Tarea 1

Carmelo García
Física Computacional

3 de noviembre de 2024

1. Problema

Considere un conjunto de N partículas que reposan en las posiciones $\vec{r}_n = x_n\hat{x} + y_n\hat{y}$, tal que:

$$x_n = -2{,}7x_{n-1}\ln(x_{n-1}), \quad n = 1, 2, \ldots, N-1; \quad x_0 = 0{,}3333$$

$$y_n = 0{,}9\sin(\pi y_{n-1}), \quad n = 1, 2, \ldots, N-1; \quad y_0 = 0{,}7453$$

La masa de la n-ésima partícula está dada por $m_n = |x_n y_n|^{1,3} e^{-r_n/\sqrt{2}}$. Para el caso $N = 10^8$, elabore un programa que evalúe para este sistema:

a) El área del mínimo rectángulo, cuyos lados son paralelos a los ejes x e y, que encierran este conjunto de partículas.

b) La masa total.

c) El centro de masas.

d) El momento de inercia alrededor del eje que descansa en el plano xy, es paralelo a la recta $y = x$ y pasa por el centro de masas.

2. Análisis del problema

Por ahora no he dimensionando el verdadero poder de `C`, no se si $N = 10^8$ partículas es bastante para este lenguaje o es un pequeño tramite. No me he querido arriesgar y es tratado de guardar lo menos posible en la memoria. Los cálculos se han hecho sobre la marcha y sòlo se ha ¨guardado"lo necesario.

La idea principal de la resolucion del problema se basa en tener un solo bucle `for` que iterará sobre todas las partículas, es decir, desde la partícula 0 hasta la partícula N. Cuando el bucle `for` se centre en cada partícula se van a hacer todos los calculos necesarios para esa partícula.

A continuación nos vamos a detener en las constantes que vamos a utilizar a lo largo del código:

```c

#include<stdio.h>
#include<stdlib.h>
#include<math.h>

// Definimos las constantes que vamos a usar.
int N = 100000000;
float x_0 = 0.3333, y_0 = 0.7453;
double x_n, y_n, m_n, r_n, x_ant, y_ant, m_t, x_cm=0, y_cm=0, r_cm[2],
       I=0, x_max, x_min, y_max, y_min, A;

```

Listing 1: Declaraciòn de Variables

- N: Número de particulas en el sistema.
- x_0 y y_0: Posición de la partícula inicial.
- x_n y y_n: Son las componentes de del vector posición.
- x_{ant} y y_{ant}: Harán el papel de x_{n-1} y y_{n-1}.
- r_n: Es el módulo del vector $\vec{r}_n$, es decir, $\sqrt{x_n^2 + y_n^2}$
- x_{cm} y y_{cm}: Son las componentes de del vector centro de masa.
- $r_{cm}[2]$: es el vector r_{cm}, visto como un arreglo de una matriz de una fila y dos columnas.
- x_{max}, y_{max} y x_{min}, y_{min}: Son las componentes del vector posición en los extremos que nos ayudaran a determinar el area del rectangulo que continene las particulas.
- m_t, I y A: Son la masa total, el momento de inercia y el area del rectangulo respectivamente.

3. Soluciones

3.1. Posiciones

El problema nos da la forma de calcular las componentes del vector posición, esto es las ecuaciones 1 y 2. Como se dijo en la sección 2, se busca hacer todos los cálculos necesarios en un solo bucle.

