Práctica 8-
Dinámica Molecular

Asignatura: *Computación Avanzada*

Profesor: *David Martín y Marero*

Luis Sánchez Izquierdo | jueves 3 de mayo de 2018

Contents

Introducción

Teoría

En esta práctica vamos a estudiar el modelo la dinámica molecular de un sistema de partículas para un sistema bidimensional. La dinámica molecular lo que hace es considerar el movimiento de las partículas siguiendo las leyes de Newton.

Para calcular las trayectorias se integran numéricamente las ecuaciones del movimiento. En este caso, el algoritmo que vamos a usar es el algoritmo de Verlet. Usamos este algoritmo frente a otros más precisos como el Runge-Kutta debido a que tiene propiedades especiales que otros integradores no tienen.

Se trata de un algoritmo cuyas ecuaciones respetan la invariabilidad del Hamiltoniano del sistema frente al cambio de dirección temporal. Esto acaba significando que la energía se mantiene constante a lo largo de la integración (excepto por los efectos de errores numéricos que aparecen debido al redondeo de la última cifra de los números decimales)

Objetivos

La práctica consiste en:

- Desarrollar el algoritmo computacional para realizar la simulación de la dinámica molecular, teniendo en cuenta las condiciones de contorno.
- Hacer un primer análisis para partículas tomadas como discos rígidos.
- Hacer un segundo análisis para partículas cuya interacción viene dada por el potencial de Lennard-Jones.
- Estudiar la distribución de velocidades para estos sistemas.

Además, realizaré un apartado extra en el que estudio la propiedad tan importante del algoritmo de Verlet que es la reversibilidad en el tiempo. Por último, estudiaré la si es caótico el movimiento de la dinámica molecular.

Desarrollo del algoritmo computacional de la Dinámica Molecular

- ## Definiciones iniciales

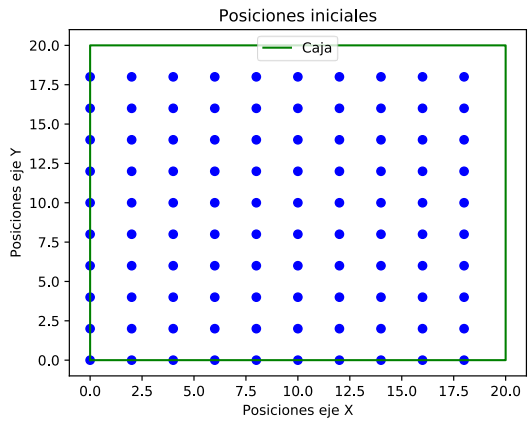
Lo primero de todo es establecer el sistema inicial. Este va a estar compuesto por un total de N_T partículas con N por lado. Es sistema es una caja bidimensional de lado $L = n \cdot a$ en donde a es la distancia inicial entre partículas (es el parámetro que define el mallado sobre el que colocaremos las partículas)

```
# INICIALIZAR
seed(1)

N = 10          # numero de particulas por lado
NT = N*N        # numero de particulas total

#distancias
sigma = 1       # distancia que define el potencial de Lennard-Jones
a = 2 * sigma   # distancia inicial entre partículas
L = N * a       # tamaño de la caja

rcorte = 3 * sigma      # distancia a partir de la cual no hay interacción
rcorte2 = rcorte**2     # es útil

dt = 0.01               # paso de tiempo en la integración
d2t = 2 * dt            # es útil
dt2 = dt**2             # es útil
```

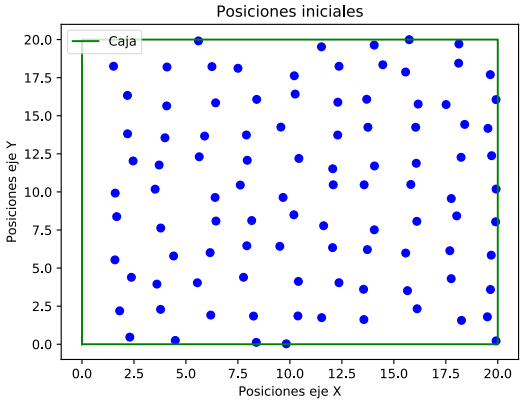
Además, establecemos las posiciones iniciales y los vectores de posiciones:

```
# vectores de posición
rx = []
ry = []

#Calculo de las posiciones:
dr = sigma/2    # distancia de variación aleatoria
for i in range(N):
    for j in range(N):
        rx.append(i*a + 2*(random()-0.5)*dr)
        ry.append(j*a + 2*(random()-0.5)*dr)

rx = array(rx)
ry = array(ry)
```

Con la parte $rx.append(i \cdot a)$ y la parte $ry.append(j \cdot a)$ se crea el mallado:



La parte de $rx.append(\ 2 \cdot (random(\ ) - 0.5) \cdot dr)$ es la que hace variar las posiciones iniciales del mallado obteniendo las posiciones iniciales aleatorias:



