Física Computacional

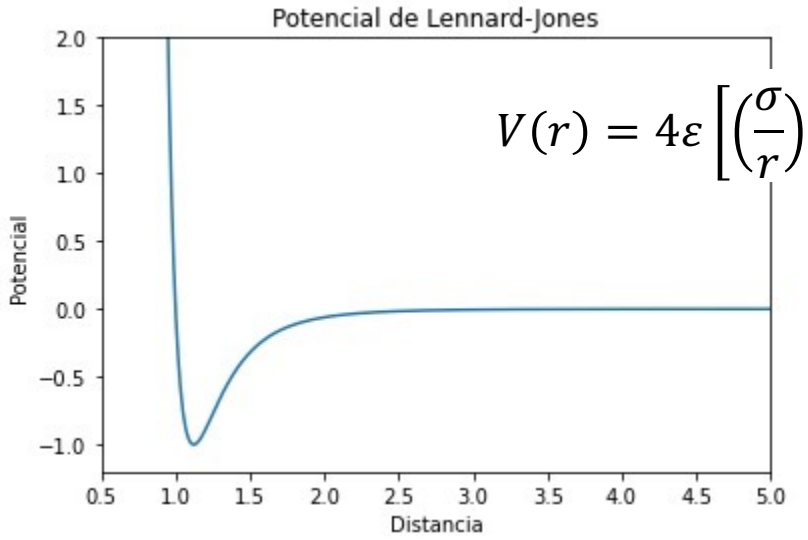
Búsqueda del estado fundamental de energía de la estructura FCC del cobre

Gabriel Simón López

Índice

Potenciales

Potencial Lennard-Jones

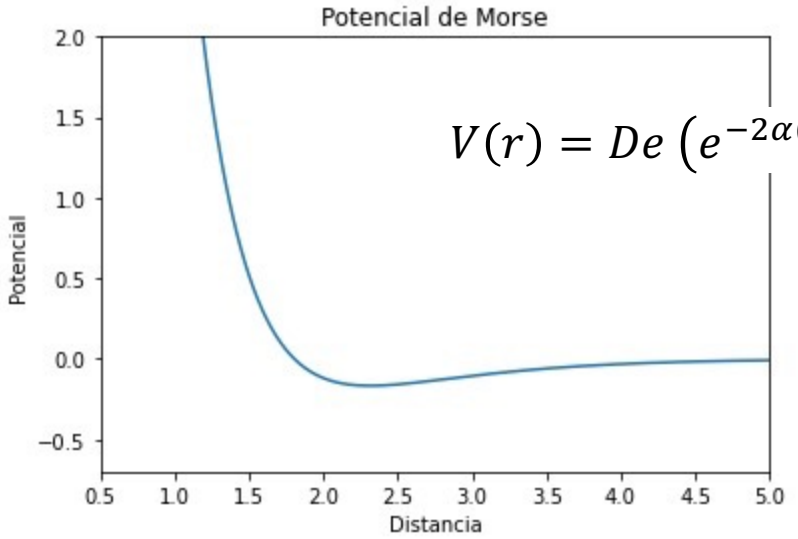


$$V(r) = 4\varepsilon\left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right]$$

```
def LennardJones(r):
    sigma = 2.3151; eps = 0.167
    return 4*eps*( (sigma/r)**12 - (sigma/r)**6 )
```

- ϵ la profundidad del potencial
- σ la distancia finita en la que el potencial entre partículas es 0

Potencial Morse



$$V(r) = De\left(e^{-2\alpha(r-re)} - 2e^{-\alpha(r-re)}\right)$$

```
def MorsePotential(r, De, a, re):
    return De * (np.exp(-2 * a * (r - re)) - 2 * np.exp(-a * (r - re)))
```

- De profundidad del pozo
- r distancia entre átomos
- re distancia al enlace de equilibrio
- α el ancho del potencial

Un vistazo al código. ¿Qué es lo que hace?

