Unidad 3

Generación de números aleatorios.

Ejercicio 1. Generadores de números aleatorios. La implementación de un buen generador de números aleatorios uniformemente distribuidos sobre el intervalo $(0,1)$ no es una cuestión trivial. En la tabla 1 se presenta algunos de los generadores disponibles en diversos paquetes de software. Para cada uno de ellos investigue los fundamentos de su implementación, su período y la manera apropiada de utilizarlo.

Ejercicio 2. Simples test estadísticos para los generadores de números aleatorios. Este ejercicio propone simples test estadísticos para verificar la confiabilidad de los generadores de números aleatorios.

a) Un primer test puede ser realizado visualmente graficando en el plano los pares ordenados (x_1,x_2), (x_3,x_4),…,(x_{n-1},x_n) de una sucesión $\{x_i\}_{i=1}^n$ de n números aleatorios. Si los mismos están uniformemente distribuidos al azar en el intervalo $(0,1)$ los puntos deberían llenar el cuadrado $(0,1)\times(0,1)$ sin ningún patrón discernible.

b) Una variable aleatoria x distribuida uniformemente en el intervalo $(0,1)$ tiene una función de densidad de distribución $f(x)=1, 0<x<1$, con lo cual su media es

$$\mu = \int_0^1 x\, f(x)\,\mathrm{d}x = \frac{1}{2},$$

y su varianza es

$$\sigma^2 = \int_0^1 (x-\mu)^2\, f(x)\,\mathrm{d}x = \frac{1}{12}.$$

En consecuencia, una muestra aleatoria de x, $\{x_i\}_{i=1}^n$ de tamaño n, tendrá una media muestral

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i,$$

cuyo valor esperado es $\mu = \frac{1}{2}$ y una varianza muestral

$$s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i-\bar{x})^2,$$

de valor esperado $\sigma^2 = \frac{1}{12}$.

c) Si x e y son dos variables aleatorias independientes uniformemente distribuidas en $(0,1)$ entonces su producto tiene una media igual a $1/4$. En consecuencia, para una sucesión de números aleatorios el estadístico

$$c_k = \frac{1}{n}\sum_{i=1}^{n} x_i\, x_{i+k}, \quad k=1,2,\ldots$$

tiene un valor esperado igual a $1/4$. Esto permite, para k pequeño, testar la ausencia o presencia de correlaciones entre valores próximos.

Utilizando el *método de congruencia lineal*

$$x_i = (ax_{i-1}+c) \mod M$$

con $a=125$, $b=0$, $M=8192$, y semilla $x_0=1$, genere una sucesión de enteros aleatorios. Obtenga la sucesión de números reales del intervalo $(0,1)$ tomando $u_i = x_i/M$. Grafique 100 pares consecutivos de esta sucesión, esto es, $(u_1,u_2),(u_3,u_4),\ldots,(u_{199},u_{200})$. ¿Se forma algún patrón obvio sobre el cuadrado unitario? ¿Qué sucede si consideramos ahora 1000 pares de puntos? Repita el experimento para otros valores de los parámetros a, b y M. Repita nuevamente el experimento para alguna de las implementaciones presentadas en la tabla 1.

Verifique si los generadores de números aleatorios utilizados cumplen los relaciones estadísticas mencionadas considerando muestras de tamaño $n=100, 1\,000, 10\,000, 100\,000$. ¿Cómo se comporta, en función de n, la desviación de los resultados respecto a los valores esperados?

Fuente	Rutina
g77/gfortran	`rand`
Fortran 95	`random_number`
Numerical Recipes	`ran1/ran2/ran3`
SLATEC	`rand`
Octave	`rand`

Tabla 1. Generadores de números aleatorios.

Ejercicio 3. Test de χ^2 sobre la “bondad” de un generador de números aleatorios. Este test estadístico permite considerar cuan buena es la hipótesis de distribución uniforme sobre el intervalo $(0,1)$ para una sucesión de n números aleatorios. El procedimiento es como sigue. Divida el intervalo $(0,1)$ en r subintervalos de igual longitud y cuente el número n_k

de números aleatorios que caen dentro de cada subintervalo $k = 1, \ldots, r$. Calcule el estadístico

$$\chi^2 = \sum_{k=1}^{r} \frac{(n_k - n/r)^2}{n/r}.$$

Sea $\chi^2_{1-\alpha}$ el fractil de la distribución χ^2 con $r-1$ grados de libertad para un nivel de significación $\alpha = 0.1$. Entonces, si la desigualdad

$$\chi^2 > \chi^2_{1-\alpha},$$

es satisfecha, la hipótesis es rechazada, esto es, la calidad del generador es cuestionable.

