Métodos matemáticos para la ciencia e Ingeniería: Método de integración de Monte Carlo y Algoritmo de Metrópolis

Fernanda Pérez *(rut: 18.769.232-6)*

17 de Noviembre, 2015

1 Pregunta1

1.1 Introducción

Se busca estimar la posición del centro de masa de un sólido descrito por la intersección de un toro y un cilindro (ver Figura 1) dados por las siguientes ecuaciones:

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

$$\text{Cilindro}: \ (x-2)^2 + z^2 \leq 1 \tag{2}$$

Con la densidad del sólido variando de la siguiente forma:

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

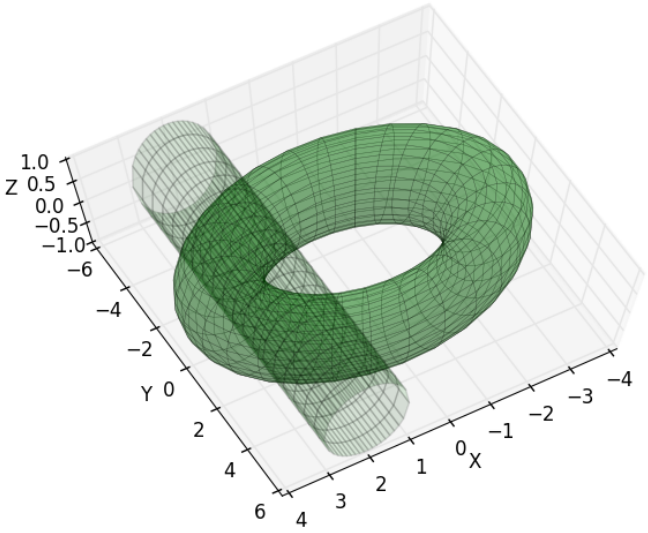

Figura 1: Gráfica de las superficies del toro y cilindro parametrizados según las ecuaciones 1 y 2, tomando la igualdad.

1.2 Procedimiento

$$I = \int_a^b f(x)dx \approx \frac{(b-a)}{N}\sum_{j=1}^{N} f_j \pm \frac{(b-a)}{\sqrt{N}} \cdot \sqrt{\frac{1}{N}\sum_{j=1}^{N} {f_j}^2 - (\frac{1}{N}\sum_{j=1}^{N} f_j)^2} \tag{3}$$

Se utiliza el método de integración simple tridimensional de Monte Carlo análogo al unidimensional mostrado en la Ecuación 1, donde $f_j = f(x_j)$ con x_j una muestra aleatoria uniforme entre a y b, para calcular las siguientes integrales:

$$I_1 = \int \rho(x,y,z)\,dx\,dy\,dz \tag{4}$$

$$I_2 = \int x\rho(x,y,z)\,dx\,dy\,dz \tag{5}$$

$$I_3 = \int y\rho(x,y,z)\,dx\,dy\,dz \tag{6}$$

$$I_4 = \int z\rho(x,y,z)\,dx\,dy\,dz \tag{7}$$

Para obtener números random se usa *numpy.random.uniform* en python y se escoge como semilla *212*. Se utiliza $N = 10^5$.
Se escoge el siguiente volumen de integración:

- $x \in [1, 3]$
- $y \in [-4, 4]$
- $z \in [-1, 1]$

Una vez obtenidas las integrales anteriormente mostradas, es posible calcular las coordenadas del centro de masa y sus respectivos errores (siguiendo la regla de propagación de errores en la división) de la siguiente manera:

$$X_{CM} = \frac{I_2}{I_1}, \quad Y_{CM} = \frac{I_3}{I_1}, \quad Z_{CM} = \frac{I_4}{I_1} \tag{8}$$

1.3 Resultados

Realizando el procedimiento descrito en la sección anterior se obtiene lo siguiente:

- $X_{CM} = 2.0803 \pm 0.01$
- $Y_{CM} = 0.0350 \pm 0.011$
- $Z_{CM} = -0.0002 \pm 0.0021$

1.4 Conclusiones

Las coordenadas obtenidas para el centro de masa tienen sentido dada la simetra que se observa en la Figura 1 y la densidad en cada punto del volumen. Se obtienen errores del orden de 10^{-2} y 10^{-3}.