Funciones Calc

```python
def CalcR(r):
    N = len(r)
    R = np.empty((N, N))
    for i in range(N):
        R[i] = np.sqrt( (r[:,0]-r[:,0][i])**2 +
                        (r[:,1]-r[:,1][i])**2 + (r[:,2]-r[:,2][i])**2)
    return R
```

Calcula la matriz de distancias entre todas las partículas del sistema

```python
def CalcV(r, factor_de_corte=3):
    sigma=2.3151
    R = CalcR(r)
    R_filtered = np.logical_and(R < factor_de_corte * sigma, R > 0)
    V_red = 0.5 * np.sum(LennardJones(R[R_filtered]))
    return V_red
```

Calcula la energía potencial del sistema. Utilizando la distancia calculada entre partículas y se aplica el potencial de Lennard-Jones.

Un vistazo al código. ¿Qué es lo que hace?

Funciones Random

```python
def random_pick_vect(n, dim):
    v = np.random.random_sample(dim)
    v = v/np.sqrt(np.sum(v**2))
    return  np.random.randint(0,n), v
```

Selecciona una partícula i aleatoria y una dirección v aleatoria

```python
def random_move(r, step, dim, L):
    while True:
        i, vect = random_pick_vect(len(r), dim)
        if (r[i]+vect*step < [L,L,L]).any() or (r[i]+vect*step > [0,0,0]).any():
            r[i] = r[i]+vect*step
            return r
```

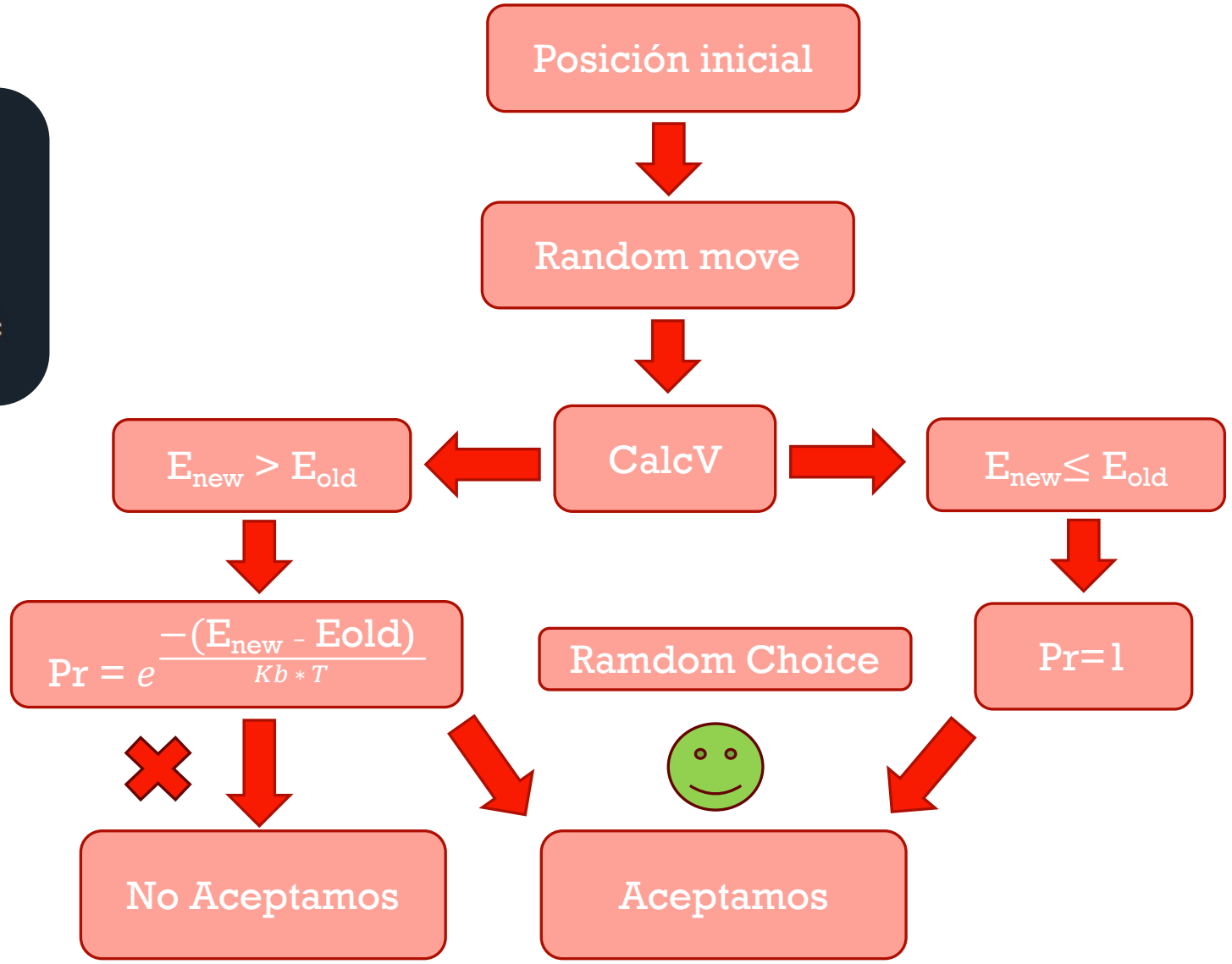
Mueve una partícula en una dirección aleatoria (dentro de los límites de la simulación)

