UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE CIENCIAS

MÉTODOS MONTE CARLO, CASOS DE ESTUDIO

T E S I S

QUE PARA OBTENER EL TÍTULO DE:

MATEMÁTICO

P R E S E N T A :

ARMANDO ESTEBAN VILLALOBOS SÁNCHEZ

FACULTAD DE CIENCIAS
UNAM

Tutor: Dr. Pedro Eduardo Miramontes Vidal

2007

Hoja de Datos de Jurado

1. Datos de alumno
   Villalobos
   Sánchez
   Armando Esteban
   55498018
   Universidad Nacional Autónoma de México
   Facultad de Ciencias
   Matemáticas
   095278638

2. Datos del tutor
   Dr.
   Pedro Eduardo
   Miramontes
   Vidal

3. Datos del sinodal 1
   Dra.
   Ana Margarita
   Guzmán
   Gómez

4. Datos del sinodal 2
   M. en A.P.
   María del Pilar
   Alonso
   Reyes

5. Datos del sinodal 3
   Dr
   María de Lourdes
   Esteva
   Peralta

6. Datos del sinodal 4
   Mat
   Luis Manuel
   Hernández
   Gallarda

7. Datos del trabajo escrito
   Métodos Monte Carlo, Casos de estudio.
   108 p
   2007

Contenido

Introducción

Los métodos Monte Carlo, son métodos númericos, útiles para resolver problemas por medio de simulación de variables aleatorias, deben su nombre a la ruleta Monte Carlo, que es una excelente generadora de números aleatorios. En situaciones donde los métodos analíticos son poco eficientes o involucran cálculos complicados, los métodos Monte Carlo pueden ser de gran ayuda, por ejemplo el espectro de velocidad de un neutrón que poseé muchos valles y crestas, es difícil de manejar de manera analítica, pero con el método Monte Carlo solo es necesario reflejar el espectro de velocidad en la distribución de probabilidad. Pueden también usarse para aproximar complicadas integrales, asi como los valores máximos o mínimos de funciones multidimensionales no necesariamente diferenciables, es útil también como método de prueba de hipótesis estadística. Algo que resulta notable en este último caso, es que la función de distribución de probabilidad no está necesariamente especificada lo cual representa una ventaja en tanto que se reducen supuestos para el análisis de datos.

En general, estos métodos, a través de múltiples ensayos, nos permiten aproximar un resultado gracias a la teoría de la probabilidad, en particular a la ley de los grandes números y al teorema del límite central, pero es muy importante tener en cuenta que se trata de una aproximación y por ende, suele usarse en problemas donde no se exige mucha precisión.

La característica fundamental de estos métodos, es que se realizan a través de la generación de números pseudo-aleatorios; actualmente, casi todas las computadoras poseen un generador de números pseudo-aleatorios bastante bueno, por lo que no es necesario desarrollarlos, pero su comprensión puede ser útil para determinadas necesidades específicas.

Creado por Jhon Von Neumann , Stanislaw Ulam y Nicholas Metropolis como un posible acercamiento estadístico para resolver el problema de la difusión de neutrones en un material fisionable,en el año de 1947 publicaron el artículo Statistical methods in neutron diffusion[13], en este trabajo simiente, se emula un núcleo esférico de material fisionable cubierto por un material protector, para ello, se asume una distribución inicial de neutrones en el espacio y sus respectivas velocidades, pero se ignoran los efectos radiactivos e hidrodinámicos.

La idea es seguir el desarrollo de grandes cadenas de neutrones individualmente como consecuencia de la absorción, fisión, escape etc. de neutrones y con ello simular una reacción nuclear para resolver diversas incógnitas referentes a los efectos radiactivos e hidrodinámicos.

Pocos años después salió a la luz el artículo "The Monte Carlo Method"[14] de los mismos autores. Durante el período posterior a la segunda guerra mundial este método además de ingenioso, resultó factible gracias al desarrollo paralelo de las computadoras electrónicas, puesto que antes de la investigación en el campo de los circuitos electrónicos, hubiera resultado casi

imposible llevar a cabo este tipo de trabajo, debido a la enorme cantidad de operaciones aritméticas y horas hombre que esto conlleva.

Posteriormente una gran cantidad de investigaciones se han desarrollado con estos métodos y han sido enriquecidos por una gran cantidad de científicos, entre otros el mismo Stanislaw Ulam, Enrico Fermi, Reuven Rubinstein, etcétera.

El método Monte Carlo ha encontrado un nicho en diversas áreas del conocimiento (física estadística, biología, química, etc) debido en parte a que muchos fenómenos naturales presentan cierto comportamiento aleatorio y a que su naturaleza sugiere una posible aproximación estadística. Por otro lado, también se ha abierto paso en el ramo industrial y financiero en el ámbito de la simulación, pues con la generación de variables aleatorias, podemos obtener diversos escenarios para la toma de decisiones.

Respecto a este último punto, la simulación de sistemas abre un campo de estudio autónomo, pues tanto el diseño de simulaciones computacionales como su implementación y comparación con respecto a un sistema real, implica también análisis y solución de problemas y situaciones previamente estudiadas, siempre con fines prospectivos.

Con el objetivo de ilustrar la utilidad y el desarrollo del método Monte Carlo, he distribuido esta tesis de la siguiente manera:

- En el primer capítulo, veremos los trabajos elementales en los que tienen aplicación los métodos Monte Carlo, entre ellos el cálculo de áreas y el método de la aguja de Buffon.

- Como parte fundamental de estos procesos, en el segundo capítulo se hace un estudio acerca de la generación de números pseudo-aleatorios por medio de computadoras electrónicas, y de algunas pruebas empleadas para evaluar la calidad de diferentes generadores, así como la manera en la que podemos imitar el coportamiento de algunas de las distribuciones de probabilidad mas empleadas con fines de análisis estadístico.

- El capítulo tres es un compendio de las aplicaciones más populares, como la integración en varias variables, la optimización de una función con el método clásico de Monte Carlo, con el algoritmo de Metropolis, comunmente empleado en aplicaciones de física estadística, prueba de hipótesis estadística, modelación de sistemas, etc., su descripción y varios ejemplos.

- Por último un comentario y el apéndice A que contiene algunos resultados y el código fuente de los programas desarrollados a lo largo de este trabajo respectivamente.

Capítulo 1

Métodos básicos

El fundamento de los métodos Monte Carlo es el cálculo de la probabilidad de un evento con variables aleatorias, aunque existen algunos que no dependen de los conceptos derivados de la probabilidad, es sumamente importante tener conocimientos elementales acerca de esto último. El apéndice A es un compendio de algunos conceptos y resultados útiles que facilitan su remembranza.

Los primeros ejemplos que se exhiben en este capítulo, pueden ser mostrados a través del cálculo de probabilidades, o como un medio para probar su efectividad y sugerir modos de empleo para la comprensión a nivel didáctico de la probabilidad.

Algunos de ellos, son previos a la era del cómputo electrónico, pero esencialmente todos emplean variables aleatorias para aproximar un resultado.

1.1 Métodos geométricos

1.1.1 Cálculo de áreas

Como se verá en el capítulo siguiente, es posible generar una sucesión de números pseudo aleatorios que imite el comportamiento de una variable aleatoria. En este capítulo, se abordan los ejemplos básicos asumiendo que es posible obtener números aleatorios por medios no necesariamente electrónicos, un ejemplo de esto último es sortear objetos muy semejantes con números asignados.

Una de las diversas aplicaciones que tiene el método Monte Carlo que en lo sucesivo designaré por las siglas MMC, es un método probabilístico, casi empírico, para calcular áreas.

Supongamos que necesitamos calcular el área contenida por una curva cerrada simple Γ, sobre la cual no se poseé información adicional además de su forma en su sentido mas elemental que es la figura que observamos.

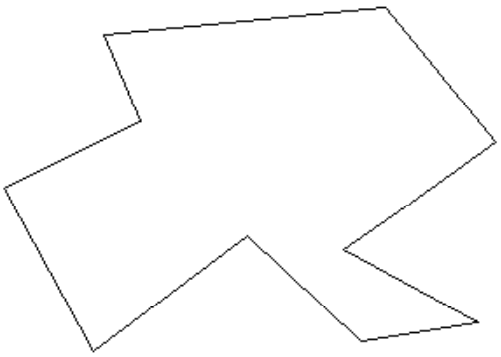


Figura 1.1: Área encerrada por una curva dada.

Si encerramos la curva Γ por otra curva de área conocida, por ejemplo un cuadrado de lado L, como en la figura 1.2, es evidente que el área encerrada por Γ, la cual se denota por $A(\Gamma)$ es igual a la razón entre ella y el área del cuadrado multiplicada por el área del cuadrado.

$$A(\Gamma) = \frac{A(\Gamma)}{L^2} L^2$$

Consideremos el evento tomar un punto "al azar" dentro de un cuadrado de longitud L como un experimento Bernoulli, el hecho de que un punto B esté contenido en Γ es tomado por éxito, por lo que en el interior del cuadrado, la probabilidad p de que el punto B esté dentro de la superficie contenida por Γ (denotada por $int(\Gamma)$) está dada por la razón:

$$p = P(B \in int(\Gamma)) = \frac{A(\Gamma)}{L^2}.$$

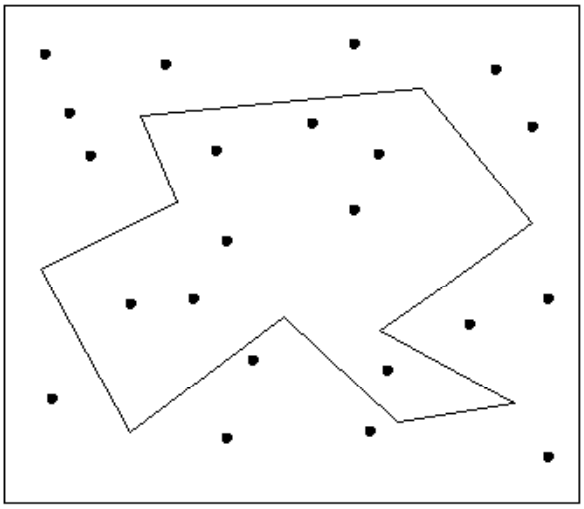


Figura 1.2 Cuadrado que encierra a Γ

de donde:

$$A(\Gamma) = pL^2$$

Por lo que al repetir el evento tomar un punto al azar en el cuadrado, se puede aproximar la razón entre el área contenida por Γ y el cuadrado, pues por la ley de los grandes números tenemos que:

$$\lim_{n\to\infty} \frac{n_e}{n} = p$$

Por tratarse de un modelo estadístico para aproximar un valor, el valor de la probabilidad p se estima empleando la razón $\frac{n_e}{n}$, para aproximar el área contenida por la curva, donde n_e es el número de éxitos obtenidos despues de realizar n experimentos Bernoulli.

Ejemplo 1.1.1 *Sea Γ un círculo de radio $r = 1.5$ y el cuadrado en el cual se encierra a Γ con lados de longitud $L = 6$, al correr el programa B.1.1., (ver apéndice B) despues de 1000 observaciones se obtiene la gráfica siguiente (figura 1.3), que representa el comportamiento de la variable área, en función del número de experimentos.*

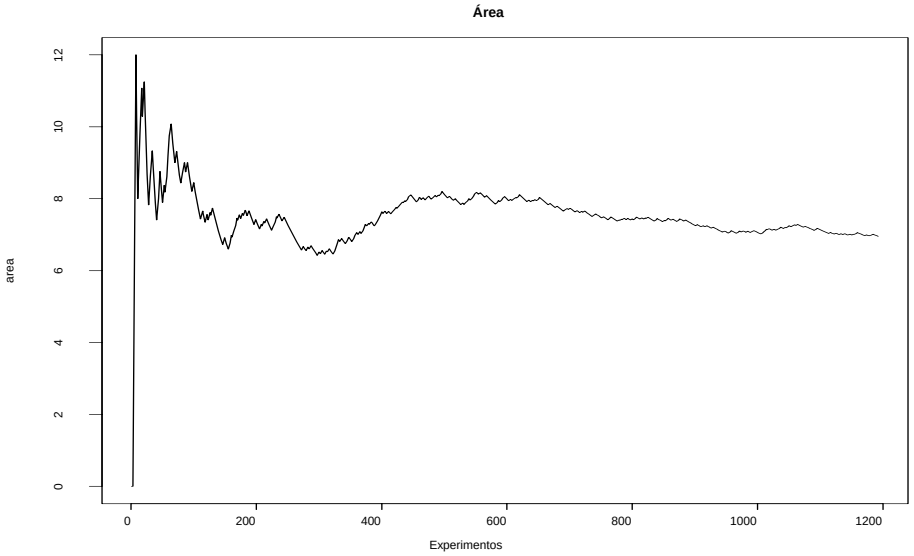


Figura 3: Área de Γ en función de n

Y como aproximación final del área se tiene que $p = \frac{n_e}{n} = \frac{193}{1000}$ por lo que el valor probabilístico del área del círculo es:

$A(\Gamma) \approx \frac{193}{1000} * 36 = 6.948$

Mientras que una aproximación analítica del área del círculo es

$A(\Gamma) = \pi * 1.5^2 \approx 7.0686.$

El error relativo es por lo tanto

$$\left|\frac{7.0686-6.948}{7.0686}\right| \approx 1.7061 \times 10^{-2}$$

del orden de 10^{-2} o bien en términos porcentuales es el 1.70%.

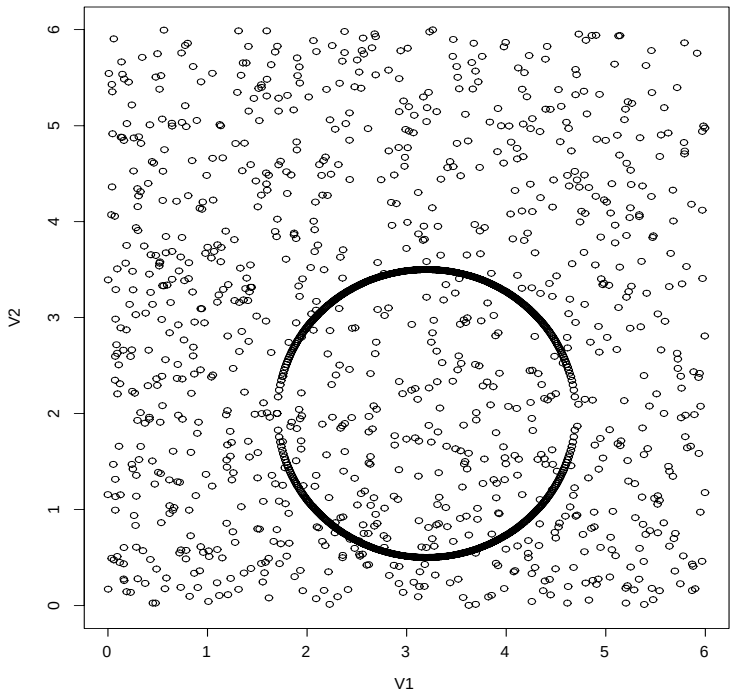


Figura 1.4: Puntos con coordenadas aleatorias.

Si incrementamos el número de experimentos a 10000 obtenemos:

$A(\Gamma) = \frac{1963}{10000} * 36 = 7.0668$ con lo que el error porcentual se reduce a $0.02\,5\%$.

El hecho de obtener mayor precisión en el resultado se traduce en incrementar el tiempo de procesamiento de datos, en este caso se emplearon dos variables y un flujo de datos muy simple, básicamente, el algoritmo analiza si una pareja ordenada poseé o no la característica de encontrarse dentro de el círculo, y la diferencia de tiempo de procesamiento puede no ser representativa, pero cuando se manejan múltiples variables y un flujo más complejo el tiempo de procesamiento y de programación computacional suele elevarse considerablemente.

Con fines didácticos, para ilustrar la utilidad de la probabilidad, el cuadrado puede ser una caja, los puntos aleatorios pueden ser representados por objetos como monedas o semillas, y con ello mostrar empíricamente conceptos elementales como el de variable aleatoria, probabilidad, o incluso mas complejos como el teorema del límite central o la ley de los grandes números.

El método empleado en el ejemplo anterior es conocido como Hit or Miss debido a su naturaleza de acierto y error en cada experimento. Por otro lado tenemos otra famosa aplicación del método Hit or Miss creada por el conde Auguste de Buffon en el Siglo XVIII publicada en el año de 1777.

1.1.2 La aguja de Buffon

El método de Buffon, es un método probabilístico para aproximar el valor de π. Supongamos la siguiente situación: en una superficie trazamos líneas paralelas separadas una distancia L entre ellas, si dejamos caer libremente una aguja de longitud L sobre la superficie ¿Cuál es la probabilidad de que la aguja intersecte a alguna de las líneas paralelas?.

Supongamos que la aguja intersecta a alguna de las líneas rectas (figura 1.5), denotemos por α al ángulo formado por la aguja y la recta intersectada. Sea $x = L\text{sen}\alpha$.

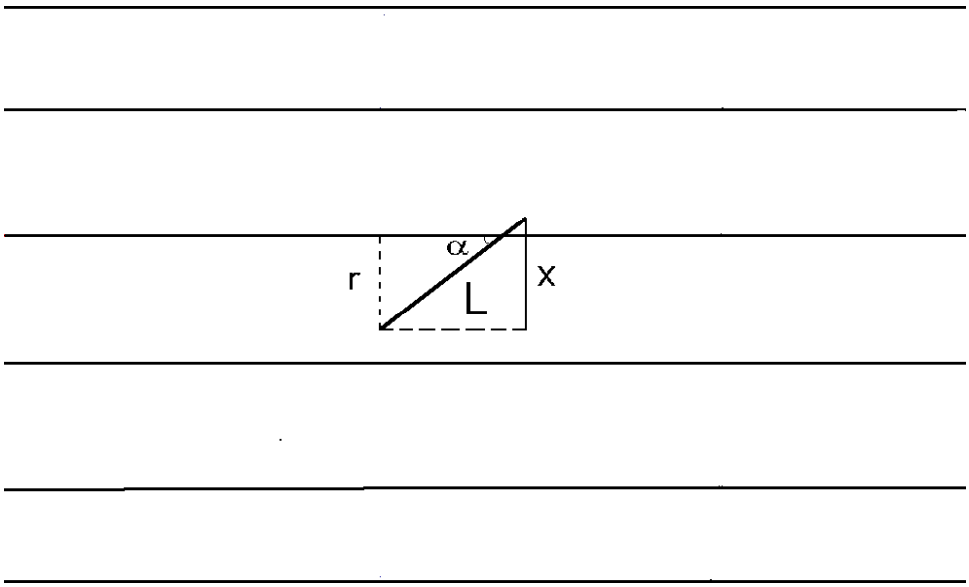


Figura 1.5 Método de Buffon

Una condición necesaria y suficiente para que la aguja intersecte a alguna de las rectas es que la distancia de ambos extremos de la aguja a una misma recta, sea menor que el seno del ángulo α.Tenemos que:

$$\text{sen}\alpha = \frac{x}{L}$$

como se observa en la figura 1.5.

La probabilidad de que la aguja intersecte a alguna de las líneas paralelas, es igual a la probabilidad de que la longitud r sea menor que x.

$$P(r < x) =$$

$$P(r < L\text{sen}\alpha) =$$

$$\frac{\int_0^{\pi} L\ \text{sen}\alpha d\alpha}{L\pi}$$

De donde :

$$\frac{\int_0^{\pi} \text{sen}\alpha d\alpha}{\pi} =$$

$$\frac{-\cos\alpha\,|_0^{\pi}}{\pi} =$$

$$\frac{2}{\pi} = P(r < x)$$

Por lo tanto:

$$\pi = \frac{2}{P(r < x)}$$

o bien, la probabilidad de que la aguja intersecte a alguna de las rectas es doble del inverso multiplicativo de π. El ejercicio de Buffon, consiste, al igual que el ejemplo anterior, en la repetición exhaustiva de un experimento Bernoulli.

Para obtener una aproximación de $P(r < x)$ con una simulación de agujas, se emplea una vez más el estimador $p = \frac{n_e}{n}$ donde n_e es el número de éxitos (intersecciones con alguna de las rectas) en n experimentos.

Ejemplo 1.1.2 *A continuación una simulación de números pseudo-aleatorios* $(r, \text{sen}\alpha)$ *bajo la condición* $r \leq L\text{sen}\alpha, L = 1$ *(figura 1.6).*

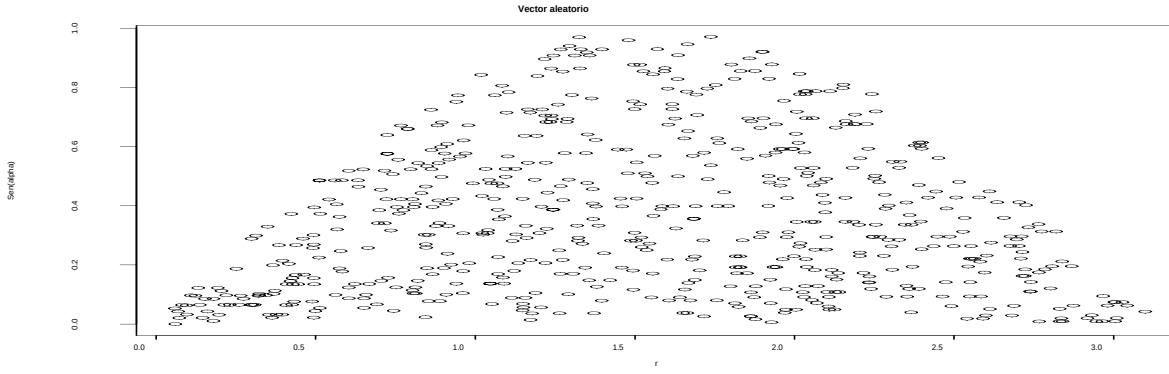


Figura 1.6: Vectores (r, α) con coordenadas aleatorias $r < \text{sen}\alpha$.

Por el teorema del límite central, para asegurarnos una buena aproximación al valor de π,es necesario repetir el experimento que consiste en arrojar la aguja a la superficie con los trazos para incrementar el número de observaciones y tomar el cociente entre el total de agujas arrojadas y las agujas que intersectan a las rectas, donde, otra vez, el ejercicio arrojar agujas es un experimento Bernoulli y se considera un éxito intersectar alguna de las rectas dibujadas.

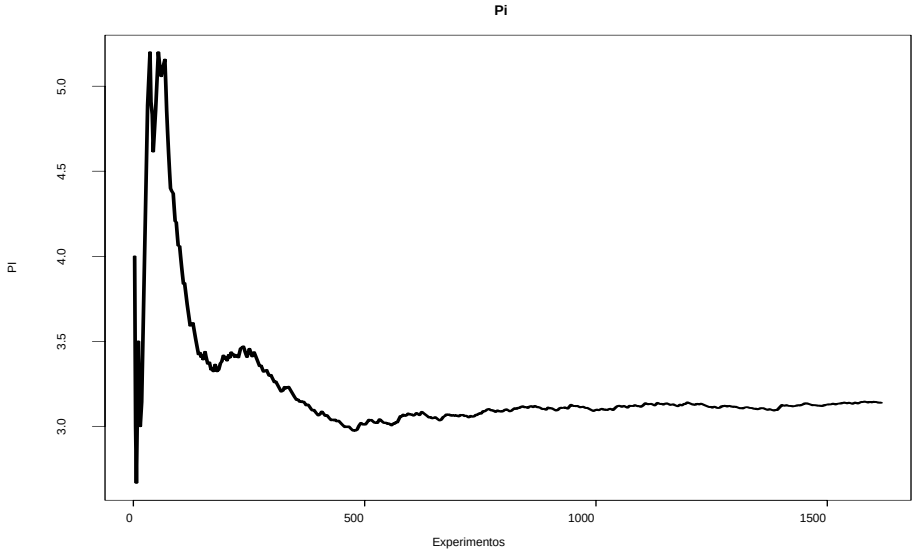


Figura 1.7: Valor de π en función de n

En la simulación anterior (Programa No B.1.2), después de 1500 experimentos (agujas) obtenemos que $Pi = 3.14013$ y tal como se observa en el histograma (figura 9), despues de correr varias simulaciones de la variable PI, su valor oscila alrededor de 3.1474 (valor medio), con una varianza igual a 0.006 y estimamos el valor del error relativo entre este cálculo y otras aproximaciones del valor de π con el cociente:
$\left|\frac{3.14159-3.14013}{3.14159}\right| = 4.6473 \times 10^{-4} = 0.04\,\%$.