2 Pregunta 2

2.1 Introducción

Se busca obtener una muestra aleatoria de números con la distribución no normalizada dada por la Ecuación 9, utilizando el algoritmo de Metrópolis con una ditribución $xp = xn + \delta \cdot r$, donde r es una variable aleatoria de la distribución uniforme $U(-1, 1)$.

$$W(x) = 3.5 \times \exp\left(\frac{-(x-3)^2}{3}\right) + 2 \times \exp\left(\frac{-(x+1.5)^2}{0.5}\right) \tag{9}$$

δ tiene un valor fijo que se debe determinar teniendo en cuenta que δ_{optimo} es aquel para el cual se aceptan aproximadamente 50% de las proposiciones. Se busca generar una muestra de 10 millones de puntos. Para comprobar que el resultado es adecuado, se pide graficar $W(x)$ y un histograma de las variables aleatorias, normalizado en ambos casos.

2.2 Procedimiento

Se utiliza el programa *Python 2.7*. Dado que se necesitan números random, se utiliza nuevamente *numpy.random.uniform* y se escoge como semilla *212*.

En primer lugar se crea la función **metropolis(w, xn, delta)**, la cual recibe como parámetros la función $W(\cdot)$, el punto de partida xn y δ. La función calcula xp tal como se define en la sección Introducción, y luego evalúa $W(xp)$ y $W(xn)$. Finalmente, calcula la división $\frac{W(xp)}{W(xn)}$ y la compara con Γ, donde Γ es una variable aleatoria de la distribución uniforme $U(0, 1)$. Si $\frac{W(xp)}{W(xn)} > \Gamma$, la función retorna xp (lo que denominamos *aceptar*), sino entonces retorna el mismo valor que se le entrega, xn, lo que denominamos *rechazar*.

Luego, se crea la función **xn_mas_1(w, xn, N, delta)**, la cual recibe los mismos parámetros que **metropolis** más uno extra, N, que corresponde a la cantidad de puntos que contendrá la muestra a generar. La función inicializa un array xn_mas_1 de tamaño N con ceros en todas sus entradas excepto en la primera que toma el valor xn (que puede ser un número o un array). La función genera las demás entradas de la siguiente manera:

$$xn\_mas\_1_{(n+1)} = metropolis(w, xn\_mas\_1_{(n)}, delta) \tag{10}$$

La función entrega el array xn_mas_1 y un porcentaje de las veces *aceptadas*.

Para estimar δ_{optimo} se genera un array de 30 posibles δ con valores entre 1 y 10. Luego se opera **xn_mas_1(w, xn, N, delta)** para cada δ_n, con $xn = 0$ y $N = 10^7$. Con lo anterior se obtiene un porcentaje de aceptación para cada δ_n, luego es posible estimar lo buscado imponiendo que $\delta_{optimo} \sim 50\%$.

Finalmente, se obtiene la muestra deseada utilizando la función **xn_mas_1(w, xn, N, delta)**, con $\delta = \delta_{optimo}$ y $N = 10^7$. Se escoge $xn = 0$. Se grafica el histograma de la muestra, tomando 50 bins de acho 0.4 entre -10 y 10, y la función $W(x)$, ambas normalizadas.

2.3 Resultados

Realizando lo mencionado en la sección *Procedimiento* acerca de δ_{optimo} se obtiene la Figura 2. Se estima $\delta_{optimo} = 3.79$.