Un vistazo al código. ¿Qué es lo que hace?

Algoritmo Metrópolis

```
def Metropolis(r, T, E_i):
    rj = random_move(r.copy(), step, dim,L)
    E_j = CalcV(rj)
    if E_j <= E_i:
        Pr = 1
    else:
        Pr = np.exp(-(E_j-E_i)/(kb*T))
    if np.random.choice((True, False), p = [Pr, 1-Pr]):
        return rj, E_j
    return r, E_i
```

¿Es realmente un mínimo?



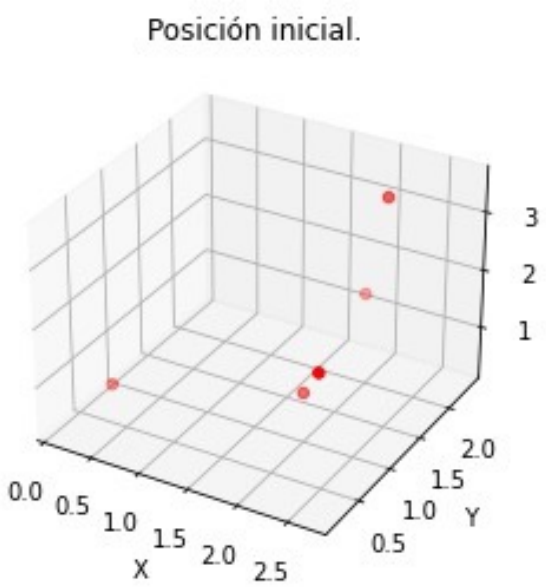
¿Es realmente un mínimo?

Probablemente no

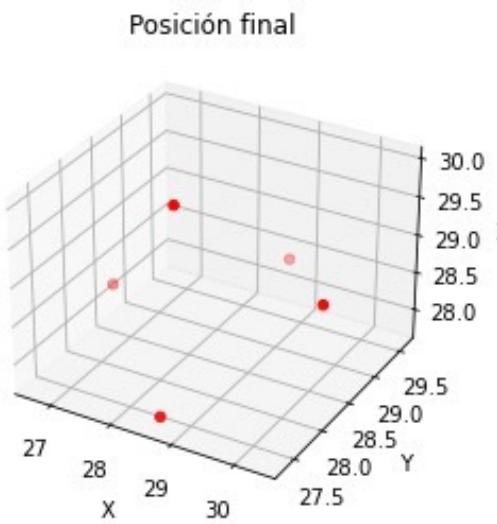
El sistema se queda estancado en los muchos mínimos locales que existen en este sistema