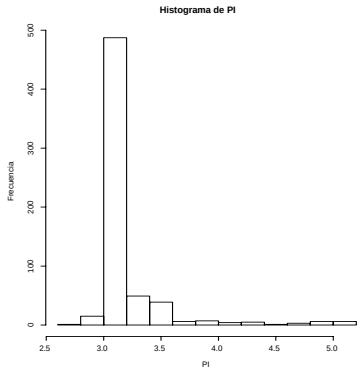


Figura 1.8
Histograma de π

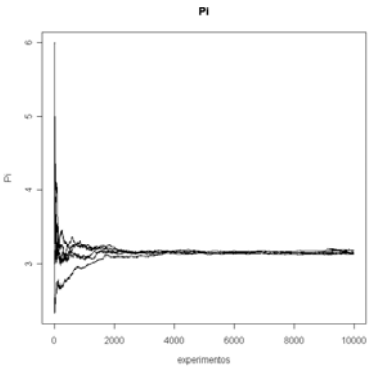


Figura 1.9
Valor probabilístico de π

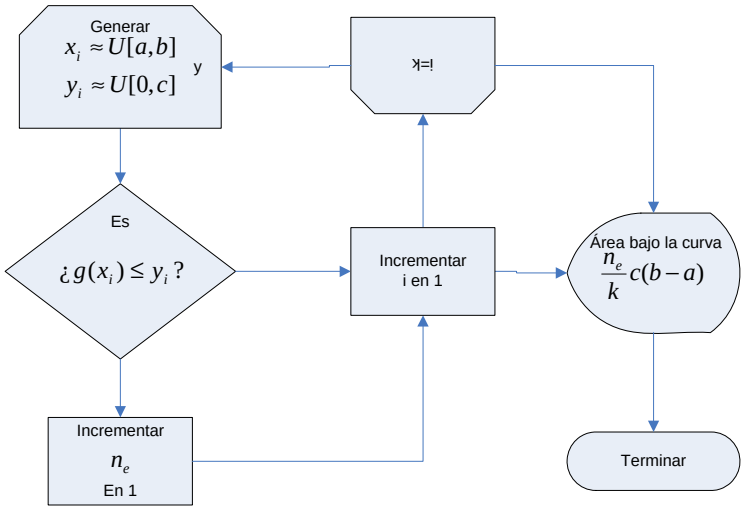
El diseño de este experimento probabilístico resulta estimulante debido a su ingeniosa presentación, y es equivalente al cálculo del área bajo la curva de la función $f(x) = \text{sen}x$ en el intervalo $[0, \pi]$, la integración con variables aleatorias, de funciones de una sola variable se puede llevar a cabo con un método muy semejante al del cáculo de áreas como veremos en la siguiente sección.

1.1.3 Integral definida

Como una generalización del ejercicio anterior, donde lo que se llevó a cabo fue una simulación de vectores aleatorios en R^2, podemos calcular el valor de la integral definida de cualquier función continua por pedazos y acotada $g : R \to R$, este método es un caso particular del primer ejemplo, pues al cálculo se realiza con experimentos Bernoulli calculando áreas con acierto y error. En el caso anterior, obtuvimos una aproximación para el valor de π por medio de la integral definida:

$$\frac{\int_0^{\pi} L\text{sen}\alpha d\alpha}{L\pi}$$

con una simulación de variables aleatorias, donde $L\pi$ es la longitud del intervalo de integración,en general para obtener una proximación de $\int_a^b g(x)dx$ todo lo que se pide para la función $g(x)$ es que sea acotada en $[a, b]$. Sea $g(x) \leq c \ \forall x \in [a, b]$. El algoritmo para aproximar el área bajo la curva definida por la función $g(x) \geq 0$ es como sigue:



Donde n_e es el número de éxitos obtenidos y k el total de experimentos a realizar, x es el valor dentro del intervalo de integración y y una variable aleatoria con la que se compara el valor de $g(x)$.

Por lo que de nuevo podemos aproximar el valor de la integral por:

$$\int_a^b g(x)dx = P(y \leq g(x))c(b-a) \approx$$

$$\frac{n_e c(b-a)}{n}$$

.

Ejemplo 1.1.3 *Sea* $g(x) = \sqrt{x}senx$ *la cual es una función que no es integrable por medio de funciones elementales.Para calcular una aproximación de la integral definida en* $[0, \pi]$ *sabemos que* $g(x) \leq \sqrt{\frac{\pi}{2}}$ *sea:*

$$c \approx \frac{\pi}{2}$$

. Con el programa No B.1.3 obtenemos:

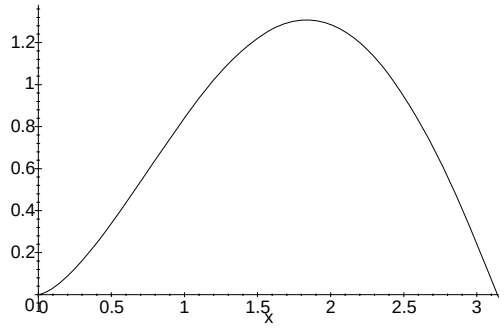


Figura 1.10
Gráfica de $g(x) = \sqrt{x}\text{sen}xdx$

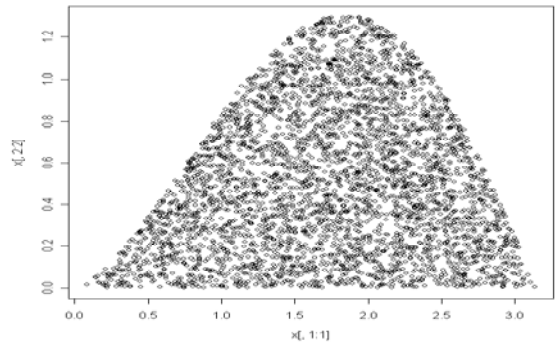


Figura 1.11
Vectores con coordenadas aleatorias

Despues de 8000 ensayos tenemos que $n_e = 3926$ por lo que

$$\int_0^{\pi} \sqrt{x}sen(x)dx \approx \frac{3926 * \frac{\pi}{2} * \pi}{8000} \approx 2.4218$$

Evaluando con otro método numérico (Simpson [9]) con 1000 elementos de la partición tenemos que

$\int_0^{\pi} \sqrt{x}senxdx = 2.4353$. Por lo que el error relativo con respecto al método en cuestión es $\left|\frac{2.4353-2.4218}{2.4353}\right| = 5.5435 \times 10^{-3} = 0.5\%$

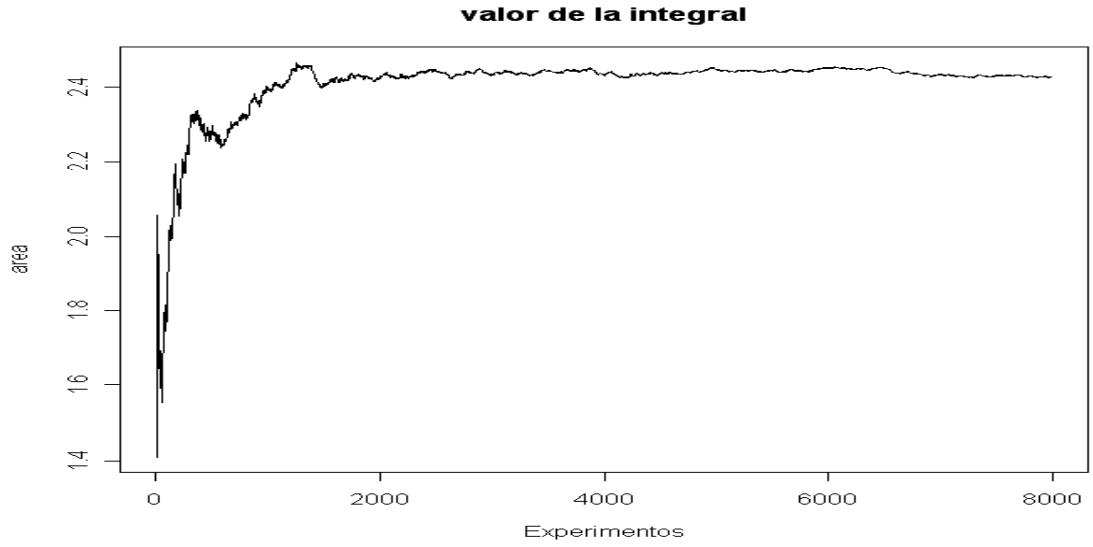


Figura 1.12

En este ejemplo, la función es continua en todo el intervalo, en el capítulo siguiente veremos que, si no tenemos esta hipótesis, se pueden hacer cambios en el algoritmo para definir los valores que puede tomar una variable aleatoria generada por medios electrónicos.

Nota 1.1.4 *En tanto que la variable p, es aleatoria en función del número de experimentos, se pueden aplicar pruebas paramétricas para construir un intervalo de confianza para p lo cual también permite construir un intervalo de confianza para el resultado.*

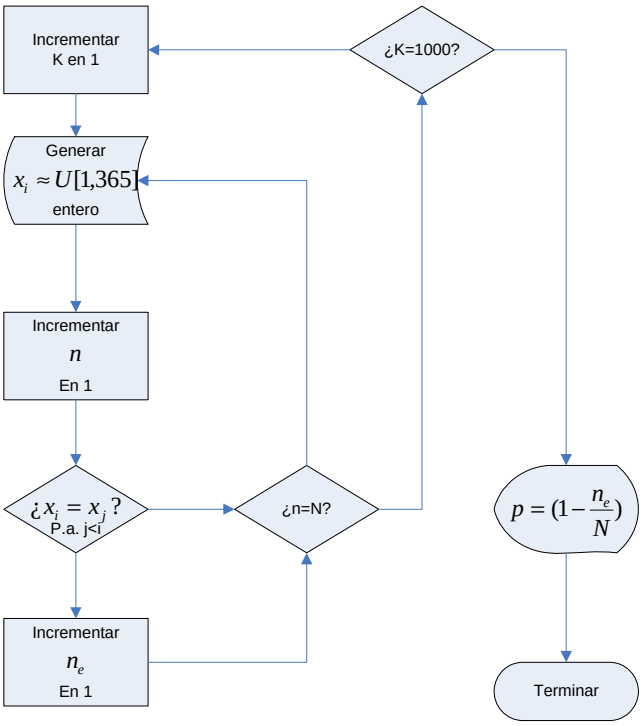
1.2 Otros modelos

Como veremos en el capítulo 2, los resultados obtenidos en los ejercicios anteriores dependerán, entre otras cosas, de la sucesión de variables aleatorias escogidas para llevar a cabo los ejercicios en forma sucesiva. Para poder aproximar con mayor seguridad un resultado es conveniente evaluar con diferentes sucesiones de números pseudo-aleatorios.

En la mayoría de los casos, no es posible evaluar todas las posibles combinaciones bajo las que un evento resulta favorable, ya que esto implica mucho tiempo máquina, por lo que es conveniente hacer una selección de manera aleatoria (muestreo) de los elementos de un espacio muestral.

1.2.1 Modelos combinatorios

Ejemplo 1.2.1 *Deseamos conocer la probabilidad de que en un grupo de N personas, al menos dos hayan nacido la misma fecha. En un modelo donde se emplea MMC para responder esta pregunta, el algoritmo podría ser como el que sigue:*



Donde x_i es la fecha de cumpleaños de la persona $i-ésima$, N es el número de personas que integran el grupo k y n_e el número de éxitos obtenidos despues de evaluar a 1000 grupos diferentes.

Empleando el programa No B.1.4 del apéndice B obtenemos $n_e = 763$, por lo que decimos que la probabilidad de que en un grupo de 30 personas al menos dos tengan la misma fecha de cumpleaños es aproximadamente 0.763. Para calcular la probablidad de que en un grupo de N personas no haya repetición de las fechas de nacimiento, podemos calcularlo de este modo, para la primera persona del grupo tenemos 365 diferentes maneras para esoger un día, para la segunda 364 en 365 y así sucesivamente para la N-ésima tenemos 365-(N-1) diferentes maneras de escoger el día de nacimiento, de modo que la probablidad p de que las fechas de nacimiento de dos personas coincidan está dada por:

$$P(x_i = x_j) = p = \frac{365!}{365^N(365-(N-1))!} \; i \neq j.$$

El valor de la probabilidad para un grupo de 30 personas tal y como fué calculada mas arriba es de:

$$p = \frac{365!}{365^{30}(365-(29))!} = 0.71$$

Si incrementamos el número de grupos a 10000 obtenemos con el mismo programa:

$$p = \frac{7161}{10000}$$

y el comportamiento de la variable probabilidad de acuerdo al número de evaluaciones:

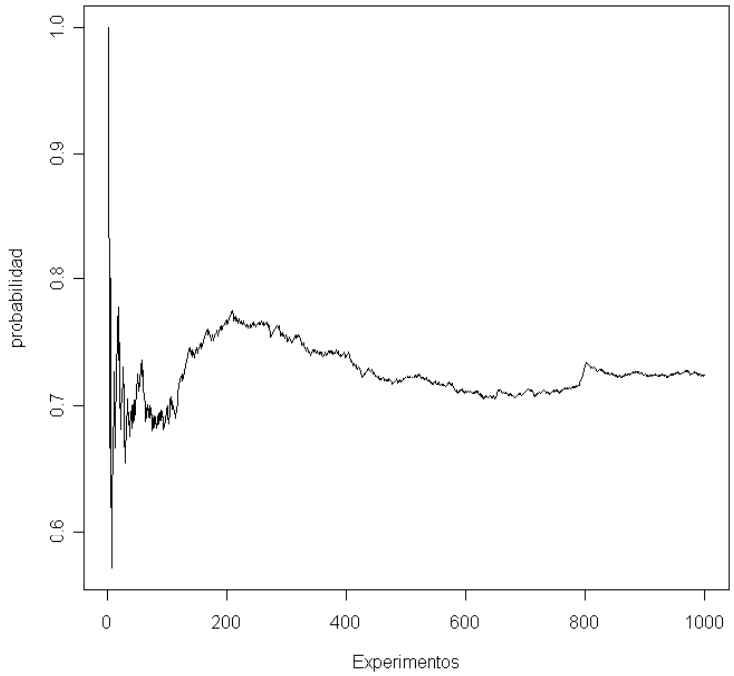


Figura 1.13

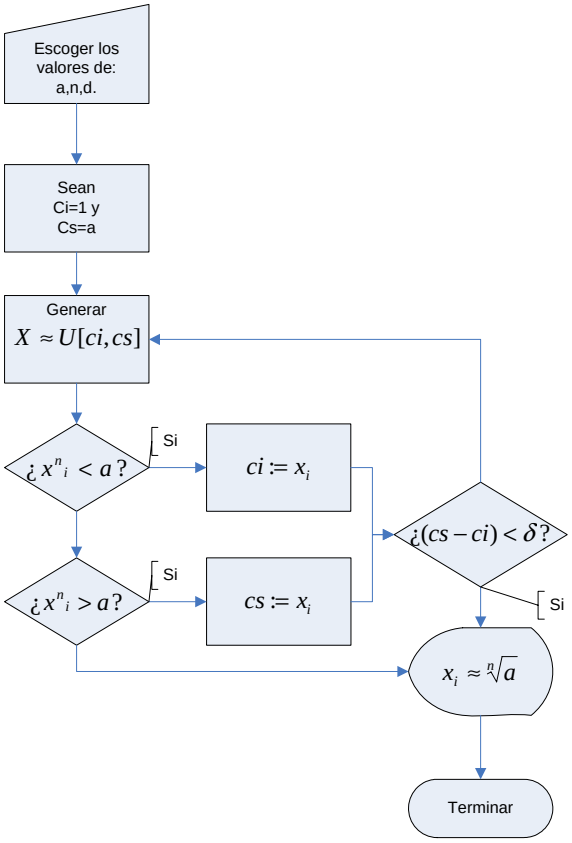
Éste es otro ejemplo de las consecuencias de evaluar la precisión, pues mientras que el primer ejercicio con 1000 iteraciones se requirieron pocos minutos para realizarse, el último con 10000 iteraciones tomó mas de 1 hora de procesamiento para poder obtener los resultados y aunado a esto hay que agregar para un problema particular el tiempo de diseño del programa o del algoritmo.

1.2.2 Aproximación de raíces

Un uso práctico para el uso de variables aleatorias o MMC, es el diseño de un algoritmo para aproximar raíces de cualquier orden, y es un ejemplo del tipo de modelos donde no se requieren conceptos o fines probabilísticos, como veremos más adelante, también es el caso de algunos algoritmos de optimización de funciones a través de MMC.

Ejemplo 1.2.2 *Deseamos obtener una aproximación de la raiz de orden n-ésimo de un número, podemos construir un método basado en la simulación de variables aleatorias de modo que se aproxime tanto como nos lo permita el orden de decimales aceptados por la computadora.*

Sea $a^{\frac{1}{n}}$ el valor del que deseamos obtener una aproximación, supongamos $n \in N$ y $a > 1$ para simplificar la construcción de nuestro modelo, con los supuestos anteriores, podemos simular X una variable aleatoria tal que, bajo la condiciones planteadas, puede tomar valores en $[1, a]$ y verificar si $x_i^n < a$, en cuyo caso decimos que x_i es la nueva cota inferior del intervalo, ahora tomamos $x_{i+1} \in [x_i, a]$ de forma análoga si $x_i^n > a$ este valor será la nueva cota inferior del intervalo en el que nuestra variable aleatoria puede tomar valores. El ciclo se termina si $|x_i - x_{i-1}| \leq \delta$. Con lo que hemos construido una sucesión que converge al valor de $a^{\frac{1}{n}}$. Veamos el algoritmo empleado para la descripción anterior.



Con el programa No. B.1.5, para $a := 1879,\ n = 10\ y\ \delta = 10^{-3}$ tenemos que $a^{\frac{1}{n}} \approx 2.12517$

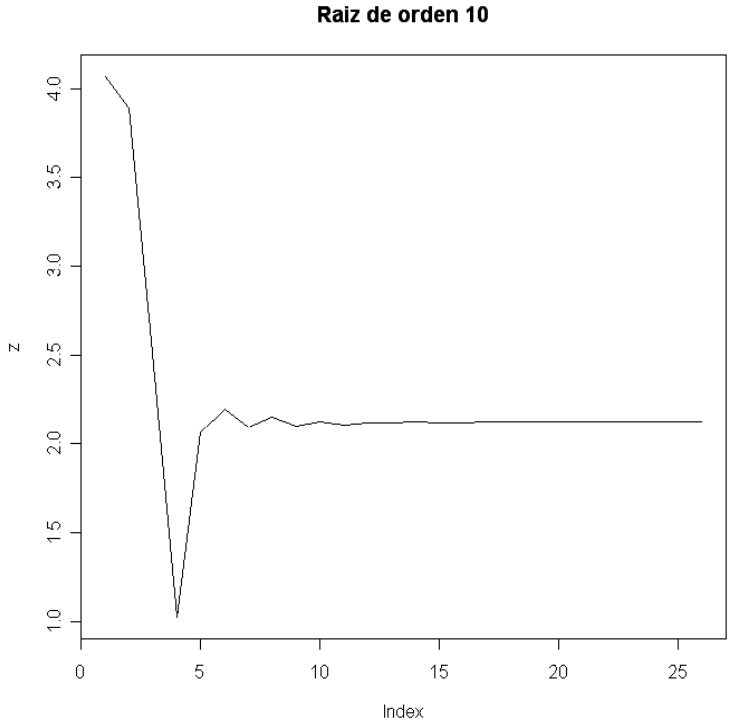


Figura 1.14

La velocidad de convergencia de estas sucesiones, aunque es relativamente alta, podemos incrementarla, ajustando mejor nuestro algoritmo, por ejemplo acotando de una manera mejor nuestros intervalos conforme se anidan.

1.3 Condiciones de Métodos Monte Carlo

Una condición necesaria tanto en los ejemplos comentados anteriormente y en general para los métodos Monte Carlo, es que los elementos aleatorios tengan una distribución conveniente.

En el caso del cálculo de áreas, podría ocurrir lo siguiente, que una cantidad representativa de los puntos escogidos estén aglomerados en una zona del cuadrado (figura 1.15), con lo cual la aproximación sufriría un sesgo.

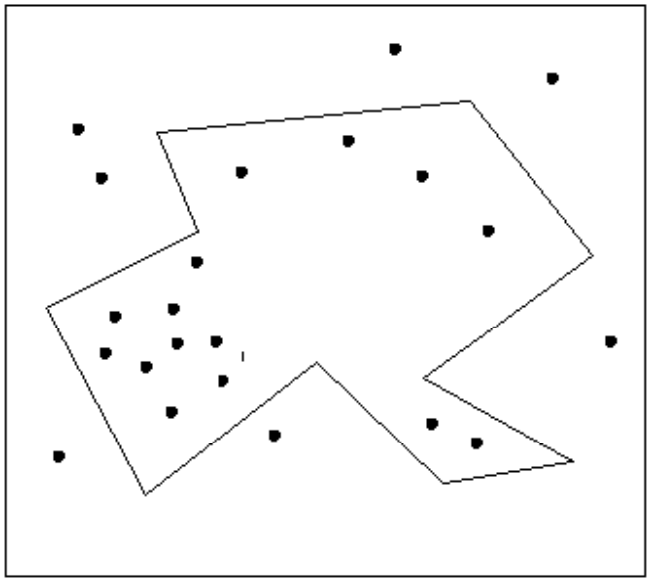


Figura 1.15

Dicho de otro modo, la distribución de los puntos deberá ser uniforme en el cuadrado $L \times L$, lo que significa que todo punto $B(x, y)$ en $L \times L$ tiene la misma probabilidad de ser escogido para obtener la aproximación del área, igualmente en el caso de Buffon, debemos esperar que las variables aleatorias de posición se distribuyan uniformemente en la superficie donde estamos simulando las líneas paralelas ya que si, por ejemplo, tomara valores más cercanos a las líneas el valor probabilístico de π se incrementaría, debido a que el área bajo la curva $senx$ es menor en esas vecindades.

Para el MMC de aproximación de raices, posiblemente sea mas conveniente una distribución con tendencia centralizada o inclusive mejor, una distribución con mayor peso en el primer medio de cada intervalo ya que si $a > 2 \Rightarrow a^{\frac{1}{n}} < \frac{a}{2} \ \forall \ n \in N$ y para cada caso ($a < 1, 1 < a \leq 2$ *etc.*) considerar una distribución de la variable aleatoria simulada que optimice el modelo.

Debemos de asegurarnos, que la distribución de la variable aleatoria, corresponda a las características del sistema a modelar o bien reduzca las operaciones necesarias en nuestro algoritmo.

Respecto a este último punto, consideramos un modelo mas eficiente cuando el error relativo se puede estimar por debajo de un parámetro de satisfacción dado, en general si el error está por debajo del 1% se considera una buena aproximación.

Debemos de tener en cuenta que el error relativo, depende de un valor obtenido a partir de otros métodos (analíticos en su mayoría).

También se pueden considerar otros factores para considerar un algoritmo mejor que otros por ejemplo el empleo de pocas operaciones para llevar a cabo cada paso del algoritmo y en general si se reduce el tiempo de

procesamiento para llevar a cabo la aproximación.

Dado que en su mayoría son modelos estadísticos, podemos realizar planteamientos con objetivos específicos, como la optimización del tiempo de procesamiento, acotar la varianza, etc.

1.3.1 Varianza y error en MMC

Como es de esperar, al incrementar el número de experimentos lo que conseguimos es disminuir la varianza de la variable aleatoria y de la variable dependiente, vemos el comportamiento de la varianza de esta última para los ejemplos anteriores (área del círculo, Las agujas de Buffon e integral definida).

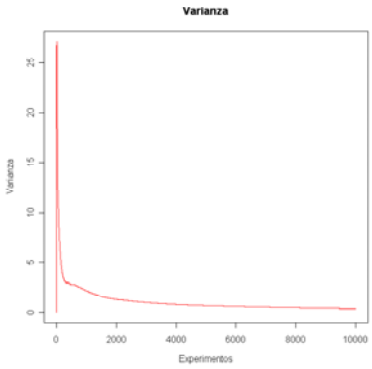


Figura 1.16
Varianza:área del círculo

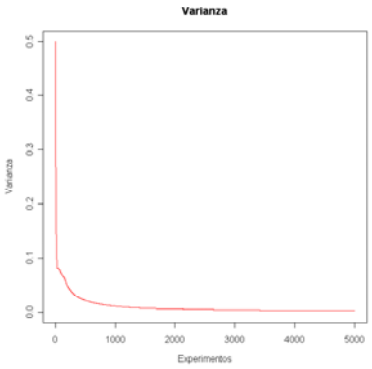


Figura 1.17
Varianza: π

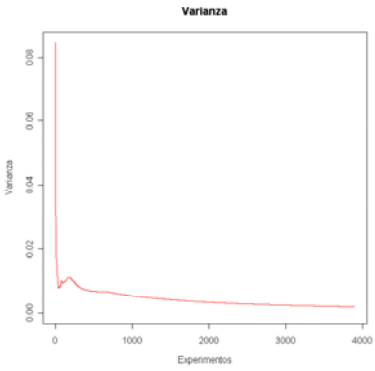


Figura 1.18
Varianza: $\int_a^b g(x)dx$

Aunque el error relativo es una medida de dispersión con respecto a un valor esperado como resultado de una aproximación, depende de la obtención del mismo valor por otros métodos, para evitar esta dependencia podemos, como en cualquier otro modelo estadístico aproximar el error sin necesidad de obtener un valor esperado de manera previa.

Si tenemos un muestreo de N variables, y una función $f = f(x_1, x_2, ...x_N)$ que depende de las observaciones de la variable aleatoria, como por ejemplo,

el valor medio de un conjunto de N observaciones, la desviación estándar, etc., a estas funciones se les conoce también como estadísticos, podemos aproximar el error de f con base en la desviación estándar.

Definimos la varianza S_N^2 de f como:

$$S_N^2 = \frac{1}{N}\sum_{i=1}^{N}\left[f(x_i) - \left(\frac{1}{N}\sum_{i=1}^{N} f(x_i)\right)\right]^2$$

La suma de las desviaciones también llamado error cuadrático medio:

$$\varepsilon^2 = NS_N^2$$

$$\varepsilon = \sqrt{NS_N^2}$$

y el promedio de estás desviaciones:

$$\varepsilon = \frac{S_N}{\sqrt{N}}$$

Lo que significa que el error esperado de $f(x)$ depende de N y de la varianza S_N^2, pero mientras que para reducir esta útima en una cifra decimal debemos incrementar N en un factor de 10, para reducir el error esperado de $f(x)$ en la misma magnitud debemos de incrementar los experimentos en un factor de 100.

Esto implica que, para asegurar mayor precisión estadística, debemos incrementar N, lo que se traduce en mayor tiempo de cómputo o bien reducción de la varianza a través de mejorías en el diseño o en el planteamiento del módelo de simulación de variables aleatorias.

