Simulación del gas de Van der Waals en Java

Manuel Eduardo Gortarez Blanco

Fausto Misael Medina Lugo

Alan David Torres Flores

2 de diciembre de 2023

Índice general

Introducción

A lo largo del curso hemos querido simular un sistema físico, esto debido a que nos interesaba unir dos materias de este semestre como lo son Programación Avanzada y Fluidos y Fenómenos Térmicos.

Decidimos simular como se comportan las moléculas cuando modificamos uno de los valores del gas de Van der Waals ya sea la presión, volumen o la temperatura. Además decidimos hacerlo en java ya que es el lenguaje que estamos aprendiendo en el curso de Programación Avanzada.

Por otra parte el decidir hacerlo de tres personas fue de interés para aprender a trabajar en un proyecto como equipo, de esta manera usamos herramientas como lo son GitHub que nos servirá para futuros proyectos.

1 Física de Gases

1.1 Gas Ideal

Cuando nos introducimos al estudio de los gases lo hicimos mediante el modelo de "gas ideal", este modelo nos permitirá ver algunos comportamientos de los gases como lo son la presión, volumen y temperatura en donde consideramos que las moléculas no interactúan entre ellas y no tienen volumen. Es mediante los experimentos que nosotros podemos plantear este modelo, en estos concluimos varias cosas como lo son:

$$P \, \alpha^{-1} \, V$$

$$P \, \alpha \, T$$

Lo que implica que

$$P = \frac{k}{V}$$

$$P = kT$$

donde en ambos casos$k$ es una constante de proporcionalidad.

Ahora como el volumen y la temperatura son independientes entre si entonces podemos concluir que:

$$P = \frac{kT}{V} \implies PV = kT$$

donde k es una constante de proporcionalidad.
Una pregunta que surge entonces es saber quien es k, pues al igual que por medio de experimentos se concluye que:

$$k = nR$$

donde n es el numero de moles y R la constante de el gas ideal.

Podemos ver que si tenemos un gas en donde n es constante entonces la presión depende solamente del volumen y de la temperatura (mismo caso para la temperatura y el volumen) por lo que es bastante sencillo de calcular, además podemos facilitar aún mas los cálculos manipulando el gas mediante tres tipos de procesos:
-Isobárico(Presión constante) $\implies V = kT$
-Isotérmico(Temperatura constante) $\implies PV = k$
-Isométrico(Volumen constante) $P = kT$

1.2 Gas de Van der Waals

Este gas no lo deducimos teóricamente sino que se nos fue presentado por el profesor como otro modelo que también existía, además de ser mas completa ya que toma en consideración que

las moléculas interactúan entre ellas y que si tienen un volumen, su ecuación es la siguiente:

$$\left[P + a\left(\frac{n}{V}\right)^2\right]\left[\frac{V}{n} - b\right] = RT$$

donde a es una constante de interacción molecular y b del volumen de las moléculas.

Se puede ver de manera fácil que tanto la presión como la temperatura se pueden despejar de manera analítica, en cambio el volumen no se puede y por lo tanto habría que calcularlo numéricamente. Si despejamos llegaremos a esta ecuación:

$$\left(\frac{P}{n}\right)V^3 - (RT + Pb)V^2 + (a\,n)V - (a\,b\,n^2) = 0$$

Esta forma es conveniente ya que es encontrar las raíces de un polinomio de grado 3, ahora nos preguntamos, ¿cuántas raíces reales tiene este polinomio? si nos basamos en un gas real, es decir una a y una b que estén basadas en un gas real y escogemos tanto presiones como temperaturas que estén en el diagrama de fase dentro del estado podemos asegurar que hay sólo una raíz positiva, por lo que podemos aproximarla con un algoritmo que encuentra raíces, en nuestro caso decidimos utilizar Newton-Raphson.

1.3 Teoría cinética molecular

La teoría cinética molecular nos dice como se manifiesta la energía cinética de las moléculas a nivel macroscópico, en nuestro caso necesitamos saber que pasa a este nivel con la presión y la temperatura para poder simularlos en un programa, con la explicación del profesor de física tenemos lo siguiente:

- Presión: Es la manifestación macroscópica del choque promedio de las moléculas con las paredes del sistema.
- Temperatura: Es la manifestación macroscópica de la energía cinética promedio que contienen las moléculas.

Con esta información ya podemos pasar a la parte de como vamos a simular a nuestro gas.