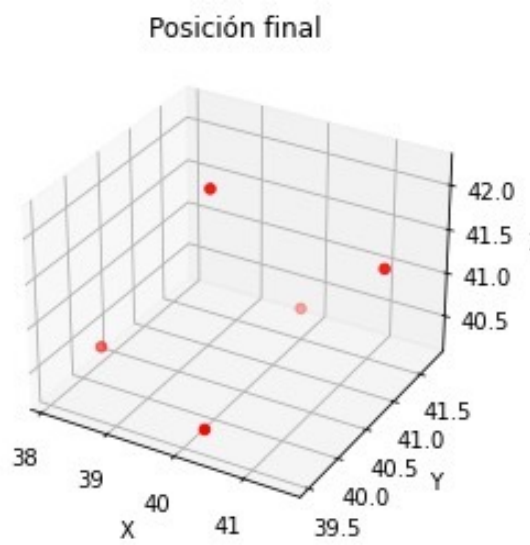
L-J

Morse



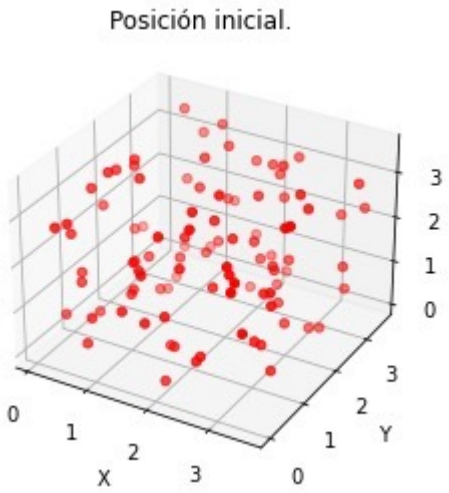
N = 5
KT = 300
Pasos = $1e^4$





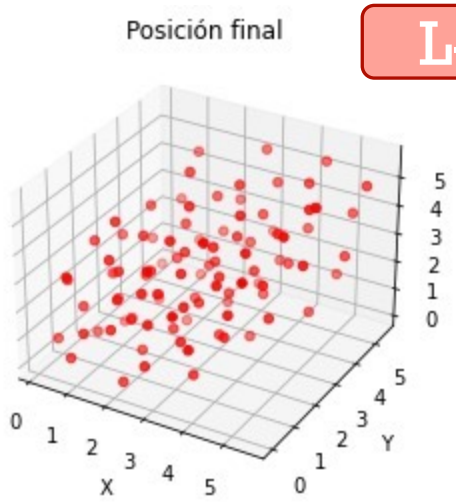

```
Progreso de la simulación 1: |####################|
Energía final:  -1.1909394795778945
Progreso de la simulación 2: |####################|
Energía final 2:  -20.799649976851843
```

¿Es realmente un mínimo?

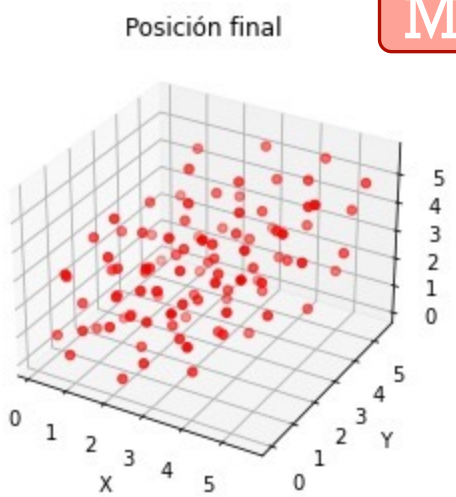


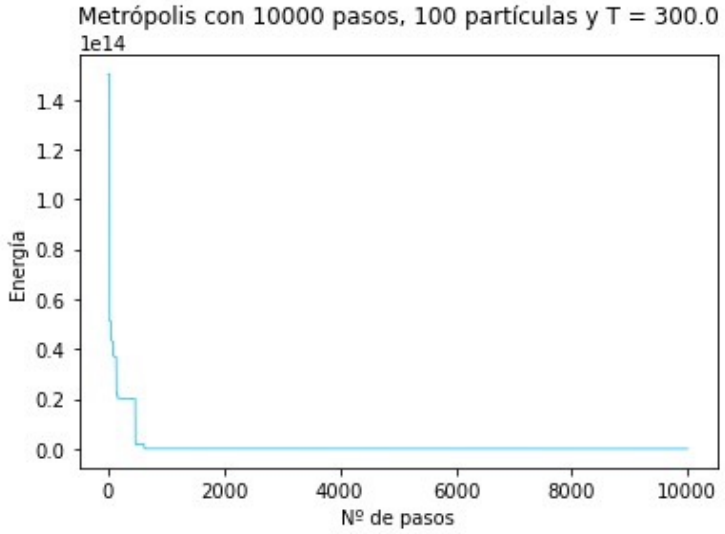
N = 100
KT = 300
Pasos = $1e^4$

L-J



Morse






```
Progreso de la simulación 1: |####################|  Completada
Energía final:  44659.29800817125
Progreso de la simulación 2: |####################|  Completada
Energía final 2:  46476.506404201704
```