Escoja algún generador de números aleatorios e implemente el test de χ^2 para $r = 10$ a 20 y $n = 100, 200, \ldots, 1000$.

Ejercicio 4. Generación de una muestra aleatoria distribuida exponencialmente. El *método de transformación integral* permite obtener una muestra aleatoria de una distribución exponencial en forma simple. Si x es una variable aleatoria cuya función de densidad de distribución f es una exponencial de media 1,

$$f(x) = e^{-x}, \quad 0 < x < \infty,$$

su función de distribución acumulada F es

$$F(x) = 1 - e^{-x}, \quad 0 < x < \infty.$$

Se sigue entonces que

$$x = F^{-1}(u) = -\log(1-u),$$

(¡verificarlo!). Así, obtenemos una muestra de x generando un número al azar u uniformemente distribuido en $(0, 1)$ y tomando $x = -\log(1-u)$. Sin embargo, como $1-u$ es también una variable aleatoria distribuida uniformemente en $(0, 1)$, $-\log(1-U)$ tiene la misma distribución de probabilidad que $-\log U$. En consecuencia, el opuesto del logaritmo de un número al azar distribuido uniformemente en $(0, 1)$ tiene una distribución exponencial de media 1.

Implementar el método anterior para generar una muestra aleatoria de tamaño $n = 1000$ distribuida exponencialmente con media 1, construya el histograma de frecuencias y compare con la distribución teórica. Generalizar la implementación para una distribución exponencial de parámetro α (media $1/\alpha$) (*Ayuda:* Considere la variable aleatoria $\hat{x} = \dfrac{1}{\alpha}\, x$).

Ejercicio 5. Generación de una muestra aleatoria con distribución de Poisson. En el caso de una variable aleatoria discreta x con función de probabilidad puntual $p_i = P(x_i)$ y función de distribución acumulada $F_i = F(x_i) = \sum_{k=0}^{i} p_k$, el proceso de inversión del *método de transformación integral* conduce a determinar en la tabla de valores de F_i el valor de $x_i = F^{-1}(u)$ tal que

$$F_{i-1} \leq u < F_i.$$

Consideremos, en particular, que x es una variable aleatoria con una distribución de Poisson de media λ:

$$p_i = \frac{\lambda^i}{i!}\, e^{-\lambda}, \quad i = 0, 1, \ldots$$

En este caso, los valores p_i satisfacen la siguiente relación de recursión (¡verificarlo!):

$$p_0 = e^{-\lambda}, \quad p_{i+1} = \frac{\lambda}{i+1}\, p_i \quad i = 1, 2, \ldots$$

lo cual permite simplificar el cómputo de los p_i durante la búsqueda secuencial de x, obteniendo así el siguiente algoritmo para obtener una muestra con distribución de Poisson de media λ:

Generar un número aleatorio u uniformemente distribuido en $(0, 1)$
Tomar $i = 0$, $p = e^{-\lambda}$, $F = p$
Mientras $u \geq F$ tomar
    $p = \frac{\lambda}{i+1}\, p$
    $F = F + p$
    $i = i + 1$
Tomar $x = i$

Implementar el algoritmo anterior. Generar una muestra de tamaño $n = 1000$ con distribución de Poisson de media $\lambda = 1$, construir el diagrama de frecuencias y comparar con la distribución teórica. Mejorar la eficiencia del algoritmo implementando una búsqueda dicotómica del valor muestral a partir de los valores más próximos al valor medio λ.