- ## Condiciones de contorno

Como se puede ver, las condiciones iniciales cumplen las condiciones de contorno:

$$CC1: \quad 0 \leq r_x < L$$

$$CC2: \quad 0 \leq r_y < L$$

Para aplicar las condiciones iniciales basta con ejecutar la siguiente sentencia:

```
51 # asegurarnos que las distancias están entre 0 y L
52 rx -= (rx//L) * L
53 ry -= (ry//L) * L
```

La explicación es la siguiente:

El valor $r_x//L$ es la división entera del vector r_x entre la longitud L. La división entera en Python funciona dándonos un valor entero. Por ejemplo, imaginemos que $L = 10$

- Si $r_{x,i} = 6.83$ entonces $r_{x,i}//L = 0$
- Si $r_{x,i} = -1.54$ entonces $r_{x,i}//L = -1$
- Si $r_{x,i} = 12.91$ entonces $r_{x,i}//L = 1$

Cuando una posición tiene $r_{x,i}//L \neq 0$ entonces significa que está fuera de la caja y para volver a meterla en la caja hay que sumar o restar L

- Si $r_{x,i}//L = 0$ entonces hay que sumar $0 \cdot L$
- Si $r_{x,i}//L = -1$ entonces hay que sumar $+1 \cdot L$ que es $-(-1) \cdot L = -(r_{x,i}//L) \cdot L$
- Si $r_{x,i}//L = 1$ entonces hay que sumar $-1 \cdot L$ que es $-(+1) \cdot L = -(r_{x,i}//L) \cdot L$

- ## Velocidades iniciales y posiciones previas

Las velocidades iniciales se establecen de forma aleatoria mediante el código:

```
64 v0 = 1
65
66 vx=[]
67 vy=[]
```

```
for i in range(NT):
    # uniform = valor aleatorio uniforme entre -v0 y v0
    vx.append(uniform(-v0,v0))
    vy.append(uniform(-v0,v0))

vx = array(vx)
vy = array(vy)
v  = sqrt(vx**2 + vy**2)
```

Y las posiciones previas vienen dadas inicialmente por

```
# posiciones anteriores
rxprev = rx - vx * dt
ryprev = ry - vy * dt

# asegurarnos que las distancias están entre 0 y + L
rxprev -= (rxprev//L) * L
ryprev -= (ryprev//L) * L
```

- ## El algoritmo de integración de Verlet

El algoritmo de Verlet lo que hace es calcular las posiciones y velocidades finales a partir de las posiciones actuales r_x y r_y, las posiciones previas $r_{x,prev}$ y $r_{y,prev}$ y las aceleraciones del sistema (iguales a la fuerza para partículas de masa reducida $m = 1$) mediante:

$$r_{x,n+1} = 2 \cdot r_{x,n} - r_{x,n-1} + f_x \cdot dt^2$$

$$r_{y,n+1} = 2 \cdot r_{y,n} - r_{y,n-1} + f_y \cdot dt^2$$

Y las velocidades vienen dadas por

$$v_{x,n+1} = \frac{r_{x,n+1} - r_{x,n-1}}{2 \cdot dt}$$

$$v_{y,n+1} = \frac{r_{y,n+1} - r_{y,n-1}}{2 \cdot dt}$$

```
def Verlet(xold,yold,x,y,vx,vy):
    #calculo de la fuerza
    Fx, Fy = Force(x,y,L)

    # calculo de las nuevas posiciones
    xnext = 2 * x - xold + Fx * dt2
    ynext = 2 * y - yold + Fy * dt2

    # calculo de las nuevas velocidades
    vxnext = (xnext - xold) / d2t
    vynext = (ynext - yold) / d2t

    # aplicar las condiciones de contorno
    xnext -= xnext//L * L
    ynext -= ynext//L * L

    # se devuelven: pos. antiguas, pos. nuevas y vel. nuevas
    return x, y, xnext, ynext, vxnext, vynext
```

- ## Obtención de la fuerza

Para calcular la fuerza tienes que recorrer todos los pares y aplicar la fórmula de la fuerza:

```
def Force(x,y,L):
    #vectores de fuerza
    fx, fy = zeros(NT,float), zeros(NT,float)
    #recorrer todos los pares
    for i in range (NT):
        for j in range(i+1,NT):   # condición para recorrer los pares
            r2min = (x[i]-x[j])**2 + (y[i]-y[j])**2
            dx, dy = x[i]-x[j], y[i]-y[j]
            # aplicar las condiciones de contorno
            for dlx in [-L,0,L]:
                for dly in [-L,0,L]:
                    r2 = (x[i]-x[j] + dlx)**2 + (y[i]-y[j] + dly)**2
                    if r2 < r2min:
                        r2min = r2
                        dx, dy = x[i]-x[j] + dlx, y[i]-y[j] + dly
            r2 = r2min

            # calculo para las que si que interactúan
            if r2 < rcorte2:
                # formula de la fuerza
                # (depende de si son discos rígidos o Lennard-Jones)
                # Caso1 : interacción de Lennard-Jones
                rm8 = 1 / r2**4
                aux = 24 * rm8 * ( 2 / r2**3 - 1 )
                fx[i] += aux * dx   # coseno
                fy[i] += aux * dy   # seno
                fx[j] -= aux * dx   # coseno
                fy[j] -= aux * dy   # seno

#                # Caso2: interacción de cuerpos rígidos
#                aux = 1 / r2**5
#                fx[i] += aux * dx
#                fy[i] += aux * dy
#                fx[j] -= aux * dx
#                fy[j] -= aux * dy
    return fx, fy
```