Un ejemplo de reducción de la varianza se obtiene al escoger adecuadamente ciertos parámetros del modelo, en el caso del cálculo de áreas, la reducción de las dimensiones del cuadrado en el que simulamos encerrar la figura, o para la integral definida, el valor escogido para $c = sup\{g(x)/x \in [a,b]\}$.

En términos mas generales, se suelen usar varias técnicas como la modificación de la función de distribución de probablidad asumida para una variable aleatoria, con lo que se modifica el muestreo, otra forma es la sustitución de algunos elementos por otros obtenidos o esperados de un modelo determinístico, o bien realizar simulaciones previas y comparar los resultados que arroje el modelo con el sistema simulado para hacer las correcciones pertinentes, entre otros.

En general, los métodos de reducción de la varianza, dependen completamente del conocimiento o información que tenemos respecto del sistema a modelar, pues solo a través de esto último, podemos ajustar mejor tanto el algoritmo como las variables en términos de su función de densidad, rango, etc. y como se aprecia en la definición el error teórico será de un orden menor.

Capítulo 2

Generación de números aleatorios

2.1 Generadores de variables aleatorias

Como veremos más adelante, la generación de variables aleatorias uniformes, es determinante para construir otras funciones de distribución de probabilidad (que a partir de aquí serán designadas por las siglas FDP), y en general para construir cualquier modelo de simulación de un proceso aleatorio, ya sea independiente o no del tiempo.

Definición 2.1.1 *Un generador uniforme de números pseudo-aleatorios es un algoritmo que, dado un valor inicial u_0 y una transformación $D : E \subseteq R \to [0,1]$, donde se obtiene la sucesión:*

$$\{u_n\} \text{ tal que } u_i = D(u_{i-1}) \text{ y } 0 \leq u_i \leq 1 \ \forall i \in N.$$

Para todo n, los valores $(u_1, u_2, ..., u_n)$ reproducen el comportamiento de una muestra de elementos independientes e identicamente distribuidos (iid sample) $(V_1, V_2, ..., V_n)$ de variables aleatorias uniformes, cuando se comparan con un conjunto usual de pruebas.

Esta definición es útil debido su funcionalidad, pero se debe hacer incapie en dos cosas principalmente, lo primero es que una sucesión de números, obtenida como resultado de una transformación con las características descritas, está sujeta a un conjunto de pruebas, esto significa que la validez de la sucesión $\{u_i\}$ como una sucesión de números aleatorios depende del rechazo o aceptación de la hipótesis:

H_0 : *Los elementos de la sucesión* $\{u_i\}$ *son identicamente distribuidos y estadísticamente independientes en el intervalo* $[0,1]$.

La razón por la que hago notar este punto, es debido a que no hay un conjunto de pruebas aceptadas de manera universal para realizar la prueba

de la hipótesis H_0, más adelante veremos algunas de las pruebas típicas para evaluar el nivel de significancia de H_0.

Esto se debe a que un generador de números pseudo aleatorios puede, bajo ciertas pruebas, ser rechazado en tanto que el valor de significancia de la hipótesis no es satisfactorio y por otras pruebas puede ser aceptada.

La otra observación que considero pertienente hacer respecto a la definición del generador de números pseudo-aleatorios, es que debemos de considerar una paradoja lógica, el hecho de pretender obtener una sucesión de números independientes e identicamente distribuidos a partir de un sucesión determinística de valores en un intervalo, en particular en $[0, 1]$, o bien una sucesión de números enteros tal como lo hace la ruleta de Monte Carlo, pero en el sentido de nuestra definición, por medio de operaciones aritméticas, en palabras de J. V. Neumann "Cualquiera que considere métodos aritméticos para obtener digitos aleatorios se encuentra claramente en un error" razón por la que definimos un generador uniforme de números pseudo-aleatorios (GNPA) y no de números aleatorios.

Es decir, un GNPA imita el comportamiento de una variable aleatoria real, pero en términos estrictos, no es realmente una de ellas. Un factor común de las transformaciones empleadas para estos fines, es el comportamiento caótico, generalmente se buscan transformaciones que presenten sensibilidad a condiciones iniciales, aunque se sabe que un proceso determinístico puede tener un periódo de evolución durante el cual presenta comportamiento irregular, típico de los procesos aleatorios.

Uno de estos últimos puede ser útil como un GNPA, antes de que presente cierta regularidad o estancamiento, pues durante el desarrollo de nuestra sucesión de elementos independientes e identicamente distribuidos, pudieramos obtener alguno de los puntos fijos de la transformación $D(u_i)$ o valores que induzcan periodicidad, aunque algunos de estos eventos se pueden corregir con modificaciones en los algoritmos, lo que para la mayoría de los casos representa un incremento en el tiempo de procesamiento.

2.1.1 Independencia estadística

La independencia estadística, es importante ya que si tenemos sucesiones dependendientes o más aún sucesiones que se encuentren autocorrelacionadas, difícilmente el GNPA resultará útil. Consideremos el siguiente ejemplo.

Ejemplo 2.1.2 *Sea S^1 la circunferencia unitaria, $x_0 \in S^1$ y $D_\alpha(\overline{x_i})$ una rotación por un ángulo α, tal que $\frac{\alpha}{2\pi} \in R - Q$.*

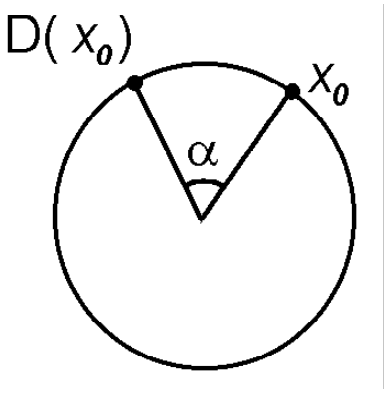


Figura 2.1

Los elementos de esta transformación están idéntica y unifórmemente distribuidos ya que como resultado de aplicar exhaustivamente la transformación D_α, para N suficientemente grande, los elementos de

$$X_N = \{x_i | x_i = D_\alpha(x_{i-1})\}$$

tendrán la misma distribución para cualquier partición discreta de S^1, más aún, si N tiende a infinito, X_N cubrirá S^1ya que $\frac{\alpha}{2\pi}$ no es racional, por lo tanto la transformación no es periódica, lo que significa que X_N es una muestra idénticamente distribuida.

Estadísticamente, para verificar la independencia (o codependencia) de dos variables aleatorias X_N, Y_N se emplea como indicador, el coeficiente de correlación (CC) dado por:

$$C = \frac{\sum_{i=1}^{N} (x_i - \overline{x})(y_i - \overline{y})}{\sigma_x \sigma_y}$$

Donde $\overline{x}$ y $\overline{y}$ son los valores medios de X_N y Y_N respectivamente y σ_x y σ_y son las desviaciones estándar de cada una de la variables aleatorias cuando las variables en cuestión tienen valores numéricos. Geometricamente este indicador es la pendiente de la recta que más aproxima a la nube de puntos, que es resultado de graficar los puntos (x_i, y_i).

Cuando el CC es menor o mayor que 0 se dice que ambas variables están correlacionadas negativa o positivamente según sea el caso, cuando es igual a 0 (o muy cercano) se dice que las variables aleatorias no presentan correlación o bien que son independientes entre sí.

Extendiendo la noción de coeficiente de correlación, definimos el coeficiente de autocorrelación de una muestra X_N :

$$f(k) = \frac{\sum_{i=0}^{N-k} (x_i - (\overline{x_i}))(x_{i+k} - (\overline{x_{i+k}}))}{\sigma_{x_i} \sigma_{x_{i+k}}} \quad k \in 1...N$$

El coeficiente de autocorrelación, es función del índice k y por ende del tamaño de la subsucesión que se toma, y se le conoce como función de autocorrelación. Del mismo modo en que el CC nos ayuda a discernir si dos variables están o no correlacionadas, decimos que una muestra tiene elementos independientes si el valor medio $\overline{f(k)} \approx 0$ y la varianza de $f(k)$ denotada por $S^2(f(k)) \approx 0$.

Al analizar la función de autocorrelación $f(k)$ de la sucesión $X_N = \{x_i | x_i = D_\alpha(x_{i-1})\}$ del ejemplo con los parámetros $\alpha = 0.9742$ y $x_0 = 0.5$ en el espacio $S^1 x S^1$ se observa:

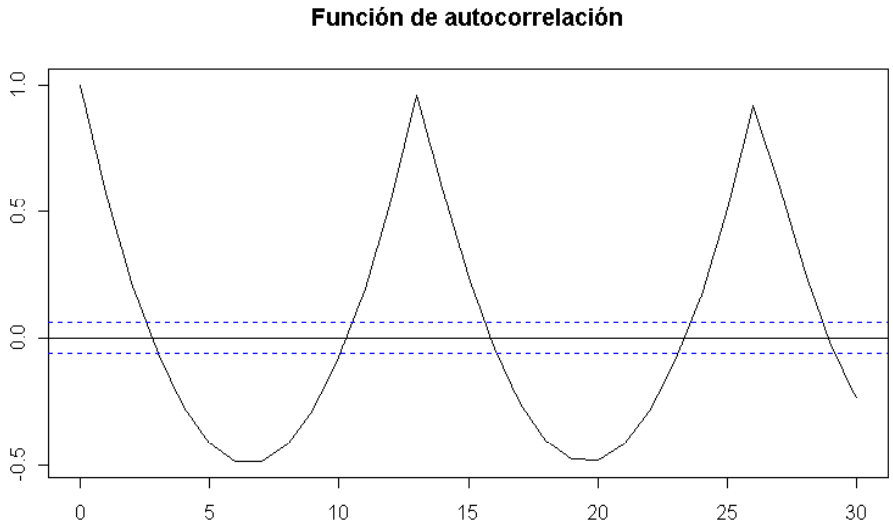


Figura 2.2 : función de autocorrelación

De lo que se puede concluir que los elementos de la sucesión no son independientes aunque tiene valor medio 0 ya que la varianza es muy grande, y con ello queda descartado como un GNPA.

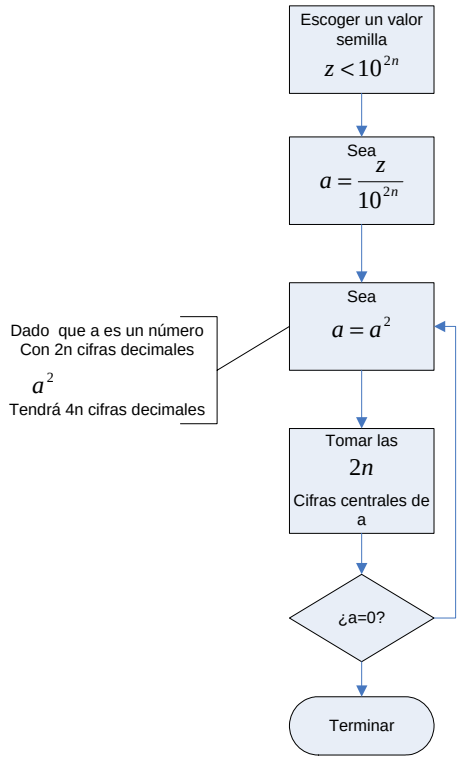
Nota 2.1.3 *Dadas dos variables aleatorias en general, el hecho de que el coeficiente de correlación sea diferente de cero no implica necesariamente que estén correlacionadas ya que depende de la naturaleza del fenómeno que se estudia y de otros factores, entre otros la metodología empleada, pero en este caso tanto el CAC como la FAC claramente determinan la independencia de los elementos de una muestra en tanto que poseemos y dictaminamos de manera determinística cada uno de ellos.*

Es importante notar que la generación de números pseudo-aleatorios, depende en principio de la transformación escogida para generarlos y del valor inicial, u_0 al que se le conoce en la literatura como semilla, puesto que dos muestras U_n, V_n de la misma extensión y con valores iniciales $u_0 = v_0$, es decir la misma semilla y la misma transformación, ambas serán exactamente

iguales. Como comentario marginal, esto último nos permite vincular los métodos Monte Carlo con otra área de las matemáticas conocida como cadenas de Markov y con ello ampliar tanto las aplicaciones como la teoría de los métodos Monte Carlo.

2.1.2 Generador de centros cuadrados (J. Von Neumann)

Uno de los primeros GNPA, fue ideado por J. Von Neumann durante los primeros años de desarrollo computacional, y es conocido como el algoritmo de los centros cuadrados:



Ejemplo 2.1.4 *(Programa No B.2.1 en apéndice B)Sea n=2 y z=2638*

a_i	a_i^2
0.2638	0.06*9590*44
0.9590	0.91*968*100
0.9681	0.93*721*761
0.7217	0.52*085*089

Aunque es ingenioso, este generador no es muy bueno, debido a que presenta muchas limitantes, en el mejor de los casos solo puede generar una

sucesión de hasta 10^{2n} números pseudo-aleatorios, de hecho solo con unos cuantos de los valores posibles de la semilla obtenemos sucesiones largas debido a que, para muchos valores iniciales la sucesión converge a cero muy rápidamente (en particular con el valor inicial z=2638 solo obtendremos 12 valores distintos antes de que esto ocurra) además de que muchas de las sucesiones que presentan extensiones grandes son periódicas (figura 2.2).

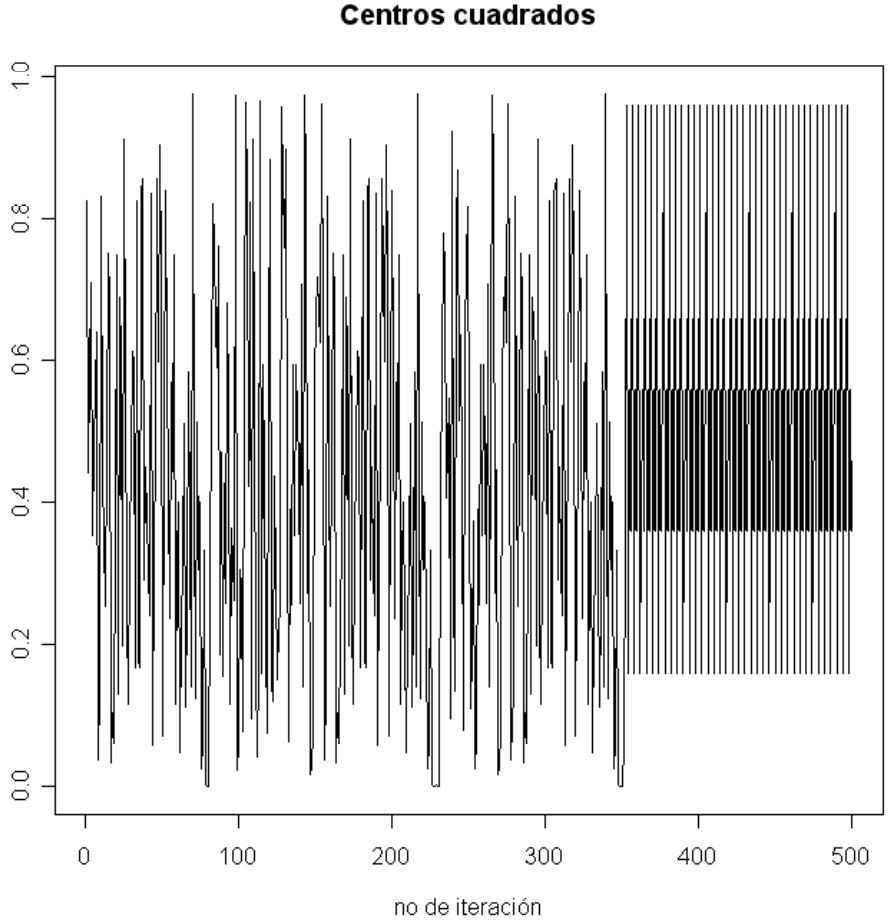


Figura 2.3.

Éstos son los problemas más comunes a los que debemos enfrentarnos cuando estamos dedicados a la programación de GNPA, es por ello que debemos de realizar pruebas para evaluar la aleatoriedad.

El algoritmo de los centros cuadrados es un ejemplo de un proceso determinístico que durante un tiempo de evolución se comporta como azaroso, pero eventualmente presenta regularidad (periodicidad o estancamiento).

Para salvar este último punto, podemos hacer una modificación en el algoritmo, de modo que el programa compare cada término con los anteriores y salga del ciclo si encuentra un valor idéntico al último generado, o inclusive para salvar el problema de la representación finita, si el elemento n-ésimo se

encuentra en un intervalo $(D^i(x) - \delta, D^i(x) + \delta)$, para alguna $i < n$, todo esto con el fin de evitar periodicidad o repetición infinita de los puntos fijos.

Esto implica el empleo de mayores recursos (almacenamiento y lectura de datos), lo cual incrementa con cada ciclo realizado por el algoritmo pues en el paso $i - ésimo$ debe de realizar $i - 1$ lecturas y comparaciones para almacenar el último dato de la sucesión, esto último puede no ser relevante si deseamos una sucesión de NPA de unos pocos miles, pero cuando elevamos nuestras necesidades a sucesiones con millones de elementos, el tiempo de procesamiento incrementa descontroladamente.

No obstante las dificultades que implica usar este GNPA, fué empleado durante algunos años en investigaciones posteriores a la primera publicación concerniente al Método Monte Carlo(del mismo título) entre otros importantes trabajos en los que se utilizó el algoritmo de los centros cuadrados se encuentra Equation of State Calculations by Fast Computing Machines por N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller 1953 donde se implementó por primera vez el conocido algoritmo de Metropolis (ver capítulo 3).

2.1.3 La Función logística

Otro generador básico de números pseudo-aleatorios es la transformación:

$D_\alpha(x) = \alpha x(1 - x)$.

En la que, para $3.57 \leq \alpha \leq 4$ se obtiene configuraciones caóticas (Ver programa No B.2.2), el valor máximo de la función logística se alcanza cuando $x = \frac{1}{2}$ y además el valor máximo es $D(\frac{1}{2}) = \frac{\alpha}{4}$ por lo que si deseamos obtener una función distribuida en $[0, 1]$ $\alpha = 4$.

Veamos el comportamiento de D cuando incrementamos el orden de composición:

Ejemplo 2.1.5 *Gráficas de la función logística*

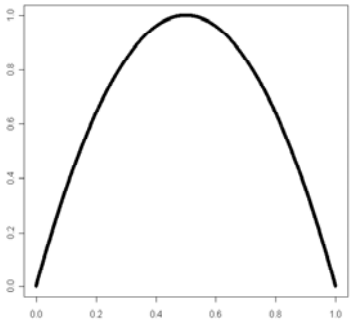


Figura 2.4
$L(x) = (D^i(x), D^{i+1}(x))$

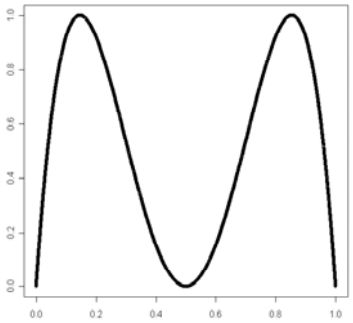


Figura 2.5
$L(x) = (D^i(x), D^{i+2}(x))$

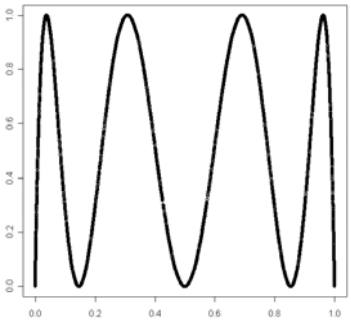


Figura 2.6
$L(x) = (D^i(x), D^{i+3}(x))$

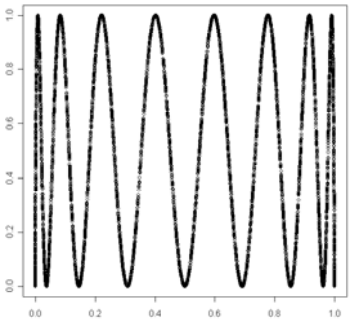


Figura 2.7
$L(x) = (D^i(x), D^{i+4}(x))$

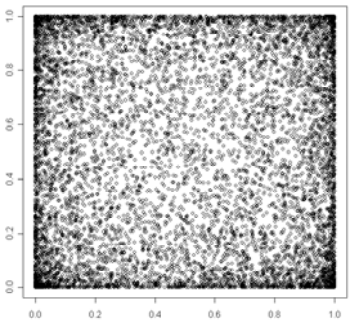


Figura 2.8
$L(x) = (D^i(x), D^{i+10}(x))$

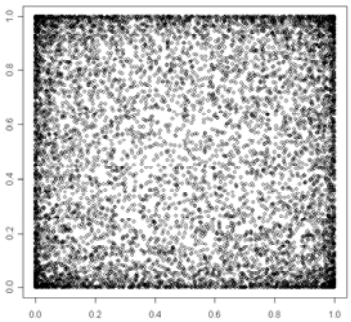


Figura 2.9
$L(x) = (D^i(x), D^{i+100}(x))$

Arriba, en las gráficas contra si misma es de esperarse que, si nuestro GNPA, es uniforme, el cuadrado $[0,1] \times [0,1]$ se llene uniformemente, sin que se observen aglomeraciones o huecos muy notables en alguna de sus partes.

Veamos una imagen de la gráfica de la función de autocorrelación de $D(x)$:

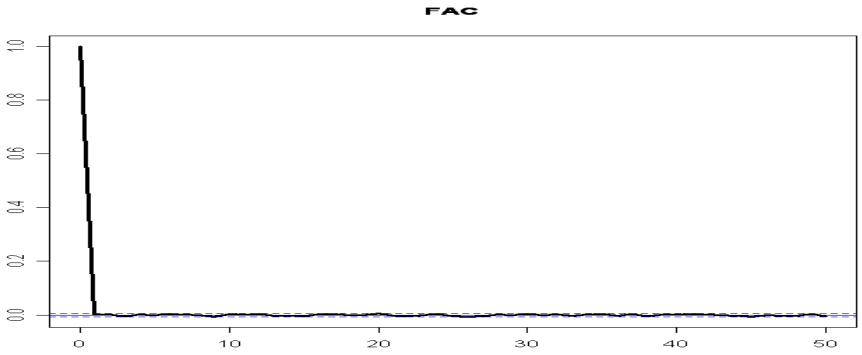


Figura 2.10 FAC: Función logística

En donde podemos reconocer la aparente aleatoriedad e independencia de la sucesión de elementos en tanto que la función de autocorrelación tiene

valor medio 0, es decir podemos pensar que hay independencia estadística, y de acuerdo a las gráficas de las figuras 2.7 y 2.8, aparentemente, este GNPA, llena el cuadrado$[0,1] \times [0,1]$ conforme incrementamos el orden de la composición en $D(x)$, pero aun no hemos comprobado la uniformidad de $D(x)$ en $[0,1]$ para ello, veamos su histograma de frecuencias (figura 2.9).

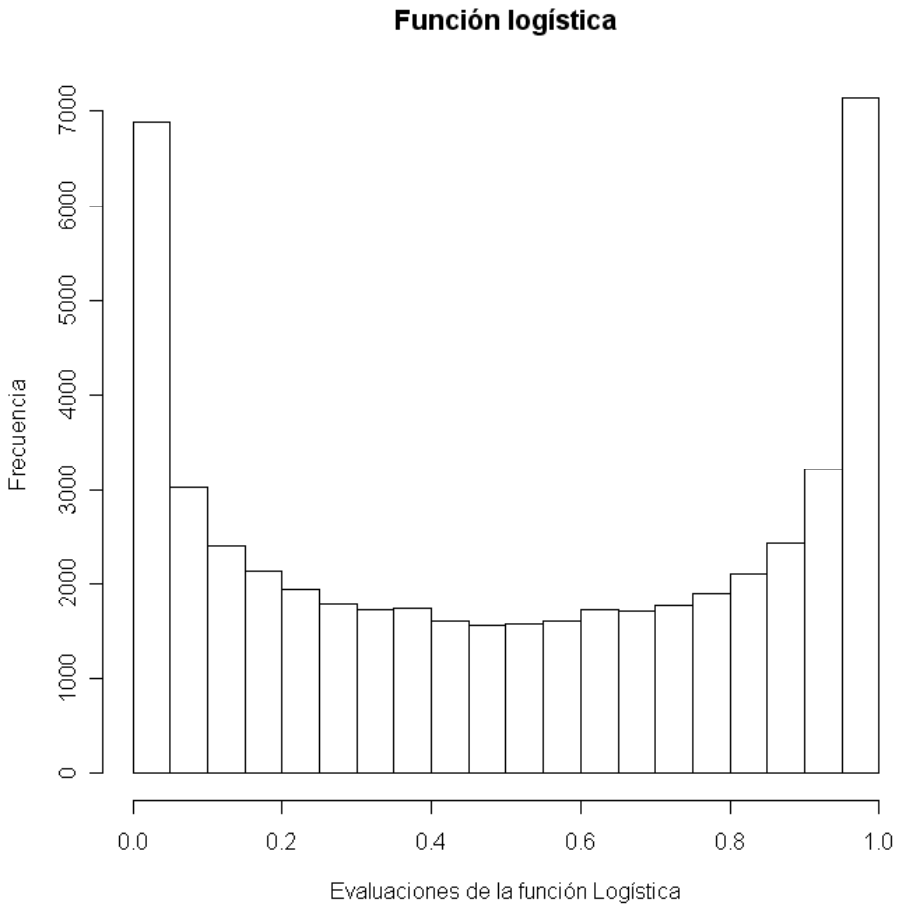


Figura 2.11

Lo que nos indica que toma más valores cercanos a los extremos del intervalo, razón por la que el cuadrado, presenta mayor densidad cerca de los puntos $(0,0),(0,1),(1,0),(1,1)$, esto quiere decir, que la distribución de la variable aleatoria que deseamos simular, no será uniforme si empleamos este generador.