$$x_n = -2{,}7x_{n-1}\ln(x_{n-1}) \tag{1}$$

$$y_n = 0{,}9\sin(\pi y_{n-1}) \tag{2}$$

Por lo tanto, afuera del bucle instanciamos las posiciones iniciales. Dentro del bucle calculamos la posición de cada partícula a partir de la 1 hasta la particula $N-1$, como vamos a hacer todos los calculos necesarios antes de que acabe la iteracion entonces no es conveniente guardar las posciones de las particulas. Por esta razon al final del bucle se le asigna a x_ant el valor de x_n para asi darle paso a la siguente particula y .olvidar la posición de la partícula anterior.

```
void main(){
    x_ant = x_0;
    y_ant = y_0;

    for (int i=1; i<N; i++){
        // componente del vector posicion en x
        x_n = -2.7*x_ant*log(x_ant);

        // componente del vector posicion en y
        y_n = 0.9*sin(M_PI *y_ant);

        //Pasar a la siguiente particula
        x_ant = x_n;
        y_ant = y_n;

        }

        }
```

Listing 2: Calculo de componentes de posición

3.2. Calculo de masa

Para el cálculo de la masa se usó la ecuación 3, donde nos damos cuenta que se necesita el módulo del vector posición.

$$m_n = |x_n y_n|^{1,3} e^{-r_n/\sqrt{2}} \tag{3}$$

Esta ecuación se usó dentro del mismo bucle del cálculo de las posiciones. No podemos inicializar la masa total en cero debido a que tenemos una partícula inicial y la masa inicial del sistema sera la masa de esa partícula. Entonces, necesitamos calcular afuera del bucle el módulo del vector posición inicial r_0 y la masa inicial $m_t = |x_0 y_0|^{1,3} e^{-r_0/\sqrt{2}}$. Dentro del bucle se hace el cálculo de la masa de cada partícula y se hace uso de la recursividad para que m_t sea la suma de todas las masas de las partículas. Esta recursividad se expresa de la forma $m_t += m_n$, que es lo mismo que decir $m_t = m_t + m_n$. En cristiano, se lee que la masa total del sistema es igual a la masa total del sistema anterior mas la masa de la partícula 'actual'.

```
void main(){
x_ant = x_0;
y_ant = y_0;
double r_0 = sqrt(x_0*x_0 + y_0*y_0);
m_t = pow(fabs(x_0*y_0), 1.3)*exp(-r_0/sqrt(2));
for (int i=1; i<N; i++){
    // componente del vector posicion en x
    x_n = -2.7*x_ant*log(x_ant);
    // componente del vector posicion en y
    y_n = 0.9*sin(M_PI *y_ant);
```

```
        //Modulo del vector posicion
        r_n = sqrt(x_n*x_n + y_n*y_n);
        // Calcular la masa
        m_n = pow(fabs(x_n*y_n), 1.3)*exp(-r_n/sqrt(2));
        m_t += m_n;
        //Pasar a la siguiente particula
        x_ant = x_n;
        y_ant = y_n;
    }
    printf("La masa total del sistema es: %lf\n", m_t);
    }

```

Listing 3: Calculo de la masa.

3.3. Calculo del Centro de Masa y Momento de Inercia.

Debido a que el centro de masa es un vector, se calculo las dos componentes por separado. Sin embargo, como se se ve en la ecuación 4, el vector esta siendo dividio por la masa total del sistema, debido a esto en el bucle solo se calculo $\sum_i m_i \vec{r}_i$. Cuando el bucle se termina, tenemos la masa total ya calculada como lo vimos antes y y es alli cuando podemos guardar en nuestro arreglo de centro de masas las variables X_{cm} y y_{cm} divididas por la masa total, de manera que: $r_{cm}[0] = x_{cm}/m_t$ y $r_{cm}[1] = y_{cm}/m_t$. De esta forma que es que calculo el centro de masa de nuestro sistema.

$$\vec{r}_{cm} = \frac{1}{M} \sum_i m_i \vec{r}_i \tag{4}$$

Para el calculo del momento de inercia se hizo prácticamente igual que para el centro de masa, utilizando la ecuación 5, con la salvedad de que se le saco el cuadrado del modulo al vector posición de cada partícula y esto se multiplico por cada masa particular y así se calculo el momento de inercia del sistema para un eje que pasa por el origen.

