Informe de Métodos Numéricos
Tarea 8: Métodos Aleatorios

Leonardo Leiva

1. Resumen

En este informe se desarrollarán los métodos de Monte Carlo para resolver integrales y de Metrópolis para obtener muestras aleatorias a partir de una distribución de probabilidad. Ambos métodos se basan en aleatoriedad, por lo que se pondrá énfasis en la utilidad de estas herramientas y se comprobará la utilidad práctica que tienen.

2. Introducción

En este informe se mostrarán algunos procedimientos numéricos para resolver problemas basados en la aleatoriedad. Antes que nada es necesario recalcar que en métodos computacionales no existe el azar absoluto ya que los computadores son deterministas. Es por esto que existen diversos algoritmos que simulan aleatoriedad pero no completamente. Deben cumplir algunos criterios especificos que no serán detallados en este informe.
A pesar de que no exista la aleatoriedad como tal, el uso de estas herramientas es muy util para resolver problemas. En esta ocasión se aplicarán los métodos de Monte Carlo para calcular integrales y el Método de Metropolis para generar una secuencia de valores aleatorios "Xçon una distribución "$W(x)$".
No se recomienda intentar implementar un método aleatorio, si no que usar aquellos que ya están implementados.

2.1. Método de Monte Carlo

Los métodos de Monte Carlo se basan en el uso de números pseudo-aleatorios para diversos cálculos. Para el caso particular de las integrales se busca resolver el siguiente problema:

$$I \equiv \int_a^b f(x)dx \tag{1}$$

Para ello se toma una muestra aleatoria de valores $\{x_i\}_{i=1,...,N}$ mediante algún método aleatorio (en este caso 'numpy.random') con una distribución $W(x)$. Por simpleza y aplicabilidad al problema planteado se usa una distribución uniforme: $W(x) = \frac{1}{b-c} = cte.$
Con estos puntos x_i generados aleatoriamente se evaluará la función y se promediará sobre los valores obtenidos $f(x_i)(b-a)$. Corresponde a una aproximación muy sencilla y con errores

más grandes que otros métodos vistos anteriormente (como el del trapecio), pero cuya simpleza en la implementación compensa la falta de precisión para regiones de integración en varias dimensiones.
De esta manera:

$$I \approx \frac{(b-a)}{N}\sum_{i=1}^{N} f_i + \frac{(b-a)}{\sqrt{N}}\sqrt{\frac{1}{N}\sum_{i=1}^{N} f_j^2 + (\frac{1}{N}\sum_{i=1}^{N} f_j)^2)} \tag{2}$$

Donde $f_i = f(x_i)$ y el segundo término es la desviación estándar, la cual indica el error de la medición.
La generalización a más dimensiones corresponde a:

$$\int_V f(\vec{r})dV \approx V\langle f\rangle \pm V\sqrt{\frac{\langle f^2\rangle - \langle f\rangle^2}{N}} \tag{3}$$

Como se dijo anteriormente, su utilidad práctica está para el cálculo de integrales en varias dimensiones y, usualmente, en dominios dificiles de parametrizar.
Algo que se debe tener en cuenta es que se prefiere escoger un volumen mínimo que contenga toda la sección a integrar, porque los puntos que pueden quedar fuera de la función serán evaluados en $f(x_i) = 0$, pero aumentarán la magnitud del error asociado al cálculo. También cabe mencionar que el error disminuye con $\sqrt{N}$, por lo que se necesitarán muchos más puntos para lograr una mejora considerable.

2.2. Algoritmo de Metrópolis

El algoritmo de metrópolis corresponde a una estrategia para generar una secuencia de valores x que distribuyen como $W(x)$ para cualquier distribución W [1]. Inicialmente se creó para generar muestras de la distribución de Boltzmann, pero luego se generalizó y hoy en día se utiliza para muchos problemas de la mecánica estadística.