Podemos observar que la distribución de los números generados por la función logística tiene función de densidad teórica (ver [3])

$$f(x) = \frac{1}{\pi\sqrt{x\,(1-x)}}$$

Para obtener una función con distribución uniforme a partir de esta

tranformación empleamos el cambio de variable (ver [3])

$$Y_n = 2\frac{arcsen(X_n)}{\pi}$$

donde $X_{n+1} = D_\alpha(X_n)$

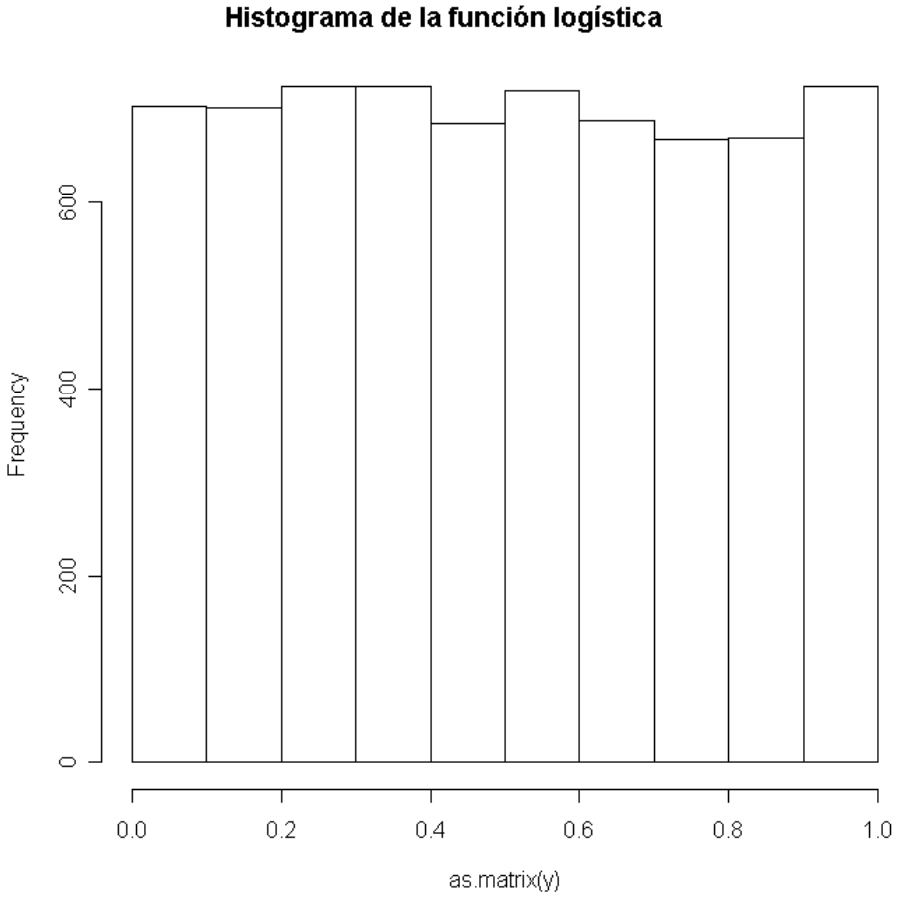


Figura 2.12: Función logística transformada

Con esta transformación (programa B.2.3), se obtiene una histograma aparentemente uniforme, hay que tener en cuenta que dado cualquier GNPA, la representación finita de números reales por la computadora, puede cambiar por mucho el comportamiento de una sistema dinámico, con las condiciones iniciales planteadas, además se requiere llamar varias veces a la función logística, lo que implica mayor tiempo de procesamiento, éstas son algunas de las desventajas de este generador si deseamos uniformidad, aunque puede resultar útil para generar una sucesión con la FDP asociada, veamos otro ejemplo de GNPA.

2.1.4 Método de congruencias (KISS)

Es muy importante señalar que en cualquier muestra o sucesión obtenida de un GNPA tiene un límite de extensión dado por M el máximo entero

aceptado por la computadora por lo que las sucesiones obtenidas en una computadora eléctrónica son finitas y definimos el período de un GNPA de la siguiente manera.

Definición 2.1.6 *El período T de un GNPA es el menor entero tal que $u_i = u_{i+T}$ para toda i es decir D^T es la función identidad.*

Si la extensión de la sucesión de números pseudoaleatorios exede a M, éste puede resultar en un descontrol (ciclos infinitos, desbordamientos, etc.)

Consideremos la función $D : [0,1] \to [0,1]$ tal que $D(x) = ax + b \ mod \ Z, \ a \neq 1$. La función $D(x)$ está definida de manera que elimina la parte entera de la imagen de modo tal, que la imagen de R es un conjunto de segmentos paralelos en $C = [0,1] \times [0,1]$, a esta transformación se le conoce con el nombre KISS. Es importante notar que dichos segmentos llenarían completamente el cuadrado unitario si la pendiente a fuera irracional, pero dada la naturaleza discreta de los elementos de la transformación en términos de cómputo electrónico, debemos de buscar parámetros óptimos.

Por lo anterior, comenzemos considerando una transformación $D : Z \to Z_m$ definida por

$$D(r) = pr + q \ mod \ m$$

Como primer objetivo, hay que tratar de obtener el máximo de elementos posibles de esta sucesión, el máximo teórico está dado por m, además deseamos que presente configuraciones caóticas.

Ejemplo 2.1.7 *Sea $D(r) = 37r + 2 \ mod \ 256$ Sea el valor semilla $r_0 = 1$.*

$$\begin{array}{ll} x_0 & 1 \\ x_1 & 39 \\ x_2 & 165 \\ x_3 & 219 \\ x_4 & 169 \\ x_5 & 111 \end{array}$$

En particular con estos parámetros la semilla genera una órbita de 128 elementos. Veamos la órbita de x_0 en la gráfica (x_i, x_{i+1}).

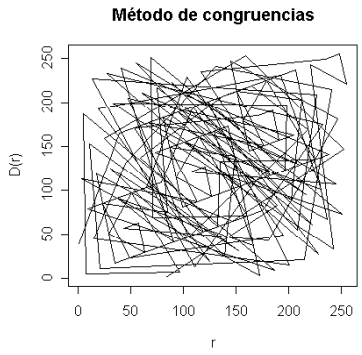


Figura 2.13
Órbita de $r_0 = 1$

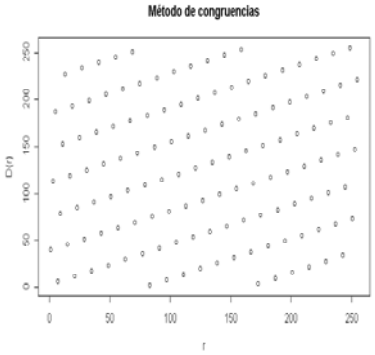


Figura 2.14
Dispersión

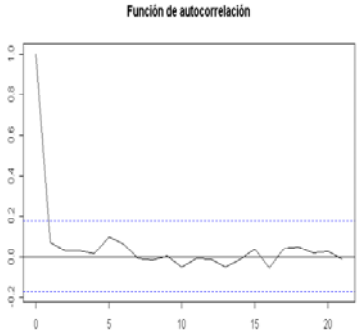


Figura 2.15
Función de autocorrelación

En la segunda gráfica se puede apreciar claramente la distribución uniforme de los elementos de la sucesión.

Por último, para obtener a una variable con distribución uniforme en $[0, 1]$ dividimos cada término por m.

Para sortear la dificultad que representa la periodicidad de una función D, el generador KISS se puede componer con otras funciones lineales para maximizar la órbita del valor semilla [3].

Ejemplo 2.1.8 *Consideremos el primer generador de KISS Programa No B.2.4 en apéndice B con los parámetros* $p = 950,\ q = 1$

De donde obtenemos las siguientes gráficas (figuras 2.16 y 2.17).

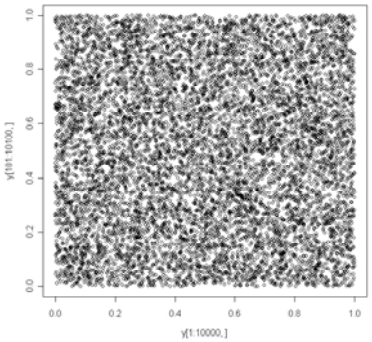


Figura 2.16

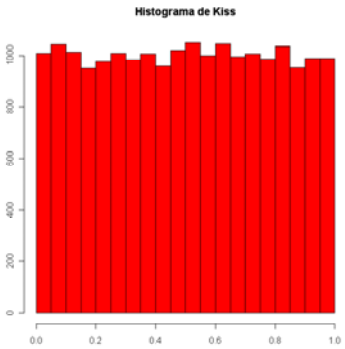


Figura 2.17

Una ventaja que presenta usar generadores de congruencias, es que se requieren pocas operaciones aritméticas para obtenerlos, comparado por ejemplo con el flujo de información de la función logística y el de centros cuadrados, el método de congruencias (KISS) es más simple en términos de cómputo y más eficiente pues esto último reduce el tiempo de operación.

Nota 2.1.9 *El término KISS son las siglas en inglés para la expresión (Keep it Simple Stupid).*

2.2 Pruebas de aleatoriedad

A pesar de que muchos paquetes, actualmente tiene un generador de números pseudo-aleatorios, siempre es conveniente aplicarles pruebas, para verificar la aleatoriedad de los mismos y las características arriba mencionadas, puesto que no necesariamente satisfacen las condiciones de un buen GNPA, existen diversos tipos de pruebas para probar la aleatoriedad de los mismos, varios de ellos los he ejemplificado anteriormente, observamos, como se va llenando el cuadrado unitario, el comportamiento de la función de autocorrelación, de la función de densidad, pero todas estas son pruebas empíricas, pues debido a, entre otras cosas, la discretización de los elementos de la sucesión, su representación finita, etc., nos impiden asegurar un comportamiento, pues por ejemplo estas causan que en términos estrictos jamás se alcance a llenar todo el cuadrado. A continuación veremos pruebas mas formales, con el fin de probar la uniformidad e independencia de un GNPA.

2.2.1 El juego del Caos

El juego del caos, es un algoritmo que puede ser empleado como una prueba de aleatoriedad. Consideremos el cuadrado $C = [0,1]\times[0,1]$ con las esquinas ordenadas $P_1 = (0,1)$, $P_2 = (0,0)$, $P_3 = (1,0)$, $P_4 = (1,1)$. Sea $U = \{u_i | u_i = D(u_{i-1})\ D : C \to C\}$ una sucesión de números pseudoaleatorios y sea $f : U \to R$ dado por:

$$f(u) = \begin{cases} P_1 & \text{si} \quad u \in [0, \frac{1}{4}) \\ P_2 & \text{si} \quad u \in [\frac{1}{4}, \frac{1}{2}) \\ P_3 & \text{si} \quad u \in [\frac{1}{2}, \frac{3}{4}) \\ P_4 & \text{si} \quad u \in [\frac{3}{4}, 1) \end{cases}$$

Definimos la sucesión $q_i := (\frac{x_{i-1}-X_i}{2}, \frac{y_{i-1}-Y_i}{2})$ donde $P_i = (X_i, Y_i)$. Si el GNPA no presenta regularidad, el llenado del cuadrado deberá ser uniforme, por otro lado si se tiene un patrón o una huella con aglomeraciones decimos que el generador no presenta uniformidad.

Ahora veamos el juego del caos para algunos generadores (Programa No B.2.5).

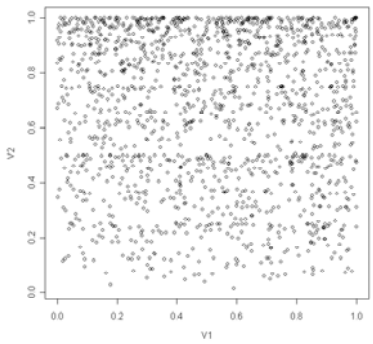


Figura 2.18
Centros cuadrados

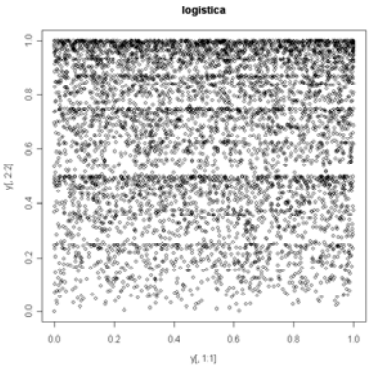


Figura 2.19
Logística

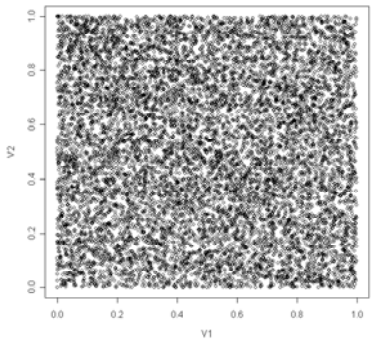


Figura 2.20
KISS

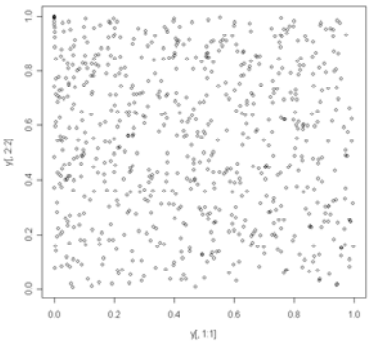


Figura 2.21
Logística transformada

En las gráficas que se encuentran arriba se observa el resultado del juego del caos en las transformaciones previamente comentadas, donde se observan algunos patrones, es decir su comportamiento presenta cierta tendencia, y aunque visualmente aparentan cubrir uniformemente $[0, 1] \times [0, 1]$ debemos de recordar que todas corresponden a sistemas caóticos, por lo que es natural observar dichos patrones.

Las siguientes pruebas son muy comunes para evaluar muestras con fines de validez estadística, es útil recordar el concepto de función de distribución acumulativa.

Definición 2.2.1 *Si $f(x)$ es una función de distribución de probabilidad de una variable aleatoria x continua decimos que*

$$g(t) = \int_m^t f(x)dx \;\; t \in [m, M] = rango\ de\ x$$

es la función de distribución acumulativa (FDA) de $f(x)$ donde m es el valor mínimo y M el valor máximo de la variable aleatoria x.

2.2.2 Prueba de Ji-cuadrada

La prueba de bondad de ajuste ji-cuadrada es de la más empleadas de las pruebas estadísticas, fue propuesta por Pearson en 1900.

Sea $X_1, X_2, ..., X_N$ una muestra de una población, de la cual desconocemos su función de distribución acumulativa (FDA) $F_X(x)$, desamos probar la hipótesis nula:

$$H_0 : F_X(x) = F_0(x), \qquad \forall\ x$$

Donde $F_0(x)$ es una FDA completamente especificada, contra la hipótesis altrernativa:

$$H_1 : F_X(x) \neq F_0(x), \qquad p.a.\ x$$

Supongamos que las N observaciones están agrupadas en k categorías mutuamente excluyentes, denotamos por N_j el número de elementos que se encuentran en la $j - ésima$ categoría y por Np_j^0 el valor esperado de esta última, cuando H_0 es verdadera.

El criterio sugerido por Pearson usa el siguiente estadístico:

$$Y = \sum_{j=1}^{k} \frac{(N_j - Np_j^0)^2}{Np_j^0} \qquad \sum_{j=1}^{k} N_j = N$$

El cual tiende a 0 cuando H_0 es verdadera, la FDP de la variable aleatoria Y es compleja, pero para muestras grandes, su distribución es semejante a ji-cuadrada con $k - 1$ grados de libertad.

Bajo la hipótesis nula esperamos que:

$$P(Y > \chi^2_{1-\alpha}) = \alpha$$

donde α es el nivel de significancia. En particular el caso de la prueba de bondad de ajuste para una muestra con distribución uniforme en $[0,1]$ (Un GNPA por ejemplo), dividimos el intervalo en k subintervalos ajenos de longitud $\frac{1}{k}$ $I_j = [\frac{j-1}{k}, \frac{j}{k})$, por lo que:

$$Np_j^0 = \frac{N}{k}$$

y en este caso tenemos que:

$$Y = \frac{k}{N} \sum_{j=1}^{k} (N_j - \frac{N}{K})^2$$

Es útil recordar que ninguna sucesión obtenida a partir de un GNPA es densa en el intervalo $[0,1]$ por lo que para emplear la prueba de ji-cuadrada como prueba de aleatoriedad, la partición del intervalo para la aplicación de la prueba de ji cuadrada, debe ser acorde a la muestra que deseamos evaluar.

Ejemplo 2.2.2 *Prueba de bondad de ajuste, para método de congruencias, Programa No B.2.6 tenemos que para una muestra de tamaño* $N = 400000$ *y* $k = 100$ *subintervalos tenemos que* $Y = 476.699$. *Por lo que al compararla con el valor esperado ji-cuadrada con 99 grados de libertad tenemos que:*

$$P(Y > \chi^2) \approx 1$$

Por lo que el generador de congruencias bajo esta prueba, es aceptado.

2.2.3 Prueba de Kolmogorov-Smirnov

Sea $X_1, X_2, ..., X_N$ una muestra de una poblaciónde la cual no conocemos la FDA, definimos la función de distribución acumulada de la muestra como:

$$F_N(x) = \frac{N_j}{N}$$

Donde N_j es el número de elementos X_i de la muestra tales que $X_i \leq x$, y denotamos

$$N_j = \sum_{i=1}^{N} I_{(-\infty,x]}(X_i)$$

Donde el término:

$$I_{(-\infty}, x](X_i) = \begin{cases} 1, \ si - \infty \leq X \leq x \\ 0, \qquad en\ otro\ caso \end{cases}$$

Es claro que para x fija, $F_N(x)$ tiene distribución Bernoulli con parámetro

$$p = \int_{-\infty}^{x} f(s)ds$$

donde f es la función de distribución de probabilidad de la variable aleatoria X, por lo cual, al ejercer esta prueba en un conjunto de observaciones debemos calcular el parámetro p para un conjunto discreto de valores de X.

No existe una regla para decidir cuantas pruebas de aleatoriedad deben de realizarse para confiar en un GNPA en particular, tampoco para considerar unas pruebas mejores que otras, mucho menos que hacer en caso de que un GNPA, pase un conjunto de pruebas y algunas no, además, las pruebas de aleatoriedad, pueden a su vez requerir mas tiempo de procesamiento que los generadores mismos. Para mayores referencias de pruebas de aleatoriedad se puede consultar R. Rubinstein Simulation and the Monte Carlo Method [9].

2.3 Funciones de densidad de probabilidad

2.3.1 Distribución de probabilidad discreta

Ejemplo 2.3.1 *Supongamos que deseamos simular una variable aletoria cuyo espacio muestral tiene unicamente 4 valores posibles,* x_1, x_2, x_3, x_4 *y sus respectivas probabilidades* $P(X = x_1) = \frac{1}{3}, P(X = x_2) = \frac{1}{4}, P(X = x_3) = \frac{1}{4}, P(X = x_4) = \frac{1}{6}$*,a diferencia de los ejemplos dados en el capítulo anterior, la variable aleatoria* X *no tiene una ditribución uniforme, para generar una varible no uniforme observemos una colección* $U = \{u_i\}$ *de números generados tal que* $U \sim u[0,1]$ *y convenir lo siguiente,*

$$f(u) = \begin{cases} x_1 & \text{si} \quad u \in [0, \frac{1}{3}) \\ x_2 & \text{si} \quad u \in [\frac{1}{3}, \frac{7}{12}) \\ x_3 & \text{si} \quad u \in [\frac{7}{12}, \frac{5}{6}) \\ x_4 & \text{si} \quad u \in [\frac{5}{6}, 1) \end{cases}$$

Con lo que hemos llevado la variable aleatoria U con distribución uniforme en X através de la transformación f. (programa B.2.6)

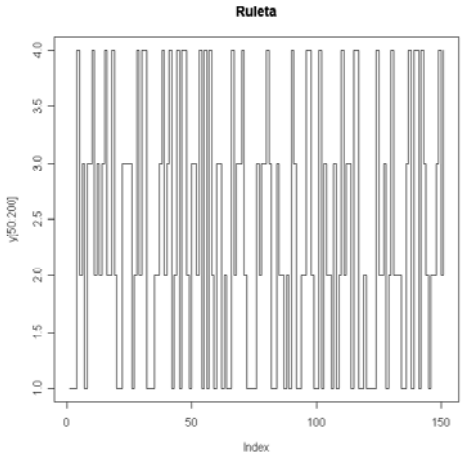


Figura 2.22 Ruleta

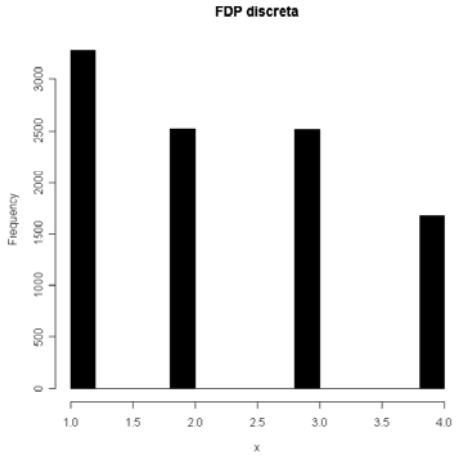


Figura 2.23: Histograma

Los métodos de simulación estan basados en la generación de variables aleatorias que están ditribuidos de acuerdo a una función de distribución no necesariamente especificada. Los primeros generadores, tienen distribución uniforme en $[0, 1]$ ya que, da la representación probabilistíca de aleatoriedad y porque podemos encontrar una transformación que nos lleve una variable aleatoria con distribución uniforme en [0,1] en otra distribución de probabilidad como podemos ver en el siguiente lema.

Definición 2.3.2 *Para una función creciente* $F : R \to R$ *la función inversa generalizada de* F*, denotada por* F^-*, es la función definida por:*

$$F^-(u) = \inf\{x; F(x) \geq x\}$$

Lema 2.3.3 *Si* U *se distribuye uniformemente en* $[0, 1]$ *entonces la variable aleatoria* $F^-(U)$ *tiene la distribución* F*.*

Demostración:
Por definición de función inversa generalizada de F tenemos que:

$$F(F^-(u)) \geq u$$

Por otro lado

$$F^-(F(x)) \leq x$$

Entonces

$$\{(u, x) : F^-(u) \leq x\} = \{(u, x) : F(x) \geq u\}$$

luego:

$$P(F^{-}(U) \leq x) = P(U \leq F(x)) = F(x)$$

■

Por lo que es suficiente generar U con distribución uniforme en $[0, 1]$ para obtener una variable aleatoria $X \sim F$ y emplear la transformación $x = F^{-}(u)$.

Es por este resultado por el que la generación de variables aleatorias distribuidas uniformemente cobra tanta importancia, puesto que otras distribuciones pueden ser representadas como una transformación determinística de variables aleatorias distribuidas uniformemente.

Inversamente, el lema también nos permite ver que si U no se distribuye uniformemente no podemos asegurar la validez externa de la simulación.

Para llevar una variable aleatoria continua de distribución uniforme en $[0, 1]$ a una variable discreta, hacemos una partición en el intervalo en el que se distribuye uniformemente la variable aleatoria generada, basicamente es lo que hace la ruleta de Monte Carlo la cual está seccionada de manera que la probabilidad de que el resultado sea cualquier número sea la misma, podemos pensar en generar una ruleta para cada distribución discreta posible:

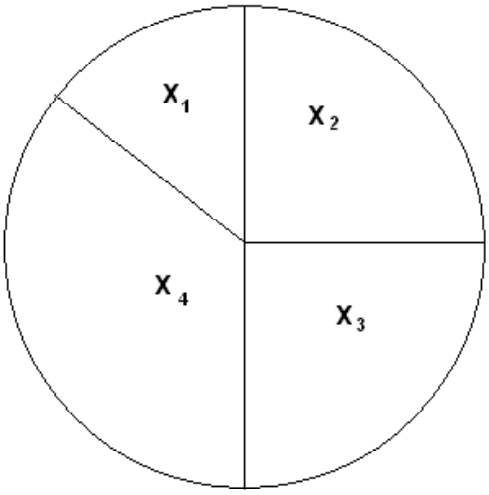


Figura 2.24: Ruleta con distribución no uniforme

Sea U distribuida uniformemente en $[0, 1]$ en general si tenemos $\{x_1, x_2, ..., x_n\}$con $P(x = x_i) = p_i$, consideramos $X = x_i$ si $\sum_{k=0}^{i-1} p_i < u < \sum_{k=0}^{i} p_i$. donde $P(x = x_0) = p_0 = 0$

Ahora veremos como transformar a las funciones de distribución de probabilidad mas empleadas en el ramo estadístico, actualmente cualquier paquete estadístico de procesamiento de datos tienen algoritmos de transformación de variables, para cada FDP puede haber varios de ellos.

2.3.2 Función de distribución de probabilidad continua

Los siguientes ejemplos, de acuerdo al lema 2.3.3 se obtienen a partir de la definición de función inversa generalizada.

Ejemplo 2.3.4 *Distribución Exponencial*

Sea U distribuida uniformemente en $[0,1]$ y $X = -Ln(U)$ entonces la variable aleatoria X tiene distribución exponencial:

$$P(X \leq x) =$$

$$P(-Ln(U) \leq x) =$$

$$P(U \leq e^{-x}) =$$

$$\int_0^x e^{-s} ds$$

que es la función de distribución de probabilidad exponencial con parámetro $\lambda = 1$ en general para un paramétro λ cualquiera debemos de emplear la transformación $X = -Ln(\frac{U}{\lambda})$.

