Curso:
Confiabilidad estructural de componentes mecánicos con daño
edición 2020
Sesión 7: Números seudo-aleatorios

docentes del curso: R. Mussini, H. Cancela

dictado semestre 1 - 2020

Contenido:

1. Números aleatorios.
2. Fuentes de números aleatorios.
3. Dispositivos de hardware.
4. Generadores software/hardware.
5. Fuentes en Internet.
6. Ejercicio.

Números aleatorios

Una de las bases esenciales para la aplicación de métodos de Monte Carlo radica en el empleo de números obtenidos mediante el sorteo de variables aleatorias independientes de distribución uniforme en $(0, 1)$. Una secuencia de estos números recibe el nombre de secuencia de números aleatorios. De manera alternativa, también recibe el nombre de secuencia de números aleatorios a una secuencia de valores obtenidos de sortear variables aleatorias independientes de distribución discreta entre los números naturales 0 y M.

Resulta entonces necesario estudiar cómo es posible disponer en una computadora de secuencias de esta naturaleza (o de comportamiento suficientemente cercano). Este problema surge en muchos otros contextos además de la aplicación de métodos Monte Carlo, quizás los de mayor impacto económico en la actualidad son las aplicaciones criptográficas, y los juegos (de azar o de otro tipo), pero también en general las simulaciones a eventos discretos, el muestreo de casos representativos, la generación de casos de prueba para testeo de software o para análisis de

tiempos de ejecución de algoritmos, la generación de escenas de realidad virtual en computación gráfica, etc. Dista mucho de ser un tema resuelto de manera totalmente satisfactoria; por el contrario, es objeto de investigación activa y en los últimos años han aparecido soluciones que han mejorado notablemente las posibilidades existentes.

Históricamente, tirar un dado, sortear bolas en una urna (por ejemplo para el juego de lotería), o hacer girar una ruleta han sido algunas de las primeras formas de obtener números aleatorios, suponiendo que estos dispositivos han sido construidos de manera tal de obtener una distribución presumiblemente uniforme.

Ya a comienzos del siglo XX, estos mecanismos eran insuficientes, y otros aparatos mecánicos o electromecánicos fueron diseñados para esta tarea; muy pronto luego de la construcción de las primeras computadoras programables, comenzaron los primeros intentos para emplearlas en la obtención de números aleatorios.

Fuentes de números aleatorios

En la actualidad, las principales formas para obtener secuencias de números aleatorios son:

1. A través de algoritmos determinísticos (software).

2. A través de dispositivos de hardware diseñados específicamente.

3. Mixto, software que emplea información proveniente del hardware estándar de una computadora.

Una cuarta forma es el empleo de números que fueron generados externamente (con alguna de las tres alternativas previas), y están disponibles a través de un dispositivo de almacenamiento o de comunicación. Históricamente, esta última alternativa estaba representada por tablas de números aleatorios publicadas en papel; posteriormente (década del 50) también existieron tablas disponibles no sólo como libros,

sino también en formato de tarjetas perforadas (ver http://en.wikipedia.org/wiki/A_Million_Random_Digits_with_100%2C000_Normal_Deviates - accedido 2019-04-22). Durante muchos años esta alternativa quedó descartada, pero en la década del 90 aparecieron tablas en CD, y en la actualidad existen además varios sitios Web que ofrecen en forma gratuita números aleatorios obtenidos a través de dispositivos de hardware específicos.

El método que resulta más rápido y satisfactorio para las aplicaciones de tipo Monte Carlo es en general el empleo de algoritmos determinísticos, que técnicamente son llamados generadores de números seudo-aleatorios (ya que conceptualmente sería una contradicción que un método determinístico generara números aleatorios). En este curso nos concentraremos en estos métodos.

Generadores de números seudo-aleatorios

Como mencionamos en la transparencia anterior, los generadores de números seudo-aleatorios son la forma más comúnmente usada para obtener secuencias de valores que puedan utilizarse en lugar de la secuencia de muestras de variables aleatorias uniformes e independientes, necesarias para todo método de Monte Carlo.

En general, todos los generadores de números seudo-aleatorios se implementan a través de una función $f : [0, Q-1]^t -> [0, Q-1]$, que recibe en entrada t valores entre 0 y $Q-1$, y genera uno adicional en este mismo rango. La función se aplica para generar una secuencia $\{Z_n, n \geq 0\}$ de números, donde el número $Z_n = f(Z_{n-t}, Z_{n-t+1}, \ldots, Z_{n-1})$ se calcula en función de los t anteriores en la secuencia, y donde $Z_0, Z_1, \ldots, Z_{t-1}$ son valores de inicialización llamados $semillas$, y que deben elegirse con tanto cuidado como la propia función f.

