Simulación en física de materiales

Daniel Vazquez Lago

25 de octubre de 2024

Índice

1. Objetivos

El objetivo de esta segunda entrega de la asignatura es llegar a la posición de equilibrio de un sistema de 500 partículas, con una densidad $\rho = 0.5$ y una energía $E = 575$ (variables reducidas) a partir de una posición inicial fcc. Sin embargo para llegar al equilibrio primero debemos avanzar en el tiempo, es decir, hacer una simulación. En este proyecto veremos qué debemos implementar para poder calcular posiciones temporales sucesivas hasta la posición deseada. En el optativo 1 dos veremos si, realmente con los pasos logrados en este ejercicio se llega o no al equilibrio.

2. Organización

En esta sección mencionare como están organizados el proyecto. En primer lugar hay que decir que, por motivos de organización personal, he creado una carpeta llamada *Datos* donde se almacenan todos los `.dat` de todos los proyectos. Sin embargo para que se guarden correctamente he supuesto que se ejecuta desde el proyecto, esto es, desde el `.exe` que se genera al construir el proyecto. De hacerlo de otro modo esto llevaría a un error. En segundo lugar, se puede ver que tal y como recomendó el profesor, hay una carpeta donde guardo todos los archivos `.f95` (llamada *Programas fuente*) y otra donde esta el proyecto `.ftn95p`, para así poder reusarlos y que esté bien ordenado. Una breve descripción de los archivos:

- **Mod_01_Def_prec**: modulo usado para definir las precisiones doble precisión y entero, permitiéndo usar datos con más cifras decimales.
- **Mod_02_Variables_comunes**: nos permite pasar las variables comunes entre las diferentes subrutinas y programas sin necesidad de definirlos en los propios programas.
- **Mod_03_Interface**: módulo que nos permite detectar errores a la hora de introducir datos en las subrutinas, así como detectarlos en las variables de salida de las mismas.
- **Sub_Potlj:** subrutina en la cual introducimos las posiciones y el número de partículas como variables de entrada y nos devuelve las aceleraciones/fuerzas, así como la energía potencial del sistema y sus derivadas respecto al volumen (primera y segunda).
- **Sub_Verlet:** subrutina la cual recibe una configuración inicial en el instante t y devuelve la nueva configuración inicial en $t + \Delta t$ (para esto esta subrutina llama a `Sub_Potlj`). Además también obtenemos con ella la energía potencial.

- **Pro_Equilibracion:** programa principal. Su principal función es leer la configuración inicial (partiendo de la fcc o bien de alguna más estable, siempre que verifique $E = -575$), y mediante un lazo iterar el número de veces que se desee la subrtuina `Sub_Verlet`, avanzando un tiempo total $k\Delta t$, donde k es el número de pasos y Δt el intervalo temporal entre cada uno. Además escribirá en el mismo archivo que se leyó la configuración final, así como en nuevos archivos de datos las energías cinética, potencial y total; así como las velocidades (entre otros datos), en pos de ver si realmente se ha equilibrado.

3. Simulación

En esta sección vamos a ver como se hace la simulación, esto es, como calculamos la posición, velocidad... en la configuración $t + \Delta t$ a partir de los datos en t (o anteriores, esto es, $t - \Delta t,\ t - 2\Delta t$). En realidad, obtener la posición temporal siguiente no es otra cosa que resolver una ecuación diferencial (ecuaciones del movimiento de las partículas) a través de metodos diferenciales finitos, que son los únicos con los que puede trabajar el ordenador. Estos métodos diferenciales finitos nos llevarán a dos errores diferentes que tendremos que minimizar todo lo que podamos con el propósito de obtener la simulación mas acertada con el modelo del que disponemos.

De esta manera en la primera parte de la sección discutiremos desde un punto de vista completamente teórico qué método vamos a usar y como vamos a minimizar el error, mientras que en la segunda parte vamos a explicar brevemente como lo implementamos en fortran.

