Métodos Numéricos para la Ciencia e Ingeniería
Informe Tarea 08

Benjamín Guerrero
18.862.370-0
17 de Noviembre, 2015

1. Pregunta 1

1.1. Introducción

Se pide estimar la posición de un sólido descrito por la intersección de los siguientes cuerpos:

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

O sea, un toro centrado en el origen de radio mayor 4 y radio menor 2, y un cilindro de altura infinita en y, generado por una circunferencia en el plano (x, z), de radio 1, y centrado en $x = 2, z = 0$.

La densidad del sólido se define de la forma siguiente:

$$\rho(x, y, z) = 0.5 * (x^2 + y^2 + z^2)$$

En este problema, se pide usar la integración de Monte Carlo.

1.2. Procedimiento

Notemos que el sólido descrito va a estar encerrado por un prisma cuadrado cuya base tiene lado 2, y una altura 8. Definiendo el volumen de esa forma, podemos generar puntos dentro de ese volumen, y ver si cumplen lo pedido.

Con esto en mente, se crea un programa llamado *MC.py*, que realizará lo pedido.

Primero, iniciemos el programa definiendo las funciones *dentro_cilindro* y *dentro_toro*, que comprueban si los puntos que se generan están dentro del sólido pedido.

Ahora, hay que ver que método de MC se va a usar. Aquí, se va a usar MC1. (Si este método es efectivo o no va a verse una vez se ejecute el programa).

Ahora, se define N_p como el número de puntos que se van a generar. En este caso, serán 100000. Con esto, se puede generar la función *generar_puntos*, que toma una

semilla y un N, y crea N puntos con el método *numpy.random.uniform*, que genera una distribución uniforme. Se usa el prisma como condición de borde.

Ahora, con esto, se puede definir la función *calculo_centro_masa*, que calcula el centro de masa de N_p puntos generados aleatoriamente. Primero, se generan los puntos (usando *generar_puntos*), y luego se abre un diccionario llamado P_adentro. Luego, se hacen N_p iteraciones para comprobar si el punto generado se encuentra dentro del sólido o no. Si se encuentra dentro, se calcula la densidad del punto (usando la ecuación dada), y se guarda en el diccionario P_adentro. Al ser de volumen nulo, calcular la densidad de un punto es equivalente a calcular su masa.

Luego, se inician los parámetros *suma_x*, *suma_y*, *suma_z* y *M* (masa). Luego, se realiza una iteración en cada definición guardada en P_adentro, y se calcula la suma de las masas en las coordenadas (x, y, z) (como *suma_x*, *suma_y*, *suma_z*), y la masa total. Finalmente, dividiendo la masa de cada coordenada por la masa total, se obtienen las coordenadas del centro de masa.

Ahora, la posición del centro de masa va a depender de los puntos generados. Por eso se programa lo anterior como una función, ya que va a tener que hacerse varias veces con diferentes semillas.

El main del programa se define de la siguiente forma:

Primero, se inician x, y, y z como arrays de numpy con N_c ceros, en donde N_c son las veces que se calculará el centro de masa usando la función anterior (En este caso, N_c es 100). Luego, esos arrays se rellenan con las coordenadas de cada centro de masa calculado. Para cambiar las semillas, se define la semilla inicial como 30, y se le suma 1 a la semilla con cada iteración.

Luego, se calcula el promedio de los valores y la desviación estándar.

1.3. Resultados

Las coordenadas del centro de masa, según el programa son, aproximadamente:

$$x_{CM} = 2.08 \pm 0.003$$

$$y_{CM} = -0.001 \pm 0.013$$

$$z_{CM} = -0.00004 \pm 0.002$$

Aquí, el número a la izquierda del signo es el promedio, y el número a la derecha es la desviación estándar.

2. Pregunta 2

2.1. Introducción

Se quiere obtener una muestra aleatoria de números con la distribución no normalizada:

$$W(x) = 3.5 \times \exp\left(\frac{-(x-3)^2}{3}\right) + 2 \times \exp\left(\frac{-(x+1.5)^2}{0.5}\right)$$

Se pide usar el algoritmo de Metrópolis con una distribución $x_p = x_n + \delta * r$, en el que r es una variable aleatoria de distribución uniforme $U(-1,1)$. La variable δ es una constante cuya definición queda a elección del usuario, pero debe aceptar aproximadamente el 50% de las proposiciones.

Se debe generar una muestra de 10 millones de puntos, y se debe graficar $W(x)$ y un histograma de las variables aleatorias apropiadamente normalizadas.

2.2. Procedimiento

Para resolver el problema, se va a crear un programa llamado *Metropolis.py*. En este, primero se van a definir 2 funciones: $W(x)$ (que retorna lo definido en el enunciado), y *proposición*, que toma un valor x_n, genera un valor aleatorio r (uniforme [-1,1]), y retorna $x_p = x_n + \delta * r$.

Ahora, se define *N* como 10 millones. Se define x como un array de $N + 1$ ceros (porque Metrópolis usa el valor $n + 1$, y se quiere evitar un error). Se inician los valores n_a (cantidad de valores aceptados, para ver si el delta funciona), y n (que sirve como contador y será el valor a iterar).

Ahora, se crea un *while loop* que aplicará el algoritmo de Metrópolis. En este, se genera *s*, un valor aleatorio uniforme entre 0 y 1, y se calcula x_p a partir de la función *proposición*, tomando $x[n]$.

