Seminario de Investigación

Método de Monte Carlo para Simulación Computacional

Gerson Esteban Valenzuela González

30 de marzo de 2010

Resumen

En el presente seminario se revisan los conceptos básicos detrás del método de Monte Carlo. Varios métodos clásicos para generar números pseudo-aleatorios son revisados en primera instancia, ya que estos constituyen el punto de partida para cualquier simulación estocástica.

La solución de integrales definidas haciendo uso de números aleatoriamente distribuidos se presenta como una herramienta poderosa en comparación a los métodos determinísticos de solución de integrales (i.e., reglas de Simpson), cuando aquellas son dobles, triples, o en general, son integrales sobre un hiper espacio. En estos casos, donde los grados de libertad aumentan considerablemente, los métodos determinísticos se tornan lentos y complicados de programar. En comparación, Monte Carlo es conservativo en el tiempo de cálculo y muy simple de programar.

El estudio de sistemas físicos mediante Monte Carlo, tal como un sistema monoatómico, es más complejo. Las posibles configuraciones de las partículas del sistema, el espacio de fase, está distribuido con densidad de probabilidad bien definida por la mecánica estadística. En particular para sistemas a temperatura constante dicha distribución no es uniforme. En este contexto, las cadenas de Markov surgen como herramienta fundamental para crear configuraciones de un sistema físico con probabilidad proporcional a la teoricamente definida. La combinación de las cadenas de Markov y el método de Monte Carlo convergen en el método de Metrópolis, el cual es la opción adecuada para simular sistemas físicos de manera estocástica.

Índice general

Capítulo 1

Introducción

Los métodos estocásticos y sus diversas aplicaciones pertenecen a un área que no tiene lugar en la formación tradicional del ingeniero químico en la Universidad de Concepción, lo cual puede ser justificado en la medida que la mayoría de los problemas de procesos químicos son resueltos por métodos determinísticos o por la experiencia. En cambio, el investigador dedicado a estudios de postgrado generalmente encuentra resultados que han sido deducidos por medio de métodos de simulación; se refiere a esto el uso extensivo de computadores para resolver problemas usando la menor cantidad de información experimental posible y, por tanto, haciendo fuerte uso de conceptos básicos.

Es así como el Departamento de Ingeniería Química, a nivel de postgrado, enfrenta el uso de métodos estocásticos en un contingente limitado e importante de investigadores que desarrollan trabajos de tesis de magister y/o doctorado. Ejemplos específicos son el trabajo de Roberto Rozas sobre modelación de permeabilidad de materiales porosos, el de Cecilia Bustos sobre modelación de flujo multifásico en materiales porosos, el de Javier Quispe sobre modelación en sedimentación y el de Luis Segura sobre simulación de secado de materiales porosos. Otros esfuerzos mediante Monte Carlo también se han dado en grupos de investigación del Profesor Andrés Mejía y Profesor Hugo Segura.

El método de Monte Carlo (MC) es un método estocástico que consiste en resolver un problema usando la fuerza bruta. Por ejemplo, si se quisiera saber la probabilidad de que como resultado de lanzar una moneda salga sello, la vía más directa es lanzar la moneda una gran cantidad de veces y contar las veces en que el resultado es sello. Naturalmente llevar a cabo

esta empresa es en la práctica tedioso, pero si se establecen en un algoritmo reglas simples que representen en forma fidedigna el experimento, un computador hace posible «simular» sin mayor esfuerzo cosas que en la práctica parecen imposible.

Para todo fin que requiera de un método de Monte Carlo, es necesario ser capaz de generar números de igual probabilidad. Números de esta naturaleza, llamados pseudo-aleatorios, representarán el concepto básico de repetir muchas veces y en forma independiente una acción (lanzar la moneda una y otra vez). Tal herramienta encuentra utilidad en la solución de integrales definidas y en la generación de números distribuidos con una probabilidad determinada entre otras.

Así, el objetivo del presente seminario de investigación es estudiar las herramientas constituyentes del método de Monte Carlo, programarlas en FORTRAN 90 y aplicar el método a simulación molecular. En primera instancia de estudiaran algunas formas de generar números pseudo-aleatorios y luego el uso de esta herramienta para solucionar integrales definidas. Como segunda aplicación de estos números se verá la forma en que es posible producir números bajo certeras densidades de probabilidad. Finalmente se presentan las bases del método de Metrópolis, que es una serie de estrategias sobre el método de Monte Carlo para su aplicación en simulación molecular, y se aplica al modelo ferromagnético de Ising.

Capítulo 2

Números pseudo-aleatorios

Muchos de los generadores de números pseudo-aleatorios (RNG) produce una secuencia de elementos por medio de una transformación lineal en alguna estructura algebraica. En este contexto los tres métodos más utilizados son: Congruencial, Algortimos de cambio de registro y generadores Fibonacci desfasado (LFG). Todos estos métodos reciben el nombre genérico de «simple RNG» [1]. Los números aleatorios así generados deben cumplir la mayor cantidad de las exigencias siguientes [2]:

- deben distribuirse en forma uniforme,
- ser no correlacionados,
- nunca deben repetirse entre si,
- deben satisfacer los test de aleatoriedad,
- ser reproducibles,
- deben ser ajustables por medio de un valor inicial (semilla),
- deben ser generados rápidamente sin consumo excesivo de memoria.

Hay varios test que permiten identificar problemas con los generadores. Uno de los más simples es dividir el intervalo [0,1] en una gran cantidad de subsegmentos, generar los números aleatorios y verificar la uniformidad en la cantidad de números en cada subsegmento del intervalo. Otro test consiste en graficar dos o tres secuencias de números aleatorios, generados con distinta

semilla, y buscar estructuras regulares [3]. La descripción de varios test y su aplicación a los generadores mencionados puede ser revisada en [1].

2.1. Congruencial

Llamado también «método multiplicativo» es una forma simple y muy popular de generar secuencias de números pseudo-aleatorios [3]. La secuencia es generada mediante la combinación

$$X_n = (cX_{n-1} + a_0)\text{MOD } m \tag{2.1}$$

Todos los elementos de la serie son números enteros, m es el módulo, c y a_0 son multiplicadores y X_n es el número aleatorio generado como el resto de la división de $(cX_{n-1}+a_0)$ y m, operación representada por la función MOD. El máximo periodo alcanzado con este tipo de generadores es m siempre que c y a_0 sean correctamente elegidos [2]. La serie necesita del valor inicial X_0 conocido como «semilla» que debe ser impar para el desempeño adecuado del generador. Números pseudo-aleatorios en el intervalo [0,1] son normalmente utilizados y se consiguen mediante la división

$$U_n = X_n/(m-1) \tag{2.2}$$

donde U_n es real en [0,1]. Según referencia [3] un buen generador multiplicativo es el «32-bit lineal congruential algorithm» (CONG)

$$X_n = (16807 \cdot X_{n-1})\text{MOD } (2^{31}-1) \tag{2.3}$$

es cual es muy fácil de implementar tal como se muestra en el código FORTRAN de la Figura 2.1. Como se ve, los parámetros de entrada son el número de términos i y la semilla X_0. Este tipo de algoritmo es fácilmente integrable como una subrutina de un programa principal pasando los números pseudo-aleatorios almacenados en U_1 para los fines que sean convenientes.

