Simulación en física de materiales

Daniel Vazquez Lago

26 de octubre de 2024

Índice

1. Objetivos

El objetivo de este primer optativo es estudiar si realmente, tras los 5×10^5 pasos, hemos llegado a una configuración de equilibrio. Para esto presentaremos 3 diferentes condiciones que un sistema aislado en equilibrio debería satisfacer ([1]):

- La energía E total debe mantenerse constante en el tiempo, a poder ser $E = -575$.
- Los valores medios de las velocidades (cada una de las componentes) deben seguir distribuciones de Maxwell.
- La función H de Boltzmann debe tender a un valor constante.

Existen mas condiciones que no trataremos aquí, no por otra cosa que la falta de información. Sin embargo, ninguna de estas (individualmente y colectivamente) nos garanticen de manera completamente inequívoca que estamos en el equilibrio.

2. Equilibración

En esta sección vamos a estudiar desde una perspectiva teórica cuales son las condiciones de equilibrio, para luego ver cuales son los resultados (y la implementación) según cada una de estas condiciones. En esta sección usaremos sobretodo el contenido que aparece en las páginas 206-209 del Haile [1].

2.1. Energía total constante

2.1.1. Teoría

Esta condición nos asegura que el si E es constante en el tiempo, de tal manera que las fluctuaciones de la energía cinética y energía potencial deben ser contrarias, esto es, cuando una de ellas aumente la otra debe disminuir y viceversa.

2.1.2. Resultados e implementación

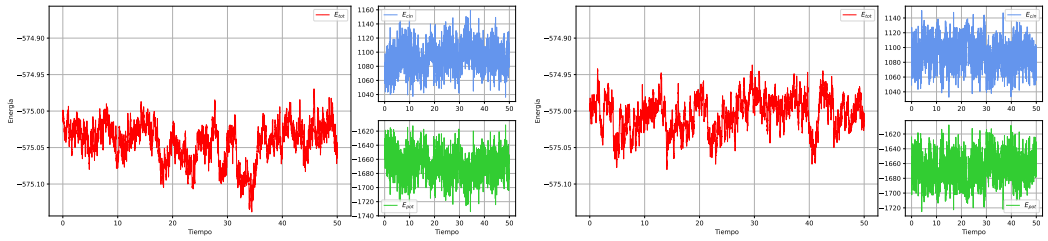
Esta es, de las tres condiciones, la que mas sencilla es de implementar, ya que en el obligatorio 2 ya se nos exigía escribir los distintos valores de la energía para cada una de las interacciones (incluida la que venía de la fcc). En este caso lo único que debemos hacer es leer los datos almacenados en el `.dat` para luego hacer las gráficas (usando matplotlib.pyplot), y obtener el valor medio de las energías, también en python. Los valores medios de las energías totales y cinéticas (las cinéticas serán interesantes para la siguiente condición de equilibrio):

	E_t	$s(E_t)$	E_{cin}	$s(E_{cin})$
1^o	-575.0422	6.9e-09	1094.15	2.5e-08
2^o	-575.0044	1.0e-08	1091.00	1.8e-08

Tabla 1: Valores de las medias para cada interacción e incertidumbre asociada a la media.

Una vez tenemos estos datos podemos entender mejor porque realizamos una segunda interacción. Dado que nosotros queremos una dinámica molecular que oscile sobre -575, queremos que el valor medio de E_t se parezca lo más posible a -575. Por ello que oscilase en la 2a cifra decimal no nos parecía suficiente.

Aún con todo, podemos ver que las oscilaciones respecto la media son muy pequeñas, por lo que en ambas configuraciones se verifica la condición de estabilidad en la energía del equilibrio, aunque no el mismo equilibrio (ambas pertenecen a colectividades microcanónicas diferentes, ya que la NVE depende de E y en E es diferente en las dos equilibraciones).



(a) Primeros 500K pasos. (b) Segundos 500K pasos.

Figura 1: Evolucion de la energía total (y cinética y potencial) para los primeros 500K pasos y para los segundos 500K pasos.

2.2. Distribución de Velocidades

2.2.1. Teoría

En este apartado vamos a seguir el desarrollo hecho en el [1] (pag. 65-66, pag. 120). Para una distribución de velocidades en el equilibrio (colectividad NVT) verifica que, para cada una de las componentes i de las velocidades (v_x, v_y, v_z), dado que son independientes:

$$f(v_i) = \sqrt{\frac{m}{2\pi kT}} \exp\left(-\frac{mv_i^2}{2kT}\right) \tag{1}$$

Aunque la distribución de Maxwell halla sido derivada en el formalismo de la colectividad canónica (NVT, PVT...), la distribuciones de cualquier sistema en el equilibrio debería ser

de este tipo, ya que en el límite termodinámico las propiedades en las diferentes colectividades se vuelven equivalentes.