El algoritmo requiere una semilla x_0 para inicializarse en el ciclo:
1) Primero se genera un x_p a partir de un x_n (para la primera iteración, $n = 0$) Debe ser simétrico, de manera que sea igualmente probable generar x_p a partir de x_n que visceversa.
2) Se usa un criterio para escoger entre x_n y x_p el cual consiste en: Si $\frac{W(x_p)}{W(x_n)} > r$, entonces $x_{n+1} = x_p$. Si no, $x_{n+1} = x_n$, donde r es un número al azar entre $[0, 1]$.
Con esto se puede observar que si $W(x_p) > W(x_n)$, x_p siempre se acepta, mientras que, para el caso contrario, existe una probabilidad de que se acepte y otra que no. Se nota que se pueden obtener números repetidos, lo cual, estadísticamente es válido y necesario.

3. Metodología

Para la resolución de dos problemas planteados se usará el algoritmo de Monte Carlo en una primera parte, y el algoritmo en la segunda.

3.1. Pregunta 1: Integral de Montecarlo

Se busca el centro de masas de un cuerpo dado por la intersección de un toro y un cilindro cuyas parametrizaciones son las siguientes:

$$\text{Toro}: \ z^2 + \left(\sqrt{x^2+y^2} - 3\right)^2 \leq 1 \qquad \text{Cilindro}: \ (x-2)^2 + z^2 \leq 1 \tag{4}$$

Donde la densidad corresponde a:

$$\rho(x,y,z) = 0{,}5 * (x^2 + y^2 + z^2) \tag{5}$$

El problema a resolver es:

$$R_{cm} = \frac{\int r\rho(x,y,z)dV}{\int \rho(x,y,z)dV} \tag{6}$$

Para comenzar, se escoge un volumen de integración que las condiciones de (4) simultaneamente. Priero, analizando para el cilindro se puede ver que no hay limitaciones para y. Para el caso de z se observa que $z \in [-1,1]$ cumple la ecuación. El caso de $x \in [1,3]$ también cumple la condición.
Para analizar la condición del toro observamos que pasa en z. Se nota que la misma condición para el cilindro se cumple para el toro. Para y observamos que el mayor intervalo se tiene cuando z^2 vale 0. De esta forma $y \in [-4,4]$ a la vez que $x \in [-4,4]$. Uniendo la condición anterior para el cilindro, el volumen que será considerado es $x \in [1,3]$, $y \in [-4,4]$ y $z \in [-1,1]$, donde se usará $V = dxdydz$.
Con el volumen definido se escogen N puntos con una distribución uniforme en el volumen, se verifica que esté dentro o no del cuerpo (viendo si cumple la condición (4) de estar dentro del cilindro y toro). Se hace una iteración en la que, si se cumple la condición de estar en el cuerpo, se calcula la densidad para ese punto. Luego se van sumando los valores para las integrales del numerador y denominador de (6). Para cada integral se calcula el valor y el error asociado y se hace la división para tener cada coordenada. Se hace el tratamiento de errores apropiado:

$$c = \langle c \rangle \pm \Delta c = \frac{\langle a \rangle}{\langle b \rangle} \pm \frac{\langle a \rangle}{\langle b \rangle} \sqrt{(\frac{\Delta a}{\langle a \rangle})^2 + (\frac{\Delta b}{\langle b \rangle})^2} \tag{7}$$

Se realiza la iteración para $N = 10^7$ para reducir el error. EL código usado está en 'parte1.py'.

3.2. Pregunta 2: Distrubución por Algoritmo de Metropolis

Se busca obtener una muestra aleatoria a partir de una distribución dada por:

$$W(x) = 3{,}5 \times \exp\left(\frac{-(x-3)^2}{3}\right) + 2 \times \exp\left(\frac{-(x+1{,}5)^2}{0{,}5}\right) \tag{8}$$

Notar que no está normalizada. Para ello se calculó con ayuda de wolframalpha[2] y su valor es:

$$\int_{-\infty}^{\infty} W(x) \approx 13{,}2516 \tag{9}$$

Luego con $N = 10^7$ se realiza la iteración. Dado un x_0 aleatorio en una distribución uniforme entre $(-8, 8)$, donde se concentra mayormente la distribución de $W(x)$. Se debe mencionar que se fija una semilla para repetir el resultado y poder analizarlo.
Para cada valor de x_n utiliza la secuencia mencionada para el algoritmo de metrópolis y evaluar la condición. La manera de generar x_p para la condición es con $x_p = x_n + \delta * r$, donde r es un valor aleatorio con distribución uniforme entre $[-1, 1]$ y δ es un valor tal que el algoritmo acepte alrededor del $50\,\%$ de los x_p. Se implementa una función que busca el δ óptimo antes de iniciar la iteración. El algoritmo usado está en 'parte2.py'.