Ejercicio 6. Generación de una muestra aleatoria distribuida normalmente. En este ejercicio consideraremos tres diferentes maneras de generar números al azar distribuidos normalmente.

a) De acuerdo al *teorema del límite central*, si $\{x_k\}_{k=1}^{n}$ es una sucesión de variables aleatorias independientes de media μ y varianza σ^2, entonces, conforme n se incrementa, la variable aleatoria

$$z = \frac{\sum_{k=1}^{n} x_k - n\mu}{\sqrt{n\sigma^2}},$$

se aproxima a una distribución normal con media 0 y varianza 1. Si los x_k están distribuidos uniformemente en $(0, 1)$, entonces $\mu = \frac{1}{2}$ y $\sigma^2 = \frac{1}{12}$, lo cual implica que

$$z = \frac{\sum_{k=1}^{n} x_k - n/2}{\sqrt{n/12}},$$

se aproxima a una distribución normal con media 0 y varianza 1. Con la elección $n = 12$, la fórmula anterior toma su forma más simple:

$$z = \sum_{k=1}^{12} x_k - 6.$$

De este modo, sumando una docena de números aleatorios uniformemente distribuidos y sustrayendo 6, obtenemos una muestra aleatoria cuya distribución se aproxima a una distribución normal con media 0 y varianza 1.

b) Si z es una variable aleatoria distribuida normalmente con media 0 y varianza 1, entonces la función de densidad de probabilidad del valor absoluto de z, $x = |z|$, es

$$f(x) = \frac{2}{\sqrt{2\pi}}\, e^{-x^2/2}, \quad 0 < x < \infty.$$

Una muestra aleatoria que sigue esta distribución puede ser obtenida a través del *método de rechazo* ó *método de Von Newmann*. Consideremos la variable aleatoria y cuya función de densidad de probabilidad $g(y)$ es una distribución exponencial de media 1:

$$g(y) = e^{-y}, \quad 0 < y < \infty.$$

Ahora,

$$\frac{f(y)}{g(y)} = \sqrt{2/\pi}\, e^{(y - y^2/2)},$$

y su máximo valor se alcanza en $y = 1$ (¡demostrarlo!). Por lo tanto tomamos

$$c = \max \frac{f(y)}{g(y)} = \frac{f(1)}{g(1)} = \sqrt{2\, e/\pi},$$

con lo cual,

$$\frac{f(y)}{c\, g(y)} = \exp\left\{-\frac{(y-1)^2}{2}\right\}.$$

Así pues, el método de rechazo procede, en este caso, como sigue: Se genera un valor de y cuya distribución de probabilidad es una exponencial de media 1 y un número aleatorio u uniformemente distribuido en $(0, 1)$. Si $u \le \exp\{-(y-1)^2/2\}$, entonces se toma $x = y$; en caso contrario, se genera otro valor de y. De este modo obtenemos una muestra con una distribución igual a la del valor absoluto de una variable distribuida normalmente. Para obtener una muestra de z, alcanza con tomar dicho valor como x ó $-x$ en igual proporción, esto es, generamos un número al azar u y tomamos:

$$z = \begin{cases} x & \text{si } u \le 1/2, \\ -x & \text{si } u > 1/2. \end{cases}$$

Nótese que la condición de aceptación para y es equivalente a $-\log u \ge (y-1)^2/2$. Como $-\log u$ tiene una distribución exponencial de media 1, para generar x podemos proceder, entonces, obteniendo dos valores y_1 e y_2 de muestras independientes de una distribución exponencial de media 1. Entonces, si $y_2 \ge (y_1 - 1)^2/2$, tomamos $x = y_1$. De lo contrario volvemos a generar otro par de valores (y_1, y_2).

c) Sean x e y dos variables aleatorias independientes con distribución normal de media 0 y varianza 1. Sean r y θ las coordenadas polares del vector (x, y) en el plano, esto es,

$$r = \sqrt{x^2 + y^2}, \quad \tan\theta = \frac{y}{x}.$$

Entonces, r^2 y θ son variables aleatorias independientes, siendo la distribución de densidad de probabilidad de r^2 una exponencial de media 2, mientras que θ está uniformemente distribuida en el intervalo $(0, 2\pi)$ (¡demostrarlo!). De este modo una muestra aleatoria distribuida normalmente para cada variable del par (x, y) es obtenida como sigue: (a) generar dos números aleatorios u_1 y u_2 uniformemente distribuidos en $(0, 1)$, (b) tomar $r^2 = -2\log u_1$ y $\theta = 2\pi u_2$, (c) asignar

$$x = r\cos\theta = \sqrt{-2\log u_1}\, \cos(2\pi u_2),$$
$$y = r\sin\theta = \sqrt{-2\log u_1}\, \operatorname{sen}(2\pi u_2).$$