La función de distribución exponencial puede ser útil para generar otras FDP. Veamos algunos casos:

Ejemplo 2.3.5 *ji cuadrada (Programa B.2.8)*

Sea $Y = -2 \sum_{j=1}^{v} Ln(U_j) \sim \chi^2_{2v}, v \in N$

Un versión mas general del ejemplo anterior es la función gama.

Ejemplo 2.3.6 *Función gama (Programa B.2.9)*

Sea $Y = -\beta \sum_{j=1}^{\alpha} Ln(U_j) \sim Ga(\alpha, \beta), \alpha \in N$

Ambos ejemplos dependen de que los parámetros (v y α respectivamente) sean valores enteros.

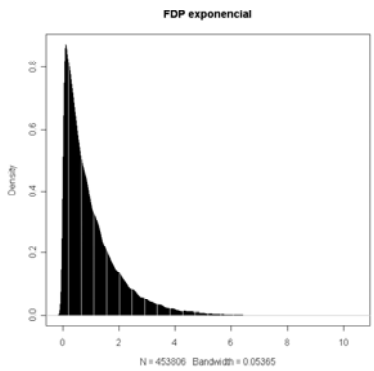


Figura 2.25
Exponencial

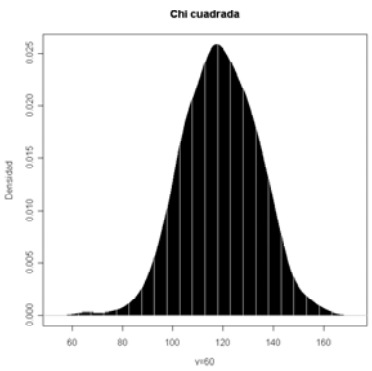


Figura 2.26
χ^2 con $v = 60$

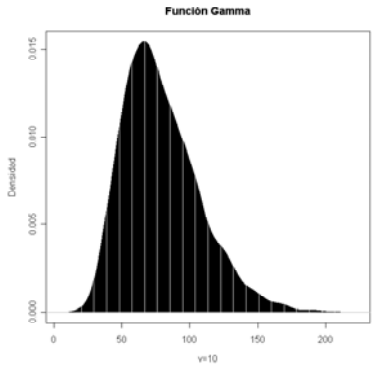
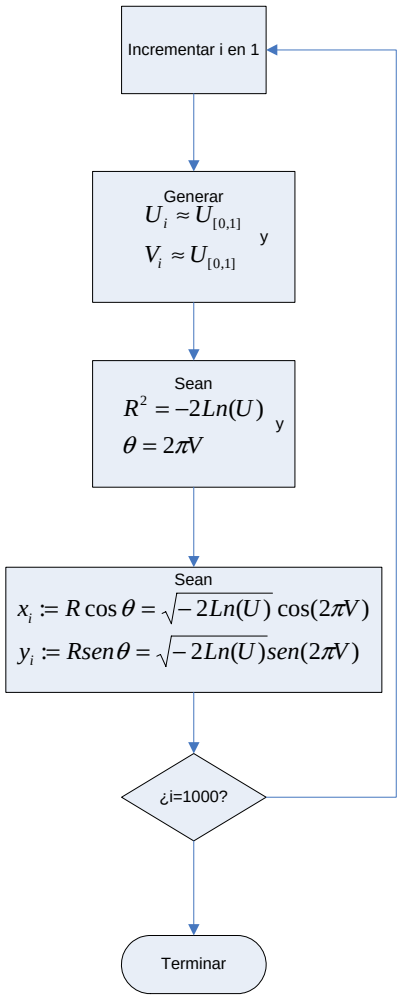


Figura 2.27
Gama con $\beta = 10$ y $v = 20$

Ejemplo 2.3.7 *Distribución normal (Programa B.2.10)*

El siguiente Algoritmo para generar una variable con distribución normal a partir de dos variables con distribución uniforme, se conoce como el algoritmo de Box-Muller o transformación polar.



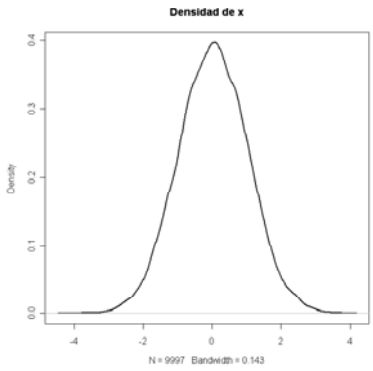


Figura 2.28
Densidad en x

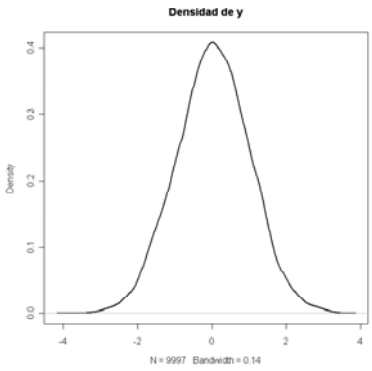


Figura 2.29
Densidad en y

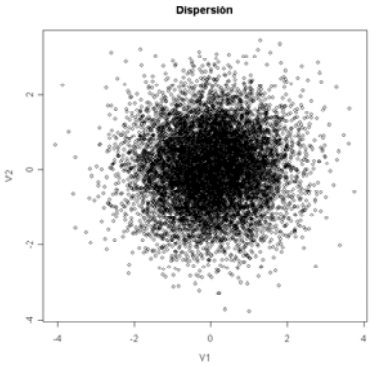


Figura 2.30

Dispersión

Ambas variables aleatorias tanto x como y presentan distribución normal, y si asignamos sucesiones (U_i, V_i) y (U_j, V_j) independientes, también podemos asegurar un coeficiente de correlación bajo:

Matriz de correlación	x	y
x	1.0000000	0.03416969
y	0.03416969	1.0000000

Proposición 2.3.8 *Si U y V son variables aleatorias con distribución uiforme, entonces las variables aleatorias $x = \sqrt{-2Ln(U)}\cos(2\pi V)$ y $= \sqrt{-2Ln(U)}sen(2\pi V)$, se distribuyen normalmente con media $\mu = 0$ y varianza $S^2 = 1$.*

Demostración: Sean $R = \sqrt{-2Ln(U)}$ y $\theta = 2\pi V$. Siguiendo el mismo razonamiento que en el caso de la variable con distribución exponencial tenemos que $X := R\cos(2\pi\theta)$ y $Y := Rsen(2\pi\theta) \Rightarrow$

$$P\left(X \leq x\right) =$$

$$P\left(\sqrt{-2Ln(U)}\cos(2\pi V) \leq x\right) =$$

$$P\left(Ln(U)\cos^2(2\pi V) \leq \frac{-x^2}{2}\right)$$

Por otro lado tenemos que

$$P\left(Ln(U)sen^2(2\pi V) \leq \frac{-y^2}{2}\right)$$

Por lo tanto

$$P\left(Ln(U)\cos^2(2\pi V) + Ln(U)sen^2(2\pi V) \leq \frac{-x^2-y^2}{2}\right) =$$

$$P\left(U \leq e^{-\frac{x^2+y^2}{2}}\right) =$$

$$P\left(U \leq e^{-\frac{R^2}{2}}\right) =$$

$$e^{-\frac{R^2}{2}}$$

■

Ejemplo 2.3.9 *Accept-Reject (método de rechazo)*

El método de rechazo es útil para una FDP continua, y no se necesita definir la función inversa generalizada (existen funciones para las que no se puede definir) solamente se necesita expresar en términos analíticos. Se puede interpretar como una variación del método Hit or Miss. Supongamos que deseamos emular una v.a. con FDP $f(x)$, sea $c \geq f(x)\ \forall\ x \in [a,b]$, generamos $U\ y\ V$ distribuidas uniformemente en $[0,1] \Rightarrow$

$$\begin{cases} Aceptar\ x\ si & f((b-a)u_i) < cv_i \\ \text{Re}chazar\ x\ si & f((b-a)u_i) \geq cv_i \end{cases}$$

Por lo que x tiene distribución:

$$P(a_i \leq x \leq b_i) =$$
$$P((b_i - a_i)f(u_i) \leq v_i(b-a)) =$$
$$c\int_{a_i}^{b_i} f(x)$$

Con el programa No B.2.12 para una FDP Normal, exponencial y polinomial respectivamente tenemos las siguientes gráficas de densidad:

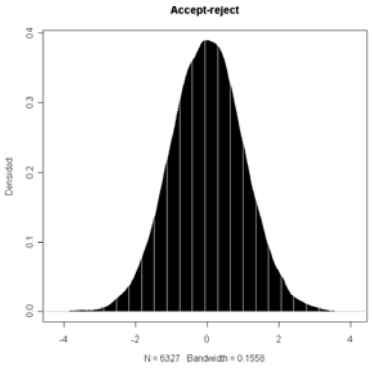


Figura 2.31
Accept-reject Normal

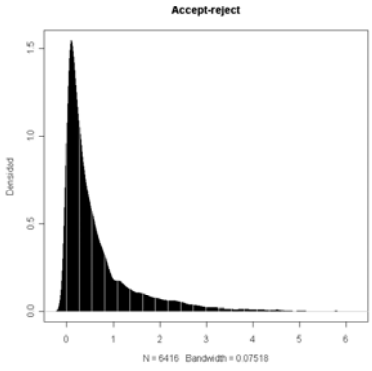


Figura 2.32
Accept-reject FDP exponencial

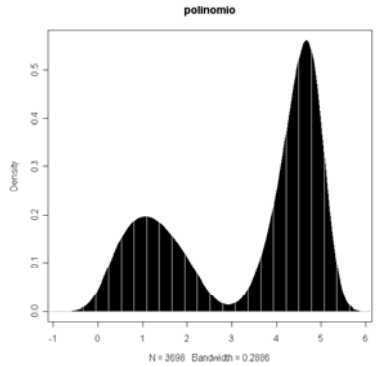


Figura 2.33
Accept-reject FDP Polinomial

Una manera de obtener una mayor cantidad de NPA con este método es escoger de manera óptima el valor c, en los casos anteriores $c = \max\{f(x)/x \in [a, b]\}$

Capítulo 3

Aplicaciones de los métodos Monte Carlo

Dos de los problemas más socorridos en el área de la simulación computacional, son la integración y la optimización de funciones, existen múltiples métodos para obtener aproximaciones en ambos casos, a continuación describiré algunos de ellos.

3.1 Integrales Monte Carlo

3.1.1 Integración Monte Carlo clásica

La integración Monte Carlo clásica, consiste en aproximar el valor de una integral definida de la siguiente manera:

Sea $h(x)$la función a integrar definimos:

$$E_f[h(x)] = \int_\chi h(x)f(x)dx$$

Donde $f(x)$ es una función de densidad con $(x_1, x_2, ..., x_m)$ elementos de $f(x)$, ahora, por la ley de los grandes números tenemos que:

$$\overline{h(x)} = \frac{1}{m}\sum_{i=1}^{m} h(x_i)$$

Converge a $E_f[h(x)]$ cuando m tiende a infinito. Más aún, si el valor esperado de h^2 bajo f, es finito, la velocidad de convergencia puede verificarse con la varianza:

$$var(\overline{h}_m) = \frac{1}{m}\int_\chi (h(x) - E_f[h(x)])^2 f(x)dx$$

la que puede estimarse con la muestra $(x_1, x_2, ..., x_m)$ con:

$$v_m = \frac{1}{m^2} \sum_{i=1}^{m} (h(x_i) - \overline{h}_m)^2$$

Para m suficientemente grande tenemos:

$$\frac{\overline{h_m} - E_f[h(x)]}{\sqrt{v_m}}$$

la cual tiene una distribución semejante a una normal con media 0 y varianza 1 y con base en esto último tenemos la posibilidad de construir pruebas de convergencia e intervalos de confianza para la aproximación de $E_f[h(x)]$, en general aplicar un conjunto de pruebas paramétricas a nuestro conjunto de datos generados a partir de las múltiples evaluaciones de $E_f[h(x)]$.

Ejemplo 3.1.1 *Calculemos la integral de* $h(x) = \sqrt{x}\text{sen}x$ *la cual no se puede integrar por medio de funciones elementales. A través del MMC, podemos obtener una aproximación del valor de:*

$$E_f[h(x)] = \int_0^{\pi} \sqrt{x}\text{sen}x dx$$

(programa B.3.1) se realizaron 100 iteraciones para tener una muestra aleatoria de 100 elementos con un millar de valores aleatorios, distribuidos uniformemente en el intervalo de integración $(0, \pi)$.

Para la variable aleatoria $h(x)$ obtenemos la siguiente gráfica correspondiente al histograma de $E_f[h(x)]$:

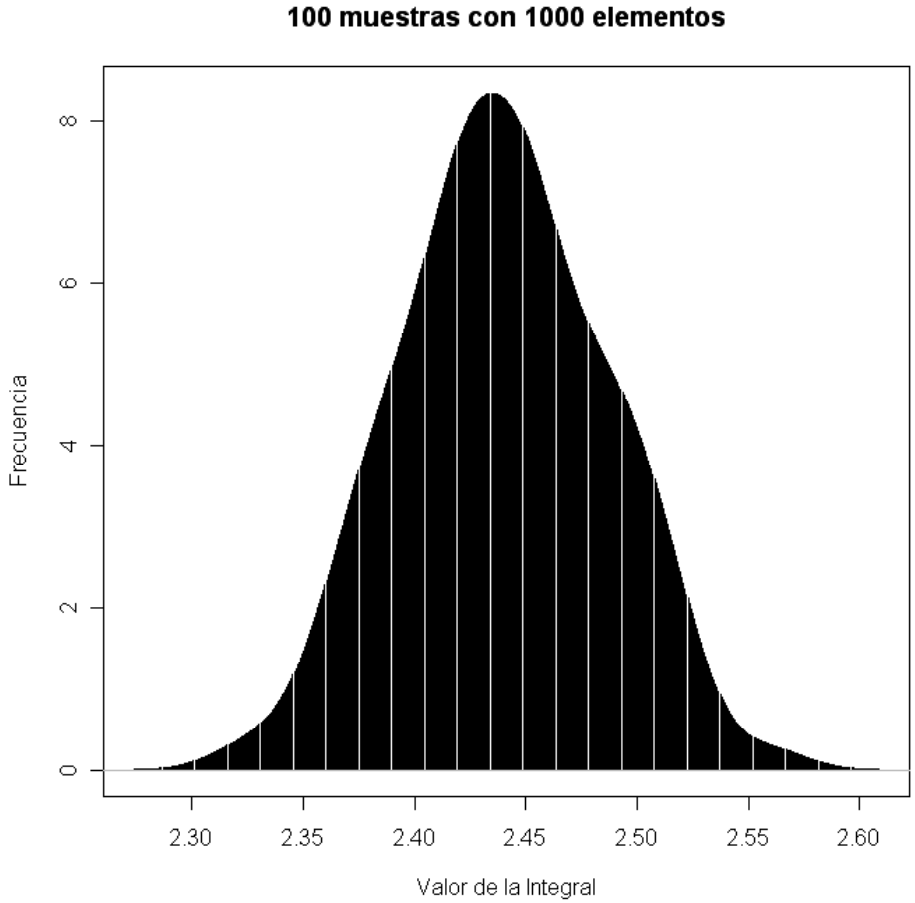


Figura 3.1

Con media $\overline{h}_{1000} = 2.4392$ y varianza= 0.002058. En este caso, nuestra aproximación del valor de la integral, es a su vez el estimador $\overline{X}$ de la media de nuestro conjunto de datos, para la media poblacional μ podemos, por ejemplo, construir el intervalo de confianza con un coeficiente $\alpha = 0.1$, por lo que $z = 1.64$, así, nuestro intervalo de confianza está dado por:

$$\begin{gathered} 2.4392 - (1.64)\sqrt{\tfrac{0.002058}{100}} < \mu < 2.4392 + (1.64)\sqrt{\tfrac{0.002058}{100}} \\ 2.4392 - 0.007 < \mu < 2.4392 + 0.007 \\ 2.\,4322 < \mu < 2.\,4462 \end{gathered}$$

Comparada con otras maneras de calcular o aproximar integrales como la exhaución, o la de Simpson, el valor obtenido presenta una diferencia porcentual de $\frac{0.004}{2.442} = 0.16\ \%$

En funciones de una sola variable, el MMC no es muy útil, pues otros métodos númericos, como la regla del punto medio son mas eficientes, debido principlamente a que con MMC la convergencia es mas lenta comparada con ellos, pero en funciones de mas dimensiones es, en algunos casos, la única manera de aproximar el valor de la integral.

A diferencia de los ejemplos comentados en el primer capítulo, La integración Monte Carlo clásica, no requiere supuestos con respecto a la función, pues implícitamente en el primer caso se pide continuidad cuando empleamos Hit or Miss, mientras que en este útimo, se puede usar en casos como el de un dominio con discontinuidades.

Con este mismo método, podemos aproximar el valor de $\int_\Omega f d\Omega$ donde $f : R^n \to R$ es acotada en Ω, usando la integración MMC clásica comentada al principio de este capítulo, simulando n variables aleatorias independientes.

3.1.2 Cálculo de integrales en múltiples dimensiones.

Este tema, es una generalización del ejemplo 1.1.1, donde el objetivo planteado es obtener una aproximación de el área encerrada por una curva simple, en general, podemos calcular el volumen de un hipersólido descrito a partir de una función determinada.

Así como el cálculo de integrales definidas, está motivado por el cálculo de áreas, donde podemos considerar la función identidad sobre una areá determinada, podemos aproximar el volumen de un hipérsólido dado por una transformación $f : \Omega \subset R^n \to R^m$,donde Ω es un subconjunto acotado de R^n.

Del mismo modo que en el caso de una dimensión necesitamos que $f(x)_\Omega < \infty$ para poder calcular la integral, y en vez de acotar f en el intervalo $[0, c]$ como en el caso de una variable, la acotamos por la bola de radio $M \geq \max(\|f(x, y, z)\| \ (x, y, z) \in \Omega)$y sin pérdida de generalidad, necesitamos que $\overline{0} \in \operatorname{Im} f(\Omega)$.

Ejemplo 3.1.2 *Sea Ω la esfera unitaria S^2, deseamos obtener una aproximación de $\int_\Omega f d\Omega : R^3 \to R^3$ dada por*

$$f(x, y, z) := (x^2, e^{xy}, \cos(zy))$$

La cual es acotada en Ω. Se propone $M = 4$ como cota para $\|f(x, y, z)\|_\Omega$, debemos generar tres variables pseudo-aleatorias de modo tal, que al ser vistas como vectores se distribuiyan uniformemente en Ω.

Una vez hecho esto, empleamos Hit or Miss al comparar con una cuarta variable pseudo-aleatoria ω, distribuida uniformemente en $[0, 4]$ para aceptar o rechazar cada uno de los elementos que resultan de evaluar la función en los puntos generados. El criterio en este caso para aceptar o rechazar un experimento como un éxito es el siguiente: si $\|\omega\| < \|f(x_0, y_0, z_0)\|$ es considerado un éxito, en caso contrario se considera un fallo.

Con este criterio podemos construir un estimador para la probabilidad p de que dado un punto $A \in B_4(0)$, $p = P(A \in \operatorname{Im}(f))$ dado por

$$p = \frac{n_e}{n}$$

Donde de nuevo n_e denota los éxitos en n experimentos Bernoulli.

En este sencillo ejemplo, parece natural llevar las variables x, y, z a su forma polar:

$$\begin{aligned} x &= r\cos\theta \text{sen}\varphi \\ y &= r\text{sen}\theta\text{sen}\varphi \\ z &= r\cos(\varphi) \end{aligned}$$

Debemos notar que aunque las variables aleatorias r, θ, φ, se distribuyan uniformemente en $[0,1]\,, [0, 2\pi]\,, [0, \pi]$ respectivamente, no implica que la distribución de las variables x, y, z también sea uniforme, ya que este cambio de variable en particular tiende a centralizar la distribución de las variables x, y, z (figuras 3.2 y 3.3).

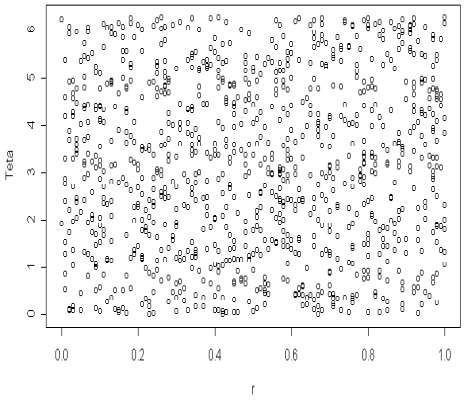


Figura 3.2
Dispersión de las variables aleatorias r, θ

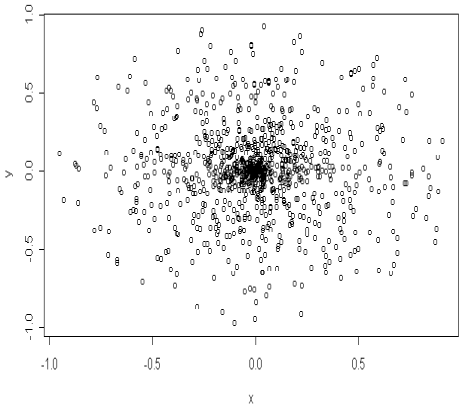


Figura 3.3
Dispersión de las variables x, y.

Como resultado de aplicar esta transformación tenemos que, al generar un conjunto de 1000 elementos para la terna (r, θ, φ) el volumen del sólido, que es $Imf(S^2)$ es aproximadamente

$$V(f_\Omega) \approx pV(B_M(0)) = 0.359 \times \frac{4\pi \times 4^3}{3} = 96.24\ u^3$$

Donde $p = 0.344$.

Al calcular el error cuadrático medio de la variable aleatoria $V(f_\Omega)$ como función de la muestra.

$$\varepsilon(V(f_\Omega)) = \frac{S}{\sqrt{N}} = 0.0749$$

Es importante recalcar que los puntos no están distribuidos uniformemente en Ω, pero el error cuadrático medio nos indica una desviación relativamente baja.

Otra idea que resulta intuitiva es generar una variable pseudo-aleatoria x que se distribuya uniformemente en $[0,1]$ y el rango de y y z dos en función de la primera del siguiente modo:

$$x \sim U_{[-1,1]}$$

$$y \sim U_{[-\sqrt{1-x^2},\sqrt{1-x^2}]}$$

$$z \sim U_{[-\sqrt{1-x^1-y^2},\sqrt{1-x^1-y^2}]}$$

En particular despues de correr el programa correspondiente con una muestra generada con mil elementos tenemos que:

$$p = \tfrac{368}{1000} = 0.368$$

El valor aproximado de la integral correspondiente:

$$V(f_\Omega) \approx 98.60\ u^3$$

Y el error cuadrático medio:

$$\varepsilon(V(f_\Omega)) = 0.0902$$

El objeto de los comentarios respecto a la distribución de las variables aleatorias, es principalmente, hacer notar, que el generar una o mas variables pseudo-aleatorias con una distribución específica (uniforme en este caso) no siempre resulta sencillo si se tienen en cuenta las modificaciones que presenta el transformar los intervalos con un cambio de variables o si hay dependencia de otros parámetros.

Una forma de evitar la construcción de la función inversa generalizada de las funciones de distribución de cada una de las variables es emplear Hit or Miss: generamos una terna (x,y,z), consideramos aceptada la terna si está en Ω, si no, es rechazada.

Dado que Ω es acotada, podemos distribuir cada variable uniformemente en un intervalo $[0,M]$, M una cota de Ω, es claro que la uniformidad se preserva.

Para este último caso tenemos que en una muestra con 1000 elementos:

$$V(f_\Omega) \approx 91.684\ u^3$$

Con el valor $p = 342$ y estimamos el valor cuadrático medio

$$\varepsilon(V(f_\Omega)) = 0.12684$$

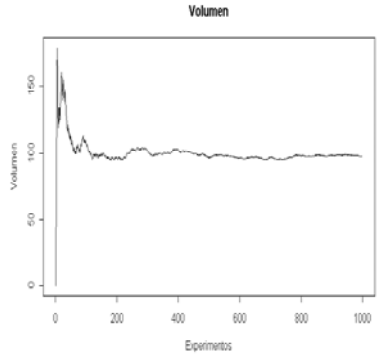


Figura 3.4
$V(f_\Omega)$ Transformación polar

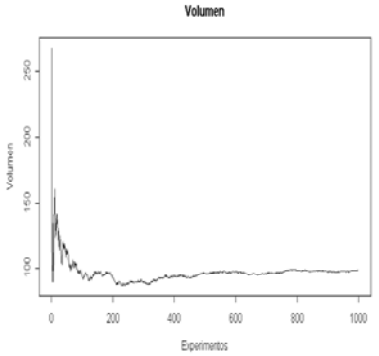


Figura 3.5
$V(f_\Omega)$ Intervalos dependientes

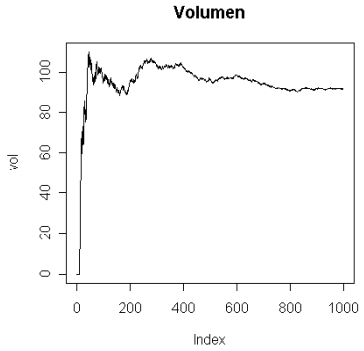


Figura 3.6
$V(f_\Omega)$ Distribución uniforme

El comportamiento de cada una de las gráficas en las figuras 3.4, 3.5 y 3.6, no es cualitativamente distinto al de los ejemplos comentados en el primer capítulo, pero la diferencia entre los resultados obtenidos en cada uno de los casos puede resultar notable si no ponemos atención en estos puntos, ya que si, por ejemplo el valor de M incrementa es natural que la desviación estándar incremente y con ello los resultados difieran aun mas entre si.