Una de las consecuencias de la distribución de velocidades es la equipartición de la energía, por lo las componentes cartesianas de la velocidades moleculares deberían tener un valor medio igual al de la temperatura del sistema:

$$\frac{1}{N}\left\langle \sum_i v_{ix}^2 \right\rangle = \frac{1}{N}\left\langle \sum_i v_{ix}^2 \right\rangle = \frac{1}{N}\left\langle \sum_i v_{ix}^2 \right\rangle = T \tag{2}$$

2.2.2. Implementación

Para implementar correctamente el programa necesitamos primero saber que tenemos y que queremos obtener. Como bien dijimos en el anterior apartado, queremos obtener una distribución de velocidades, para verificar si tiene un comportamiento gaussiano o cuasi-gaussiano (y ver si coinciden las distribuciones para los diferentes ejes, ya que en el caso de que no coincidan sabremos que la simulación no es correcta). Tenemos en total 2.5M de velocidades por eje, ya que hemos guardado 5001 interacciones (desde $t = 0$ hasta $t = 50$ teniendo en cuenta que $\Delta t = 10^{-4}$ y que almacenamos los datos de las velocidades cada 100 interacciones).

Entonces lo que debemos hacer en el programa es lo siguiente:

1. Leemos los diferentes valores de las velocidades, y las apilamos todas en un mismo array (no vamos a estudiar la evolución temporal de la distribución, si no la distribución global).
2. Definimos los parámetros del histograma:
   - Número de intervalos/bins. Nosotros usaremos la fórmula clásica: el número de intervalos será la raíz cuadrada del número total de valores. En nuestro caso entonces tendremos 1582 intervalos.
   - Límites de la muestra. Para calcular los límites de la muestra cogimos el valor mas grande (en absoluto) de todas las velocidades posibles (7.5M datos), y lo usamos tanto como valor máximo y valor mínimo de la muestra.
   - Ancho de los bins. Calculamos la anchura total de la muestra (límite superior menos límite inferior) y lo dividimos por el número de bins total.
   - Valor medio de cada bin. Valor al que asociaremos las velocidades más proximas. De define como el valor mitad entre el límite superior e inferior de un bin.
   - Frecuencias. Cada bin tendrá asociada una frecuencia en función de la cantidad de valores que haya más próximos al valor medio del mismo que a cualquier otro.
3. Una vez definidos los valores procedemos a rellenar los histogramas (esto es, definir la frecuencia de cada bin). Nosotros lo que hicimos fue hacer un bucle para cada velocidad (3 bucles), donde leemos paso por paso cada una de los valores de las velocidades (por esa misma razón las tuvimos que apilar), de tal manera que en cada interacción recorra cada uno de los bins, viendo si es más pequeño que el límite superior del bin, empezando por el más pequeño. En el momento que encontramos el bin al que asociar dicho valor, salimos del bucle. La implemetnación en fortran es pequeña:

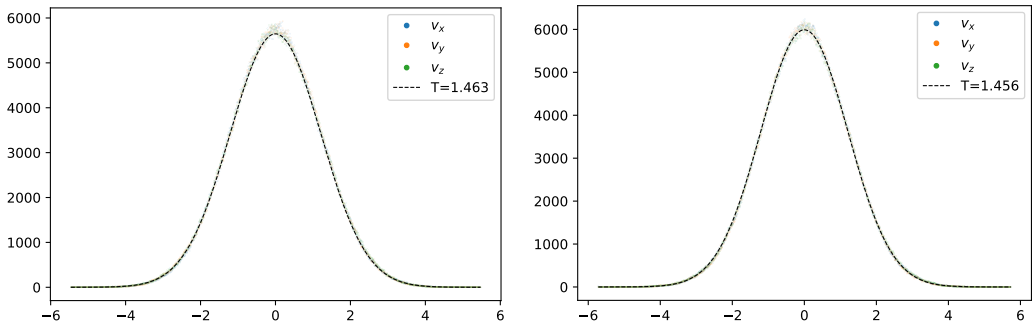
```
do i=1,500*5001
  v=vxx(i)
  do j=1,kintervalos_int
  if (v<=-distancia/2.d00+intervalo*j) then
    histx(j)=histx(j)+1.d00
    exit
  endif
  enddo
enddo
```

4. Una vez tenemos la frecuencia, guardamos en 3 archivos el valor medio de cada bin y la frecuencia asociada, de tal modo que tenemos ya nuestros histogramas. Cada archivo contiene el histograma de cada una de las velocidades.

