

Tarea 1

Carmelo García
Física Computacional

31 de octubre 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:

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

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

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:

- El área del mínimo rectángulo, cuyos lados son paralelos a los ejes x e y , que encierran este conjunto de partículas.
- La masa total.
- El centro de masas.
- 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 dimensionado el verdadero poder de \mathbb{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 sólo se ha "guardado" lo necesario.

La idea principal de la resolución 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 cálculos necesarios para esa partícula.

A continuación nos vamos a detener en las constantes que vamos a utilizar a lo largo 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],
10         I=0, x_max, x_min, y_max, y_min, A;
11

```

Listing 1: Declaración de Variables

- N : Número de partículas 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.

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

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

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 partícula $N - 1$, 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 siguiente partícula y .ºlvidar la posición de la partícula anterior.

```

1
2 void main(){
3     x_ant = x_0;
4     y_ant = y_0;
5
6     for (int i=1; i<N; i++){
7         // componente del vector posicion en x
8         x_n = -2.7*x_ant*log(x_ant);
9
10        // componente del vector posicion en y
11        y_n = 0.9*sin(M_PI *y_ant);
12
13        //Pasar a la siguiente particula
14        x_ant = x_n;
15        y_ant = y_n;
16
17    }
18
19 }
20
21

```

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.

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

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'.

```

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

```

```

12      //Modulo del vector posicion
13      r_n = sqrt(x_n*x_n + y_n*y_n);
14      // Calcular la masa
15      m_n = pow(fabs(x_n*y_n), 1.3)*exp(-r_n/sqrt(2));
16      m_t += m_n;
17      //Pasar a la siguiente particula
18      x_ant = x_n;
19      y_ant = y_n;
20  }
21  printf("La masa total del sistema es: %lf\n", m_t);
22  }
23

```

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 ve en la ecuación 4, el vector esta siendo dividido 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 es allí 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.

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

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.

$$I = \sum_i m_i r_i^2 \quad (5)$$

A continuación el extracto del código con todo lo que se ha explicado hasta ahora. Cabe destacar que en la linea 9 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      double r_0 = sqrt(x_0*x_0 + y_0*y_0);
6      m_t = pow(fabs(x_0*y_0), 1.3)*exp(-r_0/sqrt(2));
7      //printf("masa total inicial : %lf\n", m_t);
8      // limpiamos la memoria del arreglo r_cm
9      for (int i=0; i<1; i++) r_cm[i] = 0;
10

```

```

11     for (int i=1; i<N; i++){
12         // componente del vector posicion en x
13         x_n = -2.7*x_ant*log(x_ant);
14         // componente del vector posicion en y
15         y_n = 0.9*sin(M_PI *y_ant);
16         //Modulo del vector posicion
17         r_n = sqrt(x_n*x_n + y_n*y_n);
18         // Calcular la masa
19         m_n = pow(fabs(x_n*y_n), 1.3)*exp(-r_n/sqrt(2));
20         m_t += m_n;
21
22         //Calculo del centro de masas
23         x_cm += x_n*m_n;
24         y_cm += y_n*m_n;
25
26         //Calcular el momento de inercia
27         I += m_n*(x_n*x_n + y_n*y_n);
28
29         //Pasar a la siguiente particula
30         x_ant = x_n;
31         y_ant = y_n;
32     }
33     //Calculo del centro de masas
34     r_cm[0] = x_cm/m_t;
35     r_cm[1] = y_cm/m_t;
36
37     printf("La masa total del sistema es: %lf\n", m_t);
38     printf("r_cm = (%lf, %lf)\n", r_cm[0], r_cm[1]);
39     printf("El momento de inercia del sistema es: %lf\n", I);
40 }
41

```

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 = 1000000000;
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
12
13 void main(){
14     x_ant = x_0;
15     y_ant = y_0;
16     double r_0 = sqrt(x_0*x_0 + y_0*y_0);
17     m_t = pow(fabs(x_0*y_0), 1.3)*exp(-r_0/sqrt(2));
18     //printf("masa total inicial : %lf\n", m_t);
19     // limpiamos la memoria del arreglo r_cm
20     for (int i=0; i<1; i++) r_cm[i] = 0;
21
22     for (int i=1; i<N; i++){
23         // componente del vector posicion en x
24         x_n = -2.7*x_ant*log(x_ant);
25         // componente del vector posicion en y
26         y_n = 0.9*sin(M_PI *y_ant);
27         //Modulo del vector posicion
28         r_n = sqrt(x_n*x_n + y_n*y_n);
29         // Calcular la masa
30         m_n = pow(fabs(x_n*y_n), 1.3)*exp(-r_n/sqrt(2));
31         m_t += m_n;
32         //Calculo del centro de masas
33         x_cm += x_n*m_n;
34         y_cm += y_n*m_n;
35         //Calcular el momento de inercia
36         I += m_n*(x_n*x_n + y_n*y_n);
37         //Calcular el area del rectangulo
38         if (i == 1){
39             x_max = x_n;
40             x_min = x_n;
41             y_max = y_n;
42             y_min = y_n;
43         }
44         else{
45             if (x_n > x_max) x_max = x_n;
46             if (x_n < x_min) x_min = x_n;
47             if (y_n > y_max) y_max = y_n;
48             if (y_n < y_min) y_min = y_n;
49         }
50         A = (x_max - x_min)*(y_max - y_min);
51         //Pasar a la siguiente particula
52         x_ant = x_n;
53         y_ant = y_n;

```

```

54     }
55
56     //Calculo del centro de masas
57     r_cm[0] = x_cm/m_t;
58     r_cm[1] = y_cm/m_t;
59
60     printf("La masa total del sistema es: %lf\n", m_t);
61     printf("El centro de masas del sistema es r_cm = (%lf, %lf)\n",
r_cm[0], r_cm[1]);
62     printf("El momento de inercia del sistema es: %lf\n", I);
63     printf("El area del rectangulo es: %lf\n", A);
64 }
65

```

Listing 5: Calculo del centro de masa y el momento de inercia.