Generación de números aleatorios

Marcos García González (h[e]rtz)

Verano 2004

Documento facilitado por la realización de la asignatura Métodos informáticos de la física de segundo curso en la universidad autónoma de Barcelona en el transcurso del año 2003-2004.

1. Números uniformemente distribuidos

Un problema básico que nos encontramos habitualmente es el de obtener secuencias de números uniformemente distribuidos en un intervalo $[0, 1]$.

La diferentes posibilidades para resolver dicho problema son:

i) Buscar en tablas de números aleatorios publicadas (libros, internet ...);

ii) Observar un proceso físico tal como la desintegración radiactiva, el ruido eléctrico ...;

iii) Los lenguajes de programación y las hojas electrónicas incluyen una función para generarlos

iv) Mediante **algorismos de generación de números aleatorios**

Las principales **ventajas** de los generadores de números aleatorios son:

- Rapidez
- Comodidad
- Reproducibilidad
- Portabilidad

Y la **desventaja** fundamental:

- Las secuencias obtenidas no son realmente aleatorias, ya que se obtienen con operaciones deterministas. Solo podemos obtener secuencias *pseudo-aleatorias*, que a su vez satisfacen algunos criterios de aleatoriedad adecuados.

Los números generados deben cumplir ciertas características para que sean válidos. Dichas características son:

1. Uniformemente distribuidos.
2. Estadísticamente independientes.
3. Su media debe ser estadísticamente igual a 1/2.
4. Su varianza debe ser estadísticamente igual a 1/12.
5. Su periodo o ciclo de vida debe ser largo.
6. Deben ser generados a través de un método rápido.
7. Generados a través de un método que no requiera mucha capacidad de almacenamiento de la computadora.

Normalmente se utilizan números enteros, ya que su aritmética es exacta y rápida. Se generan enteros N_i entre 0 y $M-1$, y $x_i = N_i/M$ da valores reales en el intervarlo $[0, 1)$.

En general los algoritmos utilizan relaciones de recurrencia del tipo

$$N_i = f(N_{i-1})$$

en el caso de recurrencia simple, o bien

$$N_i = f(N_{i-1}, \ldots, N_{i-k})$$

para el caso de una recurrencia de orden k.

Se necesitará dar un *valor inicial* para comenzar el algoritmo (k valores para recurrencias de orden k).

1.1. Generadores de congruencia lineal (GCL)

Estos generadores son los más utilizados y los más conocidos. Se basan en la relación de recurrencia

$$N_{i+1} = (aN_i + c) \bmod m$$

donde a es el multiplicador y m el módulo.

- Hay m valores posibles de N_i, entre 0 i $m-1$.

- La secuencia es *periodica*: cuando vuelve a aparecer un número por segunda vez, la secuencia se vuelve a repetir. El periodo depende de los valores de a, c y m, así como del valor inicial; nótese que el máximo posible es m.

Recordemos que lo que nos interesa para trabajar con un buen generador de números aleatorios es que la distribución de los números obtenidos tiene que ser uniforme, no deben de haber correlaciones entre los terminos de la secuencia, el periodo debe ser lo más largo posible, y el algorismo debe ser de ejecución rápida.

1.1.1. Mejora de los generadores de congruencia lineal

Las limitaciones más importantes de los generadores son su periocidad (normalmente el periodo no suele ser más grande de $2^{32} \approx 4 \times 10^9$) y la posible presencia de correlaciones entre términos consecutivos de la secuencia. Una manera sencilla de suprimir éstas limitaciones es "desordenar" un poco la secuencia mediante el siguiente procedimiento:
Se parte de un generador que da enteros aleatorios entre 0 y $m-1$, y en primer lugar se genera con el GCL un vector que contiene una lista de N enteros aleatorios j_n, así como un entero aleatorio y. Se determina el índice $k = [y * N/m]$, entre 0 y $N-1$.
El elemento j_k de la lista se da como un nuevo nombre aleatorio, y se reasigna a la variable y el valor j_k. El valor de j_k se renueva con el GCL, y se vuelve a repetir los pasos desde la determinación del índice k.

1.2. Generadores de desplazamiento de bits

En estos generadores cada nuevo número entero aleatorio N_i, se obtiene manipulando los bits del número anterior, N_{i-1}. En lenguaje C, esto se puede hacer facilmente utilizando operadores sobre bits, $>>, <<, \wedge, |, \&$.

1.3. Generadores de Fibonacci

Las grandes ventajas de estos generadores es que son generadores muy rápidos que tienen un periodo muy largo. La fomentación teorica en la que se basan es diferente a la de los GCL. Los generadores de Fibonacci se basan en una recurrencia del tipo

$$N_i = (N_{i-r} \circ N_{i-s}) \text{mod } m$$

donde $r < s$ son enteros dados y $\circ$ denota alguna de las operaciones $+, -, \times, \wedge$. Este tipo de generador precisa iniciar (con otro generador) y mantener una lista de los últimos s números generados.

Otros tipos de generadores los podemos encontrar en:

- W.H. Press, S.A. Teukolski, W.T. Vetterling i B.P. Flannery, *Numerical Recipes in C*, Cambridge University Press.

-D.E. Knuth, *The Art of computing programming, 2: Seminumerical Algorithms,* Addison-Wesley.

Antes de aceptar un nuevo generador hace falta verificar que satisface una série de pruebas, lo que llamaremos **pruebas de aleatoriedad**.