- ### El potencial de Lennard-Jones

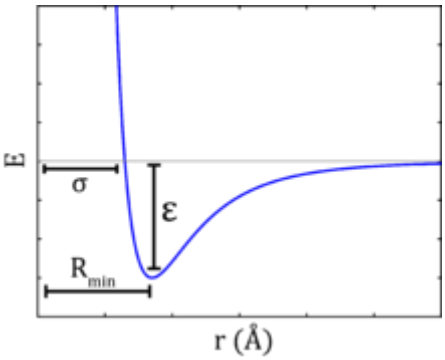
En el caso de que sean partículas que interaccionen con el potencial de Lennard-Jones entonces la energía potencial viene dada por

$$V(\vec{r}) = 4\varepsilon\left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right]$$

En donde establecemos que $\varepsilon = 1$ y también $\sigma = 1$

De modo que la fuerza viene dada por

$$\vec{F}(r) = -\vec{\nabla}V(\vec{r}) = -4\varepsilon\left[12\left(\frac{\sigma}{r}\right)^{12} - 6\left(\frac{\sigma}{r}\right)^{6}\right]\frac{1}{r}\overrightarrow{u_r}$$



- El potencial de discos rígidos

El potencial está definido como

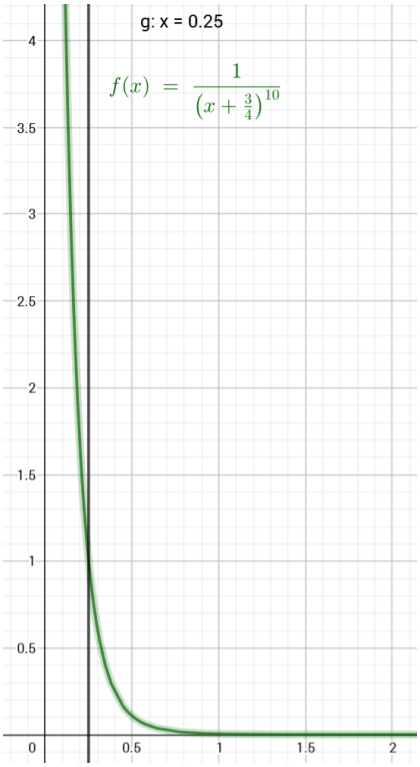
$$V(r) = \begin{cases} 0, & si\ r > \rho \\ \infty, & si\ r < \rho \end{cases}$$

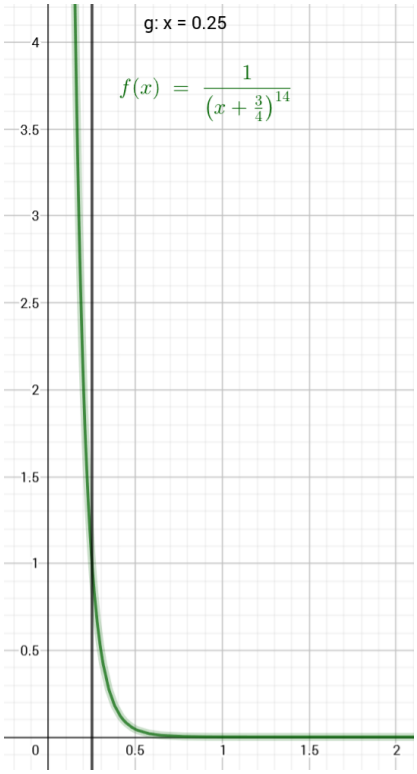
Con $\rho = \frac{\sigma}{4}$

Y por tanto la fuerza será igualmente

$$\vec{F}(r) = \begin{cases} 0, & si\ r > \rho \\ \infty, & si\ r < \rho \end{cases}$$