$$I = \sum_i m_i r_i^2 \tag{5}$$

Sin embargo, el problema es bastante explicito pidiendo el Momento de inercia para un eje que pase por el centro de masa. Para resolver este inconveniente se hará uso del **Teorema de Steiner**, también conocido como el **Teorema del Eje Paralelo**, establece que el momento de inercia de un cuerpo respecto a un eje que no pasa por su centro de masa es igual al momento de inercia respecto a un eje paralelo que pasa por el centro de masa, más el producto de la masa del cuerpo por el cuadrado de la distancia entre los dos ejes.

$$I = I_{\text{cm}} + Md^2 \tag{6}$$

donde:

- I es el momento de inercia respecto al eje paralelo.
- I_{cm} es el momento de inercia respecto al eje que pasa por el centro de masa.

- M es la masa del cuerpo.
- d es la distancia entre los dos ejes.

En nuestro caso, lo que se ha calculado es un momento de inercia respecto a un eje paralelo. Entonces se puede despejar I_{cm} de la ecuación 3.3, donde d sera el modulo de el vector $\vec{r}_{cm}$ ya que este es la distancia entre el centro de masa y el origen. Por lo tanto no quedaría lo siguiente:

$$I_{\text{cm}} = I - M|\vec{r}_{cm}|^2$$

Esto se hará después de que se termine las iteraciones del bucle debido a que debemos esperar a tener M, I y $\vec{r}_{cm}$ calculado para todas las partículas. De esta forma se puede afirmar que se esta calculando el momento de inercia indicado por el problema.

A continuación el extracto del código con todo lo que se ha explicado hasta ahora. Cabe destacar que en la linea 12 del código es para limpiar la memoria del arreglo utilizado para el vector centro de masa.

```
1
2     void main(){
3     x_ant = x_0;
4     y_ant = y_0;
5     //Le sacamos los calculos necesarios a la primera particula
6     double r_0 = sqrt(x_0*x_0 + y_0*y_0); //Modulo del vector posicion
7     m_t = pow(fabs(x_0*y_0), 1.3)*exp(-r_0/sqrt(2)); // Masa de la
    particula 0
8     x_cm = x_0*m_t; // Aportacion de la particula 0 al centro de masas
9     y_cm = y_0*m_t; // Aportacion de la particula 0 al centro de masas
10    I = m_t*(x_0*x_0 + y_0*y_0); // Aportacion de la particula 0 al
    momento de inercia
11    // limpiamos la memoria del arreglo r_cm
12    for (int i=0; i<1; i++) r_cm[i] = 0;
13
14        for (int i=1; i<N; i++){
15            // componente del vector posicion en x
16            x_n = -2.7*x_ant*log(x_ant);
17            // componente del vector posicion en y
18            y_n = 0.9*sin(M_PI *y_ant);
19            //Modulo del vector posicion
20            r_n = sqrt(x_n*x_n + y_n*y_n);
21            // Calcular la masa
22            m_n = pow(fabs(x_n*y_n), 1.3)*exp(-r_n/sqrt(2));
23            m_t += m_n;
24
25            //Calculo del centro de masas
26            x_cm += x_n*m_n;
27            y_cm += y_n*m_n;
28
29            //Calcular el momento de inercia
30            I += m_n*(x_n*x_n + y_n*y_n);
31
```

```
32              //Pasar a la siguiente particula
33              x_ant = x_n;
34              y_ant = y_n;
35              }
36          //Calculo del centro de masas
37          r_cm[0] = x_cm/m_t;
38          r_cm[1] = y_cm/m_t;
39
40           // Calculo del momento de incercia del centro de masa que pasa
    por el centro de masa usando el teorema de Steiner.
41          double Icm = I - m_t*(r_cm[0]*r_cm[0] + r_cm[1]*r_cm[1]);
42
43          printf("La masa total del sistema es: %lf\n", m_t);
44          printf("r_cm = (%lf, %lf)\n", r_cm[0], r_cm[1]);
45          printf("El momento de inercia del sistema es: %lf\n", I);
46      }
47
```

Listing 4: Calculo del centro de masa y el momento de inercia.