Uno de los principales problemas de método congruencial es que el último bit de los números producidos es correlacionado lo que puede afectar en simulaciones donde existen muchas dimensiones. Otro problema relacionado a este método es que el periodo máximo del generador, para un procesador de 32 bit, es $2^{32} \approx 10^9$ que puede ser sobrepasado en unos pocos segundos con la actual velocidad de los procesadores. [2]

```
Program CONG
implicit none
integer*8 i, X0, c, j, m, X1
double precision, dimension(:), allocatable :: U1
i = 10; X0 = 23117 ! número de términos y semilla
c = 16807; m = 2**31 - 1 ! multiplicador y módulo
allocate(U1(i))
do j = 1,i
X1 = mod(c * X0, m)
X0 = X1(j)
U1(j) = dble(X1)/dble(m - 1)
end do
end program
```

Figura 2.1: Código del algoritmo CONG en FORTRAN

2.2. Algoritmos de cambio de registro

Este tipo de algoritmos resuelven algunos de los problemas asociados al método congruencial. En primer lugar es generada una tabla de número pseudo-aleatorios (i.e. método CONG) de la cual se generan nuevos números aleatorios mediante combinación de los encontrados en la tabla:

$$X_n = X_{n-p}.\text{XOR}.X_{n-q} \tag{2.4}$$

donde el operador .XOR. es igual a $(X_{n-p} + X_{n-q})$MOD 2 cuando X_{n-p} y X_{n-q} son variables lógicas. Esta definición de la fórmula recursiva deriva de que este tipo de algoritmos son programables por *hardware* [4], en donde las variables son binarias. La programación en software mediante FORTRAN es simplificada por la función **IEOR**(I, J) la que convierte los enteros I y J al sistema binario I_{bin} y J_{bin}, evalúa $(I_{bin} + J_{bin})$MOD 2 y convierte el resultado al sistema decimal. Valores típicos de p y q son mostrados en la Tabla 2.1. Cuando $p = 250$ se obtiene el generador R250, el comúnmente usado de esta clase [3]. Un código de implementación de este generador es mostrado en la Figura 2.2 el cual es muy simple de integrar como parte de un programa de Monte Carlo.

Cuadro 2.1: Valores típicos de p y q.

p	q
98	27
250	103
1279	216, 418
9689	84, 471, 1836, 2444, 4187

```
Program R250
implicit none
integer*8 i, X0
integer*8 c, j, m,p, q
integer*8, dimension(:), allocatable :: X1
double precision, dimension(:), allocatable :: U1
i = 10000; X0 = 75911 ! número de términos y semilla
p = 250; q = 103
allocate(X1(p + i))
allocate(U1(i))
! inicio de CONG
c = 16807; m = 2**31 − 1 ! multiplicador y módulo
do j = 1, p
X1(j) = mod(c ∗ X0, m)
X0 = X1(j)
end do
! fin de CONG
do j = p + i, p + 1
X1(j) = ieor(X1(j − p), X1(j − q))
U1(j − p) = dble(X1(j))/dble(m − 1)
end do
end program
```

Figura 2.2: Código FORTRAN del algoritmo R250 de cambio de registro iniciado con el método CONG.

2.3. Generador Fibonacci desfasado

Tal como su nombre lo indica, el método proviene de la serie de Fibonacci

$$X_n = X_{n-1} + X_{n-2} \tag{2.5}$$

la cual es modificada desfasando los índices de los sumandos y cambiando la operación aritmética. En general,

$$X_n = X_{n-p} \otimes X_{n-q} \tag{2.6}$$

donde el operador $\otimes$ representa una operación aritmética binaria: adición, sustracción o multiplicación. Las dos primeras pueden ser entre números de coma flotante en tanto que la multiplicación sólo entre números enteros. La generación de números aleatorios, en forma similar al método congruencial, pasa por aplicar la operación MOD entre el resultado de $\otimes$ y el módulo m. De esta forma, se define el LFG mediante la recurrencia

$$X_n = X_{n-p} \odot X_{n-q} \tag{2.7}$$

donde $\odot$ representa al combinación de la operación $\otimes$ y la función MOD. Note que cuando $\otimes$ es multiplicación, $m = 2$ y X_i son variables lógicas, se recupera el algoritmo de cambio de registro. El algoritmo se inicia con una tabla de p número aleatorios y los nuevos números se obtienen de la combinación definida por $\odot$. Este método tiene un periodo máximo bastante mayor al obtenido con el método congruencial, en particular, si el módulo es $m = 2^b$ (donde b son los bit de precisión de las variables X_i), cuando $\odot =$ .XOR. el periodo máximo es de $2^p - 1$. Cuando $\odot$ es adición o sustracción el periodo máximo es $2^{b-1}(2^p - 1)$ y si $\odot$ es multiplicación el periodo es $2^{b-1}(2^p - 3)$ [2]. Test empíricos han mostrado que las mejores propiedades son obtenidas cuando se utiliza multiplicación, seguido de cerca por adición y sustracción dejando al método de cambio de registro en pésima posición [2, 1]. Según Marsaglia [1] la combinación $p = 17$, $q = 5$ y $\odot = +$ ha mostrado periodo máximo $2^{17}(2^{31} - 1)$. Designando esta configuración como F(17,5,+), el código R250 mostrado en la Figura 2.2 es modificado fácilmente a F(17,5,+) obteniendo mayor calidad en la generación aleatoria de números. Esto se muestra en la Figura 2.3.

```
Program F17P5
implicit none
integer*8 i, X_0
integer*8 c, j, m,p, q
integer*8, dimension(:), allocatable :: X_1
double precision, dimension(:), allocatable :: U_1
i = 15302; X_0 = 25913 ! número de términos y semilla
p = 17; q = 5
allocate(X_1(p + i))
allocate(U_1(i))
! inicio de CONG
c = 16807; m = 2**31 - 1 ! multiplicador y módulo
do j = 1, p
X_1(j) = mod(c * X_0, m)
X_0 = X_1(j)
end do
! fin de CONG
do j = p + i, p + 1
X_1(j) = mod(X_1(j - p) + X_1(j - q), m)
U_1(j - p) = dble(X_1(j))/dble(m - 1)
end do
end program
```

Figura 2.3: Código del algoritmo F(17,5,+) en FORTRAN. Pequeñas modificaciones a R250 supone, teóricamente, mejores propiedades de los números aleatorios.

2.4. Otros generadores

Se ha mostrado que mediante la combinación aditiva de dos diferentes generadores congruenciales de 32 bit, se puede producir uno que pasa todos los test conocidos con un gran periodo de alrededor 10^{18}, bastante mayor al periodo máximo de este tipo de generadores. En general, la combinación de cualquiera dos generadores mediante la operación aritmética $\odot$ mejora la aleatoriedad respecto a los métodos independientes.

Los RNG presentados dan un marco conceptual suficiente para los fines del presente seminario. Los códigos en FORTRAN presentados en las Figuras 2.1, 2.2 y 2.3 van a ser utilizados en los capítulos siguientes donde se presentan varias situaciones en las que el método de Monte Carlo es utilizable.

2.5. Distribuciones no uniformes

Para muchos casos resulta necesario disponer de números aleatorios distribuidos con cierta probabilidad no uniforme. Por ejemplo, las velocidades de los átomos o moléculas que se encuentran en un espacio definido se ajustan bien a la distribución de Maxwell-Boltzmann de velocidades, la que es una distribución Gaussiana, y la capacidad de generar números aleatorios con dicha distribución es muy útil en simulación molecular.