2 Simulación

Para simular el gas de Van der Waals decidimos hacerlo en un pistón, esto con el fin de simplificarnos la siguientes cosas: podemos manejar el volumen como queramos y la masa del gas es constante. Con esto tiene sentido que solo podamos modificar la presión, temperatura y el volumen.
Como mencionamos anteriormente simularemos el gas a nivel microscópico, entonces tendremos un pistón que dentro tendrá pelotas que rebotarán contra la pared, para ver como se relaciona esto con la teoría cinética molecular pensaremos ahora en como se vería en una simulación didáctica:

- Presión: Es la frecuencia con la que chocan las pelotas con las paredes del pistón.
- Temperatura: Es la velocidad con las que se mueven las pelotas dentro del pistón.

También debemos considerar la interacción entre las moléculas, para nuestros fines será simplemente hacer que choquen entre ellas, pero queremos que sea de forma realista, entonces para ello hay que considerar dos cosas que básicamente son principios de física:

1. La conservación de la energía (cinética en nuestro caso).
2. La conservación del momento.

Además para simplificar nuestros cálculos de movimientos tendremos en cuenta que el choque entre las moléculas es completamente elástico, es decir, no perderemos energía cinética en la deformación del objeto por ejemplo. Por último para nuestra simulación tendremos que todas las pelotas son del mismo tamaño por lo tanto sus masas son iguales.

2.1 Desarrollo de las ecuaciones de movimiento

Obtendremos las ecuaciones de los vectores de velocidad dadas dos moléculas que interactúan entre sí, entonces

$$m_1\, v_1 + m_2\, v_2 = m1\, v_1' + m2\, v_2' \tag{1}$$

$$\frac{1}{2}m_1\, v_1 + \frac{1}{1}m_2\, v_2 = \frac{1}{2}m1\, v_1' + \frac{1}{2}m2\, v_2' \tag{2}$$

Dado que se requiere mucha álgebra para resolver el sistema formado por las ecuaciones (1) y (2) nos aprovechamos de que alguien mas lo había resuelto (Reducible, 2021), por lo que el sistema resuelto queda de la siguiente manera:

$$v_1' = v_1 - \frac{< v_1 - v_2, C_1 - C_2 >}{||C_1 - C_2||^2}(C_1 - C_2)$$

$$v_2' = v_2 - \frac{< v_2 - v_1, C_2 - C_1 >}{||C_2 - C_1||^2}(C_2 - C_1)$$

Hay que especificar que C es el vector de posición de la molécula, v es el vector de velocidad actual y v' es el vector de velocidad actualizado después de la colisión.

Ahora para el choque con las paredes es algo sencillo: -Si hay colisión con las paredes verticales entonces invertimos la componente y del vector de velocidad. -Si hay colisión con las paredes laterales entonces invertimos la componente x del vector de velocidad. Con estas simples herramientas ya podemos empezar a hacer el código de nuestra simulación.

3 Programa

El lenguaje que se utilizó para la simulación fue Java. La elección de este lenguaje se debió a que es la que se nos está enseñando en programación avanzada, además, la programación orientada a objetos nos sonaba interesante para este proyecto, para esto hay que definir una estructura de proyecto, osea sus paquetes, clases, métodos y variables.

3.1 Paquetes

En total tenemos una cantidad de 4 paquetes en donde hay clases que consideramos que comparten características entre ellas, además suelen interactuar mucho entre ellas por lo que tenerlas en un mismo paquete es de gran ayuda para la organización e implementación del proyecto, los paquetes son los siguientes:

1. GUI: El paquete que alberga todas las clases que hacen el componente gráfico del proyecto (Ventanas, botones, el dibujo de los objetos, etc).

2. SistemaTermodinamico: Este se encarga de todo lo que es el sistema termodinámico que estamos simulando es decir cosas como el pistón, el gas, la molécula, etc.

3. Ciclos: Este es el encargado de animar al pistón con los ciclos como por ejemplo el de Stirling.

4. Escuchadores: La encarga de estar pendiente de cuando se hace un cambio en el estado termodinámico del sistema.

Con esta cantidad de paquetes creemos que es suficiente para clasificar correctamente a las clases.

3.2 Clases y sus métodos principales

Aquí describiremos todas las clases que están en cada uno de los paquetes, explicaremos los métodos principales que utilizan con el fin de que se pueda entender en general que hace exactamente esa clase.