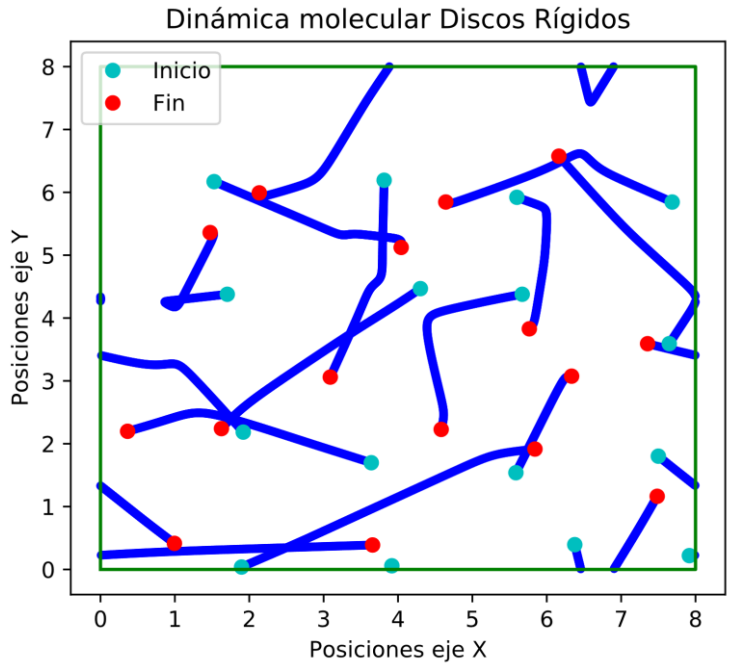
Yo la decisión que he tomado ha sido aproximar dicho potencial a un escalón. Para ello he usado la función $f(x) = \frac{1}{\left(x+\frac{3}{4}\right)^n}$ de modo que si $n \to \infty$ tendríamos dicho potencial. Yo me he quedado con el potencial dado por $n = 10$. A medida que n aumenta nos vamos acercando cada vez más al caso de cuerpo rígidos.



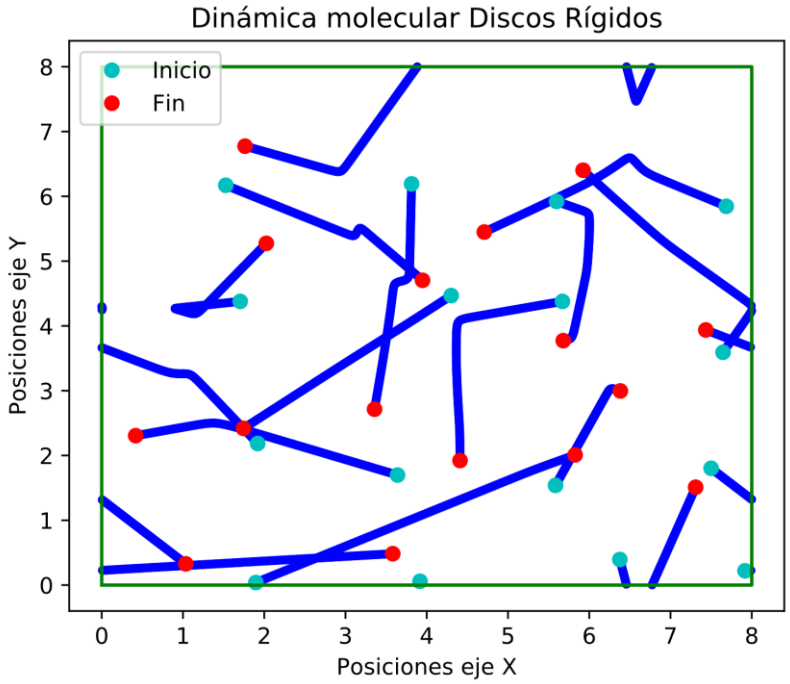


Resultados I: discos rígidos

Realizando la simulación para discos rígidos se obtiene la gráfica para un valor de la fuerza dada por $\frac{1}{r^{10}}$. Se ve que los discos no son perfectamente rígidos sino ligeramente blandos. Esto se nota en que los cambios de dirección son muy curvos, muy progresivos, mientras que si los discos fuesen 100% rígidos entonces los cambios serían con picos, no con curvas.