3.1. Introducción teórica

Para conocer la posición de las partículas en el instante de $t+\Delta t$, si Δt es suficientemente pequeña, podemos desarrollar la serie de Taylor de $\mathbf{r}(t + \Delta t)$

$$\mathbf{r}(t + \Delta t) = \mathbf{r}(t) + \dot{\mathbf{r}}(t)\Delta t + \frac{1}{2}\ddot{\mathbf{r}}(t)(\Delta T)^2 + \frac{1}{6}\dddot{\mathbf{r}}(t)(\Delta T)^3 + \dots \tag{1}$$

Conocidos los valores de $\mathbf{r}, \dot{\mathbf{r}}, \ddot{\mathbf{r}}$... podremos calcular la nueva posición. Sin embargo esto nos lleva a dos preguntas ¿A qué nos referimos con Δt suficientemente pequeña?¿Qué valor le asignamos a las variables $\dot{\mathbf{r}}$...? La primera pregunta la responderemos en el apartado 3.1.1 mientras que la segunda en el apartado 3.1.2.

3.1.1. Minimización del error y estabilidad

Existen dos tipos de errores asociados a los algoritmos de metodos diferenciales finitos: el error de trucación global (egt) y el error de redondeo global (ert).

- El error de truncación viene determinado por el método que usemos para resolver la ecuación diferencial. En general el orden del error de truncación viene depende del orden en el que cortemos el desarrollo de Taylor (1). Nosotros tendremos en cuenta hasta $(\Delta T)^2$, por lo que el error de truncamiento será generado por los términos de orden 3 o superior. Cuanto más grande sea Δt, más grande será el error de truncamiento. De hecho, si $\Delta t > 1$, tendremos que el error de truncamiento será infinito. Por eso mismo es obligatorio que se verifica que $\Delta t < 1$. Cuan más bajo sea Δt este error será mucho menor. Otra manera de minimizar este error sería coger términos de orden superior, tal y como se hace en el algoritmo Gear.

- El error redondeo depende en última instancia de las cifras tenidas en cuenta a la hora de trabajar el ordenador. Este error se hace más grande a medida que Δt disminuye, ya que llega un momento en el que el error por trabajar con números finitos se hace cada vez más significativo en Δt.

Otra elemento que afecta a las simulaciones es la estabilidad. Además de tratar de reducir el error, es importante ver como el algoritmo propaga los diferentes errores. Si el error que se propaga con cada interacción es cada vez mayor, diremos que el algoritmo es inestable. Por otro lado, si los errores no aumentan, cancelándose mutuamente, diremos que el método es estable. Dado que muchas veces la dinámica molecular trata con ecuaciones diferenciales no ordinarias, hacer una análisis soobre la estabilidad analíticamente puede ser complicado. La mejor manera de conocer el Δt crítico (aquel a partir del cual el sistema es inestable) es hacer diferentes pruebas.

Consecuentemente ni podemos tomar un Δt muy grande, ni tampoco un Δt ridículamente pequeño. La única manera de saber cual es el mejor Δt será haciendo diferentes pruebas (tanto para la estabilidad como para el error global), y ver cual funciona mejor. Nosotros usaremos $\Delta t = 0.0001$, que, como veremos, funciona bastante bien.

3.1.2. Algoritmo velocity Verlet

Para conocer los valores de los diferentes $\dot{\mathbf{r}}$ debemos usar las ecuaciones canónicas de nuestro hamiltoniano $\mathcal{H}$ (aunque en PVE el resultado sea trivial, puede que no lo sea en otras colectividades), dado por la energía cinética y un potencial que solo depende de la posición:

$$\mathcal{H} = \sum_{i=1} \frac{\mathbf{p}_i^2}{2m} + \varphi(\mathbf{r}_i) \tag{2}$$

donde $\varphi(\mathbf{r}_i)$ viene dado por el *potencial Lennard-Jones*. Si definimos la velocidad como $\mathbf{p} = m\mathbf{v}$ y la aceleración como $\ddot{\mathbf{r}}_i = \mathbf{a}_i$, tenemos que:

$$\dot{\mathbf{r}}_i = \frac{\partial \mathcal{H}}{\partial \mathbf{p}_i} = \frac{\mathbf{p}_i}{m} = \mathbf{v}_i \tag{3}$$

$$\dot{\mathbf{p}}_i = -\frac{\partial \mathcal{H}}{\partial \mathbf{r}_i} \Rightarrow m\ddot{\mathbf{r}}_i = -\frac{\partial \varphi(\mathbf{r}_i)}{\partial \mathbf{r}_i} \Rightarrow \mathbf{a}_i = -\frac{1}{m}\frac{\partial \varphi(\mathbf{r}_i)}{\partial \mathbf{r}_i} \tag{4}$$