Como en general estamos interesados en valores $U(0, 1)$, se suele dividir los valores Z_n entre Q, para obtener números (racionales) entre 0 y 1.

Esto involucra un error de aproximación, que de todas formas existe ya que en los lenguajes de programación usuales empleamos siempre una representación finita de los números reales y por lo tanto existe una pérdida de precisión.

Dado que un generador de números seudo-aleatorio es una función determinística y sobre un espacio discreto de valores, es inevitable que exista un ciclo en la misma, es decir que exista un n_0 y un n_1 tal que $(Z_{n_0-t}, Z_{n_0-t+1}, \ldots, Z_{n_0-1}) = (Z_{n_1-t}, Z_{n_1-t+1}, \ldots, Z_{n_1-1})$; esto implica a su vez que $Z_{n_0} = Z_{n_1}$, y que de allí en más todos los valores se repitan. Se llama período al largo de la secuencia sin repeticiones, y en todos los casos el mismo será menor o igual a Q^t (que es la cantidad de secuencias de largo t distintas posibles).

Propiedades deseables

Supongamos que el objetivo del empleo de generadores de números seudo-aleatorios es el de simular computacionalmente una secuencia de variables aleatorias independientes con distribución uniforme $(0, 1)$ (ésta es la utilidad que le damos en este curso, otras aplicaciones como las criptográficas pueden tener requerimientos diferentes).

Las principales propiedades que nos interesan son las siguientes:

- Período largo: dado que como acabamos de ver todo generador de números seudo-aleatorio genera una secuencia cíclica, es importante asegurarse que el largo de dicha secuencia sea lo mayor posible, y en particular que exceda los requerimientos de la aplicación que los emplea (para ejemplificar: si vamos a emplear un método Monte Carlo con N replicaciones, y cada replicación emplea M números aleatorios distintos, es necesario que el período del generador exceda NM; de no ser así, los valores comenzarían a repetirse y se perdería la validez estadística del experimento).

- Eficiencia computacional: es conveniente que el generador sea rápido (requiera el menor número de instrucciones posible para generar un valor aleatorio), y emplee poca memoria.

- Repetibilidad: el usar un generador de números seudo-aleatorio permite repetir exactamente la misma secuencia de números, esto es muy importante tanto desde un punto de vista conceptual para permitir la duplicación de un experimento (que otra persona pueda obtener el mismo resultado que el reportado en un informe científico), pero también desde una perspectiva metodológica para emplear técnicas de reducción de varianza, y ya a nivel técnico (programación) para depurar programas que si no tendrían un comportamiento distinto en cada ejecución.

- Portabilidad: es deseable que el generador funcione de la misma manera en distintos sistemas operativos y lenguajes, sin depender tampoco del hardware empleado, ya que esto permite la repetibilidad y garantiza que las conclusiones sobre propiedades estadísticas obtenidas con pruebas en

una plataforma son aplicables a todas.

- En muchos casos es necesario contar con varias secuencias independientes distintas; en muchos generadores de números seudo-aleatorios, esto se puede implementar mediante la posibilidad de calcular la secuencia a partir de un valor n arbitrario, ya que si el período es muy largo, es posible calcular entonces las subsecuencias que comienzan en distintas ubicaciones del mismo y emplearlas como generadores virtuales independientes.

- Uniformidad e independencia: claramente el conjunto de propiedades anteriores, si bien importante, no alcanza para definir un buen generador de números seudo-aleatorios (por ejemplo, el generador $f(x) = (x + 1) \text{mod } Q$ podría cumplir con todos los requisitos, pero resultaría completamente inútil en la práctica). Aunque sabemos que los números generados por un proceso determinístico no pueden cumplir en sentido estricto con estas propiedades, deseamos un generador tal que las secuencias generadas ´´parezcan´´ independientes y uniformemente

distribuidas, es decir, que tengan un comportamiento frente a diversos tests estadísticos similar al que tendría una secuencia de variables aleatorias con estas características.

Familias de generadores

Los generadores de números aleatorios más utilizados y empleados actualmente entran en alguna de las siguientes categorías:

- Basados en recurrencias lineales
  - de un paso,
  - de múltiples pasos,
  - de múltiples pasos, y módulo 2,
- Basados en combinaciones de recurrencias lineales.
- Basados en recurrencias no lineales.

Generadores basados en recurrencias lineales

Las familias de generadores más estudiadas son probablemente las basadas en la recurrencia lineal

$$Z_i = \left(\sum_{s=1}^{p} A_s Z_{i-s} \right) \text{ mod } Q,$$

donde $A_1, \ldots, A_p$, son enteros no negativos tales que $A_p > 0$ y Q es un entero.