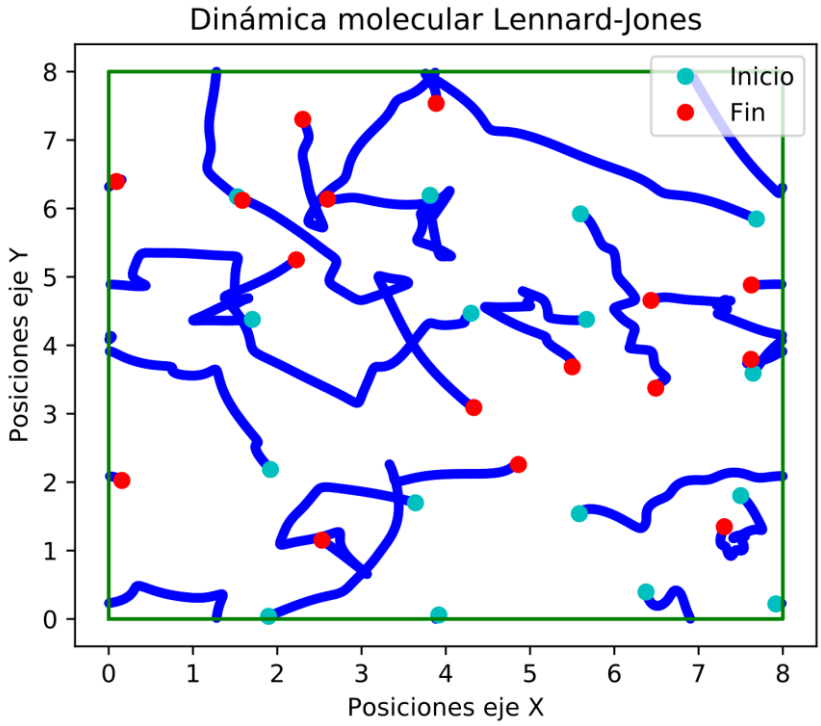


Para ejemplificar esto he hecho una simulación en la que la magnitud de la fuerza viene dada por $\frac{1}{r^{20}}$ de modo que ahora los cambios en direcciones son más esquinados y menos curvos. Nota: las condiciones iniciales son iguales. Por tanto, se ve una gran diferencia en las condiciones finales



Resultados II: potencial de Lennard-Jones

Ahora realizando la simulación para partículas que siguen interacciones de Van der Waals se ve cómo el resultado es totalmente diferente. Se trata de una evolución de un sistema gaseoso en el que las partículas siguen un movimiento caótico.



Resultado III: obtención de las distribuciones de velocidades

- Procedimiento teórico y computacional

Las partículas en el caso del potencial de Lennard-Jones siguen la distribución de velocidades de Maxwell-Boltzmann, que nos dice la probabilidad de encontrar a una partícula con módulo de velocidad igual a $|velocidad| \in (v, v + dv)$. Viene dada por

$$f_{MB}(v) = C\,\frac{1}{k_B T}\,v\,e^{-\frac{\frac{1}{2}mv^2}{k_B T}} = A\,v\,e^{-v^2}$$

En donde A es una constante que viene dada por los valores de la masa de las partículas m y la temperatura T

Para obtener las funciones de distribuciones lo que hago es realizar una simulación para un gran número de partículas de modo que haya un gran espectro de velocidades sobre las cuales hacer un histograma. Como el número de partículas es mayor, el tiempo computacional será mucho mayor. A mi me dan valores de $t_{computacional} \approx 3\ min$ para valores de $N_T = 400$

El perfil de este histograma se comparará con la función de Maxwell-Boltzmann para hallar el coeficiente A y ajustar la función a los datos experimentales.

```
# CALCULO DE LA FUNCIÓN DE DISTRIBUCIÓN DE VELOCIDADES
v = sqrt(vx**2+vy**2)

# realizar el histograma
nn = 15          # numero de intervalos
figure()
hist(v, nn, normed=1, facecolor='green')
xlabel('Intervalos de velocidad')
ylabel('Conteo de partículas')
title('Distribución de velocidades')
show()

# obtención de los datos del histograma
yh, xhaux = histogram(v, bins=linspace(0,max(v),nn), density=True)
xh = []
for i in range(len(yh)):
    xh.append( (xhaux[i] + xhaux[i+1]) / 2 )

# representar el histograma
plot(xh,yh,'-o')
```

Computacionalmente la forma de ajustar una función no lineal como la de Maxwell-Boltzmann se hace del siguiente modo:

Lo que tenemos como datos son los intervalos de velocidades $v_{intervalos} \leftrightarrow variable\ x$ y la cantidad de partículas cuya velocidad está en ese intervalo $n_{partículas} \leftrightarrow variable\ y$

La manera de hacer el ajuste es minimizar la suma de las distancias con respecto a A

$$S = \sum_i d_i^2 = \sum_i \left(y_i - f_{MB}(v_i)\right)^2 = \sum_i y_i^2 - 2 \cdot A \cdot x_i \cdot y_i \cdot e^{-x_i^2} + A^2 \cdot x_i^2 \cdot e^{-2x_i^2}$$