Reescribimos (1) como

$$\mathbf{r}(t + \Delta t) = \mathbf{r}(t) + \mathbf{v}(t)\Delta t + \frac{1}{2}\mathbf{a}(t)(\Delta T)^2 \tag{5}$$

donde hemos despreciado los términos de orden $(\Delta T)^3$ y superiores (que son los que generan el error). Conocido $\mathbf{r}(t + \Delta t)$ calcular $\mathbf{a}(t + \Delta t)$ es trivial, ya que

$$\mathbf{a}(t + \Delta t) = -\frac{\partial \varphi(\mathbf{r}_i)}{\partial \mathbf{r}_i(t + \Delta t)} \tag{6}$$

y además es trivial también el nuevo valor de la energía potencial. Para calcularlo solo tendríamos que usar la suburtina implementada en el anterior ejercicio (`Sub_Potlj`) con las nuevas posiciones.

Para obtener el valor de la velocidad podríamos usar el resultado más intuitivo según lo dicho, que sería $\mathbf{v}(t+\Delta t) = \mathbf{v}_i(t) + \mathbf{a}_i(t)\Delta t$. Sin embargo este no es estable (véase apartado 3.1.1 para saber más de la estabilidad). Un promedio que sí es estable sería:

$$\mathbf{v}(t+\Delta t) = \mathbf{v}(t) + \frac{1}{2}\left(\mathbf{a}(t+\Delta t) + \mathbf{a}(t)\right) \tag{7}$$

Este es el denominado **algoritmo velocity Verlet**. Aunque existen mas algoritmos, nosotros usaremos este porque es el más sencillo de implementar y funciona bien, sobretodo a bajas energías, como la nuestra. Escribiendo las ecuaciones todas juntas vemos que

$$\begin{aligned}
\mathbf{r}(t+\Delta t) &= \mathbf{r}(t) + \mathbf{v}(t)\Delta t + \frac{1}{2}\mathbf{a}(t)(\Delta T)^2 \\
\mathbf{v}(t+\Delta t) &= \mathbf{v}(t) + \frac{1}{2}\left(\mathbf{a}(t+\Delta t) + \mathbf{a}(t)\right) \\
\mathbf{a}(t+\Delta t) &= -\frac{\partial\varphi(\mathbf{r}_i(t+\Delta t))}{\partial \mathbf{r}_i(t+\Delta t)}
\end{aligned}$$

3.2. Resultados

Una vez corremos la simulación ($k_{\text{pasos}} = 5000$, $\Delta t = 10^{-4}$), obtenemos los resultados de la energía que vemos en la figura 1:

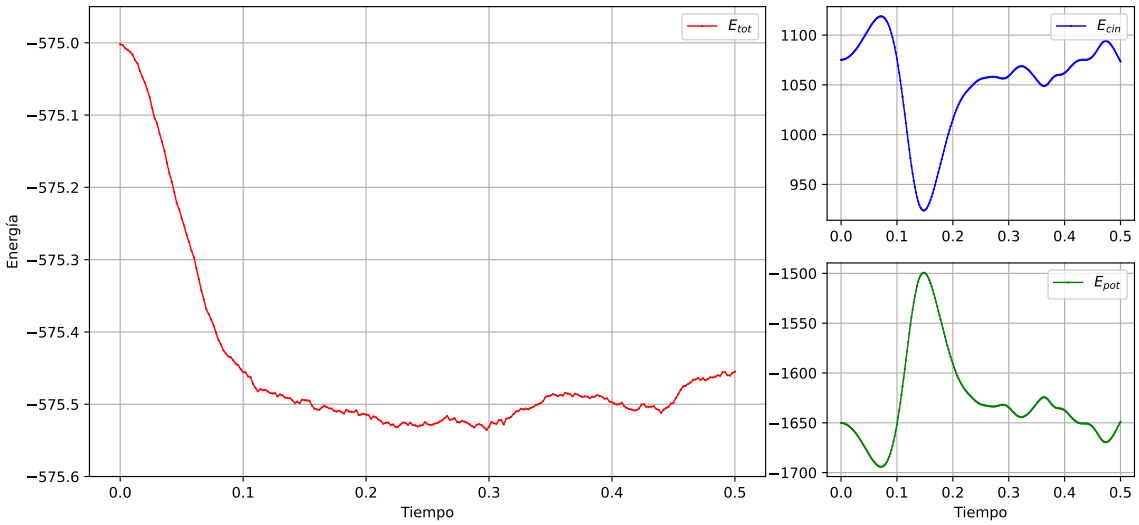

Figura 1: evolución de la energía total, cinética y potencial desde la configuración fcc y 5000 pasos.