3.2 Optimización Monte Carlo

Una aplicación típica, es la búsqueda del valor mínimo (o máximo) de una función $f : R^n \to R$.

Para la solución de este problema, existen formas con enfoques deterministas en los cuales no se requiere una interpretación probabilística o estadística de la función o del resultado obtenido, aunque en algunos como es el caso del algoritmo de Metropolis se emplean nociones de probabilidad por la naturaleza del fenómeno que motivó su desarrollo.

3.2.1 Exploración estocástica

Una manera de buscar el valor mínimo de una función $f : D \subset R^n \to R$ es considerar un subconjunto $\Omega \subset D$ del dominio y generar vectores con números aleatorios, posteriormente evaluarlos en la función y compararlos entre si, una primera aproximación de $min(f(\overline{x})/\overline{x} \in \Omega)$ puede practicarse con distribución uniforme. Una manera de ajustar este modelo a una situación particular es escoger el subconjunto Ω de interés, aun que en general esto último puede ser complicado si Ω no es conexo.

Ejemplo 3.2.1 *Exploración estocástica (programa B.3.2) sea:*

$$f(x) = \frac{[xsen(y)]^2 + ysenx * \cos(xsenx)}{xy}$$

$\Omega = \{(x, y)/\ x \leq 5\cos(y)\}$ *(Figuras 3.7, 3.8, 3.9 y 3.10)*

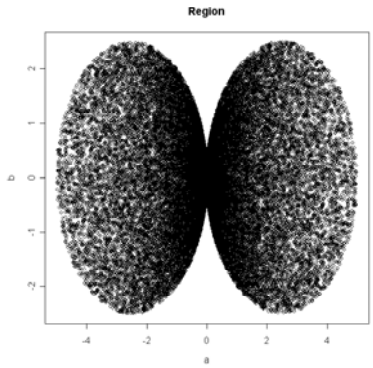


Figura 3.7
Plano xy

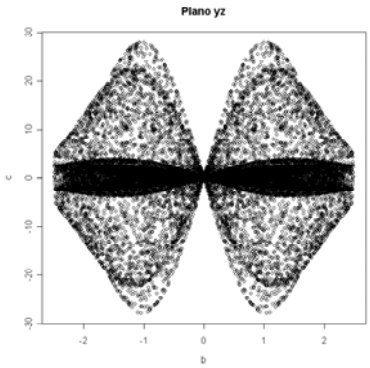


Figura 3.8
Plano yz

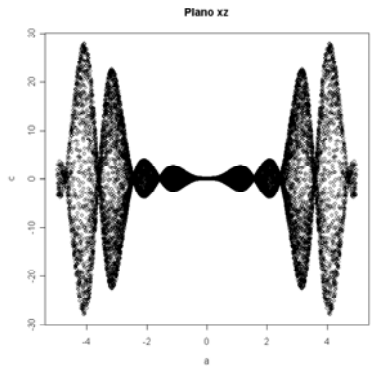


Figura 3.9
plano xz

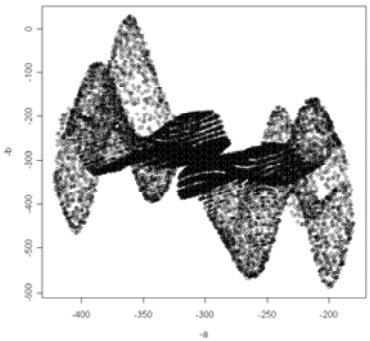


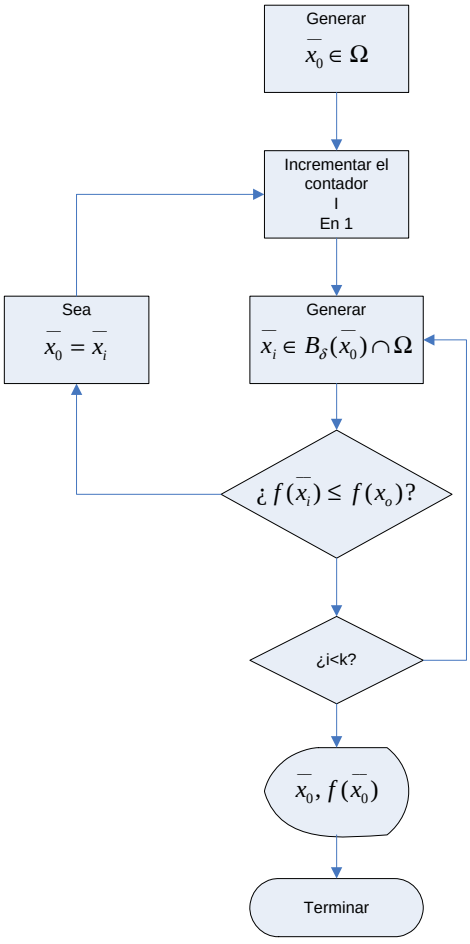
Figura 3.10
Perspectiva

En particular $f(x)$ no está definida en los ejes coordenados y esta eventualidad debemos prevenirla corrigiendo la selección de las variables aleatorias.

3.2.2 Métodos gradientes

Lo métodos gradientes escencialmente sirven para construir rutas sobre un espacio, a partir de una condición inicial y de manera aleatoria generar posibles direcciones en busca de los valores mínimos o máximos. Si deseamos optimizar una función $f(x_1, x_2, ..., x_n) : \Omega \subseteq R^n \rightarrow R$, lo que debemos de hacer, es escoger un punto $\overrightarrow{x_i} \in R^n$ como valor incial, tomar $\overrightarrow{x_{i+1}}$en una vecindad de $\overrightarrow{x_i}$, comparar las imagenes de ambos elementos en R, de modo que si despues de k intentos de escoger un nuevo elemento en la vecindad de $\overrightarrow{x_0}$, no encontramos un elemento cuya imagen sea menor que x_0 decimos que este último, es el valor mínimo de f.

Definimos el algoritmo de optimización como sigue:



Un problema con el que no podemos lidiar empleando este algoritmo es con haber escogido una mala condición inicial, supongamos que entre todas las posibles rutas $S = \{f^{\,i}(x_0)\}$ se encontrara, por ejemplo,entre x_0, y el mínimo global un mínimo local y dicha trayectoria no consiguiera rodear el mínimo local como es el caso en funciones de una sola variable.

Ejemplo 3.2.2 *Sea*

$$f(x, y) = -(x-1)^2 e^{-\frac{x^2+y^2}{2}}$$

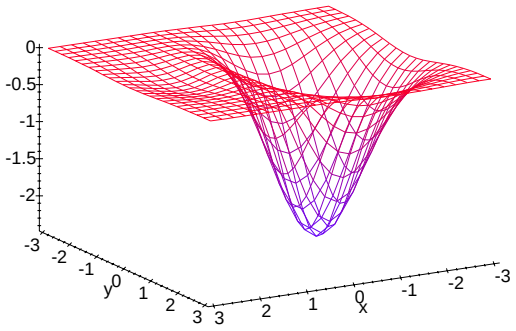


Figura 3.11
Función con mínimos locales y globales

Es claro que en funciones que tienen múltiples concavidades, como es el caso de la función del ejemplo anterior, este método puede no ser muy efectivo, pues dependerá de las condiciones iniciales y de la sucesión elegida de números pseudo-aleatorios.

Si bien hemos aproximado un elemento $\overrightarrow{x_i}$ en el dominio de la función f tal que $||f(\overrightarrow{x_i}) - f(\overrightarrow{x_{opt}})|| \leq \delta$ donde $f(\overrightarrow{x_{opt}}) \leq f(\overrightarrow{x}) \ \forall \ x \in V_\delta(\overrightarrow{x_i})$, no podemos asegurar que se trate del mínimo global de la función.

Una manera de salvar el problema de encontrarnos con un mínimo local en lugar del global, es emplear el algoritmo anterior probando con diferentes condiciones iniciales escogidas aleatoria o estratégicamente, lo que nos permite buscar en una parte mayor del dominio de la función, regularmente se escogen las nuevas condiciones inciales de manera aleatoria.

Ejemplo 3.2.3 *Sea* $f(x,y)$*:*$R^2 \rightarrow R$ *definida de la siguiente manera*

$f(x,y) = x^2 + y^2$

Con condiciones iniciales (x_0, y_0) aleatorias, $k = 10000$ y la vecindad de $\overrightarrow{x_i}$ dada por $(x \pm 1/4, y \pm 1/4)$

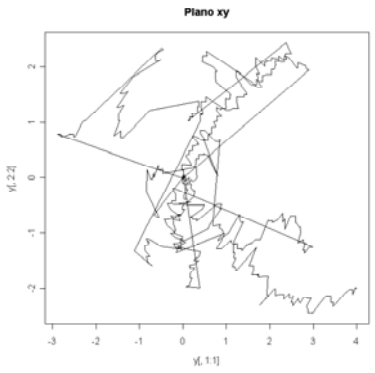


Figura 3.12
Plano x, y

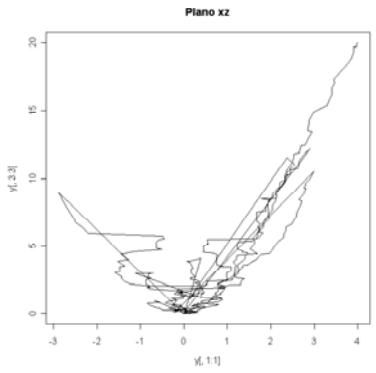


Figura 3.13
Plano x, z

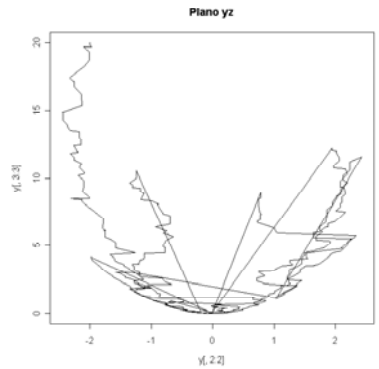


Figura 3.14
Plano y, z

En la figura 3.12 podemos ver las trayectorias resultantes tras haber utilizado el programa No B.3.3 en el plano x, y, en las figuras 3.13 y 3.14 observamos las trayectorias resultantes tras haber utilizado el mismo programa en los planos correspondientes.

Observamos trayectorias, restringidas por la condición del algoritmo que impide que el valor de $f(\overrightarrow{x_i}) > f(\overrightarrow{x_{i-1}})$, esto último abre otro campo de estudio que es conocido como caminatas aleatorias.

Este método gradiente, también implica mayores recursos y más tiempo de procesamiento, pues cada vez que se escogen condiciones iniciales, debemos de esperar que el proceso termine y a su vez compararlo con los resultados previos para quedarnos con el mejor de entre ellos.

(*Continuación del ejemplo* 3.2.2) *Sea*

$$f(x, y) = -(x-1)^2 e^{-\frac{x^2+y^2}{2}}$$

donde, empleando el mismo programa obtenemos:

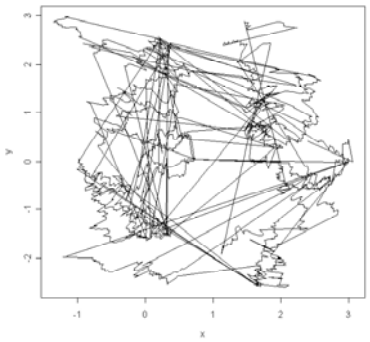


Figura 3.15
Plano x,y

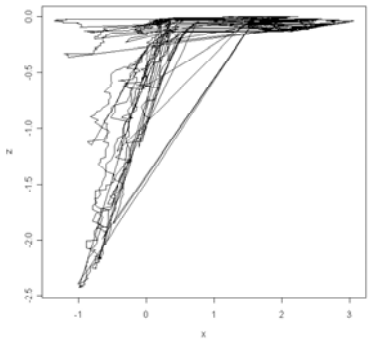


Figura 3.16
Plano x,z

en las gráficas anteriores no encontramos ninguna tendencia o convergencia de las caminatas aleatorias como en el ejemplo 3.2.3, por lo que debemos

de conservar el mejor resultado como el más cercano a nuestro valor óptimo, en esta situación particular $min\{f(x,y)\ (x,y) \in R\} = -2.4259$ con $(x,y) \approx (0,1)$.

3.2.3 Algoritmo de Metropolis (MMC mejorado)

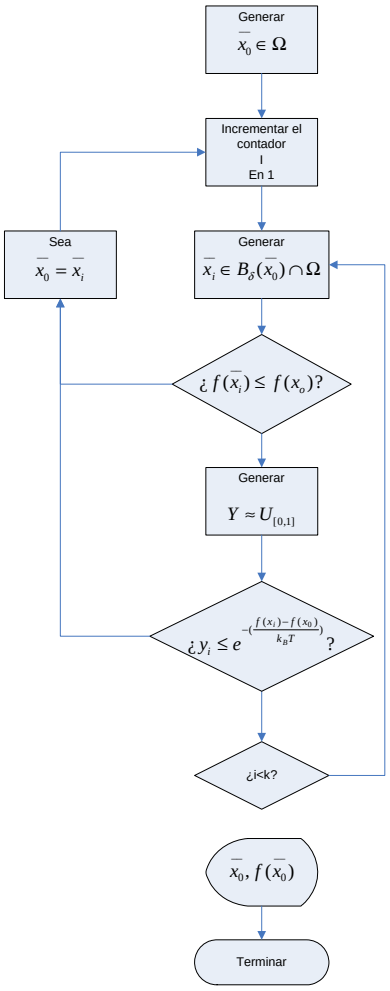
El algoritmo de Metropolis, es una mejora al método de optimización MC, es un algoritmo probabilístico para aproximar un valor óptimo global, a diferencia del MMC tradicional, el algoritmo de Metropolis, evita en la medida de lo posible encerrarnos en una vecindad de algún mínimo local, pues no se desecha en primera instancia una evaluación de $f(x_i)$ si esta resulta ser mayor a $f(x_{i-1})$ cuando estamos buscando el valor mínimo. El caso en que $f(x_i) > f(x_{i-1})$ es tratado probabilisticamente.

El algoritmo, originalmente, fué implementado para simular una colección de átomos en equilibrio a una cierta temperatura T sobre un espacio fase [11]. En cada paso para cada átomo se tiene un desplazamiento aleatorio, se calcula la diferencia de energía ΔE entre el estado inicial y el prospecto, si $\Delta E \leq 0$, la nueva configuración es tomada como la nueva configuración inicial, si $\Delta E > 0$, la probabilidad de aceptar la nueva configuración es

$$P(\Delta E) = e^{\frac{-\Delta E}{k_B T}}$$

Donde k_B es la constante de Boltzman y T es la temperatura absoluta en el sistema, con esta modificación en el algoritmo de optimización clásica, lo que buscamos es que generando una sucesión de NPA uniforme en [0,1] y comparando en cada paso con la probabilidad de aceptar la nueva configuración, eventualmente la configuración nueva es aceptada, aunque represente una solución en la que nos alejamos más del valor mínimo al menos en forma aparente.

En general, en el caso de una función cualquiera, podemos simplemente considerar la diferencia de sus imágenes como el parámetro de probabilidad de transición, es decir ΔE, ya que los parámetros T y k son constantes.



En el siguiente ejemplo, se emplea el Algoritmo de Metropolis (Programa no B.3.4) en una función de una sola variable porque es más sencillo observar los mínimos espurios, y veremos como es que el la variable aleatoria literalmente brinca el mínimo local y así logra continuar su recorrido.

Sea

$$f(x) = x^4 - 14x^2 + 8x - 1$$

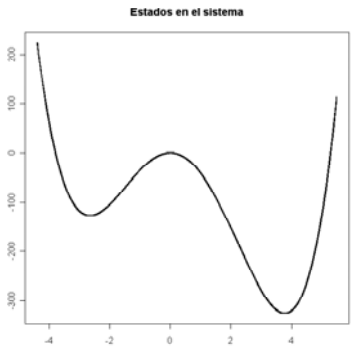
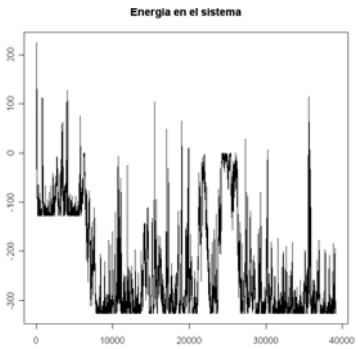


Figura 3.17
Estados del sistema



Figuera 3.18
Función potencial

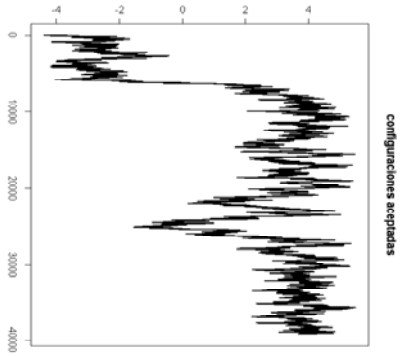


Figura 3.19
Configuraciones aceptadas

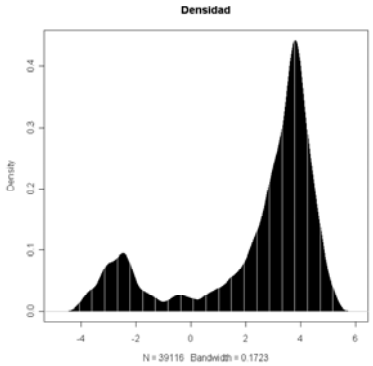


FIgura 3.20
Función de densidad x

Este método nos permite buscar el mínimo global a pesar de que en la trayectoria nos hayamos encontrado con uno local pero no podemos detener el flujo ya que, aun en el mínimo global de la función, la variable aleatoria puede saltar de la concavidad y seguir explorando, por lo que al final debemos quedarnos con el mejor resultado.

La gran ventaja de emplear este algoritmo sobre los dos comentados anteriormente, es que la aproximación no depende de las condiciones iniciales escogidas, aun que el desarrollo del mismo no puede preveer un límite para el número de iteraciones necesarias para encontrar la solución o soluciones mas próximas.

Es importante notar que en niguno de los casos anteriores se especifica que la función a optimizar sea necesarimente continua, por ende tampoco es necesario que sea diferenciable, en particular el artículo publicado por Metropolis, encuentra su motivación en un modelo discreto, la función potencial depende de la posición de un conjunto finito de partículas.

Ejemplo 3.2.4 $x^4 + (y^4 - y^3)\cos^2(xy) - e^{-(x^2+y^2)}$

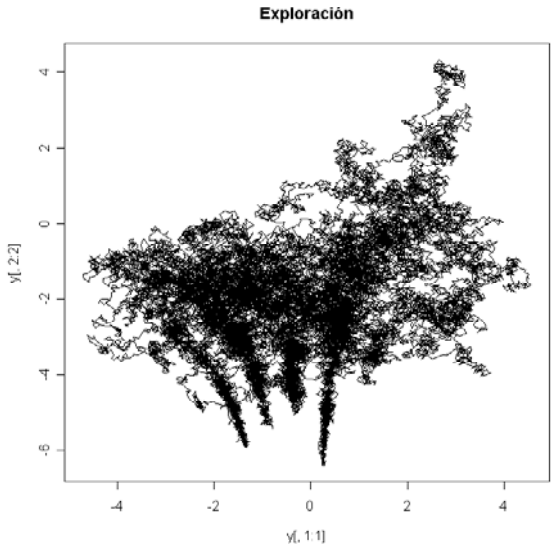


Figura 3.21: Exploración Metropolis

3.2.4 Simulated annealing (Simulación de enfriamiento lento)

La simulación de enfriamiento lento es un proceso que depende en su mayor parte de un parámetro al que se le suele llamar temperatura y que es gradualmente llevado a 0 (de ahí el nombre) buscando configuraciones con la menor energía interna posible, es basicamente, una aportación al algoritmo de Metropolis, en este último la probabilidad con la que se acepta un cambio de estado en el sistema depende enteramente de ΔE pues T y k_B son constantes. En una simulación de enfriamento el parámetro $T \to 0$, conforme al desarrollo del proceso, tenemos que

$$e^{-\frac{\Delta E}{k_B T}} \to 0$$

con lo que la probabilidad de aceptar una configuración o un estado de energía superior se reduce respecto al decremento en T con lo que la búsqueda del valor mínimo se vuelve finito.

El algoritmo de Metropolis se modifica unicamente en la actualización de T para cada iteración.

Ejemplo 3.2.5 *Sea* $f(x, y, z) = 8 - 2(z - y)x + 8y - 4yz + 4y^2$, *el parámetro* T *puede actualizarse como una función del tiempo no necesariamente lineal por ejemplo si el experimento en curso es el* $k - ésimo$*:*

$$T(k) = \frac{1}{2k}$$

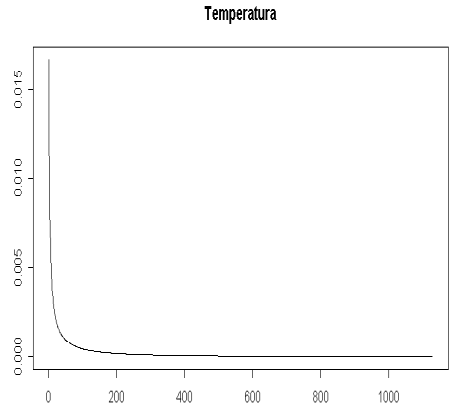


Figura 3.22
Temperatura

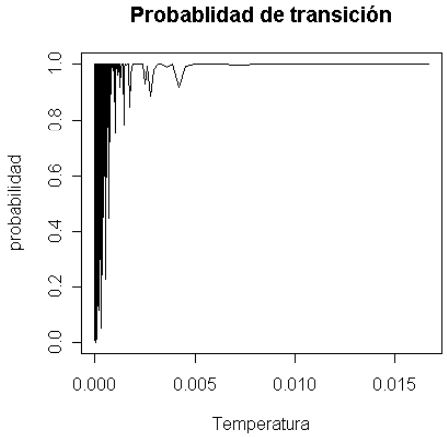


Figura 3.23
Probabilidad de transición

En las gráficas anteriores se aprecia claramente como es que la probabilidad de transición presenta un decremento en tanto que el parámetro T, tiende a 0 y para el valor mínimo w de la función $f(x, y, z)$ tenemos que el valor de $f(x, y, z)$ tiende a cero (figura 3.24):

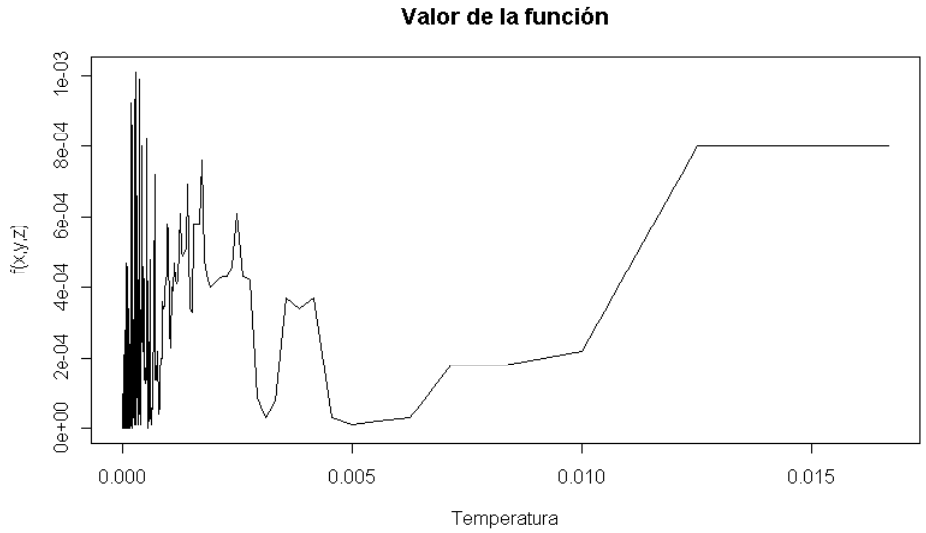


Figura 3.24

Un manera de mejorar cualquiera de estos algoritmos es crear híbridos entre ellos, de este modo podemos obtener lo mejor de cada uno.

3.3 Prueba de hipótesis estadística Monte Carlo

3.3.1 Remuestreo (Resampling)

El MMC puede ser empleado como una prueba de hipótesis, básicamente debemos de considerar todos los casos posibles de reordenamiento de un conjunto de datos, posteriormente debemos de hacer el análisis del estadístico que deseamos estudiar. esto requiere mucho tiempo de cómputo sobre todo para muestras grandes, por lo que es conveniente reducir nuestro estudio a un conjunto de casos escogidos aleatoriamente.Una de las desventajas que tiene el MMC como prueba de hipótesis estadística es que sus resultados no pueden necesariamente extrapolarse a una población, por otro lado, las grandes ventajas son que no se necesita el concepto de población y lo que resulta mas importante, es que no se requiere proponer una función de distribución de probabilidad de los elementos de la muestra para realizar dicha prueba.

Ejemplo 3.3.1 *Consideremos el siguiente grupo de datos.*

R	A
12.17	12.57
12.32	12.70
12.32	12.71
11.27	12.49
12.15	12.73
12.26	12.71
12.01	12.70
12.37	12.43
12.24	12.51

Que representan las medidas de peso de un producto plástico, para verificar que se encuentren dentro de las especificaciones determinadas por el control de la calidad que en este caso tiene como valor óptimo

$12.5 \pm 3\ gr$. La hipótesis nula (H_0) es que ambos grupos están dentro de especificaciones debido a que el promedio de la muestra de rechazados (R) depende de la muestra, la hipótesis alternativa es que existe una diferencia significativa que no depende de la muestra.