Se estanca muy pronto y no es capaz de salir del mínimo local

No estamos en el mínimo real

Barrido de temperaturas

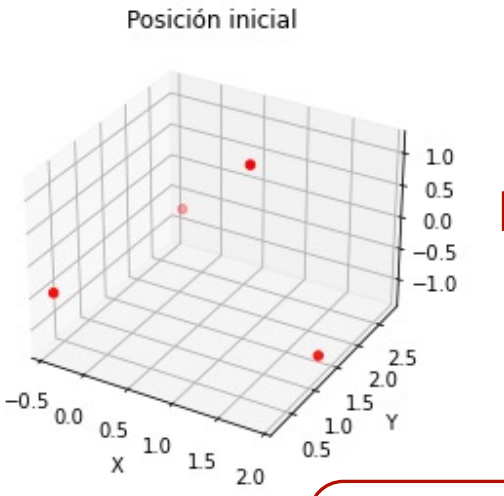
Realizaremos un barrido de temperaturas con el código

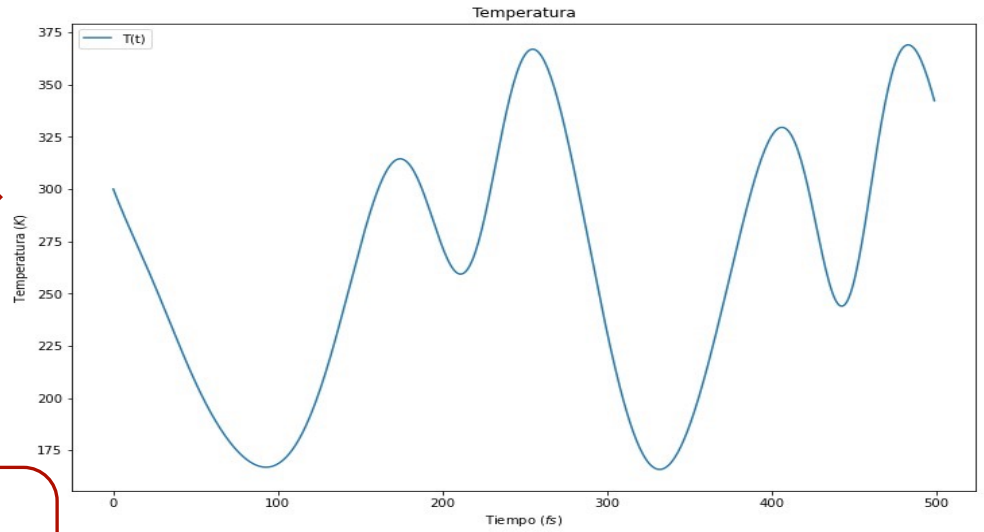
```
def random_pick_vect(n, dim):
    v = np.random.random_sample(dim)
    v = v/np.sqrt(np.sum(v**2))
    return  np.random.randint(0,n), v
```

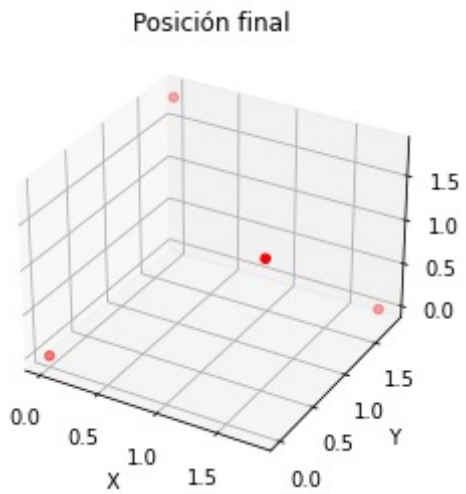
```
def Ec(v, m, Axis = 1, ConversionRate = 1/16):
    return ConversionRate*np.sum(0.5*m*np.sum(v**2, axis = Axis), axis = Axis-1)

def Calc_T(v, m, N, Kb = 8.6181024e-5, Axis = 1):
    return 2*Ec(v,m, Axis = Axis)/(Kb*N*3)
```

