UNIVERSIDAD DE
COSTA RICA

Física Computacional

Escuela de Física

Proyecto Final

Autores

Gabriel Contreras C12251

Juan Gonzáles C33326

Nicholas Snodgrass C07538

7 de julio de 2025

Índice

1. Introducción

Este código simula lo que ocurre cuando partículas chocan entre sí en un espacio. También calcula esos choques con un modelo idealista que asume colisiones elásticas y partículas con forma homogénea. El propósito de crear un modelo como este es poder simular el comportamiento de gases en condiciones controladas sin necesidad de mucho poder computacional.

2. Marco Teórico

2.1. Dinámica Molecular

Técnica computacional que simula el movimiento de partículas (átomos o moléculas) mediante integración de las leyes de Newton. Es muy util para hacer modelos de Física del estado sólido, química computacional, biofísica, entre otras aplicaciones.

2.2. Implementación Computacional

La simulación de dinámica molecular fue implementada en **Python 3**, utilizando un enfoque modular y estructurado. El proyecto hace uso de herramientas y bibliotecas científicas comunes, que permiten la ejecución, análisis y visualización interactiva de los resultados.

2.2.1. Lenguaje y herramientas

- **Python 3**: Lenguaje de programación interpretado, de alto nivel, ideal para simulaciones científicas debido a su legibilidad y vasta comunidad.
- **Jupyter Notebook**: Entorno interactivo que permite integrar código, visualizaciones y explicaciones teóricas en un mismo archivo.
- **NumPy**: Biblioteca para manipulación eficiente de arreglos multidimensionales y operaciones vectorizadas.
- **Matplotlib**: Biblioteca de visualización que permite generar gráficos estáticos y animaciones bidimensionales.
- **Grilla**: Es un tipo de computación distribuida en la que se utilizan los recursos de múltiples computadoras conectadas en red para resolver un problema común. Estas computadoras colaboran como si fueran un solo sistema. En simulaciones, visualizaciones o programación científica, una grilla es una estructura de datos en forma de malla, donde se discretiza un dominio continuo en puntos o celdas.

2.2.2. Organización del código

El código fuente se organiza en el directorio `src/`, dividido en los siguientes módulos:

- **sim.py**: Contiene la clase `Particula`, que modela las propiedades de cada partícula (masa, radio, posición, velocidad), así como la función `generar_posiciones(N)` para inicialización aleatoria sin superposición. También define los parámetros globales como el tamaño del sistema L, el tiempo total t y el paso temporal Δt.
- **colisiones.py**: Define la función `manejar_colision(p1, p2)` que implementa la detección y resolución de colisiones entre partículas, aplicando conservación de momento y energía.
- **simulador.py**: Encargado del bucle principal de la simulación, mediante la función `simular(particulas, t, dt, L)`, que gestiona la evolución temporal, detección de colisiones y almacenamiento de datos.

3. Conceptos Matemáticos

La base teórica matematica de la dinámica molecular que se está intentando simular con este modelo es la mecánica estadística y la mecánica clásica. Por tanto, se utilizará el principio de la conservación del momento lineal y la suposición de que todas las colisiones son elásticas para predecir y calcular el resultado teórico.

3.1. Conservación del Momento

El **momento lineal**, definido como $p = mv$, es una cantidad vectorial que se conserva en interacciones entre partículas en ausencia de fuerzas externas. En una colisión entre dos partículas:

$$m_1 v_1 + m_2 v_2 = m_1 v_1' + m_2 v_2' \tag{1}$$

donde:

- m_1, m_2 son las masas de las partículas.
- v_1, v_2 son las velocidades iniciales antes de la colisión.
- v_1', v_2' son las velocidades después de la colisión.

Esta ecuación es válida tanto para colisiones elásticas como inelásticas en sistemas aislados.