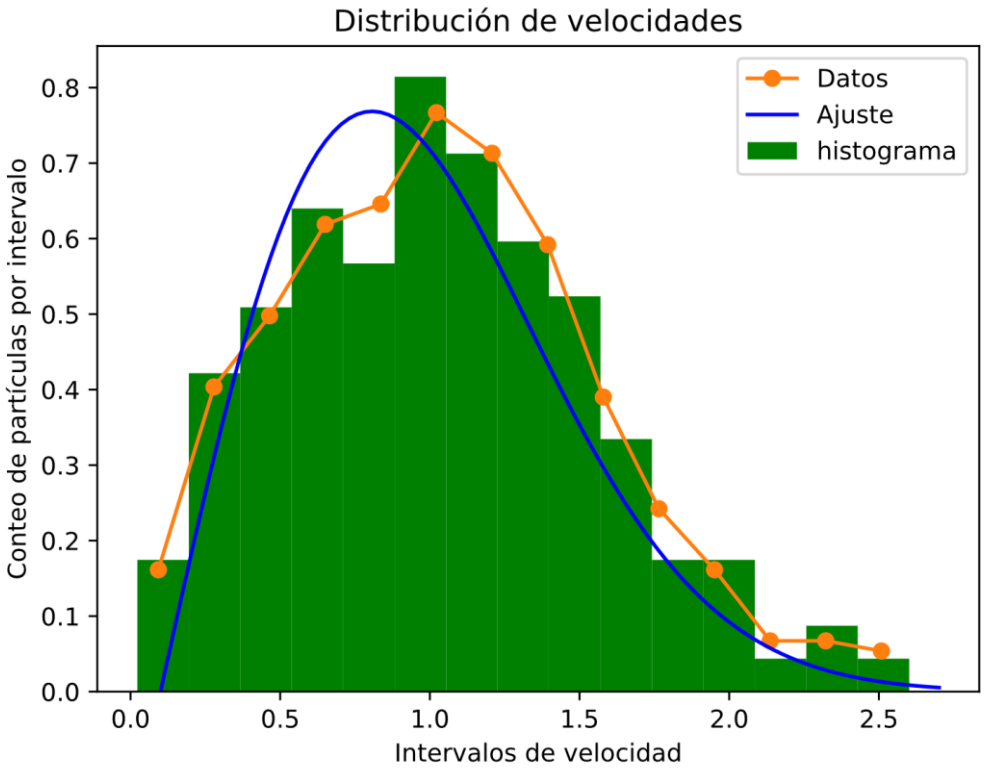
Minimizando con respecto a A al hacer $\frac{\partial S}{\partial A} = 0$ se obtiene el resultado

$$A = \frac{\sum_i 2 \cdot x_i \cdot y_i \cdot e^{-x_i^2}}{\sum_i 2 \cdot x_i^2 \cdot e^{-2x_i^2}} = \frac{S_2}{S_1}$$

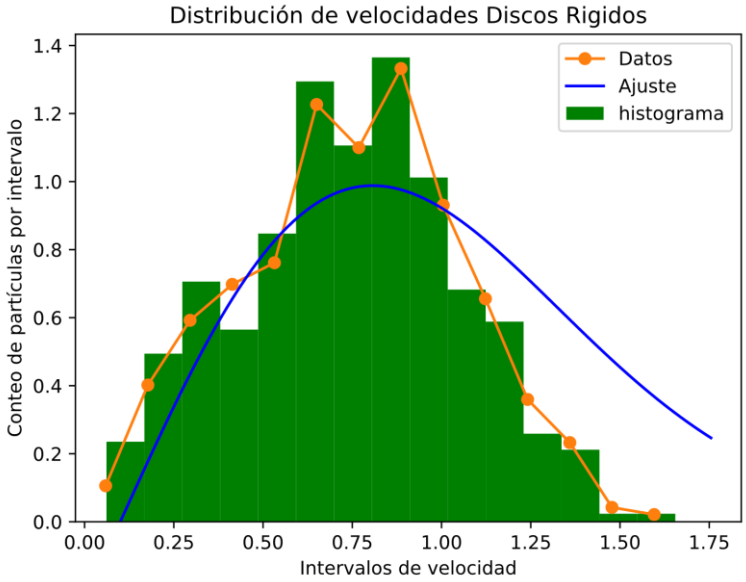
```
# CALCULO DEL AJUSTE
# http://www.sc.ehu.es/sbweb/fisica3/datos/ajuste/nolineal.html

# Obtención de las sumas
S1, S2 = 0, 0
for i in range(len(yh)):
    S1 += 2*xh[i]**2 * exp(-2*xh[i]**2)
    S2 += 2*xh[i]* yh[i] * exp(-xh[i]**2)

# obtención de A
A = S2 / S1

# representación del ajuste
xplot = linspace(0,max(v),100)
yplot = A * xplot * exp(-xplot**2)
plot(xplot+0.1, yplot)
```

- Resultados

Para las partículas que siguen la interacción de Lennard-Jones el ajuste es perfecto a una distribución de Maxwell-Boltzmann, como se puede ver a continuación:



Por otro lado, para las partículas que siguen la interacción de discos rígidos la distribución no se puede ajustar a una estadística de Maxwell-Boltzmann. Esto es debido a que en principio la distribución de velocidades permanece igual que la dada inicialmente ya que cuando hay interacción entre dos partículas lo que se hace es que se cambia la dirección, pero no el módulo de la velocidad.