Cabe destacar que nosotros no usamos el algoritmo mas eficiente posible para obtener la frecuencia de los bins, pero si el más sencillo de implementar. De tener mas datos habría que cambiar el método. En general no es un programa eficiente, pero funciona, y dado que el tiempo de cálculo es pequeño[1], no es relevante.

2.2.3. Resultados

Las gráficas obtenidas a partir de los resultados dados por fortran se hicieron en python mediante un programa que leía directamente del `.dat`. Como hemos mencionado, hicimos dos interacciones. En ambas, tal y como se puede ver en las gráficas, para ambas las distribuciones coinciden entre sí, lo cual es un buen indicio.



(a) primeros 500K pasos.

(b) segundos 500K pasos.

Figura 2: valores de la distribución de velocidades para cada eje, con la distribución de Maxwell asociada. Como podemos ver el resultado no está normalizado.

Veamos ahora si coinciden con la distribución de Maxwell asociada. Para calcular la temperatura necesaria para saber la distribución de Maxwell de las velocidades usamos la energía cinética media anteriormente calculada. Como podemos ver la distribución de Maxwell encaja casi de manera perfecta en ambas distribuciones, lo cual significa que verifican la condición previamente dicha.

$$T = \frac{2\langle E_{cin}\rangle}{fk} \tag{3}$$

donde f es el número de grados de libertad del sistema y k la constante de Boltzmann. Supondremos que es un valor suficientemente bueno.

[1] Entre 5-6 segundos.

2.3. Distribución H-Boltzmann

2.3.1. Teoría

En este apartado vamos a seguir el desarrollo hecho en el [1] (pag. 66-67, pag. 120-121). Definimos la función H de Boltzmann (función H) en un instante t como:

$$H_i(t) = \int_{-\infty}^{\infty} f(v_i)\ln(f(v_i))\mathrm{d}v_i \qquad i = x, y, z \tag{4}$$

de tal modo que podemos definir la H de Boltzmann como

$$H = \frac{1}{3}\left(H_x + H_y + H_z\right) \tag{5}$$

El **teorema H de Boltzmann** asegura que una distribución llega al equilibrio cuando se alcanza un mínimo en la función H de Boltzmann, dada por la anterior ecuación. Sin embargo nosotros no podemos implementar integrales, si no sumatorios (las integrales tienen un carácter infinitesimal que no podemos alcanzar con el ordenador), por lo que nosotros tendremos que implementar:

$$H_i(t) = \sum_{\Delta v_i} f(v_i)\ln(f(v_i))\Delta v_i \qquad i = x, y, z \tag{6}$$

donde

$$f(v_x)\Delta v_x = \frac{1}{N}\sum_{i}^{N}\delta(v_x - v_{xi})\Delta v_x \tag{7}$$

e igual para v_y e v_z.

2.3.2. Implentación

La implementación es prácticamente idéntica a la de la distribución de velocidades, salvo por el último paso. Para implementar correctamente el programa necesitamos primero saber que tenemos y que queremos obtener. Como bien dijimos en la teoría, queremos obtener la función H de Boltzmann, ya que H se mantenga constante en el tiempo es una condición para que estemos en el equilibrio. La función H de bolztmann se obtiene conociendo la distribución de de velocidades en un instante temporal determinado, por lo que lo que deberemos cambiar es el final del programa

Entonces lo que debemos hacer en el programa es lo siguiente:

1. Leemos los diferentes valores de las velocidades, y las apilamos todas en un mismo array (no vamos a estudiar la evolución temporal de la distribución, si no la distribución global).

2. Definimos los parámetros del histograma:

   - Número de intervalos/bins. En este caso bamos a disminuir el número de bins. En este caso cogeremos entorno a 200 bins, de tal modo que $\Delta x \approx 0.05$. Hacemos esto por que el Haile [1] usa este, y queremos comparar los datos obtenidos (pag. 121) con sus resultados, aunque el use un modelo de esferas duras en esa páginas.
   - Límites de la muestra. Igual que en distribución de velocidades.