3.2. Conservación de la Energía Cinética

La energía cinética se conserva en colisiones perfectamente elásticas. Para dos partículas, la condición se expresa como:

$$\frac{1}{2}m_1v_1^2 + \frac{1}{2}m_2v_2^2 = \frac{1}{2}m_1{v_1'}^2 + \frac{1}{2}m_2{v_2'}^2 \tag{2}$$

Para N partículas en una simulación generalizada:

$$\sum_{k=1}^{N}\frac{1}{2}m_kv_{ik}^2 = \sum_{k=1}^{N}\frac{1}{2}m_kv_{fk}^2 \tag{3}$$

donde v_{ik} y v_{fk} son las velocidades iniciales y finales de la k-ésima partícula, respectivamente.

3.3. Velocidades Finales después de la Colisión

Para una colisión unidimensional entre dos partículas de masas m_1 y m_2, y velocidades iniciales v_1 y v_2, las velocidades después de una colisión elástica se calculan como:

$$v_1' = \frac{(m_1 - m_2)v_1 + 2m_2v_2}{m_1 + m_2} \tag{4}$$

$$v_2' = \frac{(m_2 - m_1)v_2 + 2m_1v_1}{m_1 + m_2} \tag{5}$$

Estas fórmulas derivan de aplicar simultáneamente la conservación del momento lineal y de la energía cinética.

3.4. Ecuación de Posición

La evolución temporal de la posición de una partícula en un sistema sin fuerzas externas (o a pasos de tiempo discretos pequeños) se puede expresar como:

$$\vec{r}(t + \Delta t) = \vec{r}(t) + \vec{v}(t) \cdot \Delta t \tag{6}$$

donde:

- $\vec{r}(t)$ es la posición en el tiempo t.
- $\vec{v}(t)$ es la velocidad en el tiempo t.
- Δt es el intervalo de tiempo.

4. Clase *Partícula* y generación de condiciones iniciales

La clase *Partícula* es la base del programa, este objeto contiene los atributos necesarios para realizar la simulación, como lo son la masa de las partículas, sus radios, posiciones y velocidades, siendo estas variables la base fundamental del programa. -

4.1. Método *Mover*

El método *Mover* sirve para cambiar las posiciones según la velocidad de la partícula, esto se logra mediante la adición del vector velocidad multiplicado por el cambio del tiempo. Este método mueve la partícula en la dirección de su desplazamiento.

4.2. Método *Rebotar pared*

El método *Rebotar pared* tiene su funcionamiento en dos partes diferenciadas por los límites, por un lado se verifica si en cada eje, la resta de radio menos la posición es menor a 0, indicando que quedara fuera del cubo, por lo tanto, se multiplica la velocidad en el eje de colisión por menos uno, haciendolo ir en la dirección contraria solo en el eje de la colisión; por el otro lado se realiza el mismo método, cambiando las condiciones, para ajustar que la posición más el radio sea menor a L, donde L es la longitud del lado del cubo, y por tanto el límite máximo de posición.

4.3. Método *Rebotar partícula*

En la colisión entre partículas se utiliza un carácter vectorial para realizar los cálculos, apoyándose en la biblioteca de *Numpy* junto con su módulo de álgebra lineal. Para la verificación de las colisiones se realiza la resta de vectores, con lo que se obtiene un vector del centro de una particula al centro de la otra. Se verifica si la magnitud de ese vector es menor a la suma de los radios de las partículas, de ser así se tendrá una colisión y se procede a realizar el cálculo de la misma. Esto se realiza mediante el cálculo de la fuerza ejercida entre ellas tomando en cuenta las masas, adicionalmente se mueven ligeramente las partículas para evitar solapamientos en siguientes iteraciones.