4. Resultados y Análisis

4.1. Para la integral de Monte Carlo

Para el cálculo del centro de masas (6) se obtuvo:

$$X_{cm} = 2{,}07995 \pm 0{,}00109 \tag{10}$$

$$Y_{cm} = 0{,}0005 \pm 0{,}0012 \tag{11}$$

$$Z_{cm} = -8{,}81 * 10^{-5} \pm -0{,}00021 \tag{12}$$

Se probaron varios valores de N para observar el comportamiento de los errores asociados. Se observa que los errores disminuyen a medida que aumenta el orden de N, pero cada vez es más dificil. Esto se condice con que el error del método disminuye con respecto a $\sqrt{N}$. De cualquier forma, el error es bastante pequeño (orden 10^{-3}) aprox con respecto al orden de magnitud del problema (orden 1). De la misma forma se observan variaciones considerables antes de fijar la semilla para n pequeños, mientras que para los calculos finales, la variación es muy pequeña.
Se entiende que el algoritmo tiene asociada una imprecisión importante por construcción y que para reducir esto el gasto computacional es muy grande comparado con la ganancia, pero se observa la simpleza del algoritmo en comparación con otros métodos. Esto compensa el punto anterior, que en la práctica, debió significar parametrizar la intersección de ambos sólidos en vez de aplicar una condición que verifique su presencia en ellos. Es probable que se pueda hacer, pero refleja la utilidad del algoritmo para otros problemas mucho más complicados.

4.2. Para el Algoritmo de Metrópolis

Se probaron diversos valores de N para ver como evolucionaba la forma del histograma. Se observa una notable mejora para mayores valores. A final de cuentas se obtiene un gráfico muy parecido a la función pedida. En la figura (1) se observa lo obtenido. Además se grafica en verde la función original. Se observa un notable parecido entre el histograma y la función original.