Este procedimiento es conocido como *método de Box–Muller*. Por eficiencia computacional sería deseable evitar el cómputo de funciones trigonométricas, lo cual puede efectivamente lograrse generando pares de números aleatorios (v_1, v_2)

uniformemente distribuidos en el interior del círculo de radio unidad centrado en el origen. En tal caso, las coordenadas polares r y θ del par son variables aleatorias independientes, estando r^2 distribuido uniformemente en $(0,1)$ y θ distribuido uniformemente sobre $(0,2\pi)$ (¡demostrarlo!). De este modo, podemos obtener el seno y el coseno del ángulo aleatorio θ a partir de los valores del par (v_1, v_2) tomando:

$$\cos\theta = \frac{v_1}{r} = \frac{v_1}{(v_1^2+v_2^2)^{1/2}},$$
$$\operatorname{sen}\theta = \frac{v_2}{r} = \frac{v_2}{(v_1^2+v_2^2)^{1/2}}.$$

Entonces, generando un número aleatorio u distribuido uniformemente en $(0,1)$, obtenemos una muestra para el par de variables (x,y) distribuidas normalmente tomando:

$$x = \left(-2\log u\right)^{1/2} \frac{v_1}{\left(v_1^2+v_2^2\right)^{1/2}},$$
$$y = \left(-2\log u\right)^{1/2} \frac{v_2}{\left(v_1^2+v_2^2\right)^{1/2}}.$$

Ahora bien, puesto que $r^2 = v_1^2 + v_2^2$ está distribuida uniformemente en $(0,1)$ y es independiente del ángulo aleatorio θ, podemos utilizarlo como el número aleatorio u de las ecuaciones anteriores. De este modo, definiendo $s = r^2 = v_1^2 + v_2^2$, resulta que

$$x = v_1\left(\frac{-2\log s}{s}\right)^{1/2}$$
$$y = v_2\left(\frac{-2\log s}{s}\right)^{1/2}.$$

constituyen una muestra aleatoria para dos variables distribuidas normalmente con media 0 y varianza 1, cuando (v_1, v_2) es un punto al azar del interior del círculo de radio 1 centrado en el origen, y $s = v_1^2 + v_2^2$. Este procedimiento es conocido como *método polar*.

Implementar cada uno de los tres métodos anteriores para generar una muestra aleatoria de distribución normal de media 0 y varianza 1. Testear cada implementación generando una muestra de $n = 1000$ valores y construyendo el histograma de frecuencias con una longitud de los intervalos de clase igual a $\sigma = 1$. Compare sus resultados con la curva de la función de densidad de probabilidad normal. Experimente con otros valores de n para comparar la precisión y eficiencia de los métodos.

Generalice las implementaciones para obtener una muestra aleatoria de una distribución normal de media μ y varianza σ^2 dadas (*Ayuda:* Considere la variable aleatoria $\hat{z} = \mu + \sigma\, z$).

Ejercicio 7. Modelo de un proceso estocástico. Un *camino aleatorio* es una sucesión de pasos de longitud fija con una dirección escogida en forma independiente y al azar del paso anterior. Procesos de este tipo pueden ser utilizados para modelar varios fenómenos físicos, tales como el movimiento browniano de pequeñas partículas suspendidas en un fluido o el movimiento de los electrones en metales. En este ejercicio consideraremos un camino aleatorio en el plano. Comenzando desde el origen, consideramos una serie de n pasos $(\Delta x_i, \Delta y_i)$ de longitud unidad $(\Delta x_i^2 + \Delta y_i^2 = 1)$, $i = 1, 2, \ldots, n$. Si las direcciones en cada paso son aleatorias, entonces, para un gran número de pasos, aún cuando la distancia total caminada es n, la distancia radial al origen es, en promedio, sólo $\sqrt{n}$.