5. Simulación y recolección de datos

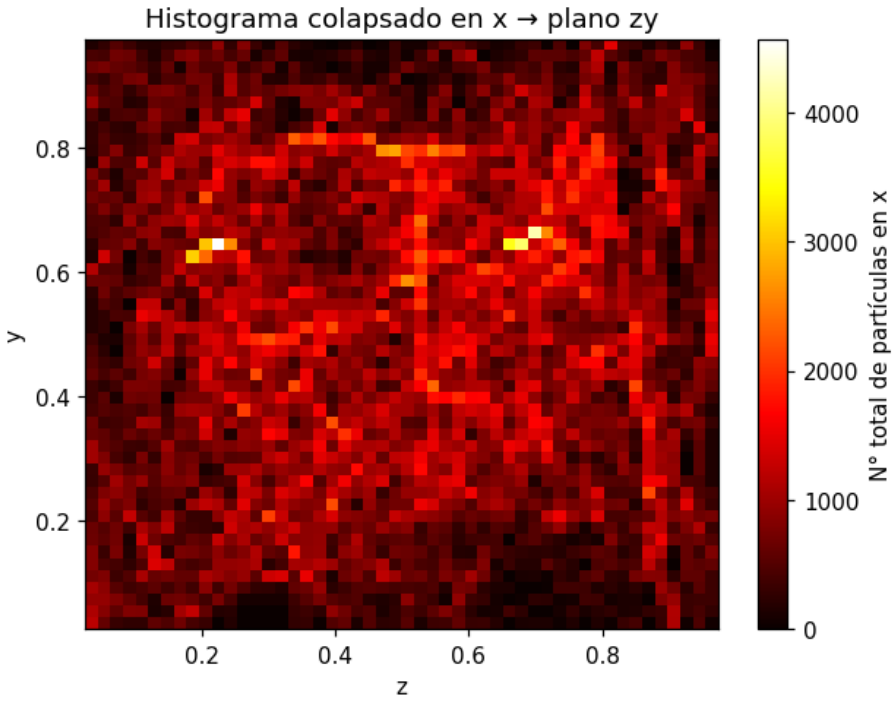

Figura 1: Histograma colapsado en la dirección x, mostrando la distribución total de partículas proyectada sobre el plano zy. El color indica la cantidad total de partículas acumuladas en cada celda del plano zy, según la barra de color a la derecha.

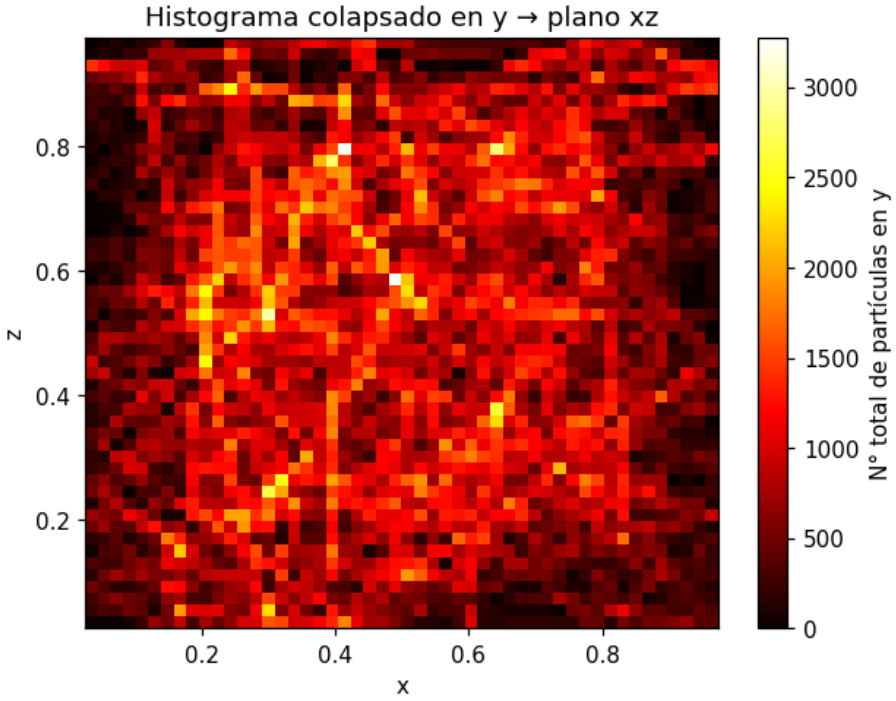

Figura 2: Histograma colapsado en la dirección y, mostrando la distribución total de partículas proyectada sobre el plano xz. El color indica la cantidad total de partículas acumuladas en cada celda del plano xz, según la barra de color a la derecha.