Selecciona una partícula i aleatoria y una dirección v aleatoria
Calculamos la energía cinética del sistema y la temperatura de esta a una velocidad dada relacionándola con el Principio de Equipartición

Barrido de temperaturas



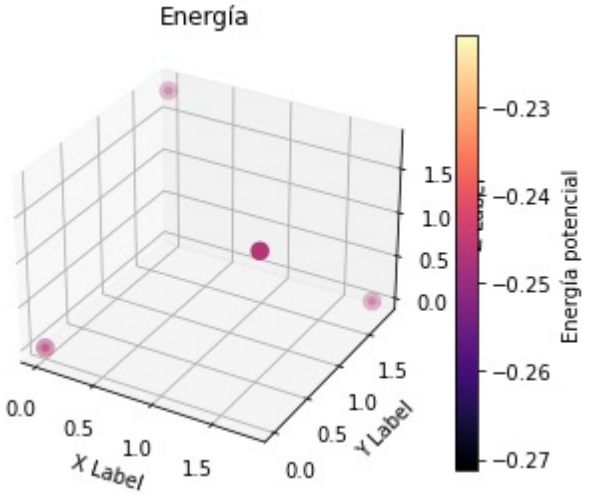




N = 4
KT inicial= 300
Pasos = $1e^4$

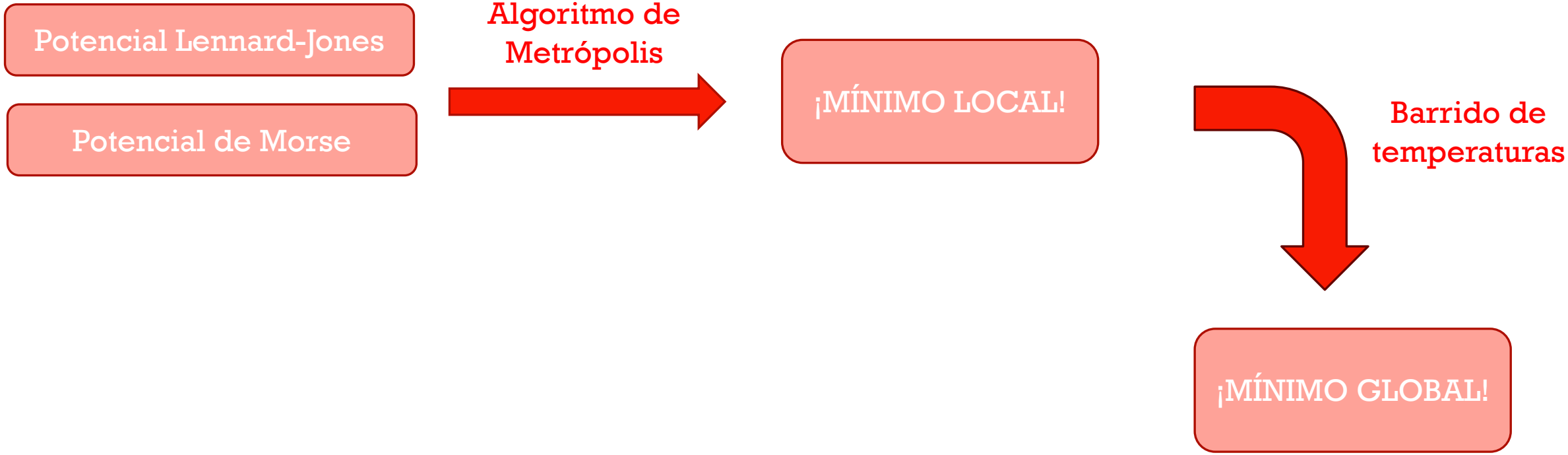
```
Progreso de la simulación: |####################|
Energía final:  -0.24417905892735964
Progreso de la simulación: |####################|
```