3.4. Calculo del área mínima del rectángulo que contiene a las partículas.

Esta parte de la tarea la deje para el final porque bajo mi parecer fue la sección mas retadora de la tarea. Como no se estaba guardando información sobre las posiciones de las partículas del sistema se tuvo que comparar cada posición de la partícula con la anterior. Esto con el fin de encontrar los extremos del 'rectángulo', es decir la coordenada en el eje y mas separadas entre si y lo mismo en el eje x. Con estas coordenadas se calculo la distancia entre estos puntos y esta distancias eran la base y la altura del 'rectángulo'. La base es la distancia entre los puntos mas extremos del eje x y la altura es la distancia de los puntos mas extremos del eje y.

Como esto es el ultimo requerimiento del problema, a continuación todo el código empleado para resolver esta tarea. El calculo del área del rectángulo esta entre las lineas 37 y 50 del código.

```
1
2  #include<stdio.h>
3  #include<stdlib.h>
4  #include<math.h>
5
6  // Definimos las constantes que vamos a usar.
7  int N = 100000000;
8  float x_0 = 0.3333, y_0 = 0.7453;
9  double x_n, y_n, m_n, r_n, x_ant, y_ant, m_t, x_cm=0, y_cm=0, r_cm[2], I
       =0, x_max, x_min, y_max, y_min, A;
10
11 void main(){
12          x_ant = x_0;
13          y_ant = y_0;
```

```
        double r_0 = sqrt(x_0*x_0 + y_0*y_0);
        m_t = pow(fabs(x_0*y_0), 1.3)*exp(-r_0/sqrt(2));
        //printf("masa total inicial : %lf\n", m_t);
        // limpiamos la memoria del arreglo r_cm
        for (int i=0; i<1; i++) r_cm[i] = 0;

        for (int i=1; i<N; i++){
            // componente del vector posicion en x
            x_n = -2.7*x_ant*log(x_ant);
            // componente del vector posicion en y
            y_n = 0.9*sin(M_PI *y_ant);
            //Modulo del vector posicion
            r_n = sqrt(x_n*x_n + y_n*y_n);
            // Calcular la masa
            m_n = pow(fabs(x_n*y_n), 1.3)*exp(-r_n/sqrt(2));
            m_t += m_n;
            //Calculo del centro de masas
            x_cm += x_n*m_n;
            y_cm += y_n*m_n;
            //Calcular el momento de inercia
            I += m_n*(x_n*x_n + y_n*y_n);
            //Calcular el area del rectangulo
            if (i == 1){
                x_max = x_n;
                x_min = x_n;
                y_max = y_n;
                y_min = y_n;
            }
            else{
                if (x_n > x_max) x_max = x_n;
                if (x_n < x_min) x_min = x_n;
                if (y_n > y_max) y_max = y_n;
                if (y_n < y_min) y_min = y_n;
            }
            A = (x_max - x_min)*(y_max - y_min);
            //Pasar a la siguiente particula
            x_ant = x_n;
            y_ant = y_n;
            }
        //Calculo del centro de masas
        r_cm[0] = x_cm/m_t;
        r_cm[1] = y_cm/m_t;

        printf("La masa total del sistema es: %lf\n", m_t);
        printf("El centro de masas del sistema es r_cm = (%lf, %lf)\n",
    r_cm[0], r_cm[1]);
        printf("El momento de inercia del sistema es: %lf\n", I);
        printf("El area del rectangulo es: %lf\n", A);
     }

```

Listing 5: Calculo del area.

4. Análisis de Resultados

Los resultados obtenidos con el código explicado en las secciones anteriores se pueden apreciar en el cuadro 1. Sin embargo, quise llevarlo mas allá y tratar de verlo gráficamente. Desafortunadamente, guardar en un archivo `.dat` las posiciones de 10^8 partículas es bastante pesado y un poco innecesario. Por lo tanto, decidí correr el programa para una muestra mas pequeña, un sistema de 1000 partículas cuyos resultados se encuentran en el cuadro 2.