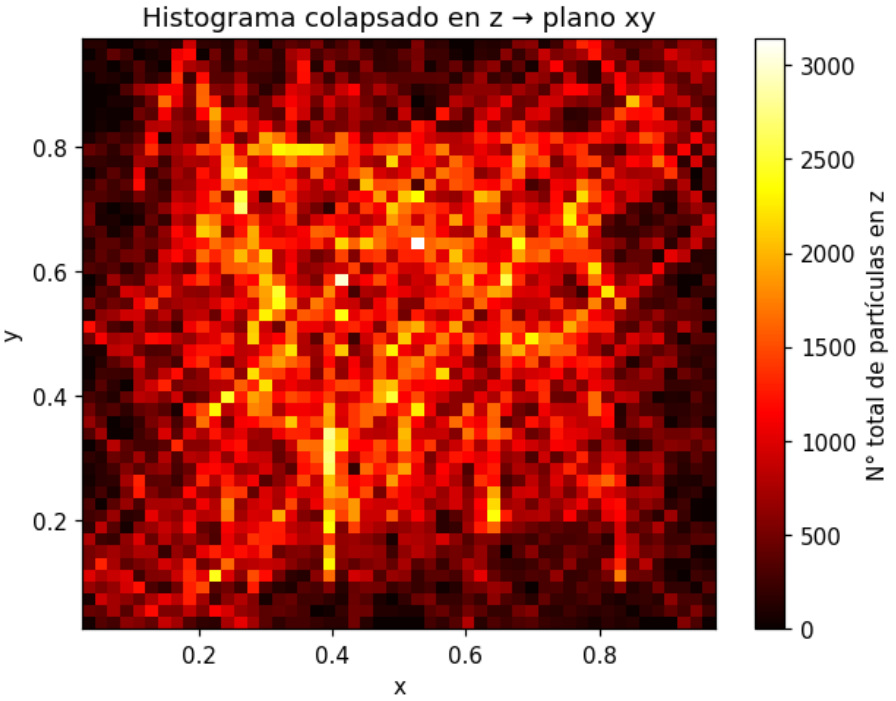

Figura 3: Histograma colapsado en la dirección z, mostrando la distribución total de partículas proyectada sobre el plano xy. El color indica la cantidad total de partículas acumuladas en cada celda del plano xy, según la barra de color a la derecha.

5.1. Análisis

Los histogramas colapsados sobre los planos xy, xz y yz permiten visualizar la densidad espacial de partículas acumulada durante la simulación. En ellos, cada color representa la cantidad de partículas proyectadas sobre una celda del plano correspondiente, proporcionando así una estimación de la distribución espacial del sistema.

Se observa una distribución aproximadamente uniforme en la mayoría del dominio, lo cual sugiere que el sistema ha evolucionado hacia un estado cercano al equilibrio estadístico, como es de esperarse en sistemas de muchas partículas con colisiones elásticas. Sin embargo, algunas regiones presentan acumulaciones ligeras, lo cual puede atribuirse a fluctuaciones estadísticas locales o a la formación temporal de cúmulos de partículas debido a múltiples colisiones sucesivas.

La simetría visual detectada en los histogramas también refuerza la hipótesis de que las condiciones iniciales fueron bien planteadas y que las condiciones de contorno periódicas o rígidas no introdujeron asimetrías en el sistema. Además, no se observan partículas fuera del dominio físico, lo que valida la correcta implementación del manejo de colisiones con paredes y entre partículas.

Desde una perspectiva computacional, los histogramas permiten evaluar la eficiencia del muestreo espacial. Se utilizó una cantidad suficiente de celdas (bins) para representar adecuadamente la resolución espacial sin introducir ruido excesivo. En particular, los histogramas fueron obtenidos a partir de una simulación con $N = 4$ partículas, un tamaño de dominio $L = 1$ y un tiempo total de simulación de $t = 10$ pasos. Estos parámetros aseguran una cobertura estadística razonable para realizar el análisis espacial.

En resumen, los resultados obtenidos no solo validan visualmente la simulación, sino que también permiten interpretar el comportamiento colectivo del sistema desde el punto de vista físico y estadístico.