A partir de un conjunto de números aleatorios con distribución uniforme cualquier otra distribución puede ser obtenida. Sea $f(x)$ una función de distribución y $F(y)$ la función acumulada de $f(x)$ entre $-\infty$ e y:

$$F(y) = \int_{-\infty}^{y} f(x)dx \tag{2.8}$$

Entonces asignando números aleatorios con distribución uniforme a $F(y)$, el conjunto de números y_i se distribuye de acuerdo a $f(x)$. Esta es la forma más general para obtener distribuciones distintas a la uniforme [3]. La distribución normal o Gaussiana está dada por:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{2.9}$$

donde σ es la desviación estandar de la distribución y μ el valor medio. La función acumulada de f es la integral

$$F(y) = \int_{-\infty}^{y} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} dx \tag{2.10}$$

Mediante el cambio de variables $s = (x - \mu)/(\sigma\sqrt{2})$ y $t = (y - \mu)/(\sigma\sqrt{2})$ la función acumulada se reduce a

$$F(t) = \frac{1}{2}[1 + \text{erf}(t)] \tag{2.11}$$

$$\text{erf}(t) = \frac{2}{\sqrt{\pi}} \int_0^t e^{-s^2} ds \tag{2.12}$$

Entonces, se asignan $F(t)_i$ como números aleatorios de distribución uniforme, se calculan los $\text{erf}(t_i) = 2F(t)_i - 1$ y los t_i se obtienen como la inversa de $\text{erf}(t_i)$. Luego se pueden calcular los x_i del cambio de variables $x_i = t_i\sigma\sqrt{2} + \mu$. La evaluación de la inversa de la función de error no es trivial y se puede recurrir a una tabla de interpolación para resolver el problema, sin embargo, algunos autores han propuesto formas más simples para obtener una distribución normal. Si x_1 y x_2 son números uniformemente distribuidos, se pueden obtener números de distribución normal y_1 e y_2 mediante el método de Box-Muller:

$$y_1 = (-2 \ln x_2)^{1/2} \cos(2\pi x_1) \tag{2.13}$$

$$y_2 = (-2 \ln x_2)^{1/2} \text{sen}(2\pi x_1) \tag{2.14}$$

de esta forma un vector $\mathbf{x}$ de números aleatorios uniformes puede ser convertido fácilmente a un vector $\mathbf{y}$ de números de distribución normal.

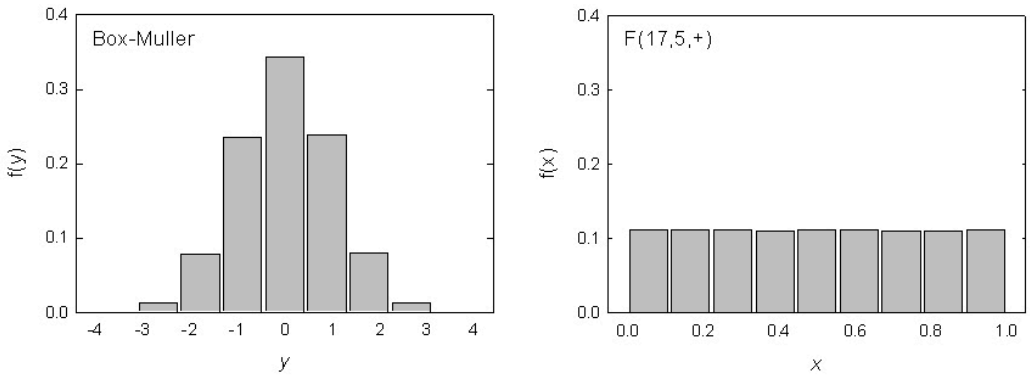

Figura 2.4: 100000 números pseudo-aleatorios distribuidos uniformemente con el generador de Fibonacci desfasado F(17,5,+) y su conversión a la distribución normal mediante el método de Box-Muller

2.6. Ejemplo: distribución de velocidades de partículas

En la simulación molecular por dinámica molecular (MD) se resuelven las ecuaciones de movimiento de Newton en el tiempo obteniendo un conjunto de posiciones y velocidades de las partículas (moléculas en el caso más general) de donde es posible calcular propiedades termodinámicas. MD es un método determinístico y como tal requiere de condiciones iniciales, tanto de posición como de velocidades de cada partícula, para la integración de las ecuaciones de movimiento. En este sentido se suele asignar las velocidades de la distribución de Maxwell-Boltzmann,

$$f(v_i) = \left(\frac{m}{2\pi k_b T}\right)^{0.5} \exp\left(-\frac{m v_i^2}{2 k_B T}\right) \tag{2.15}$$

la cual es una distribución gaussiana de la variable aleatoria v en la i-ésima dirección. La distribución tiene promedio $\langle v_i \rangle = 0$ y varianza $\sigma^2 = k_B T/m$.

$$f(v_i) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{v_i^2}{2\sigma^2}\right) \tag{2.16}$$

Con un generador de número pseudo-aleatorios uniformemente distribuidos y el método de Box-Muller, asignar velocidades v_i a un conjunto de N partículas se vuelve una tarea prácticamente trivial. Considere que $y_i = \sigma f(v_i)$ es la densidad de probabilidad de $t_i = v_i/\sigma$ que se distribuye en forma normal. Entonces t_i se obtiene con el método de Box-Muller y v_i se determina como $v_i = t_i\sigma$. Para obtener distintas velocidades en x, y y z basta con asignar una semilla distinta para el RNG en cada caso.

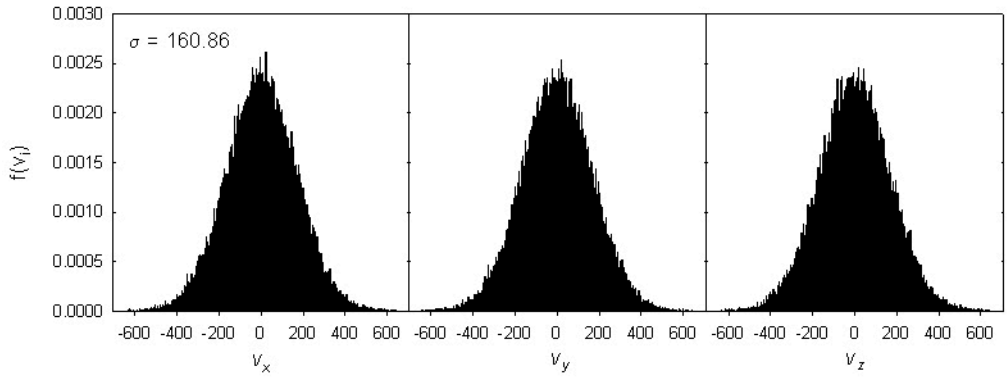

Figura 2.5: Distribución de velocidades v_x, v_y, v_z de Maxwell-Boltzmann para 100000 partículas mediante números t_x, t_y, t_z normalmente distribuidos generados con el método de Box-Muller.