Como podemos ver existe una caída de energía total hacia -575.5 sobre la cual, tras los 2000 pasos comienza a oscilar, es decir, se comienza a equilibrar en -575.5. La razón por la que sucede esta caída tan repetida es que la configuración inicial es muy simétrica, y por tanto está muy lejos del equilibrio, por lo que su evolución, y los datos que podamos obtener de ella no corresponderá a una colectividad microcanónica de $E = -575$. Entonces lo que tenemos que hacer, para que la simulación tenga la energía deseada, será reescalar las velocidades reescalar las velocidades (adecuándose a la energía cinética que necesitamos para que $E = 575$), asegurándonos que el momento total se mantenga cero, al igual que hicimos en el ejercicio anterior (mismo algoritmo mismo proceso).

Como podemos ver, a diferencia de la anterior inicialización, en esta no habrá un salto de energía, ya que al haber dejado que ocurrieran 5000 pasos (aunque fuera en otra colectividad

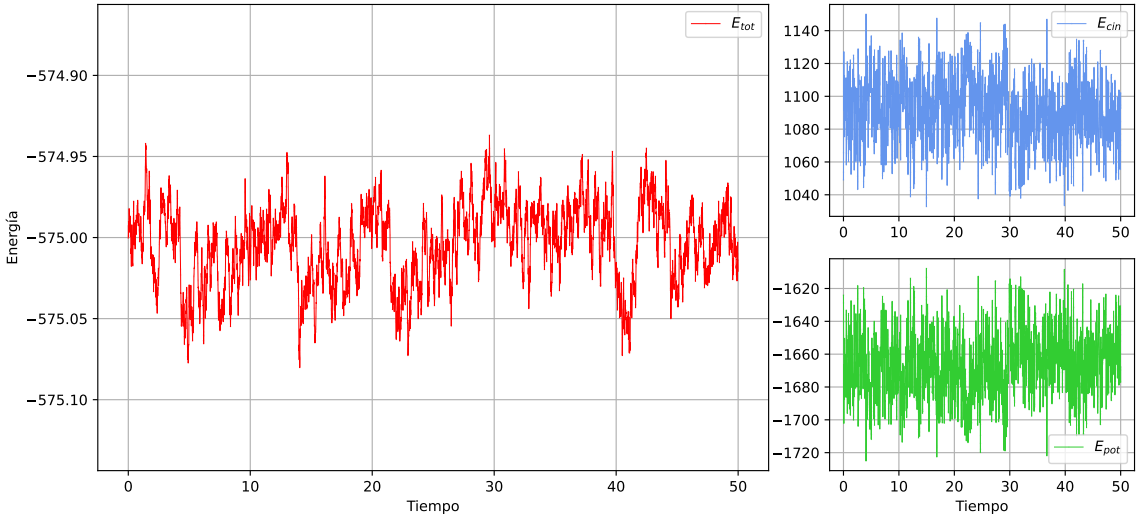

Figura 2: evolución de la energía total, cinética y potencial 500000 pasos.

microcanónica), la configuración de las partículas ya no es simétrica, y por tanto estará mucho más cerca del equilibrio en la colectividad deseada, por lo que ahora podemos hacer un número mucho más elevado de pasos. Con 5×10^5 serán suficientes. Ahora lo que resta sería usar los datos de las energías cinéticas, potenciales, totales, así como las velocidades de cada una de las 500 partículas en las 3 direcciones (entre otros) para ver si realmente hemos llegado al equilibrio.

4. Conclusiones

Una vez dicho todo esto ya podremos dar por concluido el proceso por el que llegamos a la equilibración, ya que los resultados en principio son satisfactorios, aunque la última palabra sobre si realmente hemos llegado a la equilibración la tendrá el optativo 1.

Referencias

[1] J. M. Haile. *Molecular Dynamics Simulation.*