Resultados muy similares



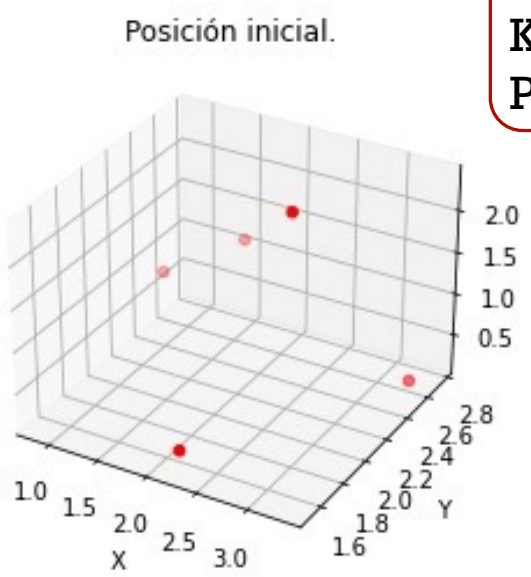

```
Energia por atomo =  -0.2465
```

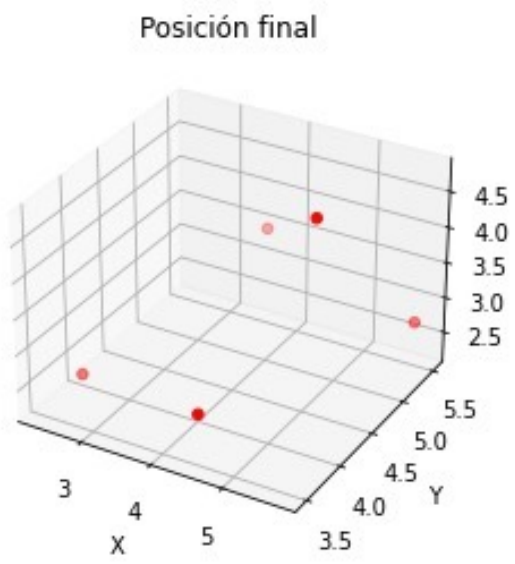
Resumen

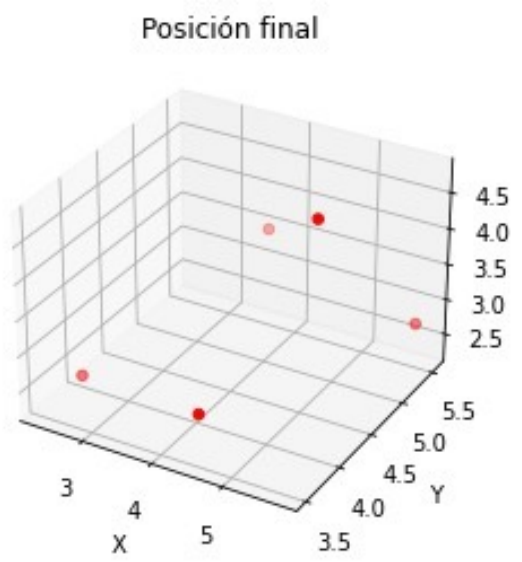

Potencial Lennard-Jones
Potencial de Morse
Algoritmo de Metrópolis
¡MÍNIMO LOCAL!
Barrido de temperaturas
¡MÍNIMO GLOBAL!

Montecarlo para temperatura baja

N = 5
KT = 10
Pasos = $1e^4$







L-J

Morse

Esto puede ser debido a las barreras de energía, en donde las partículas pueden quedar atrapadas en mínimos locales muy rápido en lugar de explorar el espacio completo de configuraciones

¡Gracias!

¿Alguna pregunta?

Bibliografía

[1] Wikipedia contributors. (2024, 15 enero). *Lennard-Jones potential*. Wikipedia. https://en.wikipedia.org/wiki/Lennard-Jones_potential

[2] Wikipedia contributors. (2023, 29 julio). *Potencial de Morse*. Wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/Potencial_de_Morse

[3] Diapositivas Física Computacional 2024 Bloque 2 Montecarlo Metrópolis

Pasos de Montecarlo