Capítulo 3

Integración numérica con Monte Carlo

La integral definida puede admitir tres tipos de solución: analítica, numérica determinística y numérica estocástica. La primera se basa en el segundo teorema fundamental del cálculo, lo que lleva a resolver la integral indefinida asociada al problema mediante reducción de la integral a formas más simples que admitan solución por reglas de antiderivación. Por ejemplo la solución de $\int_a^b \ln(x)dx$ según el segundo teorema fundamental es $f(b) - f(a)$ donde f es la antiderivada de $F(x) = \ln(x)$. La integral indefinida $\int F(x)dx$ no admite una regla de antiderivación directa y debe ser simplificada mediante el método de separación de variables. De acuerdo a ello, $\int F(x)dx = x\ln(x) - \int dx = x(\ln(x) - 1)$ en donde se aplicó una regla de antiderivación directamente a $\int dx$. Con ello, la solución de la integral definida es $b(\ln(b) - 1) - a(\ln(a) - 1)$. Muchos problemas son intratables por este método, sobre todo cuando las ecuaciones provienen de problemas físicos antes que de académicos, fundamentalmente porque no siempre existe la solución analítica a una integral indefinida. Un ejemplo clásico es $\int e^{t^2}dt$, constituyente fundamental de la función de error, $\text{erf}(x) = 2\pi^{-0.5}\int_0^x e^{t^2}dt$, presente en problemas de fenómenos de transporte, probabilidades, etc.

Los problemas sin solución analítica deben ser aproximados mediante métodos numéricos de integración. Los aquí denominados «determiníticos» se refieren a la discretización del intervalo de integración $[a, b]$ en subintervalos de igual longitud Δx. Para muchos fines es de total utilidad la regla del trapecio:

$$\int_a^b f(x)dx \approx \frac{b-a}{2n}[f(x_0) + 2f(x_1) + \cdots + 2f(x_{n-1}) + f(x_n)] \tag{3.1}$$

o la regla de Simpson compuesta:

$$\int_a^b f(x)dx \approx \frac{b-a}{3n}[f(x_0)+4f(x_1)+2f(x_2)+4f(x_3)+2f(x_4)+\cdots+4f(x_{n-1})+f(x_n)] \quad (3.2)$$

Para la regla del trapecio n es cualquier número natural, mientras que para Simpson n debe ser impar. El error de truncamiento del trapecio (T) y de Simpson (S) esta dado por las siguientes expresiones:

$$\epsilon_T = -\frac{1}{12}(b-a)f^{(2)}(\eta)(\Delta x)^2 \quad (3.3)$$

$$\epsilon_S = -\frac{1}{180}(b-a)f^{(4)}(\eta)(\Delta x)^4 \quad (3.4)$$

donde η es un número que existe en el intervalo $[a,b]$. Estas reglas son exactas para funciones lineales (trapecio) y para funciones cuya cuarta derivada anule (simpson), como polinomios de grado tres y menor. La integración numérica con estos métodos es bastante simple para integrales simples, como la función error por ejemplo, pero considere el caso en que la integral depende de más grados de libertad. Una integral triple genérica es

$$I = \int_{a_1}^{a_2}\int_{b_1}^{b_2}\int_{c_1}^{c_2} f(x,y,z)dzdydx \quad (3.5)$$

y su solución por alguno de los métodos anteriores pasa por separar la integral y resolver recursivamente,

$$I = \int_{a_1}^{a_2} f_1(x)dx \quad (3.6)$$

$$f_1(x) = \int_{b_1}^{b_2} f_2(x,y)dy \quad (3.7)$$

$$f_2(x,y) = \int_{c_1}^{c_2} f(x,y,z)dz \quad (3.8)$$

esto implica una gran cantidad de aplicaciones del método de integración. Por ejemplo si se utilizar la regla de simpson para resolver la integral triple anterior dividiendo cada intervalo x, y y z en una misma cantidad n, habría que ejecutar n^3+n^2+n veces la regla de Simpson. En general para un sistema de $\mathcal{F}$ grados de libertad la cantidad de llamadas a la regla sería $n^{\mathcal{F}}+n^{\mathcal{F}-1}+n^{\mathcal{F}-2}+\cdots+n$. Es en estos casos, donde el problema consta de muchos grados de libertad, que el método de Monte Carlo cobra importancia.

La integración de una función unidimensional se puede resolver mediante una cuadratura simple como la mostrada en la Figura 3.1.

$$\int_a^b F(x)dx = (b-a)\langle F\rangle \quad (3.9)$$

donde $\langle F \rangle$ es el valor medio de F en el intervalo [a,b]. El computo de $\langle F \rangle$ se puede conseguir evaluando a F en la mayor cantidad de puntos en el intervalo de interés, $\langle F \rangle = n^{-1} \sum_i^n F(x_i)$, donde es claro que la elección de los puntos x_i debe ser completamente independiente para resolver la integral exitosamente. Si se eligieran los puntos centrados en $x = 0.3$, por ejemplo, el valor de $\langle F \rangle$ se aproximaría a $F(0.3)$ y la integral no sería correctamente evaluada. Los puntos x_i necesariamente tienen distribución de probabilidad uniforme.

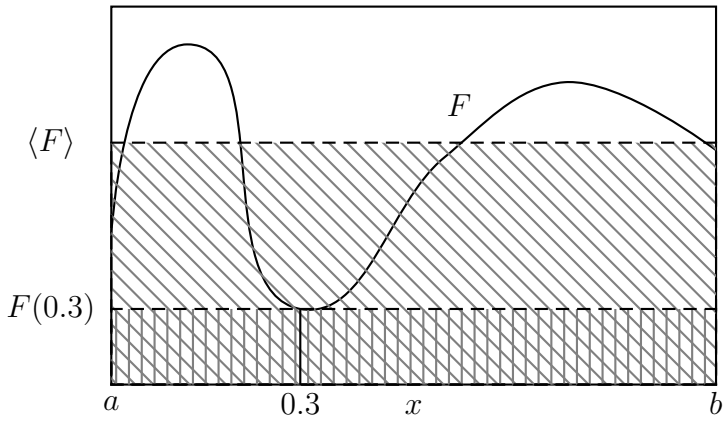

Figura 3.1: Cálculo de una integral definida por cuadratura. Área tachada con lineas oblicuas corresponde a una elección independiente de x_i. Área tachada con lineas verticales se aproxima a elegir puntos cercanos a $x = 0.3$.

Esta cuadratura se extiende fácilmente a la evaluación de la integral triple como sigue:

$$I = \int_{a_1}^{a_2} \int_{b_1}^{b_2} \int_{c_1}^{c_2} f(x, y, z) dz dy dx = (a_2 - a_1)(b_2 - b_1)(c_2 - c_1) n^{-1} \sum_i^n F(x_i, y_i, z_i) \qquad (3.10)$$

donde n corresponde al número de puntos uniformemente distribuidos. La simplicidad del método de Monte Carlo respecto al numérico determinístico es evidente conforme aumentan los grados de libertad del problema.

3.1. Integración unidimensional

Como primer paso en la comprensión del Método de Monte Carlo como integrador numérico se comparará la solución del siguiente problema

$$I = \int_0^2 [e^x + \sin(x\pi)] dx \qquad (3.11)$$

en forma analítica, con la regla de Simpson y con Monce Carlo. La solución analítica es:

$$I = e^2 - 1 \approx 6.3890561 \qquad (3.12)$$

Los métodos de Simpson y Monte Carlo se han programado en FORTRAN y los resultados son mostrados en la Tabla 3.1 en función de n, que representa la longitud del vector de números aleatorios y el número de subintervalos de discretización para Monte Carlo y Simpson respectivamente. Se ha usado el generador de números aleatorios Fibonacci desfasado $F(17, 5, +)$ con semilla X_0=27113. Inspeccionando los resultados es clara la ventaja de la regla de Simpson contra el método de Monte Carlo tanto en precisión como en tiempo computacional, sin embargo, la diferencia no es tan radical como se podría esperar y ciertamente el método de Monte Carlo es una forma eficiente para resolver una integral simple.

