# PRÁCTICA 6

December 28, 2023

## 1 Simulación de Monte Carlo de un sistema de partículas cuánticas

Este código utiliza el método de Metropolis Monte Carlo para simular un sistema de partículas cuánticas en un pozo de potencial tridimensional. El usuario debe proporcionar tres entradas:

- `N`: El número de partículas en el sistema.
- `T`: La temperatura del sistema, en unidades de energía.
- `pasos`: El número de pasos en la simulación de Monte Carlo.

El código inicializa el sistema con todas las partículas en el estado de energía más bajo. Luego, en cada paso de la simulación, elige una partícula y una dirección al azar, propone un cambio de estado que aumenta o disminuye el número cuántico en esa dirección, y decide si aceptar o rechazar el cambio basándose en la diferencia de energía y la temperatura del sistema.

[2]:
```
import numpy as np
import matplotlib.pyplot as plt
```

Se definen ahora las constantes que aparecen en las expresiones del problema. Se definen todas igual a 1, quedando T en unidades de h_barraˆ2/m*Lˆ2.

[3]:
```
hbar = 1 # Constante de Planck reducida
m = 1 # Masa de las partículas
L = 1 # Longitud de la caja
pi = np.pi # Número pi
```

Se definen dos funciones. Una para el cálculo de la energía dado un estado cuántico, definido por la terna de n,s y luego otra función para las variaciones de energía.

[4]:
```
def E(nx, ny, nz):
  return (pi**2 * hbar**2 / (2 * m * L**2)) * (nx**2 + ny**2 + nz**2)

# Definir función de cambio de energía
def deltaE(nx, ny, nz, dnx, dny, dnz):
  return E(nx + dnx, ny + dny, nz + dnz) - E(nx, ny, nz)
```

Para la variación de energía se tienen en cuenta los diferenciales de n's. No hacemos uso, directo, de las expresiones que nos proporciona el enunciado. Al final de cuentas nos dará el mismo resultado salvo pequeñas diferencias en los tiempos de ejecución. No será significativo.

En la siguiente función, tenemos en cuenta el criterio de aceptación de cada paso. Consideramos que si la variación de energía es negativa, pasamos a un estado menor de energía, entonces SIEM-

PRE aceptamos. Por ello, decimos return 1, pues posteriormente, en la iteración sacaremos aleatoriamente un valor entre 0 y 1 donde si este, de forma estricta, es menor que la salida de p_aceptar aceptará el cambio.

[5]:
```
def p_aceptar(deltaE, T):
  if deltaE < 0:
    return 1 # Aceptar siempre si la energía disminuye
  else:
    return np.exp(-deltaE / T)
```

Se definen los inputs, número de partículas (N), pasos Monte Carlo (pasos) y la temperatura estadística (T) para que la terminal se los solicite al usuario cada vez que se inicie el programa.

La variable estados es una matriz de N filas, tantas como partículas, y 3 columnas, tantas como direcciones. Esta contendrá los valores del estado en el que se encuentra cada una de las partículas. Esto es una forma de poder localizar cada partícula, cuando aleatoriamente le toque, en el bucle posterior.

[6]:
```
N = int(input('Número de partículas: ')) # Número de partículas
T = float(input('Introduce un valor de temperatura: ')) # Temperatura en␣
 ↪unidades de hbar**2 / (m * L**2)
pasos = int(input('Número de pasos Monte Carlo: ')) # Número de pasos de Monte␣
 ↪Carlo

estado = np.ones((N, 3), dtype=int)

energia = np.sum(E(estado[:, 0], estado[:, 1], estado[:, 2]))

lista_energia = [energia]
```

La variable energía nos suma la energía para, de forma independiente, todas las partículas. Esta será la energía total del sistema, se irá actualizando en cada paso Monte Carlo.

Finalmente, la lista_energia, guardará la energía, calculada en energía, para representarla posteriormente.

Se muestra el bucle para la iteración de todos los pasos Monte Carlo. En ella, para cada paso se extrae de forma aleatoria una partícula de las N, una dirección (nx,ny,nz) y el valor del cambio de energía, cambio negativo o positivo.

- Observese la linea de código donde se comprueba que el número cuántico no sea menor que 1. Expliquemos un poco el cuarto argumento que se expone cuando llamamos a deltaE. Aquí es donde utilizamos `np.eye(3)[direccion] * cambio`:
  - `np.eye(3)` crea una matriz de identidad de 3x3. Esta matriz tiene unos en la diagonal y ceros en todas las demás posiciones. En términos de vectores, esta matriz representa tres vectores unitarios en las direcciones x, y, z.
  - `np.eye(3)[direccion]` selecciona uno de estos vectores unitarios. Si `direccion` es 0, selecciona el vector unitario en la dirección x. Si `direccion` es 1, selecciona el vector unitario en la dirección y. Si `direccion` es 2, selecciona el vector unitario en la dirección

z.