Luego, si $\frac{W(x_p)}{W(x[n])} > s$, el valor x_p es aceptado, se define $x[n+1] = x_p$, y se le suma 1 a n_a. En caso contrario, se define $x[n+1] = x[n]$.

Finalmente, se le suma 1 a *n* y se reinicia el loop. Esto sigue hasta que $n = N = 10^7$.

Ahora se debe normalizar $W(x)$, o sea, encontrar una función W' tal que:

$$\int_{-\infty}^{+\infty} \frac{W(x)}{W'}\, dx = 1$$

Haciendo los cálculos, se puede definir W' como:

$$W' = \sqrt{\pi} * (3.5 * \sqrt{3} + 2 * \sqrt{0.5})$$

Hecho esto, se define x_g como 10^7 espacios entre -5 y 10. Luego, se usan los métodos incluidos en *matplotlib.pyplot* para generar el histograma y el gráfico pedidos (Se usa un bin de 100 para el histograma).

2.3. Resultados

Al definir $\delta = 3$, se aceptan 5625059 puntos, o sea, un poco más del 56%. O sea, el delta es bueno. (se probó con $\delta = 5$, pero no se aprobaron 50% de los puntos)

Con esto, se genera el siguiente gráfico:

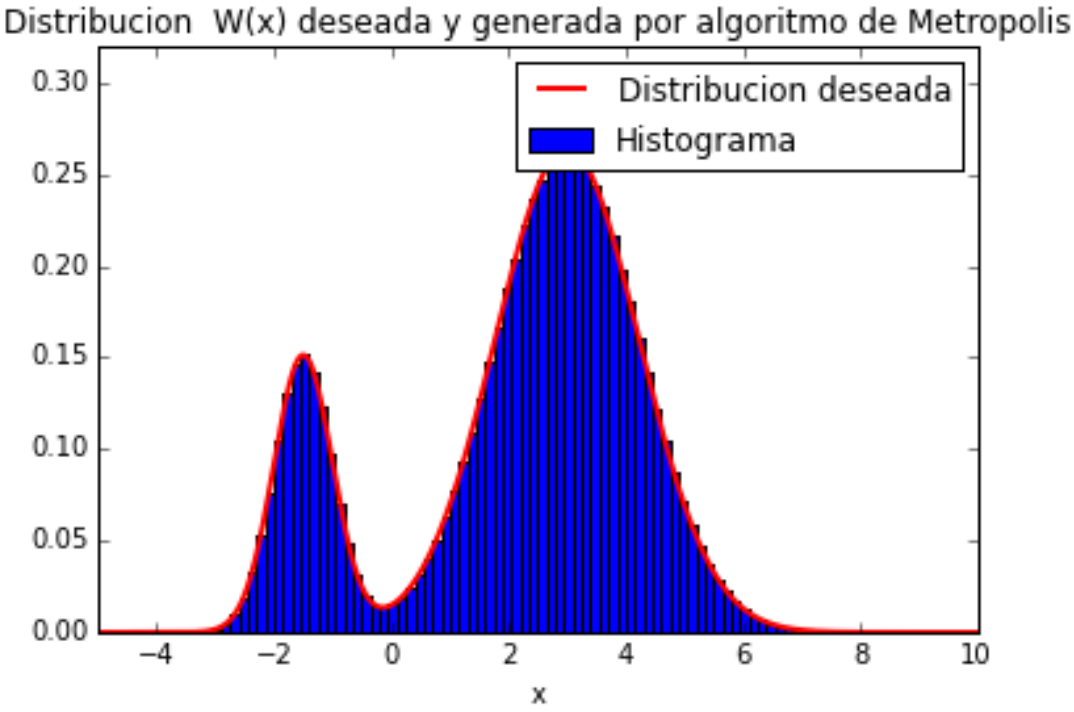

Fig 1: Distribución W(x)

3. Conclusiones

Al ver los resultados de la P1, se puede deducir que el centro de masa del sólido pedido se encuentra más o menos en el punto (2, 0, 0), es decir, en el centro del cilindro. Esto se debe a que la función de distribución de puntos se "balancea" con la función de densidad (hay más puntos cerca del origen, pero los puntos más alejados del origen pesan más).

Ahora, el margen de error se debe a 2 cosas. Primero, es el hecho de que MC funciona usando variables aleatorias. Segundo es el hecho de usar MC1 en una función que varía en el espacio. La variación no es tan significativa, pero es suficiente para generar variaciones. El margen de error hubiera sido menor si la densidad hubiera sido constante en el espacio.

Así, si bien usar Monte Carlo 1 generó algunas inexactitudes, también da una buena aproximación del centro de masa.

Al ver los resultados de la P2, se ve que el delta tiene que ser menor a 5 para que se acepten al menos el 50% de los puntos (se deduce que si delta es igual a 4, se van a aceptar aproximadamente el 50% de los puntos, y también sirve). Viendo el gráfico, se aprecia que el histograma generado con el algoritmo de Metrópolis concuerda, más o menos, con el gráfico que genera la distribución normalizada.

Así, usar el algoritmo de Metrópolis sirve para generar una buena aproximación de la distribución normalizada.

Una última nota, es que, debido a la enorme cantidad de puntos generados, ambos programas tardan varios minutos en completarse. En específico, generar 1 histograma usando *Metropolis.py* tarda al menos 5 minutos.