. Asimismo compruebe si la generación del tamaño del paquete del ejercicio 5 supera el *test de Pearson*.

Observaciones:

- Para realizar la evaluación de los generadores puede emplearse cualquier paquete de software adicional, si bien se aconseja el uso de *matlab* por su sencillez y enfoque. Para agilizar la realización de los ejercicios es conveniente realizar una función de *matlab* que secuencie las pruebas a realizar. Muchas de las pruebas descritas pueden llevarse a cabo de forma sencilla con este paquete matemático
- Para generar semilla válidas podéis emplear la herramienta *seedtool*, que se ejecuta desde consola. Su mecanismo de utilización es:

  seedtool g valor_inicial separación_entre_semillas número_de_semillas

  Por ejemplo *seedtool g 1 10000000 5* genera 5 semillas con 10000000 de valores entre ellas partiendo del valor inicial 1
- *OMNeT++* guardará las muestras generadas en un fichero llamado *omnetpp.vec*. Este fichero consta de 3 columnas: id de la variable guardada, instante temporal en el que se grabó la muestra y valor de la muestra. Incluye además una cabecera por lo que tendrá que ser procesado para eliminar tanto las columnas que no interesen como la cabecera. Para obtener un fichero con las muestras únicamente puede emplearse el comando

 # awk '{if ($1==0) print $3}' < omnetpp.vec > samples.txt

que selecciona la columna *3* (valor de las muestras) de la variable con identificador 0 (la única en este caso) del fichero *omnetpp.vec* y la guarda en el fichero *samples.txt*. Dicha selección puede hacerse también en *matlab*

- Las tablas de la función de χ^2 se pueden encontrar junto a la documentación de la práctica
- Para los test de bondad de ajuste emplead un α de 0.05, a menos que se indique lo contrario

10 Anexo A: Distribuciones utilizadas

Distribución	fdp	Media	Varianza
Uniforme	$f(x)=\frac{1}{(b-a)}$	$E(x)=\frac{(b-a)}{2}$	$Var(x)=\frac{(b-a)^2}{12}$
Exponencial	$f(x)=\lambda e^{-\lambda x}$	$E(x)=\frac{1}{\lambda}$	$Var(x)=\frac{1}{\lambda^2}$
Normal	$f(x)=\frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$	μ	σ^2
Gamma	$f(x)=\beta^{\alpha}e^{-\beta x}\left[\frac{(x)^{\alpha-1}}{\Gamma(\alpha)}\right]$ $\Gamma(\alpha)=(\alpha-1)!, \text{con } \alpha \text{ entero}$	$E(x)=\frac{\alpha}{\beta}$	$Var(x)=\frac{\alpha}{\beta^2}$
Chi-2	$f(x)=\frac{(1/2)^{\frac{k}{2}}}{\Gamma(k/2)}x^{\left(\frac{k}{2}-1\right)}e^{-\frac{x}{2}}$	$E(x)=k$	$Var(x)=2k$