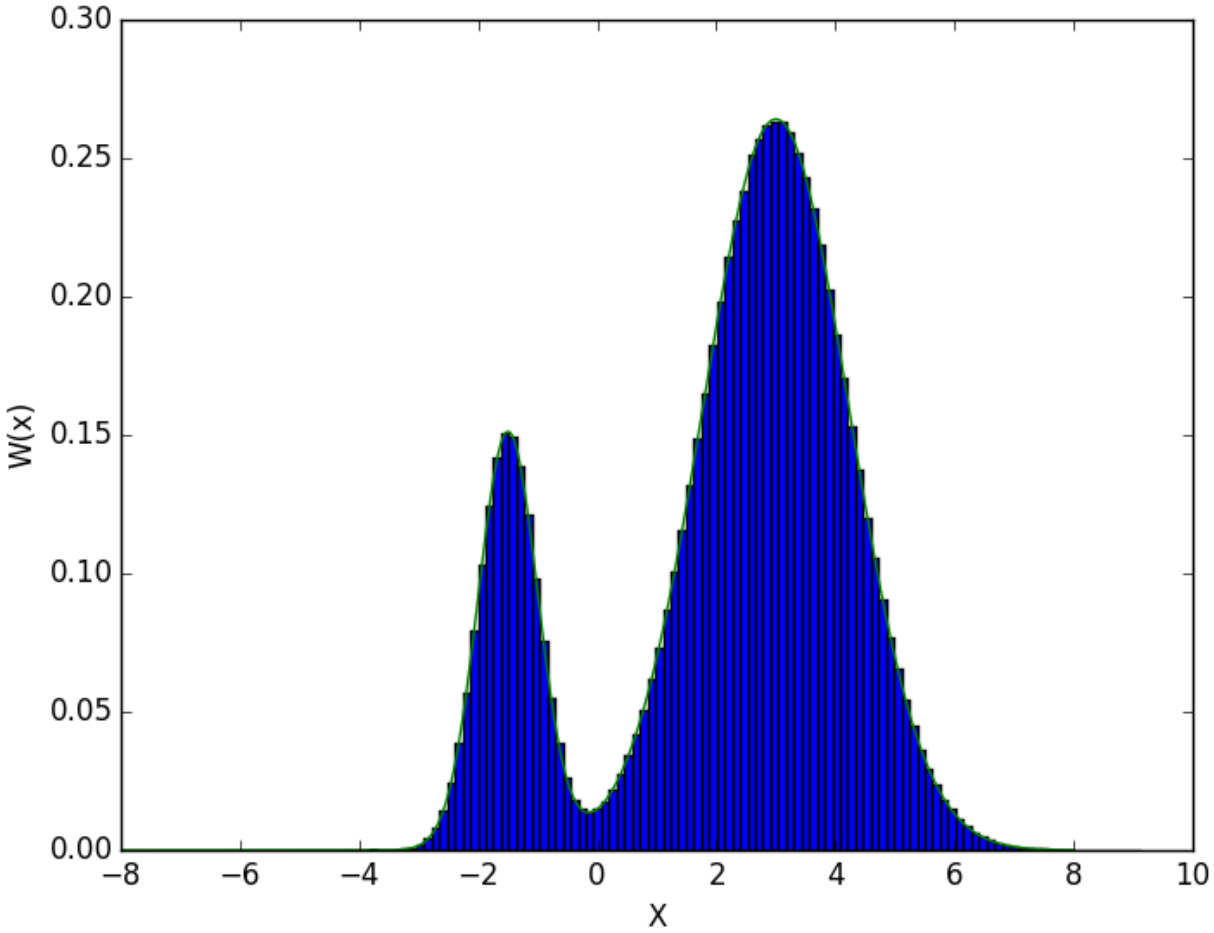

Figura 1: Histograma con la distribución sintética obtenida a partir del algoritmo de Metrópolis. Se alcanza a notar levemente en verde la curva correspondiente a la distribución que se busca alcanzar

Cabe mencionar que se escogieron 100 bins para reducir el ruido en la muestra. Debería obtenerse un resultado similar si se aumentan los bins siempre y cuando no se pase cierto umbral: es esperable que para muchos bins la función no quede bien representada. Se entiende que esto es una consecuencia del tipo de datos tratados y que para obtener una muestra exacta se deberían hacer infinitas iteraciones, lo cual es, computacionalmente imposible.
Resulta de este algoritmo una manera favorable de generar datos .ªleatorios"para una distribución. Se tiene un código bastante sencillo y compacto en el cual basta reescribir la ecuación de la distribución para poder representar otras mucho más complicadas. Las posibilidades que se generan desde acá son muy variadas, sobre todo ahora que se ha descubierto la aleatoriedad de la naturaleza vía mecánica cuántica (un mundo desconocido hasta ahora, revelado por el ramo Física Moderna). Permite tener una idea aproximada de la solución de problemas cuánticos que manejan probabilidades. Además, por supuesto, es una herramienta

útil para simulaciones experimentales en mecánica estadística.