Cuando $p = 1$, tenemos generadores basados en recurrencias de un paso, que durante muchos años fueron el método de elección (y aún hoy predominan en las bibliotecas estandarizadas de lenguajes como C y Java).

Dentro de estos métodos, tenemos los métodos multiplicativos congruenciales,

$$Z_i = AZ_{i-1} \text{ mod } Q,$$

y los métodos mixtos congruenciales (o lineales congruenciales),

$$Z_i = AZ_{i-1} + C \text{ mod } Q,$$

introducidos por D.H. Lehmer en 1949.

Estas funciones son de las más simples imaginables, lo que ha permitido el estudio teórico y empírico de su comportamiento, con resultados que permiten conocer con mucha precisión en qué condiciones estos generadores tienen buenos y malos resultados, y los límites teóricos de su precisión. Quizá sorprendentemente, con una buena elección de los valores de A, C y Q, es posible tener generadores de ciclo Q o $Q-1$, y con un comportamiento muy bueno frente a tests de independencia y uniformidad.

En particular, un generador mixto congruencial tiene período Q sí y solo sí

1. C y Q son primos entre sí;

2. $A-1$ es múltiplo de p, para todo primo p que divide a Q;

3. $A - 1$ es múltiplo de 4, si Q es múltiplo de 4.

Y si Q es primo, un generador multiplicativo congruencial tiene período $Q - 1$ sí y solo sí A es una raíz primitiva de Q, es decir que

1. $A^{Q-1} - 1 \bmod Q = 0$;

2. para todo entero $I < Q - 1$, $(A^I - 1)/Q$ no es entero.

Estas condiciones, necesarias y suficientes para garantizar los máximos períodos alcanzables, no son suficientes para garantizar el buen comportamiento de los generadores del punto de vista de la distribución de los valores en las secuencias generadas; es más, existen numerosos ejemplos de implementaciones realizadas con valores inadecuados de estos parámetros, que han resultado en generadores con muy malas características que pueden invalidar los resultados de experimentos realizados en base a las secuencias por ellos generadas.

Existen también apreciaciones respecto a la eficiencia computacional y facilidad de implementación teniendo en cuenta la precisión (tamaño de palabra) de las computadoras existentes, que hace que estos métodos resulten especialmente rápidos para ciertas combinaciones de valores.

La sección "LCG: Linear congruential generators" del reporte "A collection of selected pseudorandom number generators with linear structures", por Karl Entacher (1997), referencia `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.53.3686&rep=rep1&type=pdf` (lectura opcional - último acceso 2020-04-15) contiene información sobre generadores de esta familia que han sido propuestos y utilizados en diversos contextos.

Empleando múltiples pasos en la recursión lineal (es decir, cuando $p > 1$) es posible lograr períodos mayores y también mejores comportamientos en cuanto a la distribución de la secuencia generada. Estos generadores reciben muchas veces el nombre de generadores recursivos múltiples (Multiple Recursive Generators, MRG). Si bien su teoría también ha sido desarrollada de manera detallada, los resultados obtenidos muestran que la

elección de buenos parámetros de un generador puede ser un problema intratable en tiempos razonables.

Sin embargo, cuando se considera el caso especial en que $Q = 2$ (es decir, cuando lo que generamos son bits aleatorios), es posible salvar estos problemas y desarrollar generadores de excelente calidad. Los generadores de la forma $B_i = (\sum_{s=1}^{p} A_s B_{i-s}) \mod 2$ donde A_i y B_i son números binarios se conocen por el nombre de Feedback Shift Register Generators (este nombre surge porque sería posible implementar un generador de este tipo directamente en hardware a través de un circuito de diseño estándar, conocido por feedback shift register). Con una correcta elección de los A_i, es posible obtener períodos de largo $2^p - 1$ (el máximo alcanzable con una recursión de esta forma).

Dado que estos generadores proveen secuencias de bits, si necesitamos obtener números en un rango $(0, 2^w)$ alcanza con tomar w bits consecutivos del generador. Formalmente, cada número aleatorio Z_i tendrá la representación binaria $B_{wi} B_{wi-1} \ldots B_{w(i-1)+1}$.

Una generalización del concepto resulta en los llamados Generalized

Feedback Shift Register Generators (GFSRG), que utilizan la misma formulación para generar los bits aleatorios, pero con reglas más generales para formar los números aleatorios como secuencias de los bits generados.

Lectura opcional: artículo "A New Class of Linear Feedback Shift Register Generators", Pierre L'Ecuyer and Francois Panneton, Proceedings of the 2000 Winter Simulation Conference, Dec. 2000, 690–696, accesible en `http://www.informs-sim.org/wsc00papers/091.PDF` (último acceso 2020-04-14).