Cuadro 3.1: Comparación en la solución de $I = \int_0^2 [e^x + \sin(x\pi)]dx$ obtenida con Monte Carlo (MC) y con la regla de Simpson (S).

n	I_{MC}	I_S	CPU_{MC}, segundos	CPU_{S}, segundos
101	6.6223	6.2454	$2.6 \cdot 10^{-5}$	$3.2 \cdot 10^{-5}$
1001	6.3721	6.3745	$7.1 \cdot 10^{-5}$	$9.1 \cdot 10^{-5}$
10001	6.3740	6.3878	$7.1 \cdot 10^{-4}$	$5.1 \cdot 10^{-4}$
100001	6.3922	6.3891	$5.5 \cdot 10^{-3}$	$4.3 \cdot 10^{-3}$
1000001	6.3889	6.3893	$5.5 \cdot 10^{-2}$	$3.7 \cdot 10^{-2}$
10000001	6.3894	6.3893	$5.6 \cdot 10^{-1}$	$3.8 \cdot 10^{-1}$

3.2. Integración bidimensional

En el siguiente ejemplo se muestra la eficiencia del método de Monte Carlo para el cálculo del volumen generado por los ejes coordenados y la función $f(x, y) = 2x^2 + 3y^2$ en los intervalos $[0, 2]$ y $[0, 5]$ para x e y respectivamente.

$$V = \int_0^2 \int_0^5 [2x^2 + 3y^2]dydx \approx 276.666 \tag{3.13}$$

Como ya se ha comentado, la regla de Simpson se aplica en forma recursiva:

$$V_S \approx \int_0^2 g(x)dx \tag{3.14}$$

$$g(x) \approx \int_0^5 [2x^2 + 3y^2]dy \tag{3.15}$$

de forma que en primer lugar se calcula, con la regla de Simpson, $g(x)$ para cada x en el intervalo $[0, 2]$. Construida g se calcula V_S. En el método de Monte Carlo se resuelve:

$$V_{MC} \approx (2-0)(5-0)\frac{1}{n}\sum_{i=1}^{n}[2x_i^2 + 3y_i^2] \tag{3.16}$$

En la Tabla 3.2 se muestran los resultados mediante ambos métodos y se puede notar un aumento significativo en el consumo de tiempo de CPU utilizando Simpson compuesto para resolver la integral doble. Comparando con la solución de la integral simple de la sección anterior, Monte Carlo resulta conservativo en el consumo computacional con un error que en muchas aplicaciones puede considerarse aceptable. Otra ventaja es la programación del método; mientras que Simpson comienza a volverse engorroso, en Monte Carlo únicamente hay que definir un vector adicional de números aleatorios para solucionar el problema.

Cuadro 3.2: Comparación en la solución de $V = \int_0^2 \int_0^5 [2x^2 + 3y^2] dydx$ obtenida con Monte Carlo (MC) y con la regla de Simpson (S).

n	V_{MC}	V_S	CPU_{MC}, segundos	CPU_{S}, segundos
101	316.0113	271.6478	$1.8 \cdot 10^{-5}$	$1.5 \cdot 10^{-4}$
1001	273.0381	276.1615	$6.9 \cdot 10^{-5}$	$9.4 \cdot 10^{-3}$
10001	274.9171	276.6161	$9.2 \cdot 10^{-4}$	$8.0 \cdot 10^{-1}$
100001	277.0332	276.6616	$6.8 \cdot 10^{-3}$	79.7
1000001	276.8941	-	$7.3 \cdot 10^{-2}$	-
10000001	276.7538	-	$7.1 \cdot 10^{-1}$	-
100000001	276.6855	-	15.2	-

Con el ejemplo anterior ya se establece las ventajas y el tipo de problemas que Monte Carlo esta orientado a resolver.

3.3. Estimación de error en la integración

Una forma de estimar el error en la integración con Monte Carlo esta dado por la siguiente fórmula [5]:

$$\sigma_m \approx \sigma/\sqrt{n} \tag{3.17}$$

en donde σ_m es el error del método y σ es la desviación estandar de la integral calculada por Monte Carlo. Por ejemplo, para una integral simple $I = \int_a^b f(x)dx$

$$\sigma^2 = \langle I^2 \rangle - \langle I \rangle^2 \tag{3.18}$$

donde

$$\langle I \rangle = (b-a)\frac{1}{n}\sum_{i=1}^{n} f(x_i) \tag{3.19}$$

$$\langle I^2 \rangle = (b-a)^2\frac{1}{n}\sum_{i=1}^{n} f(x_i)^2 \tag{3.20}$$

Para una integral doble $V = \int_{a_1}^{a_2}\int_{b_1}^{b_2} f(x,y)dydx$

$$\sigma^2 = \langle V^2 \rangle - \langle V \rangle^2 \tag{3.21}$$

$$\langle V \rangle = (a_2-a_1)(b_2-b_1)\frac{1}{n}\sum_{i=1}^{n} f(x_i,y_i) \tag{3.22}$$

$$\langle V^2 \rangle = (a_2-a_1)^2(b_2-b_1)^2\frac{1}{n}\sum_{i=1}^{n} f(x_i,y_i)^2 \tag{3.23}$$

y la extensión a más grados de libertad se vuelve intuitiva. Para el ejemplo de las Sección 3.2 se tienen los siguientes resultados:

Cuadro 3.3: Error estimado con Monte Carlo el la solución de $V = \int_0^2\int_0^5[2x^2+3y^2]dydx$. La solución analítica es $V = 276.666$.

n	V_{MC}	error real	error estimado
101	316.0113	39.54	21.71
1001	273.0381	3.62	7.03
10001	274.9171	1.74	2.25
100001	277.0332	0.37	0.71
1000001	276.8941	0.23	0.25
10000001	276.7538	0.088	0.071

Se observa que el error del método es bien estimado por la Ecuación 3.17 conforme $n \to \infty$ y que tiende a disminuir considerablemente mientras mayor es el número de evaluaciones.