- Ancho de los bins. Igual que en distribución de velocidades.
- Frecuencias. Igual que en distribución de velocidades.

3. Una vez definidos los valores procedemos a rellenar los histogramas (esto es, definir la frecuencia de cada bin). Nosotros lo que hicimos fue hacer un bucle para cada velocidad (3 bucles), donde leemos paso por paso cada una de los valores de las velocidades (por esa misma razón las tuvimos que apilar), de tal manera que en cada interacción recorra cada uno de los bins, viendo si es más pequeño que el límite superior del bin, empezando por el más pequeño. Dado que tenemos que implementar la evolución temporal, lo que haremos, a diferencia del anterior, es incluir un if al final del segundo lazo, de tal manera que si llevamos ya 500 partículas ponga a cero el histograma, y guarde el valor de $H_i(t)$. Podemos ver de manera sencilla la implementación:

```
do i=1,500*5001
  v=vxx(i)
  do j=1,kintervalos_int
    if (v<=-distancia/2.d00+intervalo*j) then
      histx(j)=histx(j)+1.d00
      exit
    endif
  enddo
  if (modulo(i,500).eq.0) then
    do j=1,kintervalos_int
      if (histx(j).ne.0.d00) then
        Hx(int(i/500))=Hx(int(i/500))+intervalo*(log(histx(j)/np))*
    histx(j)/np
      endif
    enddo
    histx=0.d00
  endif
enddo
```

4. Una vez tenemos $H_x(t), H_y(t), H_z(t)$ los escribimos en el `.dat`.

Cabe destacar que nosotros no usamos el algoritmo mas eficiente posible para obtener la frecuencia de los bins, pero si el más sencillo de implementar. De tener mas datos habría que cambiar el método. En general no es un programa eficiente, pero funciona, y dado que el tiempo de cálculo es pequeño[2], no es relevante.

2.3.3. Resultados

Los valores de la dispersión Δ de $H(t)$ los valores vienen dados por

$$\Delta_1 = 0.000583 \qquad \Delta_2 = 0.000599 \tag{8}$$

como podemos ver son muy pequeños, casi podemos asumir $H(t)$ constante. En las gráficas también se puede ver como estos valores oscilan respecto a un mismo valor, incluso para los primeros 500K pasos desde el principio, lo cual es muy sorprendente, ya que no debería estar tan próximo al equilibrio al principio. En cualquier caso, es evidente que se mantiene con el tiempo en ambas configuraciones, por lo que ambas cumplen esta condición.

[2] Entre 5-6 segundos.

