Métodos Numéricos para la Ciencia e Ingeniería
FI3104-1

Tarea 8

Maximiliano Dirk Vega Aguilera
18.451.231-9

1 Introducción

La tarea 8 consistió en utilizar algoritmos estadísticos para estimar la posición del centro de masa de la intersección de dos sólidos, y obtener una muestra aleatoria a partir de una distribución dada. Los algoritmos utilizados fueron el método de integración de Monte Carlo y el algoritmo de Metrópolis. Adicionalmente, se debió escribir el programa de forma que cumpliera las reglas de PEP8.

El método de integración Monte Carlo es un método estadístico que consiste en aproximar la integral de una función ($f(x)$) mediante la generación de un gran número de valores (N) uniformemente distribuidos, dentro de un volumen (V) que contenga la integral buscada. Esta se obtiene de la siguiente relación:

$$I = \frac{V}{N} \sum_{i=1}^{N} f(x_i) \tag{1}$$

Que es equivalente a:

$$I = V \langle f(x) \rangle \tag{2}$$

Donde $\langle f(x) \rangle$ es el valor esperado de $f(x)$.

En la pregunta 1, se nos pidió estimar la posición del centro de masa de un solido compuesto por la intersección de los siguientes cuerpos:

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

$$\text{Cilindro}: \ (x-2)^2 + z^2 \leq 1 \tag{4}$$

Los que corresponden a un toro centrado en el origen de radio menor 2 y radio mayor 4, y un cilindro cuyo eje de simetría esta a lo largo del eje y y esta centrado en $x = 2$ y $z = 0$.

La densidad de este cuerpo varia según la siguiente expresión:

$$\rho(x, y, z) = 0.5 * (x^2 + y^2 + z^2) \tag{5}$$

La posición del centro de masa viene dado por:

$$r_{cm} = \frac{\int_V r\rho(r)dV}{\int_V \rho(r)dV} \tag{6}$$

Donde la integral inferior corresponde a la masa total del cuerpo y la integral superior la masa según la coordenada r. Estas integrales se calculan utilizando el método de Monte Carlo.

La segunda parte de la tarea consistió en obtener una muestra aleatoria a partir de una distribución conocida pero no normalizada, utilizando el algoritmo de Metrópolis. La distribución es:

$$W(x) = 3.5 \times \exp\left(\frac{-(x-3)^2}{3}\right) + 2 \times \exp\left(\frac{-(x+1.5)^2}{0.5}\right) \tag{7}$$

El algoritmo de Metrópolis genera una secuencia de números a partir de una distribución conocida salvo por la normalización. Se parte con valor conocido x_n y se generan valores de prueba x_p, estos valores se prueban con el criterio de Metrópolis, si son aceptados serán el término x_{n+1} y si no, el término x_n se repetirá como x_{n+1}. Los términos x_p se generan a partir de una distribución propuesta, en esta tarea los x_p se generaran a partir de $x_p = x_n + \delta r$, donde r es una variable aleatoria uniforme en (-1,1) y δ un parámetro que se debe determinar.

El criterio de Metrópolis consiste en evaluar $\frac{W(x_p)}{W(x_n)}$ y se comparan con un valor r' uniforme en (0,1). Si la fracción es mayor, el valor x_p se acepta, si no, se rechaza.

2 Desarrollo

Para el desarrollo de esta tarea se utilizó el lenguaje de programación python y sus paquetes numpy, matplotlib y scipy.

Para la primera parte, se construyó el algoritmo de Monte Carlo para una caja de 2x8x2, correspondiente a la caja más pequeña que contiene el objeto de interés. Se generaron puntos (x, y, z) de manera aleatoria dentro de la caja y se preguntó si ese punto pertenecía al toro y al cilindro. De pertenecer, se contaba y se seguía con el siguiente. El procedimiento se realizo para 100.000 puntos.

Como resultado se obtuvo que el centro de masa se encuentra en la posición aproximada (2.0, 0.0, 0.0) y la masa total del cuerpo es de aproximadamente 70. Estos valores son aproximados, se realizó el cálculo varias veces sin fijar la semilla y se concluye que son una buena aproximación. También se varió el número de muestras y se notó que para unos 1000 puntos ya hay convergencia para la caja dada, si la caja aumenta, se requerirá de más puntos para la convergencia.