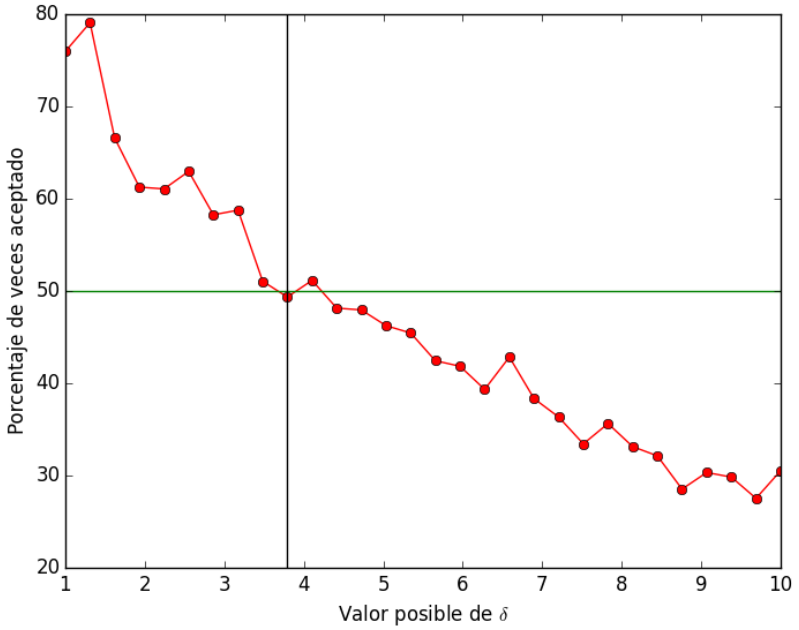

Figura 2: Gráfica de δ vs porcentaje de aprobación al operar *xn_mas_1(w, xn* $= 0$, $N = 10^3$, δ*)* para cada posible valor de δ.

Utilizando el valor anterior se calcula la muestra buscada y se gráfica el histograma de la Figura 3.

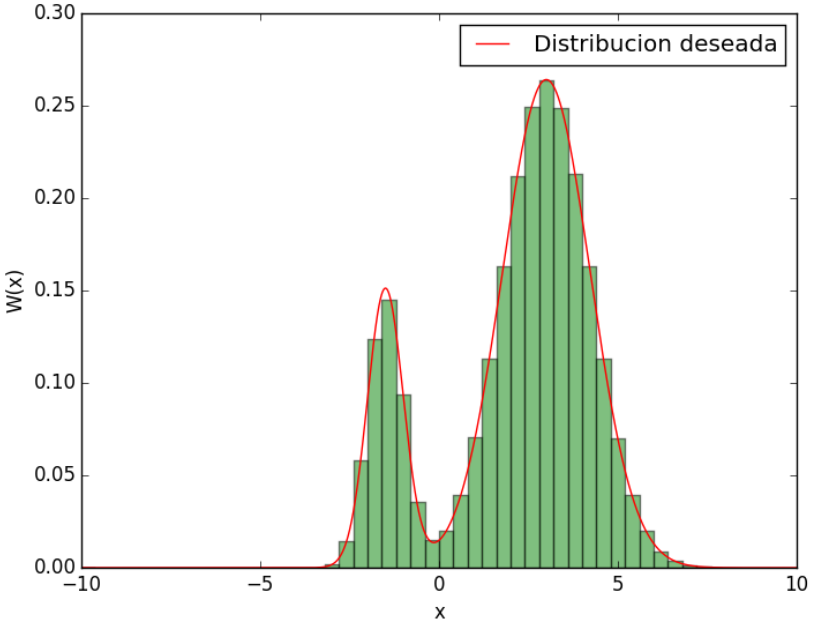

Figura 3: Histograma (en verde) de la muestra, de 10 millones de puntos, obtenida utilizando el algoritmo de Metrópolis. En rojo, la gráfica de la función distribución deseada $W(x)$ de la Ecuación 9.

2.4 Conclusiones

En la Figura 2 se observa que existe una relación inversamente proporcional (a modos general, no estricto) entre el valor de δ (entre 1 y 10) y el porcentaje de veces que se acepta en el programa.

En la Figura 3 se apreciar cómo la muestra obtenida mediante el algoritmo de Metrópolis se ajusta a la distribución deseada.

2.5 BONUS

2.5.1 Introducción

Se busca determinar la incertidumbre asociada a cada bin del histograma de la Figura 3 y graficar el histograma con las barras de error asociadas.

2.5.2 Procedimiento

Se repite 101 veces el procedimiento completo del problema, pero cada vez se utiliza un punto de partida (xn) distinto. Se utiliza xn como un array con 101 valores entre -10 y 10.
Se ocupan los mismos bins que fueron descritos anteriormente.
Se define el error para el valor de cada bin como la desviación estándar de los 101 valores que se obtienen.