La prueba de hipótesis estádistica se basa en el reordenamiento (resampling) de los datos obtenidos, o mas claramente de un reagrupamiento.

De el conjunto presentado anteriormemnte podemos obtener $C_9^{18} = \frac{18!}{9!9!}$: 48620 diferentes muestras de tamaño 9, lo que deseamos ver es que bajo el

agrupamiento dado de rechazados y aprobados existe una fuerte tendencia a suponer que los promedios no serán semejantes para cualquier reagrupamiento de los datos en general, cuyos promedios respectivos son $\overline{R}$=12.23 y $\overline{A}$=12.62 y la diferencia de ambos promedios es $\overline{R}$-$\overline{A}$=0.39.

Reordenamos aleatoriamente los elementos de ambos grupos, de modo que conformamos dos nuevos conjuntos, uno de rechazados y otro de aprobados de manera aleatoria, y se compara con respecto a la distribución de las diferencias de los reordenamientos posibles.

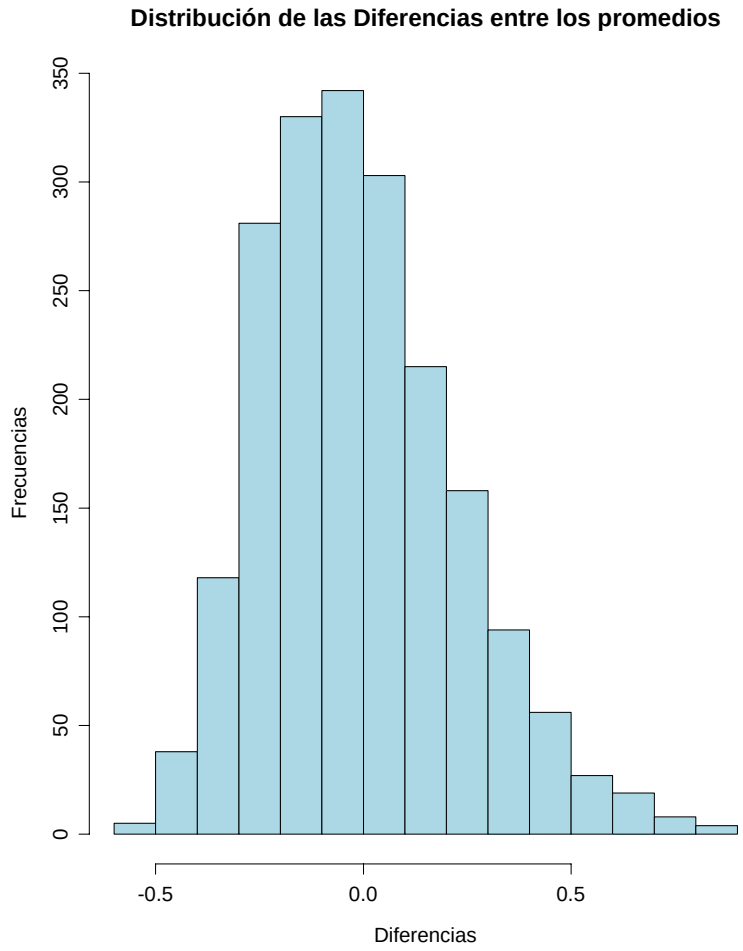


Figura 3.25

Despues de 2000 reagrupamientos de la muestra dada (figura 3.25), se observa que, la diferencia entre los promedios de ambas observaciones es poco usual, por lo que decimos que la hipótesis alternativa tiene mayor probabilidad de ser aceptada, en este ejemplo solo 96 de 2000 reordenamientos son mayores que 0.39 por lo que decimos que se apoya la hipótesis alternativa con 1-$\frac{96}{2000}$ = 95.2% de nivel de significancia.

3.3.2 Bootstrap

El método conocido como Bootstrap, fué introducido inicialmente por Bradley Efron en el año de 1979, consiste en generar muestras del mismo tamaño que la muestra original pero con reemplazo. Supongamos que tenemos N valores de una variable aleatoria, entonces se escogen aleatoriamente N de entre ellos sin importar las repeticiones, con lo que obtenemos un espacio muestral de N^N muestras posibles, se hace una selección aleatoria de entre estas y para cada una de ellas se calculan los estadísticos de interés, con lo que podemos obtener un estimador de la varianza de estos estadísticos.

3.3.3 Jacknife

Este método, consiste ene la construcción de muestras de menor tamaño que la que poseemos. Supongamos que tenemos una muestra de tamaño N. Primero se calcula el estadístico de interés, después, aleatoriamente, eliminamos uno de los valores de la variable aleatoria o uno de los sujetos según sea el caso, y con ello obtenemos una muestra con $N-1$ elementos, se vuelva a calcular el estadístico de interés, repetimos el proceso $N-1$ veces de modo que con este método obtenemos $N!$ diferentes muestreos posibles y 2^N pseudo valores del estadístico, por lo que es conveniente considerar una muestra aleatoria de los mismos para obtener un estimador del estadístico en cuestión.

Al valor medio de los pseudo valores del estadístico, se le conoce como estimador Jacknife, y para este se construyen intervalos de confianza para el mismo.

3.4 Simulación de sistemas

En el área industrial, el MMC puede ser empleado como una herramienta muy útil creando escenarios posibles de un sistema cualquiera o bien de una parte de el.

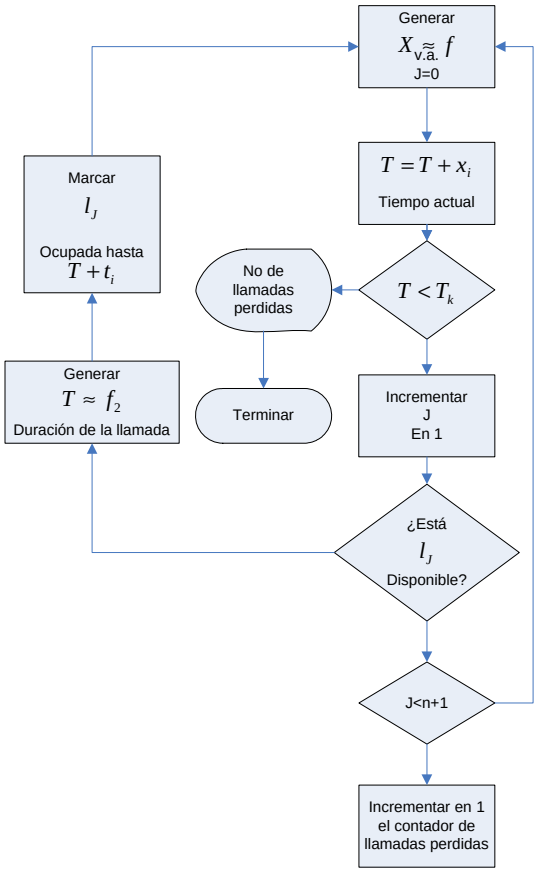
Existen diversas razones por las que una simulación es valiosa, entre otras cosas indica posibles resultados ante la toma de decisiones, se analizan casos positivos y adversos, esto último es particularmente interesante pues del mismo modo en que las probabilidades de que un sistema se encuentre en un estado óptimo son bajas, igualmente de que se encuentre en el peor estado posible.

Uno de los elementos más importantes de una simulación de un sistema, es el modelo computacional, conceptualmente, a veces esta técnica puede verse como un simple ejercicio de programación, pero puede convertirse en una manera de reducir costo y tiempo de operación.

Una práctica que resulta común, es introducir un primer modelo que represente unicamente una parte del sistema, posteriormente verificar la semejanza de los resultado obtenidos con la realidad. Veamos un ejemplo.

3.4.1 Simulador de servicios

Supongamos que en el departamento de ventas de una empresa, se cuenta con n líneas telefónicas, y sabemos que el tiempo máximo de comunicación entre el operador y el cliente es t minutos, así mismo,el tiempo promedio que tarda en entrar una llamada es de m minutos. Podemos diseñar un simulador de conmutador de modo que en un rango de h horas reproduzca el comportamiento de las operaciones diarias, un algoritmo con los supuestos anteriores es:



Donde X es la varible de control que funje como el tiempo entre cada llamada, y T la duración de las mismas.

Con lo cual podemos resolver varias preguntas, entre otras ¿Cómo está repartida la carga de trabajo entre las líneas telefónicas? ¿Cuántas llamadas se pierden al día? y con base en estas, algunas mas complejas ¿es conveniente contratar otra linea telefónica con el fin de evitar pérdidas? ¿como se debe

de planificar la rotación de personal para hacer mas equitativa la carga de trabajo?

Este algoritmo al igual que cualquier modelo es suceptible de ser enriquecido, en el caso del programa B.3.7 en el apéndice B se supone distribución uniforme en los tiempos de operación, pero podemos empiricamente fundamentar nuestro modelo modificándolo de acuerdo a datos históricos levantados previamente. Podemos definir en general funciones de distribución para cada variable de control y de este modo, ajustar aun mas la dinámica del mismo al sistema real o bien, agregar un tiempo de espera para cada llamada que no puede ser atendida en el momento de entrada de manera que no sea rechazada automaticamente por el conmutador si no hay líneas disponibles. Las líneas telefónicas pueden a su vez ser sustituidas por otro tipo de servicio (horas máquina, transportes, mano de obra,etc) en general puede aplicarse como un análisis de tiempos de operación con el fin de optimizar las actividades de un sistema. En particular, este ejemplo puede ser empleado para el cálculo de los costos de operación, para hacer estimaciones del costo de hora máquina por ejemplo o de mano de obra, etc.

Capítulo 4

Conclusiones

Los métodos númericos, son una alternativa práctica en la solución de problemas, en particular, con las simulaciones creadas con base en variables aleatorias, obtenemos aproximaciones que pueden ser mas realistas que al considerar modelos determinísticos, pues con los métodos Monte Carlo podemos emular fluctuaciones que regularmente son difíciles de describir, mientras que asumiendo una cierta función de densidad de probabilidad para estas fluctuaciones el sistema puede ser imitado aunque presente un comportamiento ruidoso.

Otra de las razones por las que una simulación puede ser plausible, es que el costo que tiene su realización, puede ser más bajo en muchos casos, sobre todo cuando es necesario levantar una gran cantidad de datos, o bien cuando estos son inaccesibles o implican destrucción de los objetos de estudio o en un caso extremo, cuando el experimento implica daño a un ser vivo como es el caso de la investigación de los efectos de la radiación en los aparatos empleados para la medicina.

Gracias a los avances tecnológicos en el área de cómputo durante el último medio siglo, es posible hacer simulaciones dinámicas e inclusive existe paquetería orientada a estos fines aunque es importante destacar, que una simulación de un problema particular depende en gran medida del conocimiento que se tenga acerca de el mismo pues, como comenté en el cápitulo uno, gracias a ello podemos obtener módelos y algoritmos mas eficientes.

Los Métodos Monte Carlo, no pueden sustituir a otros métodos de investigación, pero pueden ser útiles cuando el trabajo se vuelve muy complejo por métodos analíticos o como en el caso de la prueba de hipótesis estadística, cuando las operaciones a realizar se vuelven tediosas.

Una manera de mejorar el desempeño de estos métodos es empleando métodos de procesamiento paralelo, así mismo el diseño de algoritmos adecuados a ellos, además existen variantes, que incluyen por ejemplo la teoría de cadenas de Markov (Métodos Markov-Monte Carlo).

Por último, considero que el auténtico valor de los Métodos Monte Carlo radica en su versatilidad, ya que inicialmente fueron empleados para resolver

problemas en el área de la física estadística, que incluyen difusión de neutrones, optimización de función potencial, permeabilidad en medios porosos, etc., pero el día de hoy, se encuentran publicaciones con orientaciones diversas, además de las que comenté en este trabajo como por ejemplo modelos de reacciones químicas, problemas de física cuántica, física médica, biología estadística, modelos macroeconómicos, aproximación de soluciones de ecuaciones diferenciales, los cuales por atender a campos de estudio particulares en los que habría que detallar tanto la motivación como parte de la teoría de fondo empleada para resolverlos y en muchos casos por la complejidad de los mismos, no fueron tratados en esta tesis, sin embargo reitero, las aplicaciones de los Métodos Monte Carlo seguiran siendo tan amplias como las dificultades que representa obtener una solución numérica precisa en un problema particular y como lo permita la imaginación de cada individuo que tenga contacto con ellos.

Apéndice A

Apendice

Lema A.0.1 *Ley de los grandes números: si $x_{1,}x_2, ...,$ es una secuencia infinita de variables aleatorias que son independientes e idénticamente distribuidas con esperanza $E(x_i) = \mu < \infty$ entonces $P(\lim_{n\to\infty} \overline{X}_n - \mu < \epsilon) = 1$*

Definición A.0.2 *Sea $f(t) : [a, b] \to R$ continua y X una variable aleatoria tal que $M = max(x_1, x_2, ..., x_n)$. Definimos la probabilidad p de que X tome un valor menor a $f(t)$ en $[a, b]$ como*

$$P(X = x | x < f(t)) = \frac{\int_a^b f(t)dt}{M(b-a)}.$$

En el ejemplo de Buffon tenemos la función seno y la variable aleatoria no podrá exceder el valor L tenemos que en la gráfica de Lsinx, la probabilidad está dada por el área bajo la curva de la función.

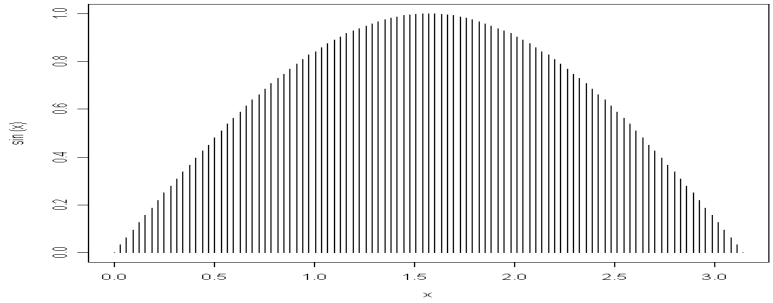


Figura A.1 area bajo la curva $y = senx$

Y la probabilidad p de que un punto (x, y) esté por debajo de la curva $senx$ es:

$$P(y \leq senx) = \frac{\int_0^\pi senxdx}{\pi}$$

Definición A.0.3 *Una función de distribución de probabilidad (FDP) para x variable aleatoria continua es una función que satisface:*

$$\begin{cases} a)\ f(x) \geq 0\ \forall x \in R \\ b)\ p(a \leq x \leq b) = \int_a^b f(x)dx \\ c)\ \int_{-\infty}^{\infty} f(x) = 1 \end{cases}$$

para x variable aleatoria discreta

$$\begin{cases} a)\ f(x_i) \geq 0\ \forall i \in 1, 2, ..., N \\ b)\ \sum_{i=1}^{N} f(x_i) = 1 \end{cases}$$

Definición A.0.4 *Sean $X_1 X_{2....} X_n$ variables aleatorias, un vector aleatorio de dimensión n, es una n-ada de observaciones de n variableas aleatorias, lo denotamos por:*

$$\vec{v_a} = (x_{1_i}, x_{2_i}, ..., x_{n_{i,}})$$

Donde x_{k_i} es una observación de la variable X_k

Definición A.0.5 *X se disribuye uniformemente en $[a, b]$ si*

$$P(X = x_i) = P(X = x_j)$$

para todos $x_i, x_j \in [a, b]$ $i, j \in I$.

A continuación se enlistan los programas y el código fuente empleados en el lenguaje PASCAL para la generación de números aleatorios expuestos en los capítulos anteriores, posteriormente con la salida en texto plano, las gráficas y los análisis estadísticos fueron generaron en R.

A.1 Capítulo 1

A.1.1 Área del círculo

```
{Programa para Calcular el area de un circulo}
Uses Crt;
Const
r=1.5; {radio del circulo}
h=3.4; {Coordenada x del centro}
m=2.0; {Coordenada y del centro}
L=6; {Lado del cuadrado}
```

```
Var
x,y,x1,y1,Ch:real;
i,j,k,npi:longint;
F,G,T:text;
Begin
clrscr; {CONDICIONES INICIALES}
Assign(F, 'kiss1.txt');{KISS como GNPA}
Reset(F);
i:=0;
j:=100;
k:=1;
x:=2;
y:=-1;
x1:=2;
y1:=-1;
Assign(G,'area.txt');
Rewrite(G);
repeat {circulo}
x:=h+r-(i/200);
y:=sqr(r)-sqr(x-h);
Append(G);
Writeln(G,x:2:5,' ',sqrt(y)+m:2:5,' ',L*L*(npi/k):1:5,' ',npi,' ',k);
Writeln(G,x:2:5,' ',(m-sqrt(y)):2:5,' ',L*L*(npi/k):1:5,' ',npi,' ',k);
Close(G);
i:=i+1;
until i=600;
i:=1;
Repeat
Read(F, Ch);
Clrscr;
Writeln('Calculo: ',i+1);
{Generando los vectores aleatorios con muestreo sistematico sobre KISS}
Repeat
j:=j-10
until j<10;
case j of
0: Begin
{Distribuir x y y por todo el cuadrado}
x:=L*Ch;
End;
9:Begin
y:=L*Ch;
if ((x-h)*(x-h)+(y-m)*(y-m))<(r*r) then
Begin
npi:=npi+1;
```

```
End;
k:=k+1;
Append(G);
Writeln(G,x:2:5,' ',y:2:5,' ',L*L*(npi/k):1:5,' ',npi,' ',k);
close(G);
End;
End;
i:=i+1;
j:=i;
until i=100000;
Writeln('Exitos=',npi,' fracasos=',k-npi);
Writeln('Valor probabilistico del area=',L*L*(npi/k):1:3);
Writeln('Valor de real=',PI*r*r:1:3);
Writeln('Diferencia=',L*L*(npi/k)-PI*r*r:1:3);
Readln;
```

A.1.2 La aguja de Buffon

```
{Programa para Calcular la pi con el metodo de buffon usando KISS}
Uses Crt;
Var
x,y,x1,y1,Ch:real;
i,j,k,npi:longint;
F,G:text;
Begin
clrscr; {SELECCIONAR CONDICIONES INICIALES}
Assign(F, 'kiss1.txt');
Reset(F);
i:=100;
j:=100;
k:=1;
x:=2;
y:=-1;
x1:=2;
y1:=-1;
Assign(G,'buffon.txt');
Rewrite(G);
Repeat
Read(F, Ch); {Generando los valores aleatorios sobre KISS}
Repeat
j:=j-10
until j<10;
case j of
4: Begin
x:=Ch;{Distancia del extremo de la aguja a la linea}
```

```
End;
7:Begin
y:=Ch*PI;
k:=k+1;
if sin(y)>x then {y es alpha en el programa, es decir el valor del ngul
Begin
npi:=npi+1;
Append(G);
Writeln(G,x:2:5,' ',y:2:5,' ',2/(npi/k):1:5,' ',npi,' ',k);
close(G);
End;
End;
End;
i:=i+1;
j:=i;
until i=100000;
Writeln('Exitos=',npi,' fracasos=',k-npi);
Writeln('Valor probablistico de PI=',2/(npi/k):1:3);
Readln;
End.
```

A.1.3 Integral Definida Hit or Miss

```
{Programa para Calcular la integal definida con hit or miss}
Uses Crt;
Var
x,y,x1,y1,Ch,h,l:real;
i,j,k,npi:longint;
F,G:text;
Begin
clrscr; {SELECCIONAR CONDICIONES INICIALES}
Assign(F, 'kiss1.txt');{KISS com GNPA}
Reset(F);
i:=100;
j:=100;
k:=1;
h:=pi/2;
x:=2;
y:=-1;
x1:=2;
y1:=-1;
Assign(G,'int01.txt');
Rewrite(G);
Repeat
Read(F, Ch); {Generando los valores aleatorios sobre KISS}
```

```
Repeat
j:=j-10
until j<10;
case j of
0: Begin
y:=h*Ch; {Genera una variable aleatoria distribuida uniformemente en el rectangulo}
End;
7:Begin
x:=Ch*PI; {Evaluar si el vector x,y) esta bajo la grafica de g(x)}
if sqrt(x)*sin(x)>y then
Begin
k:=k+1;
npi:=npi+1;
Writeln('Vector aleatorio(',x:2:2,',',sqrt(y)*sin(y):2:2,') exito');
Append(G);
Writeln(G,x:2:5,' ',y:2:5,' ',(npi/k)*pi*h:1:5,' ',npi,' ',k);
close(G);
End;
k:=k+1;
End;
End;
i:=i+1;
j:=i;
until i=80000;
Writeln('Exitos=',npi,' fracasos=',k-npi);
Writeln('Valor probablistico de la integral=',(npi/k)*pi*h:1:4);
Readln;
End.
```

A.1.4 Cumpleaños

```
{Programa para calcular la probabilidad de que en un grupo de N
personas al menos dos tengan la misma fecha de Cumplea os}
Uses Crt;
Const
N=30;
var
F,G:text;
persona,bdpers,ch,ne,i,k:integer;
Begin
Clrscr;
Assign(G,'cumplea.txt');
Rewrite(G);
Close(G);
For k:=1 to 1000 do
```

```
Begin
Randomize;
Assign(F,'birthd.txt');
Rewrite(F);
Close(F);
persona:=0;
Repeat
Persona:=persona+1;
bdpers:=random(364);
Reset(F);
i:=0;
Repeat
i:=i+1;
read(F,ch);
If bdpers=ch then
Begin
ne:=ne+1;
i:=persona-1;
Persona:=N;
End
Else
Begin
End;
Until i=persona-1;
Append(F);
Writeln(F,bdpers);
Close(F);
Until persona=N;
Append(G);
Writeln(G,ne,' ',ne/k:2:4);
Close(G);
writeln(k,' ',ne/k:2:3);
End;
Writeln('Probablidad:=',(ne/k):2:3);
Readln;
End.
```

A.1.5 Aproximación de raíces

```
{Raiz n-esima}
Uses Crt;
Var
F:Text;
n,i,j:longint;
r,s,a,ci,cs:real;
```

```
Begin
Clrscr;
Assign(F,'raiz.txt');
Rewrite(F);
randomize;
n:=10; {Orden de la raiz}
a:=1879; {valor del que deseamos calcular la raiz}
cs:=a/2; {Cota superior asumiendo que n>1}
ci:=1;{Cota inferior minima bajo la misma suposicion}
Repeat
r:=(1-random)*(cs-ci)+ci;
s:=r;
For j:=1 to n-1 do
Begin
r:=r*s;
End;
If r<a then
BEgin
ci:=s;
End
Else
Begin
cs:=s;
End;
Append(F);
Writeln(F,s:2:4,' ',ci:2:3,' ',cs:2:3,' ',r:2:3,' ',i);
Close(F);
i:=1+i;
Writeln(s:2:5);
Until sqr(a-r)<0.000001;
Writeln('Listo raiz de orden ',n,'= ',s:2:5);
Readln;
End.
```

A.2 Capítulo 2

A.2.1 Centros cuadrados

```
{Generador de numeros aleatorios de Von Neumann}
uses crt;
Var
i,j,k:longint;
a,b:real;
F:text;
```

```
Begin
randomize;
clrscr;
Assign(F, 'jvn.TXT');
Rewrite(F);
For j:=1 to 10 do
Begin
a:=random(10000);
b:=a;
i:=0;
Repeat
a:=sqr(a);
If a>999999 then
repeat
a:=a-1000000;
until a<1000000;
a:=int(a/100)/10000;
Append(F);
Writeln(F, a:2:6,' ',i);
Close(F);
i:=i+1;
k:=i;
a:=a*10000;
if a=0 then i:=10000;
if a=0.61 then i:=10000;
Until i=10000;
Writeln('a:=' ,b:4:1,' genero:',k);
End;
readln;
End.
```

A.2.2 Función logística

```
{Ilustra el comportamiento caotico de la funcion logistica cuando alpha
Uses crt;
const
alpha=4;
Var
x,y:real;
i:longInt;
F:text;
Begin
Clrscr;
Assign(F,'logistica.txt');
Rewrite(F);
```

```
Close(F);
x:=0.0001;
Repeat
x:=alpha*x*(1-x);
 Append(F);
 Writeln(F,x:2:5);
 Close(F);
writeln(x:2:5);
y:=x;
i:=i+1;
until i=100000;
readln;
End.
```

A.2.3 Transformación de la función logística

```
{Ilustra el comportamiento caotico de la funcion logistica compuesta
con la funci\UNICODE{0xf3}n de densidad te\UNICODE{0xf3}rica}
Uses crt;
Var
alpha,x,y:real;
i,j,k:longInt;
F:text;
Begin
j:=101;
i:=201;
Assign(F,'log02.txt');
Rewrite(F);
Close(F);
x:=0.86;
alpha:=4;
Repeat
x:=alpha*x*(1-x);
Repeat
j:=j-100;
if j=2 then
Begin
k:=k+1;
y:=(2/pi)*Arctan(x/(sqrt(1-sqr (x))));
If y>0.5 then
Begin
Append(F);
Writeln(F,y/0.5-1:2:5);
Close(F);
writeln(y:2:5,' ',k);
```

```
End;
End;
Until j<10;
i:=i+1;
j:=i;
until i=400000;
readln;
End.
```

A.2.4 Método de congruencias

```
{KISS ax+b, genera una recta compactada en el[0,1]x[0,1]}
uses crt;
Var
a,b,D:real;
i:longint;
F:text;
Begin
clrscr;
Assign(F, 'KISS1.TXT');
Rewrite(F);
a:=950.01;
b:=0.99;
D:=0.02;
Repeat
 If a*D+b<1 then
 Begin
 D:=a*D+b;
 End
 Else
 Begin
 D:=a*D+b;
 Repeat
 D:=D-1;
 Until D<1;
 End;
 Append(F);
 Writeln(F, D:2:6);
 Close(F);
i:=i+1;
Until i=1000000;
writeln('Listo');
readln;
End.
```