3.4. Importance sampling

En la sección anterior se mostró que el error de la integración con Monte Carlo es proporcional a σ, la desviación estandar de la evaluación de la integral. El error del método puede ser disminuido y aún requerir menos números de evaluaciones de la integral para obtener un resultado satisfactorio mediante la técnica de importance sampling. Sea p una función de densidad de probabilidad en el intervalo $[a, b]$,

$$\int_a^b p(x)dx = 1 \tag{3.24}$$

y una integral I a calcular en el mismo intervalo,

$$I = \int_a^b F(x)dx \tag{3.25}$$

La ecuación anterior se puede escribir en forma equivalente como

$$I = \int_a^b \left[\frac{F(x)}{p(x)}\right] p(x)dx \tag{3.26}$$

Ahora, calculando el racional como un valor promedio en el intervalo $[a, b]$ la integral tiene solución aproximada I_n

$$I_n = \int_a^b \left\langle \frac{F}{p} \right\rangle p(x)dx = \left\langle \frac{F}{p} \right\rangle \int_a^b p(x)dx = \left\langle \frac{F}{p} \right\rangle \tag{3.27}$$

$$\left\langle \frac{F}{p} \right\rangle = \frac{1}{n}\sum_{i=1}^{n} \frac{F(x_i)}{p(x_i)} \tag{3.28}$$

La esencia del método es determinar una función p tal que el cuociente F/p sea una función suave y lo más constante posible en el intervalo $[a, b]$. Como siempre, los x_i son números de distribución uniforme. Cuando $p(x) = C$ con $C =$ constante, de Ecuación 3.24, $C = (b-a)^{-1}$ con lo cual $n^{-1}\sum_{i=1}^{n} F(x_i)/p(x_i) = (b-a)n^{-1}\sum_{i=1}^{n} F(x_i)$ de modo que se recupera la cuadratura analizada en los capítulos anteriores. Como ejemplo de la técnica de importance sampling se estudia la solución de la integral $I = \int_0^1 \mathrm{e}^{-x^2}dx$ utilizando dos funciones de distribución distintas: $p_1(x) = C_1$ y $p_2(x) = C_2\mathrm{e}^{-x}$. Para el primer caso se tiene:

$$\int_0^1 C_1 dx = 1 \Rightarrow C_1(1-0) = 1 \Rightarrow C_1 = 1 \tag{3.29}$$

de modo que

$$\left\langle \frac{F}{p} \right\rangle = \frac{1}{n}\sum_{i=1}^{n} \mathrm{e}^{-x_i^2} \quad 0 \le x_i \le 1 \tag{3.30}$$

Para el segundo caso:

$$\int_0^1 C_2 \mathrm{e}^{-x} dx = 1 \Rightarrow C_2(-\mathrm{e}^{-1} + \mathrm{e}^0) = 1 \Rightarrow C_2 = \frac{1}{(-\mathrm{e}^{-1} + \mathrm{e}^0)} \approx 1.58 \tag{3.31}$$

de modo que

$$\left\langle \frac{F}{p} \right\rangle = \frac{1}{n} \sum_{i=1}^{n} \frac{\mathrm{e}^{-x_i^2}}{1.58\mathrm{e}^{-x_i}} \quad 0 \leq x_i \leq 1 \tag{3.32}$$

Los x_i se distribuyen uniformemente en el intervalo $[0, 1]$. La solución de la integral se ha programado en FORTRAN y los resultados son mostrados en la Tabla 3.4. Se puede observar que el método de importance sampling, para este ejemplo, es un orden de magnitud menor a Monte Carlo respecto el tamaño del vector de número aleatorios, el tiempo empleado y el error estimado.

Cuadro 3.4: Solución de la integrál $\int_0^1 \mathrm{e}^{-x^2} dx$ con el método de importance sampling usando $p(x) = C1$ (Monte Carlo normal) y $p(x) = C_2 e^{-x}$ (Monte Carlo optimizado)

	Monte Carlo normal	importance sampling
n	10000	1000
I_n	0.749	0.748
error	0.002	0.002
CPU time (s)	$5.7 \cdot 10^{-4}$	$6.7 \cdot 10^{-5}$
n	100000	10000
I_n	0.7476	0.7491
error	0.0006	0.0005
CPU time (s)	$4.5 \cdot 10^{-3}$	$8.2 \cdot 10^{-4}$

Se deduce del ejemplo anterior que la técnica de importance sampling puede incrementar considerablemente la calidad de la integración con Monte Carlo. La idea de introducir la densidad de probabilidad $p(x)$ es que el cuociente $g(x) = F(x)/p(x)$ sea una función casi constante en el intervalo de integración $[a, b]$. Para el caso anterior, $g_1(x) = \mathrm{e}^{-x^2}$ y $g_2(x) = \mathrm{e}^{-x^2}/\mathrm{e}^{-x}$ son obviamente funciones de distinto comportamiento. En la Figura 3.2 se han graficado ambas funciones en el intervalo $[0, 1]$ de donde se deduce, con cierta facilidad, que al elegir números x_i uniformemente distribuidos en el intervalo $[0, 1]$ para calcular el valor esperado de g_1 se obtendrá mayor desviación estandar de los datos respecto al mismo experimento con g_2 por

el simple motivo que g_1 cubre mayor rango de la ordenada que g_2. La elección de la densidad de probabilidad depende del intervalo de integración, en efecto, la densidad $p(x) = \mathrm{e}^{-x}$ no da buenos resultados para el intervalo $[0, 2]$ o cualquier otro distinto al $[0, 1]$. Pese a lo anterior, siempre es posible normalizar una función F en el intervalo genérico $[a, b]$ hasta el $[0, 1]$.

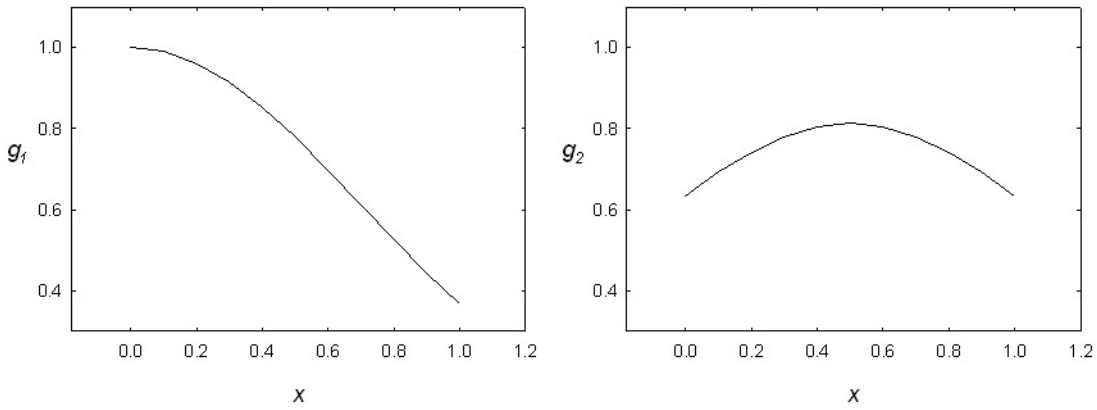

Figura 3.2: Diferencias al utilizar una densidad de probabilidad como importance sampling en el método de Monte Carlo.

Capítulo 4

Monte Carlo en simulación molecular

Con los capítulos anteriores se espera haber entregado una idea mediana, pero clara, de lo que es el método de Monte Carlo en general. El objetivo del presente capítulo es utilizar el método para resolver problemas moleculares, relacionados fuertemente con la mecánica estadística, para lo cual resulta indispensable presentar algunos conceptos que serán de total utilidad.

4.1. Cadenas de Markov

Se entiende una cadena de Markov como la dependencia que tiene la probabilidad de un evento estocástico respecto al ocurrido inmediatamente antes [3]. En forma matemática es una probabilidad condicional,

$$p(X_{t_n} = S_{i_n} | X_{t_{n-1}} = S_{i_{n-1}}) \tag{4.1}$$

donde S_1, S_2,... son los posibles estados de un sistema, en tanto X_t es el estado que adquiere el sistema en el instante t.