3.2.1 GUI

1. VentanaDibujo: Se encarga de dibujar todo lo gráfico como lo son las moléculas, el pistón y el gas. Su método principal:

    - public void paintComponent(Graphics grafico): este método es el encargado de dibujar todo en pantalla

2. PanelVariables: clase que se encarga de la interacción del usuario con las variables termodinámicas del sistema, hereda de Thread. Su método principal:

- public void run(): se encarga de que almacenar y modificar los valores que el usuario vaya seleccionando.

3. Figura: es una interfaz que implementaran todas las clases de pistón, gas y molécula. Su único método es:

    - public void pintar(Graphics gráfico): Método que informará al método de paintComponent como debe de ser pintado lo que implementa esa interfaz.

3.2.2 SistemaTermodinamico

1. Gas: es el encargado de dibujar la ”nube” que esta dentro del pistón. También es el encargado de administrar el calculo de las variables cuando el usuario interactúa con ellas, además hereda de Thread. Sus métodos principales son:

    - private double aproximarVolumen(): Calcula el volumen numéricamente con Newton-Raphson.
    - private GradientPaint cambiarColor(): Cambia el color de forma gradiente en función de la temperatura.
    - public void calcularVariables(String, int): se encarga de calcular las variables en función del tipo de proceso en que se esta modificando al sistema.
    - public void pintar(Graphics): pinta la nube del gas.
    - public void run(): repinta la nube del gas.

2. Molecula: simula a una molécula en forma de pelota, además extiende de Thread. Sus métodos principales son:

    - public void pintar(Graphics): dibuja un circulo con contorno negro.
    - private Color cambiarColor(): actualiza el color de la molécula en función de la temperatura.
    - public void actualizarVelocidad(): actualiza la velocidad en función de la temperatura, además hace que rebote en las paredes de forma realista.

- public void run(): llama a actualizarVelocidad() y repinta a la molécula.

3. Piston: se encarga de dibujar al pistón, extiende de Thread. Sus métodos principales son:

   - public void setVolumen(double, double): se aplasta o expande en función del volumen calculado.
   - public void pintar(Graphics): se encarga de pintar un pistón.
   - public void run(): repinta al pistón.

4. Colision: se encarga de administrar las colisiones entre las moléculas, extiende de Thread. Sus métodos principales son:

   - private boolean verificarColision(Molecula, Molecula): predicado que nos dice si dos moléculas están en colisión o no.
   - modificarVelocidad(Molecula, Molecula): realiza la colisión de forma realista entre dos moléculas.
   - public void run: administra todo el proceso de las colisiones entre todas las moléculas.

3.2.3 Ciclos

1. Lector: Se encarga de realizar la lectura de archivos para poder realizar el ciclo de Erricsson y Stirling.

2. Estado: Administra los ciclos para que se realice una animación de ellos.

3.2.4 Escuchadores

1. EscuchadorVentana: Se encarga de realizar acciones cuando el usuario interactúa con la ventana, extiende a WindowAdapter. Se encarga de iniciar, congelar o parar todos los hilos del programa dependiendo del estado de la ventana como por ejemplo cuando se minimiza.

Conclusión

Realizar este proyecto nos resultó profundamente satisfactorio. Nos fue todo un reto el realizar una simulación de forma realista (tanto los cálculos como las colisiones) pero al final pudimos resolverlo de alguna manera. También aprendimos a usar la plataforma de GitHub de una manera mas eficiente teniendo como consecuencia el aprender a realizar un proyecto, junto con la documentación de este mismo. Por último pero no menos importante consolidamos nuestros conocimientos sobre la física de gases así como los de la programación orientada a objetos, y no solo eso, logramos conjugarlos para un mismo fin, osea realizar este proyecto.

Nos gustaría en un futuro cuando tengamos mayores conocimientos de estructura de datos implementar un algoritmo de optimización para las colisiones, necesitamos una estructura llamada quadtree pero debido al tiempo y la dificultad no pudimos implementarlo. De haberlo hecho nuestros cálculos de colisiones hubiera pasado de $\mathcal{O}(n^2)$ a $\mathcal{O}(n \log n)$ donde n es el número de moléculas, de esta manera lograríamos ejecutar el programa en máquinas menos potentes.

Bibliografía

Reducible (2021, 19 de enero). *Building Collision Simulations: An Introduction to Computer Graphics* (Video). YouTube. https://www.youtube.com/watch?v=eED4bSkYCB8.