Parámetro	**Valor**
Masa	13493401.457418
$\vec{r}_{cm}$	(0.767452 , 0.711379)
I_{cm}	1124470.810726
Área	0.606448

Cuadro 1: Resultados para un sistema de 10^8 partículas.

Parámetro	**Valor**
Masa	138.668537
$\vec{r}_{cm}$	(0.769525, 0.717268)
I_{cm}	11.065602
Área	0.606412

Cuadro 2: Resultados para un sistema de 1000 partículas.

Entonces, para comprobar que el código funciona correctamente se graficó todos los puntos correspondientes a las partículas, ademas del centro de masa y el rectángulo con las dimensiones usadas para calcular el área. Se puede notar en la figura 1 que las partículas efectivamente están contenidas en un rectángulo. Uno se podría imaginar que el centro de masa estaría mas al centro del rectángulo, sin embargo en la imagen se puede observar que esta mas arriba y a la derecha.

El la figura 2 podemos observar el comportamiento de la masa en función de las posiciones, y nos damos cuenta que en los rangos de posiciones que manejamos dentro del rectángulo de las partículas la masa crece mientras mas a la derecha y arriba las partículas estén dentro del rectángulo. Por lo tanto las partículas de esa región del rectángulo tienen un aporte mayor en el calculo del centro de masa, por esta razón el centro de masa esta mas a la derecha de lo que imaginaríamos.

Cabe destacar que estas gráficas se hicieron a través de `Python`. Solo las gráficas, todos los cálculos y el archivo `.csv` con las posiciones de las partículas se hicieron en `C`.