A.2.5 El juego del caos

```
Uses Crt;
Var
x,y,x1,y1,Ch,z:real;
i,j:longint;
F,G:text;
Begin
clrscr;
Assign(F, 'log02.txt');
Reset(F);
i:=100;
j:=101;
{SELECCIONAR CONDICIONES INICIALES}
{x:=0;
y:=0;}
x1:=0;
y1:=0;
Assign(G,'Caos03.txt');
Rewrite(g);
Repeat
Read(F, Ch);
{Generando los vectores aleatorios con muestreo sistemtico sobre KISS}
 Repeat
 j:=j-2
 until j<2;
 if j=1 then
 Begin
 If Ch>1/4 then
 Begin
 x1:=0;
 y1:=0;
 If Ch>3/4 then
 Begin
 x1:=1;
 y1:=1;
 End
 Else
 Begin
 If Ch>1/2 then
 x1:=1;
 y1:=0;
 End;
 End
 Else
```

```
 Begin
 x1:=0;
 y1:=1;
 End;
x:=(x1+x)/2;
y:=(y1+y)/2;
Writeln('(',x:2:5,',',y:2:5,')');
Append(G);
Writeln(G,x:2:5,' ',y:2:5,z);
Close(G);
 End;
i:=i+1;
j:=i;
until i=10000;
Readln;
End.
```

A.2.6 Prueba de bondad de ajuste

```
{prueba de bondad de ajuste ji-cuadrada
para una distribucion uniforme}
Uses Crt;
Var
F,G:text;
N,k,Np,i,j,l:Longint;
X,Y,ch:Real;
Begin
N:=400000;
k:=100;
Clrscr;
Assign(F,'kiss1.txt');
Assign(G,'pbachi.txt');
Rewrite(G);
For j:=1 to k do
Begin
Reset(F);
l:=0;
 For i:=1 to N do
 Begin
 Read(F,ch);
 If ch<=j/k then
 Begin
 If ch>(j-1)/k then
 Begin
 l:=l+1;
```

```
 End;
 End;
 End;
Y:=Y+(k/N)*sqr(l-(N/k));
Writeln('subintervalo:',j,': ',l,' ',l-(N/k):2:5);
Append(G);
Writeln(G,j,' ',l,' ',Y);
Close(G);
End;
Writeln(y:2:5);
Readln;
End.
```

A.2.7 FDP Discreta

```
{genera FDP discreta}
Uses Crt;
Var
F,G:Text;
j:Longint;
ch:Real;
i,x:shortint;
Begin
Clrscr;
Assign(F,'kiss1.txt');
Reset(F);
Assign(G,'FDPD.txt');
Rewrite(G);
For j:=1 to 10000 do
Begin
i:=1;
 Repeat
 i:=i+1;
 Read(F, ch);
 Until i=11;
 if ch<1/3 then
 begin
 x:=1;
 Append(G);
 Writeln(G,x);
 Close(G);
 End;
 If ch>1/3 then
 begin
 If ch<7/12 then
```

```
 Begin
 x:=2;
 Append(G);
 Writeln(G,x);
 Close(G);
 End
 Else
 Begin
 If ch>7/12 then
 Begin
 If ch<5/6 then
 Begin
 x:=3;
 Append(G);
 Writeln(G,x);
 Close(G);
 End
 Else
 Begin
 x:=4;
 Append(G);
 Writeln(G,x);
 Close(G);
 End;
 End;
 End;
 End;
Writeln(x,' ',ch:2:5);
End;
Readln;
End.
```

A.2.8 FDP ji-cuadrada

```
{Programa para obtener una FDP ji cuadrada}
Uses Crt;
Var
x,chi,ch:real;
j,v:integer;
i:Longint;
F,G:text;
Begin
clrscr; {SELECCIONAR CONDICIONES INICIALES}
x:=0;
Assign(F, 'kiss1.txt');
```

```
Reset(F);
Assign(G,'chi01.txt');
Rewrite(g);
Repeat
Read(F, Ch); {Generando los vectores aleatorios con muestreo sistemtico sobre KISS}
 Repeat
 j:=j-10
 until j<10;
 if j=1 then
 Begin
 x:=x+ln(Ch);
 v:=v+1;
 IF v=59 then
 begin
 Append(G);
 writeln(G,-2*x:2:5);
 Writeln(-2*x:2:5);
 v:=0;
 x:=0;
 end;
 End;
i:=i+1;
j:=i;
until i=400000;
Readln;
End.
```

A.2.9 FDP Gama

```
{Programa para obtener una FDP Gama}
Uses Crt;
Var
x,chi,ch,beta:real;
j,v:integer;
i:longint;
F,G:text;
Begin
clrscr;
x:=0;
Assign(F, 'kiss1.txt');
Reset(F);
{SELECCIONAR CONDICIONES INICIALES}
Assign(G,'gamma.txt');
Rewrite(G);
beta:=10;
```

```
Repeat
Read(F, Ch);
{Generando los vectores aleatorios con muestreo sistemtico sobre KISS}
 Repeat
 j:=j-10
 until j<10;
 if j=8 then
 Begin
 x:=x-ln(Ch);
 v:=v+1;
 IF v=9 then
 begin
 Append(G);
 writeln(G,beta*x:2:5);
 Close(G);
 Writeln(beta*x:2:5,i);
 v:=1;
 x:=0;
 end;
 End;
i:=i+1;
j:=i;
until i=400000;
Readln;
End.
```

A.2.10 Transformación de Box-Muller

```
{Algoritmo para generar una variable con distribucion normal}
Uses Crt;
Var
F,G:Text;
i,j,k:longInt;
ch,x,y,U,V:real;
Begin
Clrscr;
Assign(F,'kiss1.txt');
Reset(F);
Assign(G,'boxmull.txt');
Rewrite(G);
Close(G);
i:=31;
j:=5;
Repeat
i:=i+1;
```

```
Read(F,ch);
 Repeat
 j:=j-5;
 If j=4 then
 Begin
 U:=-2*Ln(ch);
 {Append(G);
 Writeln(G,x:2:5,' ',y:2:5);
 Close(G);}
 End;
 If j=3 then
 Begin
 V:=2*pi*ch;
 y:=sqrt(U)*sin(2*pi*V);
 x:=sqrt(U)*cos(2*pi*V);
 writeln('Proceso: ',(i-4)/10:4:0);
 Append(G);
 Writeln(G,x:2:5);{,' ',y:2:5);}
 Close(G);
 End;
 Until j<5;
j:=i;
Until i=400000;
Readln;
End.
```

A.2.11 Accept Reject

```
{Algoritmo para generar una variable con distribucion f usando Accept-Reject}
Uses Crt;
Var
F,G,h:Text;
i,j,k:longInt;
a,b,c,ch,U,V:real;
Function FDP:real;{FDP}
Begin
FDP:=(U*U*U-6*U*U+9*U)/25; {log=exp(-U);}{Normal(exp(-sqr(U)/2))/(2*pi);}
End;
Begin
Clrscr;
Assign(F,'kiss1.txt');
Reset(F);
Assign(G,'ar01.txt');
Rewrite(G);
Close(G);
```

```
Assign(H,'ar02.txt');
Rewrite(H);
Close(H);
i:=31;
j:=5;
b:=5;
a:=0;
c:=0.8;{1/(2*pi);}
Repeat
i:=i+1;
Read(F,ch);
 Repeat
 j:=j-5;
 If j=4 then
 Begin
 U:=(b-a)*ch+a;
 FDP;
 If FDP<=V
 Then
 Begin
 Append(G);
 Writeln(G,U:2:6,' ',V:2:6);
 Close(G);
 End
 Else
 Begin
 k:=k+1;
 Append(H);
 Writeln(H,U:2:6,' ',V:2:6);
 Close(H);
 Writeln(k,'.-rechazados ',FDP:2:3,'<',V:2:3);
 End;
 End;
 If j=3 then
 Begin
 V:=c*ch;
 End;
 Until j<5;
j:=i;
Until i=100000;
Readln;
End.
```

A.3 Capítulo 3

A.3.1 Integración Monte Carlo

```
Begin
clrscr;
Assign(F, 'kiss1.txt');
Reset(F);
Assign(T,'int02.txt');
Rewrite(T);
{SELECCIONAR CONDICIONES INICIALES}
x:=0;
limsup:=pi;
liminf:=0;
dmin:=(limsup-liminf)/10000;
Assign(G,'int01.txt');
Rewrite(G);
For k:=0 to 0 do
Begin
i:=0;
j:=500;
Reset(F);
intgx:=0;
Repeat
Read(F, Ch);
{Generando los vectores aleatorios con muestreo sistemtico sobre KISS}
Repeat
j:=j-25;
until j<25;
if j=k then
Begin
x:=(limsup-liminf)*(Ch)+liminf;
gx:=sqrt(x)*sin(x);
Append(G);
Writeln(G,x:2:5,' ',gx:2:5,' ',((gx*dmin)+intgx):2:5,' ',gx:2:5,' ',i);
Close(G);
intgx:=intgx+(gx*dmin);
End
Else
Begin
Clrscr
writeln('Ensayo No: ' i+1);
End;
i:=i+1;
j:=i;
```

```
until i=250000;
Writeln('Valor integral ',k,'=',' ',intgx:1:3);
Append(T);
Writeln(T,intgx,' ',k);
Close(T);
End;
Readln;
End.
```

A.3.2 Exploración estocástica

```
{Algoritmo para explorar una seccion del plano}
Uses Crt;
Var
F,G:Text;
i,j,k:longInt;
ch,r,t,x,y,opt:real;
Function Fxy:real;{Funcion de x y y}
Begin
fxy:=(sqr(x*sin(y)+y*sin(x))*cos(sin(x)*x))/y*x;
End;
Begin
Clrscr;
Assign(F,'kiss1.txt');
Reset(F);
Assign(G,'expest.txt');
Rewrite(G);
Close(G);
i:=31;
j:=5;
opt:=10000;
Repeat
i:=i+1;
Read(F,ch);
 Repeat
 j:=j-5;
 If j=4 then
 Begin
 r:=5*ch*cos(t);
 x:=r*cos(t);
 y:=r*sin(t);
 fxy;
 Append(G);
 Writeln(G,x:3:6,' ',y:3:6,' ',fxy:3:6,' ',opt:3:5);
 Close(G);
```

```
 Writeln(fxy:3:6,' ',i);
 If fxy<opt then
 Begin
 opt:=fxy;
 End;
 x:=-r*cos(t);
 y:=-r*sin(t);
 fxy;
 Append(G);
 Writeln(G,x:3:6,' ',y:3:6,' ',fxy:3:6,' ',opt:3:5);
 Close(G);
 Writeln(fxy:3:6,' ',i);
 If fxy<opt then
 Begin
 opt:=fxy;
 End;
 End;
 If j=2 then
 Begin
 t:=2*pi*ch;
 End;
 Until j<5;
j:=i;
Until i=100000;
Writeln('listo');
Readln;
End.
```

A.3.3 Optimización Monte Carlo

```
{Programa para optimizar una funcion atravs del mtodo montecarlo}
Uses Crt;
Var
x,y,x1,y1,Ch,opt:real;
i,j,l:longint;
k:integer;
F,G:text;
Function guarda:string;
Begin
Assign(G,'Optim01.txt');
Append(G);
Writeln(G,x:2:5,'    ',y:2:5,'    ',opt:2:5);
Close(G);
End;
Function fxy:real;
```

```
Begin
fxy:=-sqr(x-1)*exp(-(sqr(x)+sqr(y))/2);
End;
Begin
clrscr;
Assign(F, 'kiss1.txt');
Reset(F);
x:=3;
y:=2;
{SELECCIONAR CONDICIONES INICIALES}
   For k:=0 to 8 do
   Begin
   opt:=10000;
   write(x:1:1,'   ',y:1:1,'   ');
   i:=20;
   j:=20;
      Repeat
      Read(F, Ch);
      {Generando los vectores aleatorios con muestreo sistemtico sobre
          Repeat
          j:=j-5
          until j<5;
             case j of
             1: x:=x+(Ch/4)-1/2;
             2: y:=y+(Ch/4)-1/2;
             {Definimos la funcion f(x,y)}
             End;
            fxy;
            if fxy<opt then
            Begin
            l:=l+1;
            opt:=fxy;
            guarda;
            End;
          i:=i+1;
          j:=i;
      Until i=30000;
   writeln(opt:2:5,'   ',x:1:1,',',y:1:1);
   Read(F,ch);
   x:=(4*ch)-2;
   Read(F,ch);
   y:=(4*ch)-2;
   l:=0;
   End;
   Readln;
```

```
{   assign(G,'optim01.txt');
   Rewrite(G);}
End.
```

A.3.4 Algoritmo de Metropolis

```
{Programa para optimizar una funcion atravs del algoritmo de Metropolis}
Uses Crt;
Var
x,y,x1,x2,y2,z,fxy,Ch,opt,p,ropt,d:extended;
i,j:longint;
F,G:text;
Begin
clrscr;
Assign(F, 'kiss1.txt');
Reset(F);
{SELECCIONAR CONDICIONES INICIALES}
Assign(G,'Optim02.txt');
Rewrite(G);
x1:=-2;
y2:=-2;
opt:=10000;
ropt:=opt;
i:=20;
j:=20;
Repeat
Read(F, Ch);
{Generando los vectores aleatorios con muestreo sistemtico sobre KISS}
 Repeat
 j:=j-10;
 until j<10;
 case j of
 1:Begin
 x:=x1+Ch/4-(1/8);
 fxy:=sqr(sqr(x))+(sqr(sqr(y))-y*y*y)*sqr(cos(x*y))-exp(-(sqr(x)+sqr(y)));{2*x*x*x*x-
 fxy:=fxy/10000;
 {para generar las condiciones iniciales en cada iteracion}
 if fxy<=opt then
 Begin
 x1:=x;
 y2:=y;
{ Writeln(i);}
 Append(G);
 Writeln(G,x:2:5,' ',y:2:5,' ',fxy:2:5,' ',1.00000);
 close(G);
```

```
 opt:=fxy;
 End
 Else
 Begin
 d:=sqrt(sqr(x1-x)+sqr(y2-y));
 P:=Exp(-(fxy-opt)/d);{Boltzman:=1.3806503x10-23}
 { writeln(p:4:5,' ',d:2:5);}
 read(F,ch);
 z:=ch;
 If z<P Then
 Begin
 x1:=x;
 y2:=y;
 Append(G);
 Writeln(G,x:2:5,' ',y:2:5,' ',fxy:2:5,' ',p:2:5);
 close(G);
 opt:=fxy;
 End;
 End;
 If opt<ropt then
 Begin
 ropt:=opt;
 x2:=x;
 y2:=y;
 End;
 End;
 7:Begin
 y:=y2+ch/4-1/8;
 {Definimos la funcion f(x,y)}
 End;
 End;
i:=i+1;
j:=i;
WRiteln(i);
until i=400000;
Writeln(x2:2:5,' , ',y2:2:5,' ',ropt:2:5);
Readln;
End.
```

A.3.5 Simmulated Annealing

```
{Programa para optimizar una funcion a traves del algoritmo de menfriam
Uses Crt;
Var
x,y,x1,x2,y2,z,z2,Ch,opt,p,ropt,d,T,fxy:extended;
```

```
i,j:longint;
F,G,H:text;
{Function fxy:extended;
Begin
fxy:=sqr(sqr(x))+(sqr(sqr(y))-y*y*y)*sqr(cos(x*y))-exp(-(sqr(x)+sqr(y)));{2*x*x*x*x-3
{fxy:=fxy/10000;
End;}
Begin
clrscr;
Assign(F, 'kiss1.txt');
Reset(F);
{SELECCIONAR CONDICIONES INICIALES}
Assign(G,'simann.txt');
Rewrite(G);
Assign(H,'plotsa.txt');
rewrite(H);
x1:=0;
y2:=0;
z2:=0;
opt:=sqr(sqr(x))+(sqr(sqr(y))-y*y*y)*sqr(cos(x*y))-exp(-(sqr(x)+sqr(y)));
ropt:=opt;
i:=20;
j:=20;
T:=6.5;
Repeat
Read(F, Ch);
{Generando los vectores aleatorios con muestreo sistemtico sobre KISS}
 Repeat
 j:=j-10;
 until j<10;
 case j of
 0:Begin
 x:=x1+Ch/4-(1/8);
 fxy:=8-2*(z-y)x+8*y-4*y*z+4*sqr(y);
 fxy:=fxy/10000;
 {para generar las condiciones iniciales en cada iteracion}
 if fxy<=opt then
 Begin
 x1:=x;
 y2:=y;
{ Writeln(i);}
 Append(G);
 Writeln(G,x:2:5,' ',y:2:5,' ',fxy:2:5,' ',1.00000,' ',t:2:5);
 close(G);
 Append(H);
```

```
 Writeln(H,x:2:0,' ',y:2:0,' ',fxy:2:0);
 close(H);
 opt:=fxy;
 End
 Else
 Begin
 d:=sqrt(sqr(x1-x)+sqr(y2-y));
 P:=Exp(-(fxy-opt)/T);{Boltzman:=1.3806503x10-23}
 { writeln(p:4:5,' ',d:2:5);}
 read(F,ch);
 z:=ch;
 If z<P Then
 Begin
 x1:=x;
 y2:=y;
 Append(G);
 Writeln(G,x:2:5,' ',y:2:5,' ',fxy:2:5,' ',P:2:5,' ',t:2:5);
 close(G);
 opt:=fxy;
 Writeln(i,' ',opt:2:5,' ',z:1:6,' ',P:1:5);
 End;
 End;
 If opt<ropt then
 Begin
 ropt:=opt;
 x2:=x;
 y2:=y;
 End;
 End;
 3:Begin
 z:=z2+ch/4-1/8;
 End;
 6:Begin
 y:=y2+ch/4-1/8;
 {Definimos la funcion f(x,y)}
 End;
 End;
i:=i+1;
j:=i;
T:=1/(2*i);
until i=350000;
Writeln(x2:2:5,' , ',y2:2:5,' ',ropt:2:5,' ',p:2:6,' ',t:2:6);
Readln;
End.
```

A.3.6 Prueba de hipótesis MC

```
{Programa para optimizare una funcion atravs del mtodo montecarlo}
Uses Crt;
Const{valores de la muestra}
v1=12.17;
v2=12.32;
v3=12.32;
v4=11.27;
v5=12.15;
v6=12.26;
v7=12.01;
v8=12.37;
v9=12.24;
v10=12.57;
v11=12.70;
v12=12.71;
v13=12.49;
v14=12.73;
v15=12.71;
v16=12.70;
v17=12.43;
v18=12.51;
Var
prom,pob:real;
i,j,k,l1,l2,l3,l4,l5,l6,l7,l8,l9:integer;
G:text;
Begin
Randomize;
clrscr;
pob:=(v1+v2+v3+v4+v5+v6+v7+v8+v9+v10+v11+v12+v13+v14+v15+v16+v17+v18)/18;
Assign(G,'hip01.txt');
Rewrite(g);
Repeat
j:=Random(18);
case j of
1:begin l1:=j; prom:=prom+v1/9; end;
2:begin l2:=j; prom:=prom+v2/9;end;
3:begin l3:=j; prom:=prom+v3/9;end;
4:prom:=prom+v4/9;
5:prom:=prom+v5/9;
6:prom:=prom+v6/9;
7:prom:=prom+v7/9;
8:prom:=prom+v8/9;
9:prom:=prom+v9/9;
```

```
10:prom:=prom+v10/9;
11:prom:=prom+v11/9;
12:prom:=prom+v12/9;
13:prom:=prom+v13/9;
14:prom:=prom+v14/9;
15:prom:=prom+v15/9;
16:prom:=prom+v16/9;
17:prom:=prom+v17/9;
0:prom:=prom+v18/9;
else
Writeln('fallo por j=',j);
End;
k:=k+1;
If k=9 then
Begin
k:=0;
Writeln(G,prom:2:5,' ',2*pob-prom:2:5,' ',2*(pob-prom):2:5);
prom:=0;
End;
i:=i+1;
Until i=18000;
Readln;
End.
```

A.3.7 Conmutador

```
{Simulacion de conmutador}
Uses Crt;
Const
mediallamadas=22;
desvesta=15;
tiempoentre=7;
desvestllam=2;
Var
tiempo,hora,H1,H2,H3,H4,t1,t2,t3,t4:real;
j,k,n:integer;
l1,l2,l3,l4:boolean;
conmutador:text;
Function varaleat:real; {Genera una variable aleatoria con distribucion
Var
x,y:real;
i,n1:integer;
Begin
x:=tiempo+0.017;
 For i:=1 to n do
```

```
 Begin
 x:=975*x+0.15;
 while x>1 do
 Begin
 x:=x-1;
 End;
 y:=965*x+0.15;
 while y>1 do
 Begin
 y:=y-1;
 End;
 End;
varaleat:=(x-y);
End;
Function varaleat2:real; {Genera una variable aleatoria con distribucion dada}
Var
z:real;
i,n1:integer;
Begin
z:=tiempo+0.24;
 For i:=1 to n do
 Begin
 z:=945*z+0.18;
 while z>1 do
 Begin
 z:=z-1;
 End;
 End;
varaleat2:=-ln(z/desvestllam)+tiempoentre;
End;
BEGIN
Clrscr;
assign(conmutador,'conm.txt');
rewrite(conmutador);
Writeln('Dia Linea1 Linea2 Linea3 Linea4 Perdidas');
for n:=1 to 22 do
Begin
j:=1;
h1:=0;
h2:=0;
h3:=0;
hora:=0;
l1:=true;
L2:=TRUE;
L3:=TRUE;
```

```
L4:=TRUE;
Write(n,' ');
 Repeat
 tiempo:=varaleat2;
 hora:=hora+tiempo;
 if hora>h1 then
 begin
 l1:=True;
 end
 else
 begin
 l1:=False;
 end;
 if hora>h2 then
 begin
 l2:=True;
 end
 else
 begin
 l2:=False;
 end;
 if hora>h3 then
 begin
 l3:=True;
 end
 else
 begin
 l3:=False;
 end;
 if hora>h3 then
 begin
 l3:=True;
 end
 else
 begin
 l3:=False;
 end;
 if l1=TRUE then
 BEGIN
 H1:=Hora+(varaleat*desvesta+mediallamadas);
 t1:=t1+h1-hora;
 append(conmutador);
 Writeln(conmutador,j+1,' ',hora:2:2,' ',1,' ',h1-hora:2:2,' ',varaleat
 close(conmutador);
 END
```

```
 Else
 begin
 If l2=True then
 begin
 H2:=Hora+(varaleat*desvesta+mediallamadas);
 t2:=t2+h2-hora;
 append(conmutador);
 Writeln(conmutador,j+1,' ',hora:2:2,' ',2,' ',h2-hora:2:2,' ',varaleat2);
 close(conmutador);
 end
 else
 begin
 if l3=True then
 Begin
 H3:=Hora+(varaleat*desvesta+mediallamadas);
 t3:=t3+h3-hora;
 append(conmutador);
 Writeln(CONMUTADOR,j+1,' ',hora:2:2,' ',3,' ',h3-hora:2:2,' ',varaleat2);
 close(conmutador);
 End
 Else
 Begin
 if l4=True then
 Begin
 H4:=Hora+(varaleat*desvesta+mediallamadas);
 t4:=t4+h4-hora;
 append(conmutador);
 Writeln(CONMUTADOR,j+1,' ',hora:2:2,' ',4,' ',h4-hora:2:2,' ',varaleat2);
 close(conmutador);
 End
 Else
 Begin
 k:=k+1;
 append(conmutador);
 Writeln(CONMUTADOR,j+1,' ',hora:2:2,' ',0,' ',0);
 close(conmutador);
 End;
 end;
 end;
 End;
j:=j+1;
until hora>=480;
writeln(t1/60:2:2,' ',t2/60:2:2,' ',t3/60:2:2,' ',t4/60:2:2,' ',k);
End;
Readln;
```

```
END.
```

Bibliografía

[1] Sóbol I. *Método de Monte Carlo* MIR Lecciones populares de matemáticas

[2] Metropolis N. The Beginning of the Monte Carlo method Los Alamos Science Special Issue 1987

[3] Robert C. et al.*Monte Carlo Statistical Methods*
1999 Springer-Verlag, New York

[4] Patrick K. *Optimization by Simulated Annealing* Science 1983 Mayo 13 Vol 220

[5] Manly B. *Randomization and Monte Carlo Methods in Biology.* Chapman and Hall

[6] Rubinstein R. *Simulation and The Monte Carlo Method* 1981 1a edición Wiley & Sons, Inc

[7] Falcioni M., Vulpiani et al. 2006 *Properties making a chaotic system a good Pseudo Random Number Generator*

[8] Yamane T.*Estadística* 1979 3a edición Mexico Harla.

[9] Courant R.*Introducción al cálculo diferencial e integral 1974* $NewYork\ Wiley - Interscience$

[10] Cochran W. *Técnicas de muestreo* 1980 Compañía editorial continental.

[11] Metropolis N. et al.*Equation of State Calculations by fast Computing Machines.*
The Journal of Chemical Physics Vol. 21 No. 6 Junio 1953

[12] Peitgen H.Chaos and Fractals 1993 1a edición Springer.

[13] Ulam S. et al.1947. *Statistical methods in neutron diffusion.* Los Alamos Scientific Laboratory report LAMS551.

[14] Metropolis N. et al 1949 *The Monte Carlo Method* Journal of the American Statistical Association No. 44