Se realiza el procedimiento para dos casos:

- Muestra de 100 puntos
- Muestra de 10 millones de puntos

Cabe decir que los valores asociados a cada bin en cada repetición son guardados en los archivos *datos_100_puntos.txt* y *datos_10_millones_puntos.txt*, respectivamente. Luego leídos en otro script que los procesa. Esto dado que para el segundo caso el tiempo que demora el programa en obtener los datos es de aproximadamente 3.5 hrs.

2.5.3 Resultados

Realizando lo explicitado en la sección anterior, se obtienen las figuras 3 y 4.

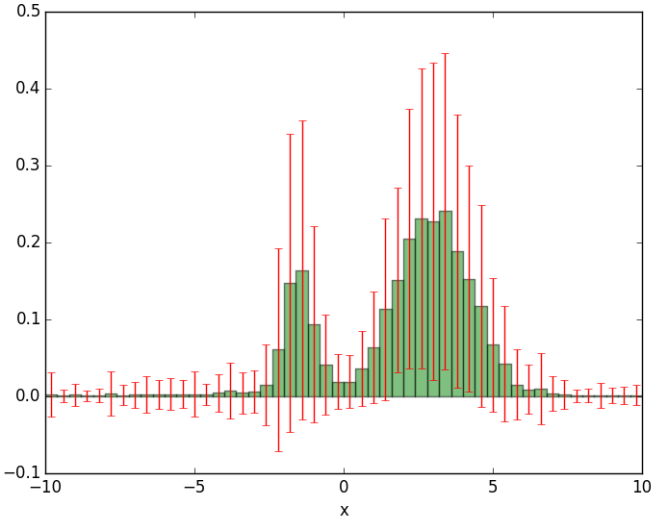

Figura 4: Caso 1 : Muestra de 100 puntos. Histograma donde la altura de las barras verdes corresponden al valor promedio asociado a cada bin en las 101 repeticiones. Las barras rojas muestran el error asociado.

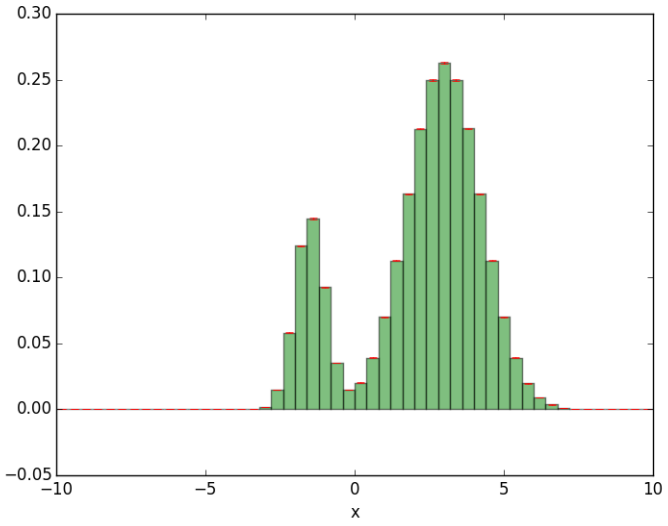

Figura 5: Caso 2 : Muestra de 10 millones de puntos. Histograma donde la altura de las barras verdes corresponden al valor promedio asociado a cada bin en las 101 repeticiones. Las barras rojas muestran el error asociado.

Los errores asociados a cada bin (desviación estandar) van del órden de 10^{-1} a 10^{-3} y de 10^{-4} a 10^{-8}, para el primer y segundo caso respectivamente. Estos datos están guardados en los archivos *desviacion_estandar_100.txt* y *desviacion_estandar_10_millones.txt*.

2.5.4 Conclusiones

A partir de las figuras 4 y 5 se aprecia la relación de que con muestras más pequeñas se obtienen errores más grandes, lo que es esperable teóricamente.

En la Figura 5 las barras de error no entran en un rango observable, lo que dice que una muestra de 10 millones de puntos es lo suficientemente grande para obtener errores despreciables (de órdenes de 10^{-4} a 10^{-7}).