1.4. Pruebas de aleatoriedad

Éstas consisten basicamente en realizar dos tipos de pruebas, empíricas y teoricas. Si los generadores superan estas pruebas, podremos asegurar que estamos ante un generador de números aleatorios bastante competente.

Pruebas empíricas (sobre la muestra de la secuencia)

- Test de uniformidad: hace falta que los valores esten uniformemente distribuidos en $[0, 1]$. Se puede realizar un test χ^2. Alternativamente, podemos estimar los momentos de orden k, $X^k = 1/N \sum_i (x_i)^k$ y comprobar que se aproximan a sus correspondientes valores teoricos, $1/(k+1)$.

- Test serial: se generan parejas de valores (x_1, x_2), y se comprueba si se distribuyen uniformemente en el cuadrado $[0, 1] \times [0, 1]$.

- Test de correlaciones: se determina la correlación entre números separados k lugares en la secuencia, $C(k) = 1/N \sum_i x_i x_{i+k}$. Su valor tendría que acercarse a $1/4$.

Pruebas teoricas (sobre toda la secuencia): La sencillez de los generadores de congruencia lineal permiten demostrar propiedades importantes:

- Para determinar valores de a, c i m se obtienen secuencias de periodo máximo m. Por ejemplo, si m es una potencia de 2, bastará con que c sea impar y a sea igual a un múltiple de 4 mas 1.

- Test espectral: si se forman vectores con k valores consecutivos,

$$\mathbf{u}_n = (x_n, x_{n+1} \dots x_{n+k}),$$

estos forman hiperplanos paralelos en el espacio k-dimensional. La separación d_k entre los planos tiene que ser la mínima posible.

2. Distribuciones no uniformes

El problema a tratar será el de obtener una secuencia con densidad de probabilidad dada $w(y)$, definida en el intervalo (y_{min}, y_{max}) a partir de una secuencia de números con distribución uniforme $U(0, 1)$.

Para resolver este problema utilizaremos el **método del cambio de variable**. Que concretamente consistirá en buscar una transformación $y = f(x)$ para obtener la distribución deseada. Si la densidad de probabilidad de x es $p(x)$, tenemos:

$$P(X \in (x, x + \mathrm{d}x)) = P(Y \in (y, y + \mathrm{d}y));$$

$$p(x)\mathrm{d}x = w(y)\mathrm{d}y;$$

Si $x \in U(0, 1)$ la integración de esta ecuación resulta ser

$$\int_0^x \mathrm{d}x = \int_{y_{min}}^{y} w(y')\mathrm{d}y'.$$

Si sabemos calcular la integral, obtenemos una relación del tipo $X = g(Y)$, que se tiene que invertir para poder obtener $Y = f(X)$.

2.1. Ejemplos:

Distribución exponencial. Para obtener Y con distribución

$$w(y) = \begin{cases} 0, & y < 0 \\ \lambda e^{-\lambda y}, & y \in [0, +\infty) \end{cases} \tag{1}$$

entonces nos queda la ecuación

$$x = \int_0^y \lambda e^{-\lambda y'} \mathrm{d}y' = 1 - e^{-\lambda y'}$$

por lo tanto, el cambio adecuado será $y = -\frac{1}{\lambda}\ln(1-x)$.

Distribución normal. Para obtener $Y \in N(0,1)$, nos queda la siguiente ecuación

$$x = \int_{-\infty}^{y} e^{-y'^2/2} \frac{\mathrm{d}y'}{\sqrt{2\pi}}$$

La integral no se puede resolver analíticamente, y menos aun invertir la relación para obtener $Y = f(X)$. La alternativa es aplicar el **algoritmo de Box-Muller.** Que permite obtener 2 valores independientes x_1, x_2 con distribución $N(0,1)$: si x_1, x_2 son las coordenadas de un punto del plano, la densidad de probabilidad de sus coordenadas polares es

$$w(r,\theta) = p(x_1,x_2)\left|\frac{\partial(x_1,x_2)}{\partial(r,\theta)}\right| = \frac{1}{2\pi} r e^{-r^2/2}, \quad \theta \in [0, 2\pi], \quad r \in [0, +\infty)$$

Por lo tanto, $\theta \in U(0, 2\pi)$ i r tiene la densidad de probabilidad $re^{-r^2/2}$. Se pueden obtener a partir de 2 números $z_1, z_2 \in U(0,1)$ con las transformaciones $\theta = 2\pi z_1$, $r = \sqrt{-2\ln z_2}$. Por lo tanto x_1, x_2 se pueden obtener con las transformaciones

$$\begin{cases} x_1 = \sqrt{-2\,\ln(z_2)}\,\cos(z_1) \\ x_2 = \sqrt{-2\,\ln(z_2)}\,\sin(z_1) \end{cases} \tag{2}$$

Método de aceptación-rechazo: Es un método sencillo y general, aunque en ocasiones no muy eficiente.
Sea $p(x)$ definida en un intervalo finito, (a,b), y M una cota superior de $p(x)$. Se generan dos valores aleatorios $x \in U(a,b)$ y $p \in U(0,M)$. Entonces:

$$\begin{cases} \text{se acepta } \ x \text{ si } p(x) \geq p \\ \text{se rechaza } \ x \text{ si } p(x) < p \end{cases} \tag{3}$$

De esta manera el valor x aparece con una densidad de probabilidad $p(x)$, aunque no se aprovechan todos los valores que obtenemos en la realización del método. De todos modos, este es el único método disponible cuando la distribución de probabilidades es complicada. De hecho es la base del **algoritmo de Metropolis**, posiblemente el más utilizado en física computacional.