Implemente un camino aleatorio en dos dimensiones para un número de pasos n de longitud unidad. Para escoger cada dirección arbitraria genere un ángulo aleatorio θ en el rango $(0, 2\pi]$ y tome $\Delta x = \cos\theta$, $\Delta y = \operatorname{sen}\theta$. Alternativamente, con el fin de evitar la evaluación de funciones trigonométricas, considere dos números aleatorios $\Delta\hat{x}$ y $\Delta\hat{y}$ en el rango $(-1,1)$ y tome $\Delta x = \Delta\hat{x}/L$, $\Delta x = \Delta\hat{y}/L$, siendo $L^2 = \Delta\hat{x}^2 + \Delta\hat{y}^2$. Efectúe simulaciones para $n = 10$, 100, $1\,000$ y $10\,000$, promediando la distancia radial al origen al final de cada camino sobre varias pruebas para cada valor de n. Compare sus resultados, en función de n, con la predicción teórica.

Ejercicio 8. Estimación de una cantidad determinista por un modelo estocástico. Considere un círculo de radio unidad centrado en el origen de coordenadas e inscrito en el cuadrado de área 4 centrado en el origen. Genere una sucesión de n pares (x_i, y_i) de números uniformemente distribuidos sobre el intervalo $(-1,1)$ y cuente el número m de pares que caen dentro del círculo. La probabilidad de que un par dado caiga dentro del círculo es

$$\frac{\text{Area del círculo}}{\text{Area del cuadrado}} = \frac{\pi}{4}$$

la cual, para n grande, puede aproximarse por la frecuencia relativa m/n.

Implemente el proceso anterior para aproximar π. Experimente con diversos valores de n.

Ejercicio 9. Generación de condiciones iniciales. Utilice el siguiente programa para la construcción con $N = 250\,000$ cuerpos de una esfera de Plummer de masa $M = 1.0$ y radio crítico $R = 1.0$ (en el sistema de unidades donde $G = 1$). Debido a que la configuración se extiende a infinito, considere un *radio de corte* $= 20$. Compare el histograma de la distribución de densidad de masa de la configuración generada con la distribución conocida.