Para la segunda parte, se implementó el algoritmo de metrópolis dividido en dos funciones, una que prueba un punto contra el criterio de Metrópolis y la otra que lo itera para $N = 10^7$ puntos. El histograma se hizo con la función *hist* de matplotlib y se graficó junto con

la función distribución $W(x)$ normalizada. Para normalizarla se calculó la integral mediante el método del trapecio, usando la función de scipy, obteniéndose el gráfico de la figura 1. En el se aprecia que la distribución generada por el algoritmo de Metrópolis se ajusta bastante bien a la distribución original normalizada con la cantidad de puntos tomada. Durante el proceso, se probaron N más pequeños y se notó que el algoritmo se ajustaba a grandes rasgos, pero no tenia una forma tan limpia.

Para encontrar el δ óptimo, es decir, el δ para el cual el método de Metrópolis acepta el 50% de los valores propuestos x_p, se construyó una función que genera varios histogramas a partir del método de Metrópolis, variando δ para cada histograma y contando el porcentaje de aceptación para uno, obteniéndose el gráfico de la figura 2. En este gráfico que aprecia que el valor para el cual se acepta el 50% de los valores de x_p es aproximadamente 3.9, el gráfico de la figura 1 se obtuvo con $\delta = 3.9$.

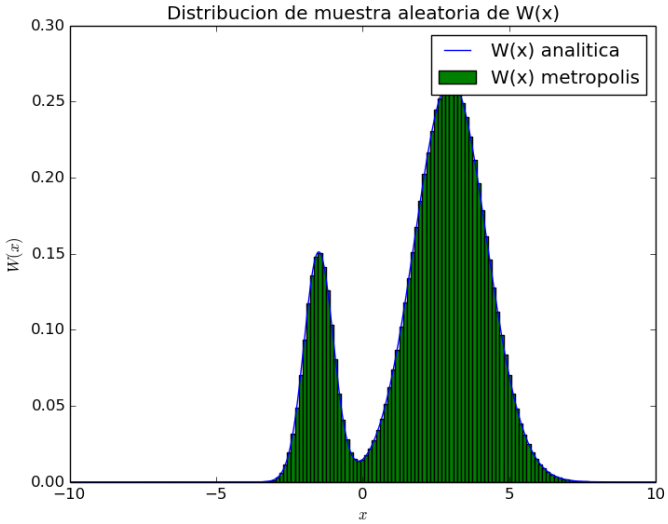

Figure 1: Distribución de números aleatorios generados a partir del método de Metrópolis a partir de W(x). Se utilizó como punto de partida $x_n = 0$

3 Conclusión

Los algoritmos estadísticos se adaptan fácilmente a cualquier tipo de problema y funcionan bastante bien en la medida que se use un gran número de datos. Sin embargo, esto hace que sea poco eficiente y mucho más tiempo de lo deseado, lo que podría ser un problema para el usuario.

4 Puntos Extra

En esta parte se nos pidió calcular la incertidumbre de cada bin del histograma de la figura 1. Para ello se repitió el proceso $N_{MC} = 100$ veces, pero considerando 200 bins y una muestra

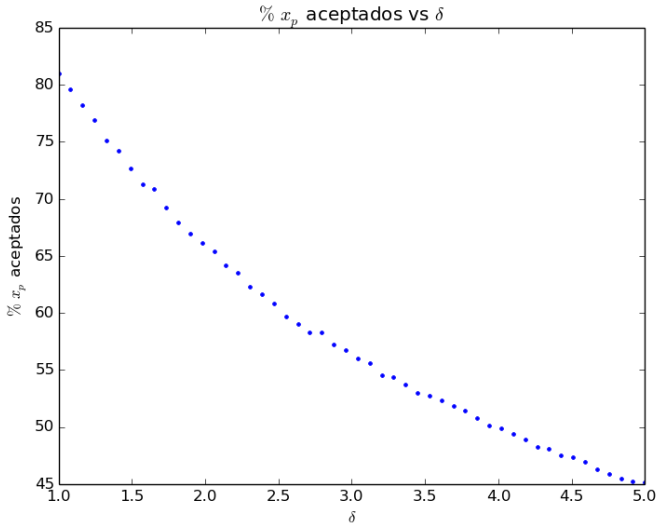

Figure 2: El gráfico indica el porcentaje de aceptación de los x_p para distintos valores de δ.

aleatoria de N=10^5 puntos, esto debido a problemas con el computador. Cada histograma se generó usando el algoritmo de Metrópolis, pero con distintos puntos de partida x_n, aleatorios uniformes en el intervalo [-10, 10], para cada histograma .