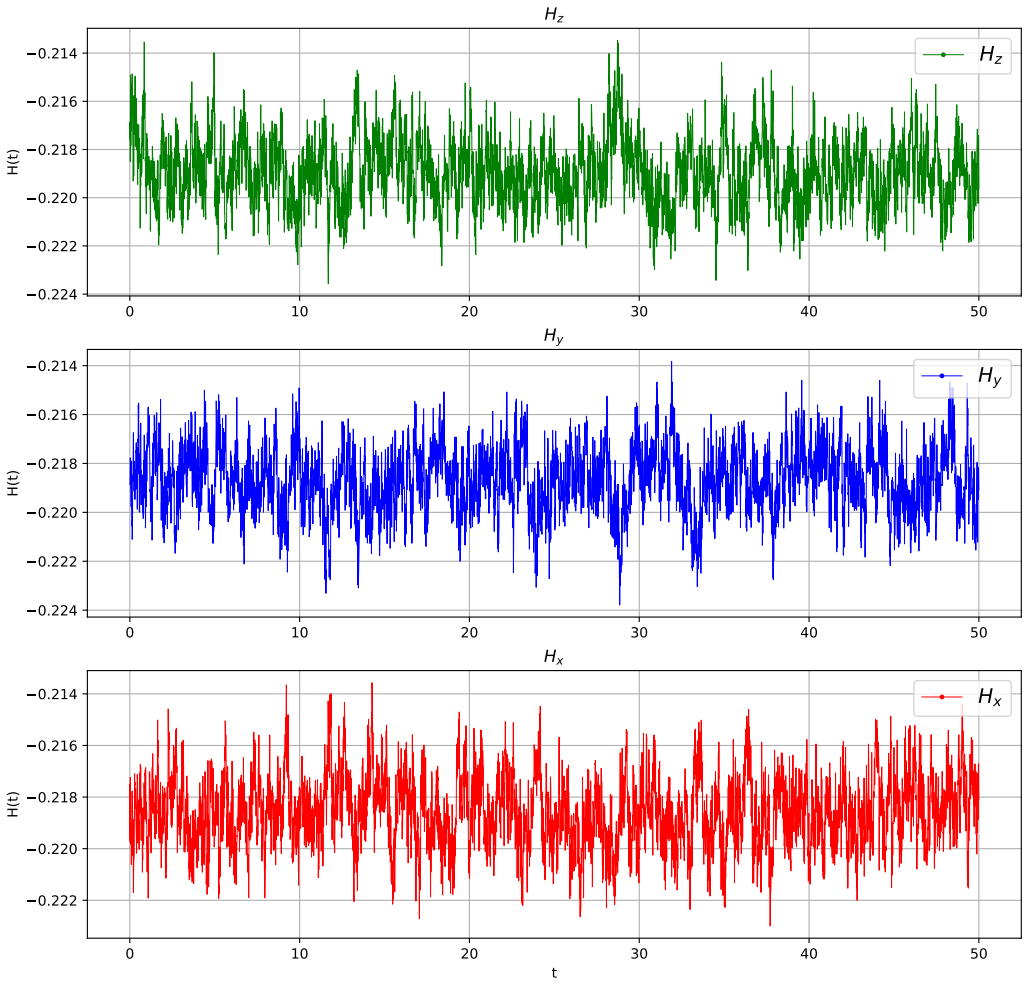


Figura 3: valores de las funciones de Maxwell H_x, H_y y H_z con el tiempo.

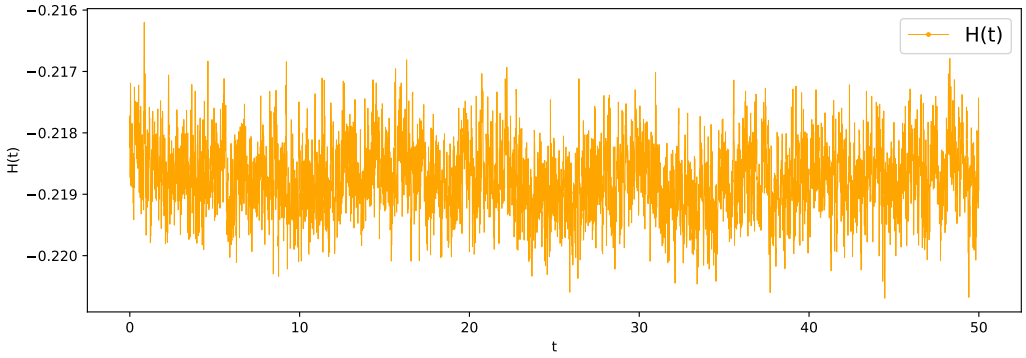


Figura 4: Valor de la función de Maxwell $H(t)$ con el tiempo .

3. Organización

En este programa hemos usado 2 programas de fortran y 3 de python. Los programas de fortran son:

- **Pro_Funcion_H**: lee los datos de las velocidades para cada interacción y genera un archivo con las H de Boltzmann para dichas velocidades.