5. Sobre el Bonus

Se pedía obtener el promedio y error asociado de los valores obtenidos para el histograma de la parte 2, realizando el algoritmo para 100 valores inciales diferentes o semillas distintas. Se escoge como punto de partida una semilla diferente y una posición inicial diferente. Se usa una la función 'linspace' para escoger ambas. Para el caso de la semilla se sabe que esto no tiene importancia (se tomaron valores del 1 hasta el 100 de 1 en 1), pero para el caso de las posiciones se propone escoger una distribución aleatoria. Por tiempo no se alcanza a hacer la modificación dado que el programa tarda mucho tiempo en correr y ver que los datos cambian en tales situaciones (aleatoria y no aleatoria) no se alcanza a hacer.
A partir de la semilla y el valor inicial x_0 escogido se realiza la misma iteración que en la parte 2. Se propone eliminar el cálculo del δ óptimo. No se hizo porque la idea surgió después de empezar a correr el programa.
Se presentaron más problemas para tener una forma adecuada de graficar que para el algoritmo en sí, ya que estaba casi hecho. Se escoge una manera diferente de graficar, porque para los casos anteriores bastaba tener el arreglo de x_n para hacer el histograma. En este caso se necesitaba la cantidad de x_n que caían en cada bin para poder graficar. Se implementa una función que calcula este numero para cada iteración y luego hace un promedio con todos los valores para cada bin, junto con su respectiva desviación. No está claro si alcanzará a terminar de correr el algoritmo antes del tiempo de entrega de la tarea, por lo que se hará un análisis corto en base a los resultados obtenidos de los tests y una estimación del resultado.
Para implementar el código se usó 10 valores iniciales de semilla y posición y se iteró para $N = 1000$. Se obtuvo un gráfico con errores considerables, pero a pesar de esto, se alcanzaba a asemejar a la forma de la función. Habían barras de error que, incluso, iban más abajo que 0 en valor mínimo. Para comprobar que el programa corriera para 100 datos iniciales se corrió, pero esta vez para $N = 100$ en vista del tiempo y para iniciar lo más pronto posible la iteración final. Se observaron barras de error mucho más grandes debido a la disminución del parámetro N.
Para el caso final se espera un error casi imperceptible en cada bin debido a la gran precisión del algoritmo observada en la figura (1): Al ser muy parecido a la distribución, se espera que cada iteración pase lo mismo, logrando variaciones pequeñas en comparación al orden de magnitud del problema.

6. Conclusiones

Se concluye que los métodos aleatorios tienen gran aplicabilidad en diversos problemas de la física (y otras áreas también), en el sentido de que permiten calcular problemas muy complejos con algoritmos bastante sencillos (Integrales de Monte Carlo) y la obtención de

muestras aleatorias para hacer simulaciones (Metropolis).
Si bien ambos algoritmos tienen desventajas tanto en su aleatoriedad (dado que tiene un error muy grande) como en que en la práctica no son del todo aleatorios (no pueden reflejar plenamente la esencia cuántica de la naturaleza), sus ventajas son sustanciales a la hora de resolver ciertos problemas: Por un lado, las integrales de Monte Carlo pueden ser un buen punto de partida para un cálculo complejo, pero que puede ser parametrizado y resuelto por algoritmos más precisos, mientras que también pueden ser el cálculo final para el caso de regiones tremendamente complejas y su precisión siempre puede ser mejorada si se tiene suficiente paciencia.
En especial llama la atención como una metodología basada en la aleatoriedad puede obtener resultados tan robustos. Es un área muy interesante en la cual profundizar al considerar que, en esencia, la naturaleza se comporta así, con cierta incertidumbre. Claramente para sistemas suficientemente macroscópicos, efectos cuánticos son despreciables, pero en dichos sistemas también se observan errores y distribuciones en cada fenómeno debido a factores que son dificiles de controlar durante un experimento (desde temperatura, presión, etc, hasta las herramientas usadas para medir). Conocer esto y como simular este comportamiento puede resultar muy ventajoso para interpretar datos experimentales.
Se concluye, finalmente que los métodos aleatorios son parte importante de la naturaleza y poder modelarlos permite entenderlos mejor y controlar el azar. A un costo importante de tiempo y recursos computacionales se pueden resolver muchos problemas. Queda pendiente, en general, mejorar la velocidad de los algoritmos y obtener algo más cercano a la verdadera aleatoriedad. Son metas muy lejanas, incluso pueden ser imposibles, pero se pueden lograr muchos avances con intentarlo.

Referencias

[1] Métodos Computacionales en Física - Patricio Cordero. Versión 14 de julio 2009.
[2] Para calcular la integral (8)