Medidas de bondad y pruebas estadísticas

Para estudiar el comportamiento de los generadores de números seudo-aleatorios, hay dos enfoques que se complementan. Por un lado, el estudio teórico permite calcular ciertas medidas que reflejan que tan "bueno" es un generador (en el sentido de comportarse de manera similar a una secuencia aleatoria). Por otro, es posible realizar tests empíricos, bajo la forma de prueba de hipótesis, que permiten si el generador no es adecuado, refutar la hipótesis nula "el generador se comporta como una secuencia de valores uniformes e idénticamente distribuidos".

En relación a los llamados tests teóricos, los dos más importantes se basan en los conceptos de discrepancia (una medida de k-equidistribución, relacionada a la diferencia entre la distribución esperada en un hipercubo de dimensión k y y la distribución empírica), y de test espectral (relacionado con la inversa de la mínima distancia entre hiperplanos que contienen soluciones). También se estudian otra medidas tales como el mínimo número de hiperplanos necesarios para cubrir todos los puntos de la secuencia, y la mínima distancia entre puntos en el hiper-espacio de

dimensión k.

Respecto a los tests empíricos, es posible pensar y aplicar una cantidad muy grande de pruebas posibles. Las dos más clásicas son quizá el uso de tests de Kolomgorov Smirnov (junto con las estadísticas de χ^2) para verificar si un generador cumple con la uniformidad en su salida, así como la ausencia de correlación entre pares de valores sucesivos.

Una discusión detallada de los mismos escapa a los objetivos del curso.

Existen diversos paquetes y blbliotecas para realizar pruebas a generadores de números aleatorios, damos una lista a continuación (no es obligatorio entrar a cada uno):

- Set de tests Diehard, elaborado por George Marsaglia, `https://webhome.phy.duke.edu/~rgb/General/rand_rate/rand_rate.abs`. (último acceso:2020-04-15) `http://www.staff.science.uu.nl/~sleij101/Opgaven/LabClass/site/asm_diehard.php`. (último acceso:2020-04-15)

- TestU01 - Empirical Testing of Random Number Generators, elaborado por Pierre L'Ecuyer, `http://simul.iro.umontreal.ca/testu01/tu01.html`. (último acceso:2020-04-15)

- The ENT test program, `http://www.fourmilab.ch/random/`. (último acceso:2020-04-15)

- Random Number Generation and Testing - NIST (con énfasis en aplicaciones criptográficas): `http://csrc.nist.gov/groups/ST/toolkit/random_number.html`(último acceso:2020-04-15)

La biblioteca científica de GNU, GSL, provee implementaciones de varios buenos generadores (así como de otros clásicos, aunque con defectos conocidos): `http://www.gnu.org/software/gsl/manual/html_node/Random-Number-Generation.html`.(último acceso:2020-04-15)

En la Wikipedia hay descripciones de varios de los métodos actualmente en

uso: http://en.wikipedia.org/wiki/List_of_pseudorandom_number_generators.(último acceso:2020-04-15)

Preguntas para auto-estudio

- ¿Cuáles son las principales fuentes de números aleatorios?
- ¿Que es el período de un generador de números seudo-aleatorios? ¿Cuál es el máximo período posible para un generador que recibe t valores entre 0 y $Q-1$ en entrada?
- ¿Cuáles son las propiedades deseables en un generador de números seudo-aleatorios?
- ¿Qué medidas permiten evaluar la bondad de un generador de números seudo-aleatorios?

Ejercicio

- Buscar información sobre la generación de números seudoaleatorios en el lenguaje de programación que emplee para este curso (típicamente funciones rand, random, etc.). Esta función devuelve valores uniformes (0,1), o números discretos entre dos cotas? Verifique si es posible dar una semilla inicial que inicialice la secuencia de números (puede ser una función seed, o un argumento al llamar la función random la primera vez, etc.), o si la semilla es generada por el sistema cada vez que se ejecuta el programa para dar un punto de inicio distinto

- Hacer un programa que genere 10 números seudoaleatorios o variables uniformes (0,1) y las imprima. Correrlo dos veces distintas.

- Modificarlo para que el generador se inicialice con una semilla dada (por ejemplo 97531). Correrlo dos veces distintas.

- Comparar y comentar los resultados.

- Comentar las siguientes tiras:
  `https://dilbert.com/strip/2001-10-25`,
  `https://dilbert.com/strip/2016-04-01` (accedidas 2020-06-20).
  ¿Qué propiedades de los números aleatorios les parece ha utilizado el autor para generar un efecto cómico?