Extra: reversibilidad de las ecuaciones

La reversibilidad de las ecuaciones es una de las principales ventajas del algoritmo de Verlet. Esta irreversibilidad consiste en cambiar $t \to -t$ y se obtienen las mismas ecuaciones. Esto se puede ver directamente en la fórmula de Verlet:

$$x_{n+1} = 2 \cdot x_n - x_{n-1} + f_n \cdot (dt)^2$$

Al cambiar $t \to -t$ se obtiene:

$$x_{n-1} = 2 \cdot x_n - x_{n+1} + f_n \cdot (-dt)^2$$

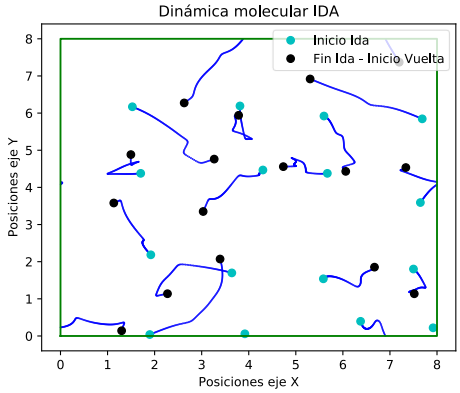
Y reordenando se obtiene de nuevo la ecuación inicial:

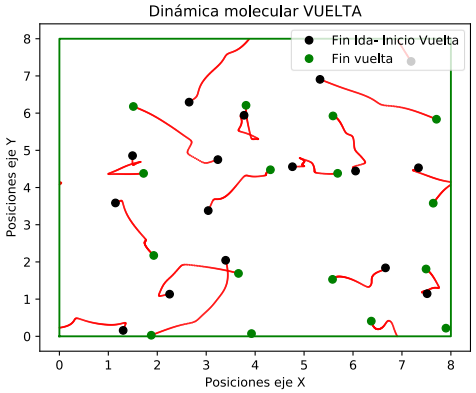
$$x_{n+1} = 2 \cdot x_n - x_{n-1} + f_n \cdot (dt)^2$$

Para comprobar la reversibilidad basta con realizar un camino de ida y el camino de vuelta. Esto se implementa con:

```
156 #dar la vuelta
157 vx = -vx
158 vy = -vy
159
160 rx_aux = rx
161 ry_aux = ry
162
163 rx = rxprev
164 ry = ryprev
165
166 rxprev = rx_aux
167 ryprev = ry_aux
```

Hay que realizar una simulación para la ida y una vez que se ha llegado al punto final, dar la vuelta y realizar la simulación de la vuelta. Los resultados indican que las ecuaciones son reversibles:





Extra II: cálculo del caos del movimiento

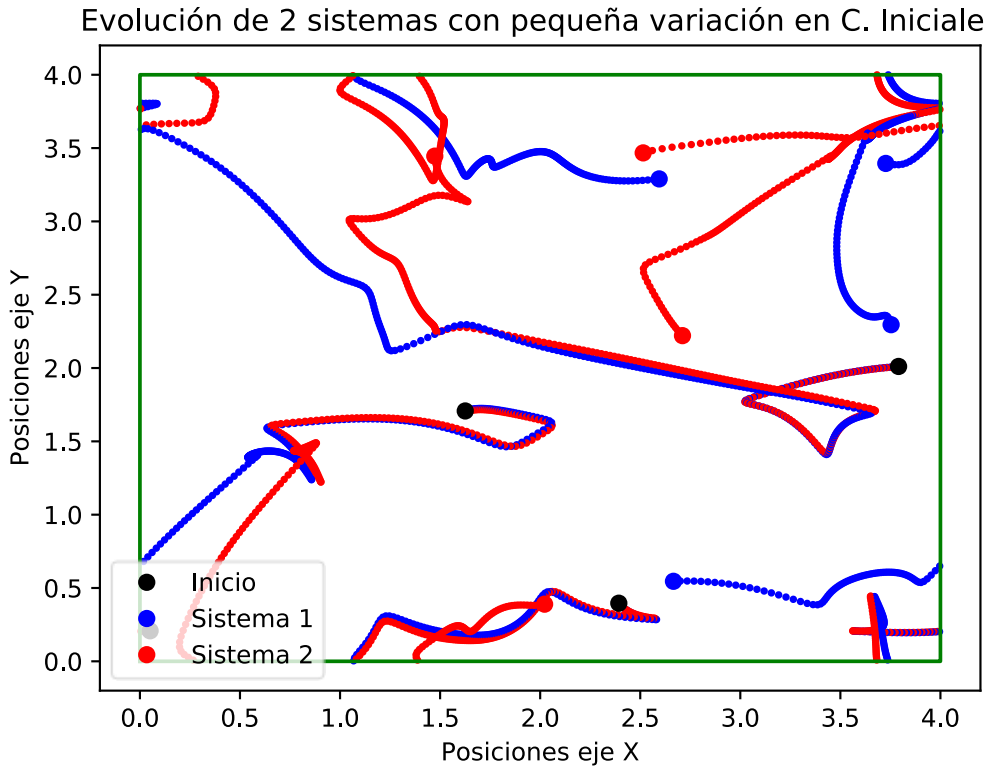
Para calcular eso calculo la evolución de dos sistemas con condiciones iniciales similares, variadas en centésimas de unidades.