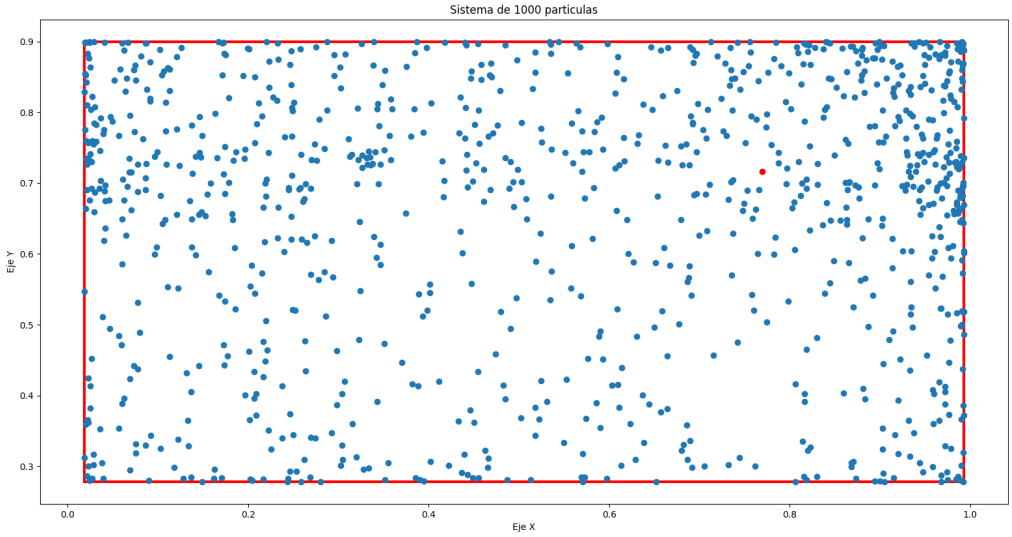

Figura 1: Sistema de 1000 partículas

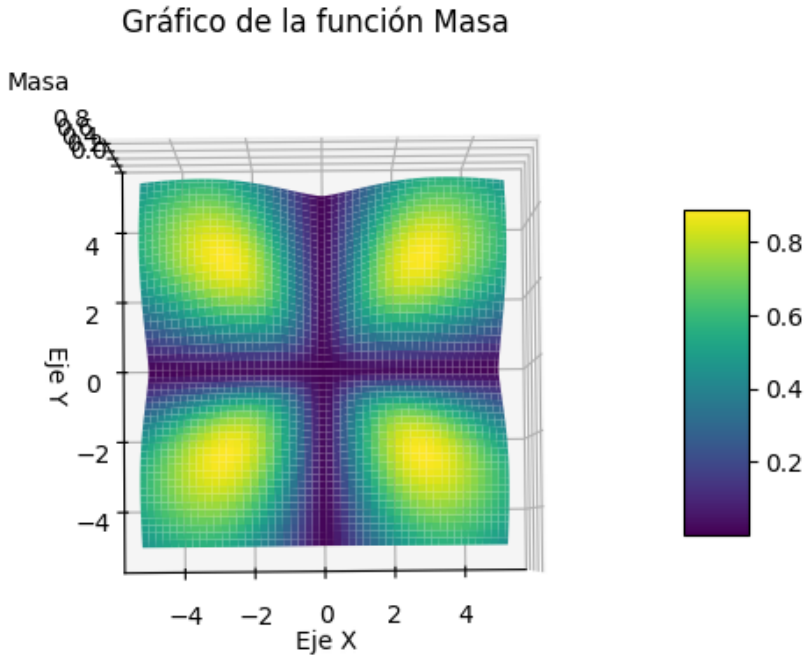

Figura 2: Comportamiento de la masa en función de la posición.

5. Anexos

A continuación los programas usados para graficar.

```

import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.patches as patches


datos = pd.read_csv('tarea_1/particulas.csv')

x_cm, y_cm = (0.769527, 0.716770)

df_datos = pd.DataFrame(datos)
# Calcula b y h
b = df_datos.x.max() - df_datos.x.min()
h = df_datos.y.max() - df_datos.y.min()
A = b*h
# Crear una nueva figura y un eje
fig, ax = plt.subplots()
# Crear y a adir el rect ngulo al gr fico
rect = patches.Rectangle((df_datos.x.min(), df_datos.y.min()), b, h,
    linewidth=3, edgecolor='r', facecolor='none')
ax.add_patch(rect)
# Gr fico de dispersi n de los datos
ax.scatter(df_datos.x, df_datos.y)
# Gr fico de dispersi n del centro de masas
ax.scatter(x_cm, y_cm, c='red')
# A adir etiquetas y t tulo
plt.xlabel('Eje X')
plt.ylabel('Eje Y')

plt.title('Sistema de 1000 particulas')

# Mostrar el gr fico
plt.show()

print(A)
```

Listing 6: Graficador de particulas.

```

import numpy as np
import matplotlib.pyplot as plt

# Definir la funci n
def funcion(x, y):
    return np.abs(x * y)**1.3 * np.exp(-np.sqrt(x**2 + y**2) / np.sqrt(2))

# Crear una malla de puntos
x = np.linspace(-5, 5, 100)
y = np.linspace(-5, 5, 100)
x, y = np.meshgrid(x, y)
z = funcion(x, y)
```

```
14
15 # Crear la figura
16 fig = plt.figure()
17 ax = fig.add_subplot(111, projection='3d')
18
19 # Graficar la superficie
20 surface = ax.plot_surface(x, y, z, cmap='viridis')
21
22 # A adir una barra de color
23 fig.colorbar(surface, ax=ax, shrink=0.5, aspect=5)
24
25 # A adir etiquetas y t tulo
26 ax.set_xlabel('Eje X')
27 ax.set_ylabel('Eje Y')
28 ax.set_zlabel('Masa')
29 ax.set_title('Gr fico de la funci n Masa')
30
31 # Mostrar el gr fico
32 plt.show()
```

Listing 7: Graficador de la Masa.