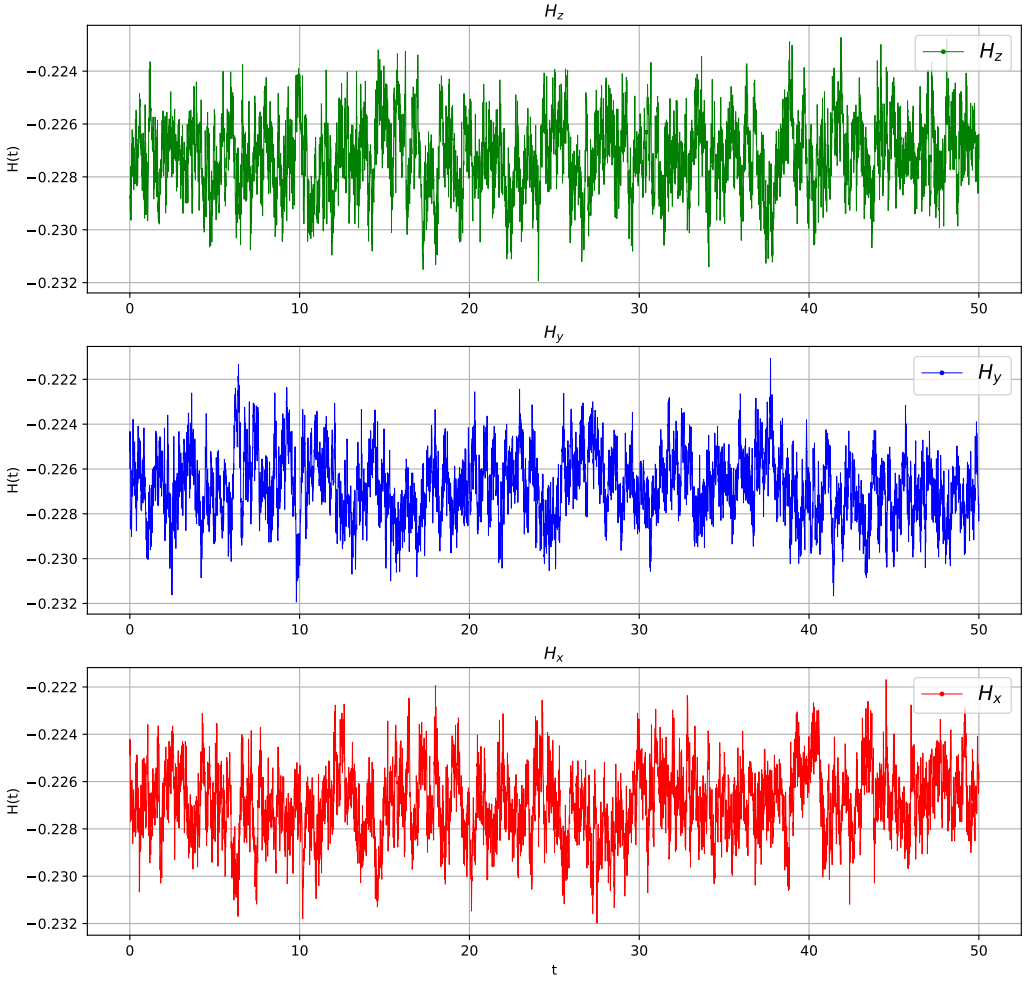


Figura 5: valores de las funciones de Maxwell H_x, H_y y H_z con el tiempo.

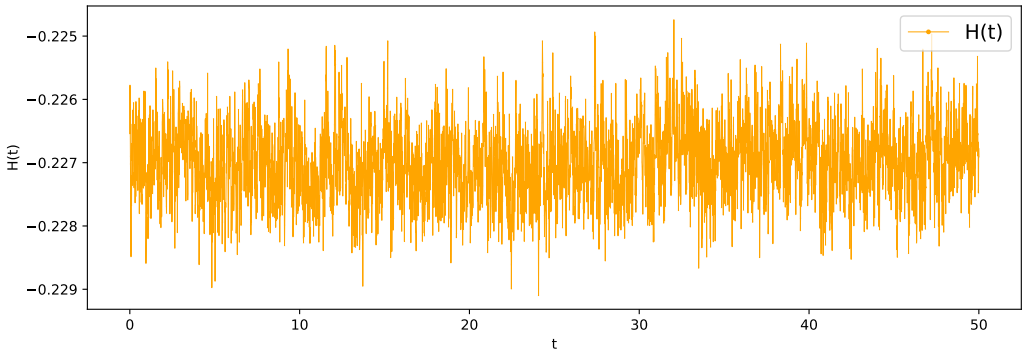


Figura 6: Valor de la función de Maxwell $H(t)$ con el tiempo .

- **Pro_Distribucion_velocidades**: lee los datos de las velocidades para cada interacción y genera un archivo con los histogramas de cada una de las velocidades.

Los programas de python se usan únicamente para hacer las gráficas y dar los valores medios.

4. Conclusiones

Las conclusiones son sencillas: las dos simulaciones verifican las condiciones de equilibrio, lo cual es sorprendente, ya que la primera, aún viniendo de las posiciones de la equilibración de 5000 pasos, llega ya en equilibrio, verificando incluso una equilibración más estable. Cabe destacar que, a pesar de que si verifica la equilibración, no vamos a usarla para la posterior dinámica molecular, ya que la energía está bastante alejada de -575 (véase tabla 1).

Es importante mencionar que el estudio de las condiciones de equilbiración es bastante escueto, ya que quizás lo ideal sería hacer más pasos y estudiar un conjunto de datos más extenso. También, debido a la cantidad de tiempo que requeriría su implementación, quedan sin hacer mejoras en la eficiencia de los algoritmos, así como el estudio de la distribución para v módulo entre otras.

Referencias

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