Para ello primero añado un nuevo sistema y sus condiciones iniciales variadas ligeramente (ver que la variación viene dada por $ddr = sigma/100$)

```
# vectores de posición para el segundo sistema
rx_2 = []
ry_2 = []

#Calculo de las posiciones:
ddr = sigma/100    # distancia de variación aleatoria con respecto al caso inicial
for i in range(NT):
    rx_2.append(rx[i] + 2*(random()-0.5)*ddr)
    ry_2.append(ry[i] + 2*(random()-0.5)*ddr)

rx_2 = array(rx_2)
ry_2 = array(ry_2)

# asegurarnos que las distancias están entre 0 y L
rx_2 -= (rx_2//L) * L
ry_2 -= (ry_2//L) * L

vx_2 = vx
vy_2 = vy

# posiciones anteriores
rxprev_2 = rx_2 - vx_2 * dt
ryprev_2 = ry_2 - vy_2 * dt

# asegurarnos que las distancias están entre 0 y + L
rxprev_2 -= (rxprev_2//L) * L
ryprev_2 -= (ryprev_2//L) * L
```

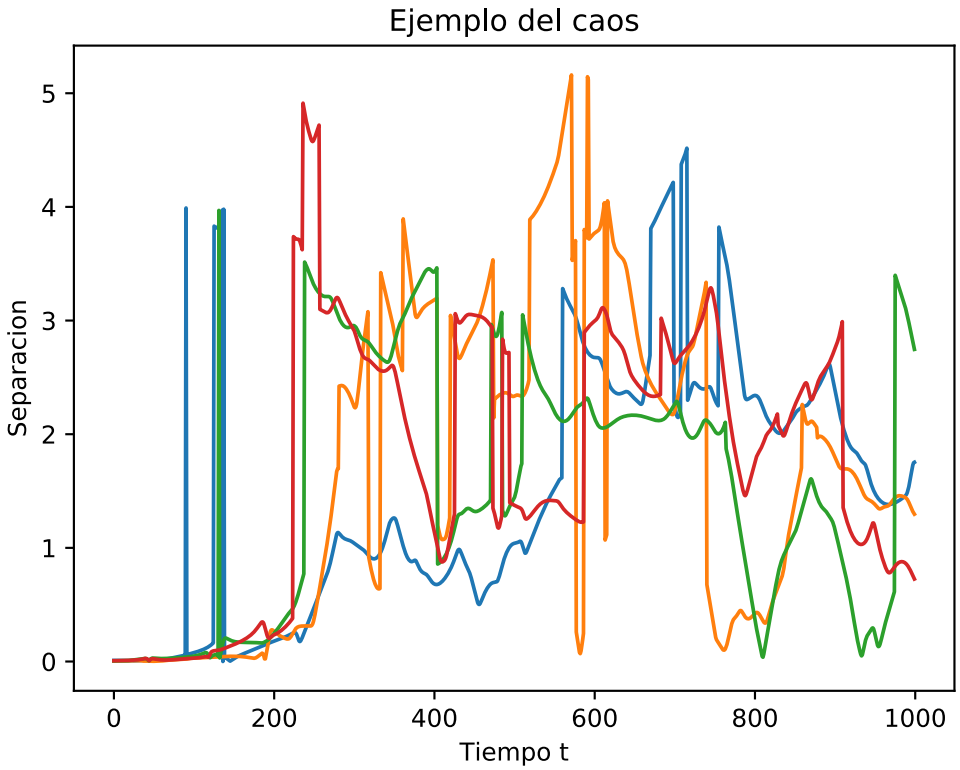
Tras calcular la evolución se puede ver cómo los movimientos de las partículas de un sistema se alejan con respecto a los movimientos de las correspondientes partículas del otro sistema.

Nota: he escogido un sistema de $NT = 4$ para que se pueda seguir con los ojos las trayectorias.



Si represento la separación de las partículas en un sistema y otro vemos como esta separación aumenta. Si la separación fuese próxima a cero no habría caos. Sin embargo, en este sistema si que hay caos.

Como el movimiento está acotado a la caja, las mayores separaciones serán del tamaño de la diagonal de la caja $\sqrt{2}\,L \approx 5.4$ y si las separaciones son mayores que σ es que el movimiento ya es caótico y no se debe al error numérico.



Conclusiones

En esta práctica he realizado el algoritmo para estudiar la dinámica molecular de un sistema bidimensional.

He estudiado dos tipos de interacciones: para el caso de interacciones como discos rígidos se puede observar en las gráficas que las trayectorias son rectas con cambios de dirección bruscos (formando picos). Por otro lado, para el caso de interacción dada por el potencial de Lennard-Jones se puede ver que los movimientos son más suaves. Además, en este último caso se aprecia que el sistema sigue una distribución de velocidades dada por la distribución de velocidades aleatorias de Maxwell.

Finalmente, he estudiado dos apartados fundamentales del sistema de Verlet. Uno de ellos es la reversibilidad de las ecuaciones, lo que lo convierte en un sistema adecuado para hacer simulaciones en las que se tiene que conservar la energía. El otro factor importante es el caos del sistema de dinámica molecular, observando el efecto de variar ligeramente las condiciones iniciales.