Luego de generar los histogramas, se calculó la desviación estándar de cada bin usando la función de numpy *std* y se usaron estos valores como el error de cada bin, obteniéndose la el gráfico de la figura 3. Analizando este gráfico, junto a la figura 4 y 5, se observa que la incertidumbre es bastante pequeña.

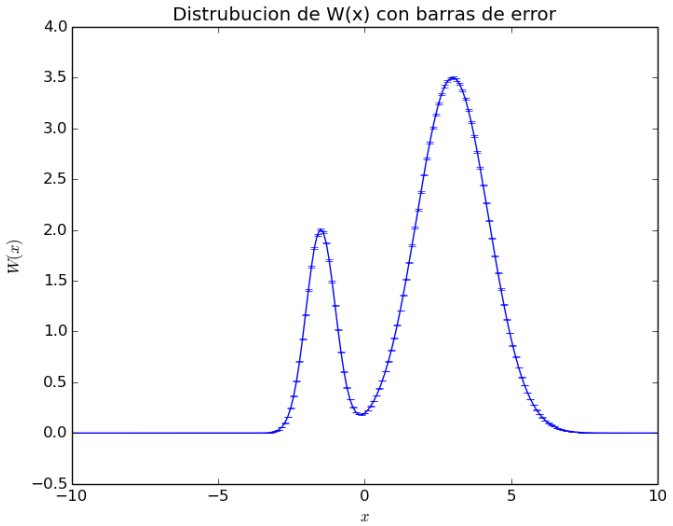

Figure 3: Gráfico de la distribución W(x) junto a las barras de error por bin.

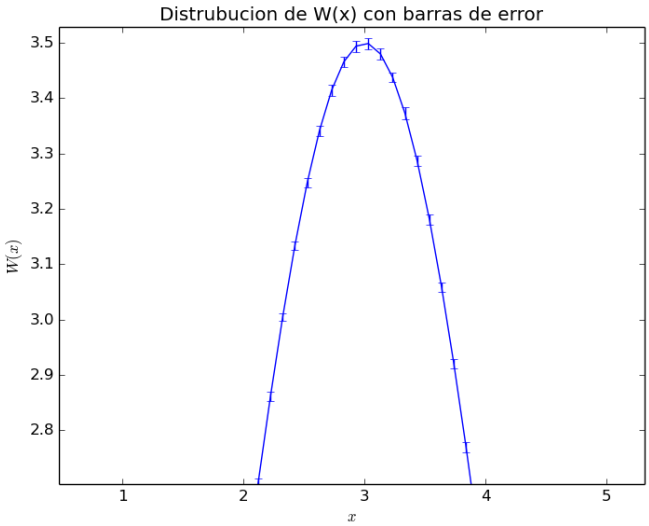

Figure 4: Zoom a la cima mayor del gráfico de la figura 3.

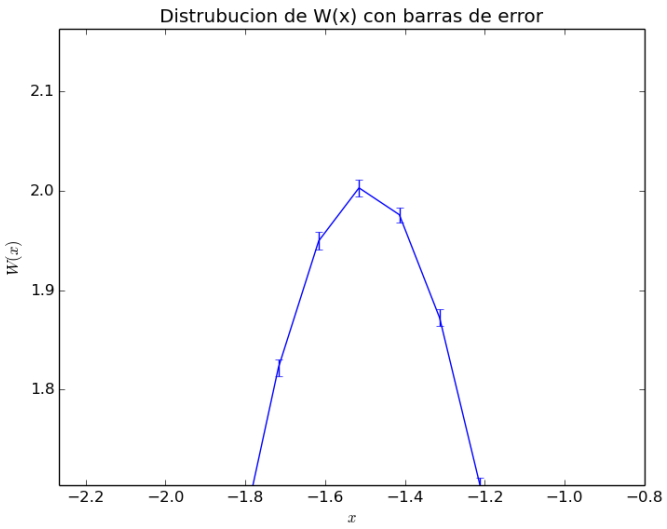

Figure 5: Zoom a la cima menor del gráfico de la figura 3.