En contraste con el lanzamiento de una moneda, donde cada lanzamiento es independiente del anterior, la probabilidad de que una persona compre una guitarra dependerá, intuitivamente, de la compra que se haya realizado antes. La persona como tal se dice es el «sistema» que tendrá distintos estados S_1, S_2,... (i.e. distintos matices entre necesidades y cantidad disponible de dinero). Este sistema no permanece constante en el tiempo, sino que va cambiando entre estados de acuerdo a su decisión de compra: hay una transición constante entre estados,

$$W_{ij} = p(X_{t_n} = S_j | X_{t_{n-1}} = S_i) \tag{4.2}$$

La Ecuación 4.2 es una interpretación práctica de la Ecuación 4.1, en efecto el concepto de transición entre estados es central en la simulación de MC como se verá más adelante. La probabilidad de que ocurran dos eventos consecutivos, $p(X_{t_n} = S_j)$, es por definición de probabilidad condicional,

$$p(X_{t_n} = S_j) = p(X_{t_n} = S_j | X_{t_{n-1}} = S_i) p(X_{t_{n-1}} = S_i) \tag{4.3}$$

$$p(X_{t_n} = S_j) = W_{ij} p(X_{t_{n-1}} = S_i) \tag{4.4}$$

4.2. Sistema molecular y mecánica estadística

Un sistema molecular, en el contexto de este trabajo, se entiende como un sistema compuesto de átomos que pueden o no estar enlazados entre si formando distintas moléculas. Estos átomos son modelados como partículas situadas en un espacio dimensional definido que pueden interactuar entre si de acuerdo a alguna ley, un potencial de interacción semi-empírico, que en primera instancia define la naturaleza del sistema. Toda la materia, en la escala de nanómetros, puede ser modelada en dicha forma, así pueden ser estimadas sus propiedades con alto grado de certeza.

Las partículas se encuentran necesariamente confinadas en un espacio dimensional y se mueven vertiginosamente cambiando continuamente de posición ($\mathbf{r}$) y momentum ($\mathbf{p}$), como resultado, se genera el espacio de fase ($\mathbf{r}^N$, $\mathbf{p}^N$) de acuerdo a las restricciones que estén sometido el sistema, por ejemplo, condiciones definidas de temperatura y presión.

El espacio de fase de un sistema que ha alcanzado el estado de mínima energía libre, en acuerdo a las restricciones y la naturaleza del sistema, se distribuye con densidad de probabilidad [6],

$$p_i = \frac{exp(-\mathcal{H}_i / k_B T)}{Q} \tag{4.5}$$

en donde la función de partición es,

$$Q = c \int exp(-\mathcal{H}/k_B T) d\mathbf{r}^N d\mathbf{p}^N \tag{4.6}$$

donde c es una constante. Aquí el Hamiltoniano, $\mathcal{H} = \mathcal{H}(\mathbf{r}^N, \mathbf{p}^N)$, engloba la contribución energética del sistema aislado. Cualquier propiedad del sistema puede ser calculada como

$$\langle A \rangle = \frac{1}{Q} \int A(\mathbf{r}^N, \mathbf{p}^N) exp(-\mathcal{H}/k_B T) d\mathbf{r}^N d\mathbf{p}^N \tag{4.7}$$

El objetivo último de simular un sistema molecular es calcular sus propiedades de acuerdo a la Ecuación 4.7, el objetivo de este capítulo es examinar cómo es posible realizar esta tarea utilizando procedimientos estocásticos, lo que es ilustrado con uno de los sistemas más simples de estudiar con MC: el modelo de Ising ferromagnético.

4.3. Modelo de Ising ferromagnético: Introducción

Este consiste de partículas situadas en un látice y que interaccionan con sus vecinos más cercanos de forma que lo único que son capaces de hacer es cambiar de spin [3]. En la figura a continuación se bosqueja el sistema para un látice cuadrado bidimensional.

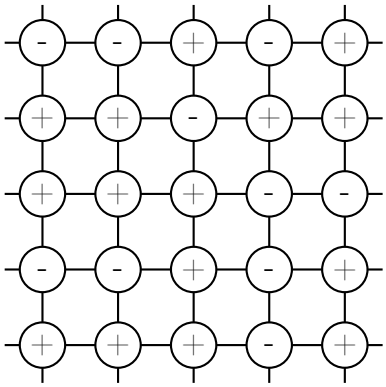

Figura 4.1: Esquema del modelo de Ising bidimensional de 5 X 5

El Hamiltoniano de este sistema es,

$$\mathcal{H} = -\mathcal{F}\sum_{i,j}\sigma_i\sigma_j - H\sum_i \sigma_i \tag{4.8}$$

donde $\mathcal{F} > 0$ es una constante de interacción, H es el campo magnético y $\sigma_i = \pm 1$ representa el spin de la partícula i-ésima. El espacio de fase de este sistema esta formado por las distintas configuraciones de spin de las partículas y se distribuye de acuerdo a Ecuación 4.5. Una propiedad de interés en este sistema es la magnetización $M = \sum_i \sigma_i$.

En principio, si se quiere resolver el problema en el espacio bidimensional, es factible situar partículas en puntos fijos de un látice de L X L y comenzar a cambiar los spin de estas aleatoriamente, utilizando para ello un generador de números aleatorios. El problema con ese

experimento es que el espacio de fase realmente obedece a una densidad de probabilidad bien definida por la Ecuación 4.5, en tanto que las configuraciones que somos capaces de hacer pudieran ser aquellas de menor probabilidad, o bien de otra densidad distinta, y por tanto no contribuir con certeza al cálculo de las propiedades del sistema (i.e., magnetización).

Para sortear este problema es conveniente utilizar el método de Metrópolis, que consiste en generar configuraciones del espacio de fase utilizando una cadena de Markov. Esto es, cada configuración se va generando a partir de la anterior de acuerdo a una probabilidad de transición W_{ij}. Con ello no todas las configuraciones que pudiéramos construir aleatoriamente son aceptadas, si no sólo aquellas que representan efectivamente al sistema. Una expresión adecuada para la probabilidad de transición entre estados, propuesta por Metropolis et al., es,

$$W_{ij} = \begin{array}{ll} exp(-\Delta\mathcal{H}/k_BT), & \Delta\mathcal{H} > 0 \\ 1, & \Delta\mathcal{H} < 0 \end{array} \tag{4.9}$$

donde es claro que se aceptan todas las configuraciones tales que su energía sea menor a la configuración precedente. En caso contrario, la nueva configuración es aceptada si un número aleatorio uniformemente distribuido resulta menor a W_{ij}.

Con el criterio anterior se asegura que las distintas configuraciones del sistema, generadas aleatoriamente y aceptadas de acuerdo a la probabilidad de transición, se distribuyen con probabilidad proporcional a la Ecuación 4.5. Por lo tanto, cualquier propiedad $\mathcal{A}$ del sistema puede ser estimada como,

$$\langle\mathcal{A}\rangle = \sum_i \frac{n_i}{\tau} A_i \tag{4.10}$$

donde n_i es la cantidad de configuraciones i idénticas, τ el número total de configuraciones y $\mathcal{A}_i$ el valor de la propiedad $\mathcal{A}$ en la configuración i.

4.4. Modelo de Ising ferromagnético: Simulación

La simulación de este sistema consiste en los siguientes pasos:

1. **Situar las partículas en un látice de L X L.** Las partículas en el látice pueden ser representadas como una matriz cuadrada $\mathbf{z}(i,j)$ de tamaño L X L que contiene los valores de spin σ_k de la partícula k-ésima con posición (i,j), de tal forma que $\sigma_1 = \mathbf{z}(1,1)$, $\sigma_2 = \mathbf{z}(1,2)$, $\sigma_3 = \mathbf{z}(1,3)$, etc.