Programa para generar una esfera de Plummer

```
      program esfera_de_plummer
*     ----------------------------------------------------------------
*     Descripción: genera una distribución de N partículas en equilibrio
*     de acuerdo a la distribución de masa de la llamada ESFERA DE
*     PLUMMER.
*
*     Formato del archivo de entrada:
*     ---------------
*     Archivo salida
*     plummer.sal
*     Constate de gravitacion universal
*     1.0
*     Numero de cuerpos
*     250000
*     Radio de corte
*     20
*     Masa, radio critico
*     1.0 1.0
*
*     La salida consiste de la masa, posición (x,y,z) y velocidad
*     (vx,vy,vz) de cada una de las N partículas del modelo.
*     ----------------------------------------------------------------
*     Bloque de declaración de variables
*     ----------------------------------------------------------------
      implicit none
      character(80)  infodato,     ! Nombre descriptivo
     &               nombreentrada,! Archivo de datos de entrada
     &               nombresalida  ! Archivo de datos de salida

      integer i,j,ok
      real PI,K                 ! Constantes
      parameter (PI=4.0*atan(1.0),K=3.0*PI/64.0)
      real GRAV,                ! Constante de gravitacion universal
     &     NCUERPOS,            ! Número de cuerpos del sistema
     &     RADIODECORTE,        ! Radio de corte de la configuración
     &     MTOT,                ! Masa total del sistema
     &     ERRE,                ! Radio critico
     &     ENERGIA,             ! Energía del sistema
     &     r,                   ! Módulo del vector posición
     &     a0,a1,a2,            ! Números al azar en [0,1]
     &     f,                   ! Función del test de Von Newmann
     &     ve,                  ! Velocidad de escape
     &     v,                   ! Módulo de la velocidad
     &     C1,                  ! Constante
     &     masa,                ! Masa de las partículas
     &     vector_r(3),         ! Vector posición
     &     vector_v(3)          ! Vector de velocidad
*     ---------------------------------------------------------------
*     Bloque de lectura de datos
*     ---------------------------------------------------------------
      if (command_argument_count().eq.0) then
           write(0,*) "Error: Indicar el archivo de datos"
           stop
      endif
      call get_command_argument(1,nombreentrada)
*     ---------------------------------------------------------------
      open(8,file=nombreentrada,status='old',iostat=ok)
```

```
      if (ok.ne.0) then
           write(0,*) 'El archivo de datos no puede ser leído'
           stop
      endif
*     ---------------------------------------------------------------
      read(8,'(A)')  infodato
      write(*,'(A)') infodato
      read(8,'(A)')  nombresalida
      write(*,'(A)') nombresalida
*     ----------
      read(8,'(A)')  infodato
      write(*,'(A)') infodato
      read(8,*)  GRAV
      write(*,*) GRAV
*     ----------
      read(8,'(A)')  infodato
      write(*,'(A)') infodato
      read(8,*)  NCUERPOS
      write(*,*) NCUERPOS
*     ----------
      read(8,'(A)')  infodato
      write(*,'(A)') infodato
      read(8,*)  RADIODECORTE
      write(*,*) RADIODECORTE
*     ----------
      read(8,'(A)')  infodato
      write(*,'(A)') infodato
      read(8,*)  MTOT,ERRE
      write(*,*) MTOT,ERRE
*     ----------
      close(8)
      ! Chequear mínima consistencia de los datos
      if (MTOT.lt.0.0.or.ERRE.lt.0.0) then
         write(0,*) 'Valores fuera de rango!'
         stop
      endif
*     ---------------------------------------------------------------
*     Bloque de procesamiento
*     ---------------------------------------------------------------
      ! Calcular energía total del sistema
      ENERGIA=-K*GRAV*MTOT**2/ERRE
      write(*,*) 'Energia = ',ENERGIA
*     ----------
      ! Generar cuerpos
      masa = MTOT/real(NCUERPOS)
      RADIODECORTE = RADIODECORTE/ERRE

      open(8,file=nombresalida,status='unknown',iostat=ok)
      if (ok.ne.0) then
           write(*,*) 'El archivo de salida no puede ser escrito'
           stop
      endif

      i=1
      C1= sqrt(2.0*(GRAV*MTOT/ERRE))
      do while(i.le.NCUERPOS)
         ! Generación del modulo del radio vector
         call random_number(a0)
```

```
            r=1.0/sqrt(a0**(-2.0/3.0)-1.0) ! r/R
            if(r.le.RADIODECORTE) then
               ! Computar velocidad de escape
               ve=C1*((1.0+r**2)**(-0.25))
               ! Generar modulo de la velocidad
               ok=1
               do while (ok.ne.0)
                  call random_number(a1)
                  call random_number(a2)
                  f=(1.0-(a1**2))**(3.5)
                  if(a2.le.f) then
                     v=a1*ve
                     ok=0
                  endif
               enddo
               r=r*ERRE  ! Dimensionar correctamente
               ! Generar vectores posicion y de velocidad
               call generavector(r,vector_r)
               call generavector(v,vector_v)
               ! Imprimir datos del cuerpo generado
               write(8,'(7(1pe13.6,1x))') masa,
     &                    (vector_r(j),j=1,3),
     &                    (vector_v(j),j=1,3)
               ! Seguir con el siguiente cuerpo
               i=i+1
            endif
         enddo
         close(8)
         stop
         end
*     ===============================================================
      subroutine generavector(modulovector,vector)
*     -----------------------------------------------------------------
*     Subrutina para generar al azar las componentes x,y,z de un vector
*     de módulo dado
*     -----------------------------------------------------------------
*     Argumentos de la subrutina
*     -----------------------------------------------------------------
      implicit none
      real modulovector,      ! Módulo del vector
     &     vector(3)          ! Vector
*     -----------------------------------------------------------------
*     Variables locales
*     -----------------------------------------------------------------
      real rn,r
      real DBLE_PI
      parameter (DBLE_PI=8.0*atan(1.0))
*     -----------------------------------------------------------------
*     Bloque de procesamiento
*     -----------------------------------------------------------------
      call random_number(rn) ! Obtener número al azar en [0,1]
      vector(3)=(1.0-2.0*rn)*modulovector ! Componente z
      call random_number(rn) ! Otener número al azar en [0,1]
      r=sqrt((modulovector**2-vector(3)**2))  ! Proyección en el plano
      vector(1)=r*cos(DBLE_PI*rn) ! Componente x
      vector(2)=r*sin(DBLE_PI*rn) ! Componente y
      return
      end
```