2. **Asignar una configuración inicial de spin a cada partícula**. Para este sistema se proponen dos tipos de configuraciones.

   Tipo I: completamente magnetizada con $\sigma_k = 1$ para todo k.

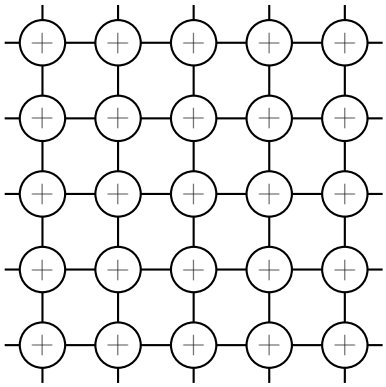

Figura 4.2: Ejemplo de configuración inicial completamente magnetizada.

   Tipo II: Aleatoria de baja magnetización.

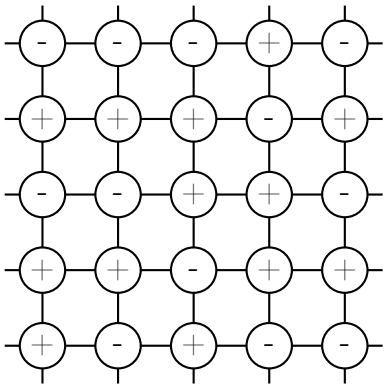

Figura 4.3: Ejemplo de configuración inicial aleatoria de baja magnetización.

3. **Calcular la energía del sistema según** $\mathcal{H}_1 = -\mathcal{F}\sum_{i,j}\sigma_i\sigma_j - H\sum_i \sigma_i$. El parámetro de interacción $\mathcal{F}$ es un valor constante que depende del sistema real que se quiera modelar por con el modelo de Ising, lo importante es el parámetro definido como $j = J/k_BT$. Dado j para cualquier valor fijo de $\mathcal{F}$ se calcula la temperatura.

4. **Cambiar el spin de una partícula elegida aleatoriamente**. Para ello se utiliza un RNG que definirá el valor de i y j. Luego se cambia el spin como $\mathbf{z}(i,j) = -1 \times \mathbf{z}(i,j)$. Calcular la energía del sistema según $\mathcal{H}_2 = -\mathcal{F}\sum_{i,j}\sigma_i\sigma_j - H\sum_i \sigma_i$,

5. **Definir si aceptar o no la nueva configuración**. Si $\mathcal{H}_2 - \mathcal{H}_1 < 0$, hacer $W = 1$. Si $\mathcal{H}_2 - \mathcal{H}_1 > 0$ calcular $W = exp(-\Delta\mathcal{H}/k_BT)$.

   Generar un número pseudo-aleatorio r, luego si $r < W$ la nueva configuración es aceptada y se asigna $\mathcal{H}_1 = \mathcal{H}_2$. Volver al paso 4. Si por el contrario la configuración es rechazada, volver el spin a su valor original y volver al paso 4. Se rechazan las configuraciones de igual energía.

Con los pasos mencionados se ha escrito un programa en FORTRAN 90, del cual sólo se ha calculado la magnetización del sistema. Es sabido que el modelo bidimensional presenta transición de fase en $j \approx 0.44$ para $H = 0$. Cuando la temperatura disminuye (j aumenta) el sistema tiende a magnetizarse en forma espontánea, por el contrario, a alta temperatura predomina la aleatoriedad de los spin por lo que el sistema tiende a desmagnetizarse.

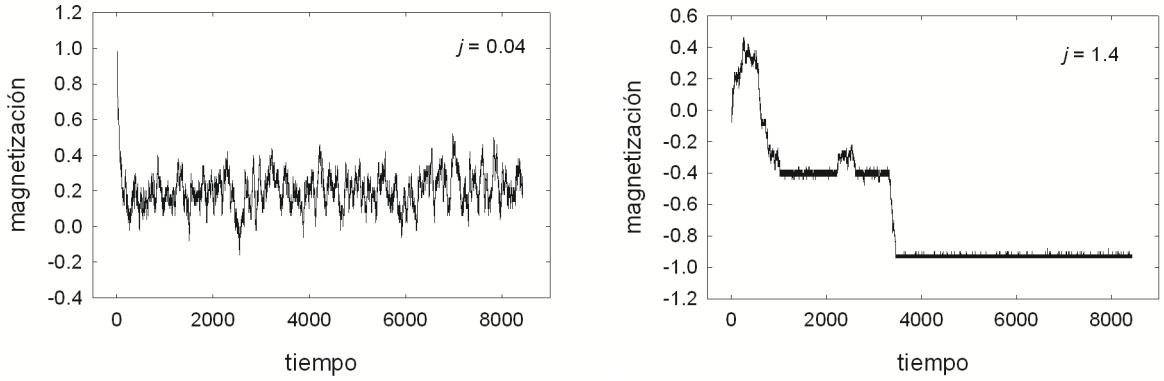

Figura 4.4: Magentización para las siguientes simulaciones: $j = 0.04$ configuración inicial magnetizada, $j = 1.4$ configuración inicial desmagnetizada. Sistema de $L = 10$.

En la Figura 4.4 se presentan los resultados de magnetización en el tiempo. Nótese que el tiempo se entiende aquí no en el sentido temporal, si no en la sucesión de cadenas de Markov aceptadas. Las simulaciones han sido sobre un total de 1000000 de configuraciones probadas de las cuales solo una fracción han sido aceptadas. Para el caso $j = 0.04$ el 90 % de las configuraciones probadas fueron aceptadas, se explica esto en la alta temperatura del sistema, lo

que aumenta el valor de la probabilidad de transición W. Esta simulación fue iniciada con una configuración totalmente magnetizada y, de acuerdo a las restricciones del sistema (alta temperatura), hay una clara tendencia a desmagnetizar. En cambio, la simulación con $j = 1.4$ se inició con configuración inicial desmagnetizada y el sistema espontáneamente tiende a magnetizarse. En este caso las configuraciones aceptadas alcanzaron tan solo el 2.5 % del total.

Este ejemplo refleja que al momento de utilizar el método de Monte Carlo para un sistema físico, que tiene descripción en la mecánica estadística, no es suficiente con generar configuraciones aleatoriamente. Es necesario que las configuraciones sean consistentes con la densidad de probabilidad del espacio de fase. Por ello el método de Metrópolis aquí estudiado es el adecuado para este tipo de estudios.

Bibliografía

[1] Marsaglia G. A current view of random number generators. *Computer Science and Statistics*, 1984.

[2] Coddington P. Random number generators for parallel computers. 1997.

[3] Landau D. and Binder K. *A guide to Monte Carlo simulations in statical physics*. Cambridge, second edition, 2000.

[4] Vehmanen L. Saarinen J., Tomberg J. and Kaski K. VLSI implementation of tausworthe random number generator for parallel processing. *IEE PROCEEDING-E*, 138(3):138, 1991.

[5] Gould and Tobochnik. *An introduction to computer simulation methods*, volume 2. Addison Wesley, 1988.

[6] Berend.S. Frenkel.D. *Understanding Molecular Simulation*. Academic Press, 2002.