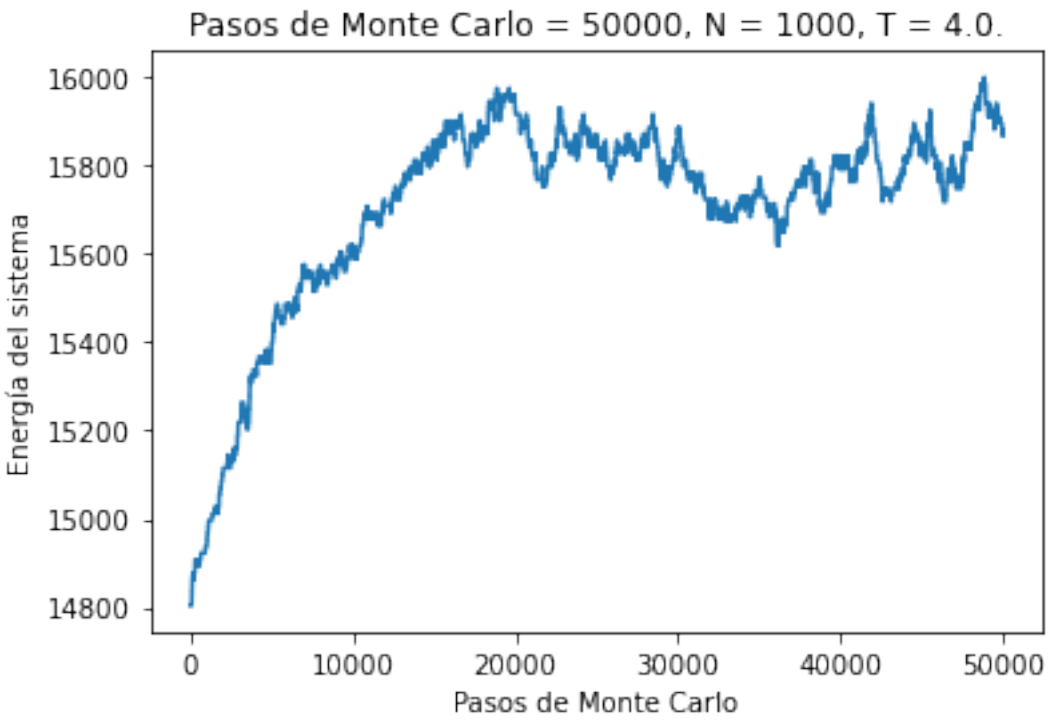
- `np.eye(3)[direccion] * cambio` multiplica este vector unitario por el cambio propuesto. Si `cambio` es +1, el vector se mantiene igual. Si `cambio` es -1, el vector se invierte.
- Finalmente, el `*` antes de la expresión es un operador de desempaquetado. Esto significa que los elementos del vector se pasan como argumentos separados a la función `deltaE`.

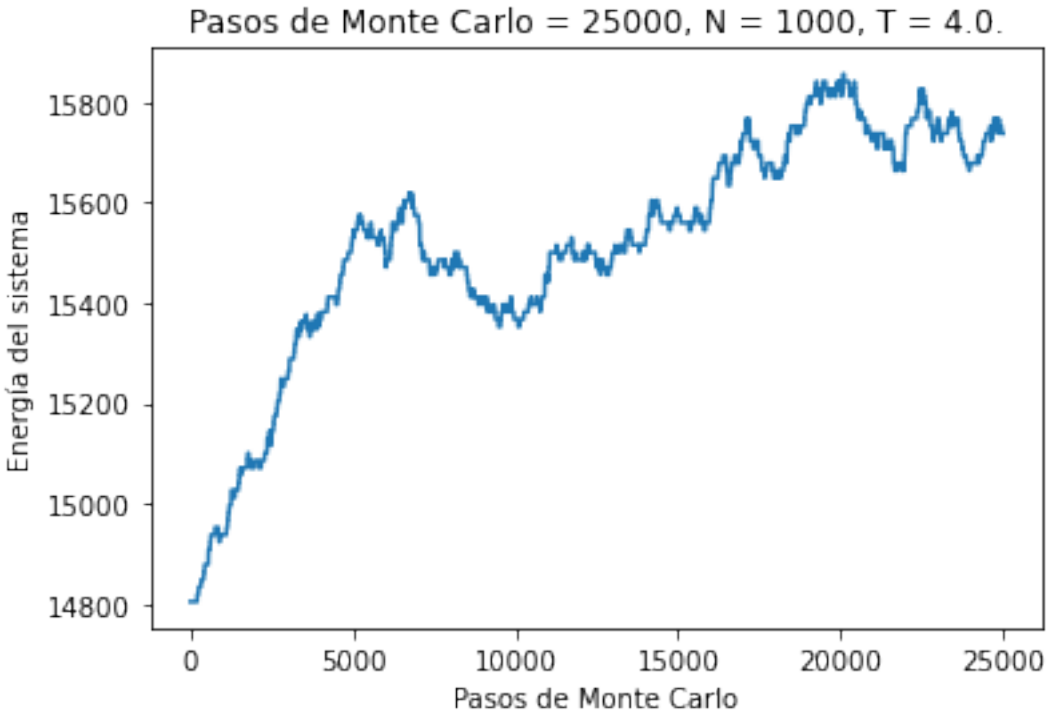
Por lo tanto, esta expresión está calculando el cambio en la energía si se incrementa o decrementa el número cuántico en la dirección seleccionada.

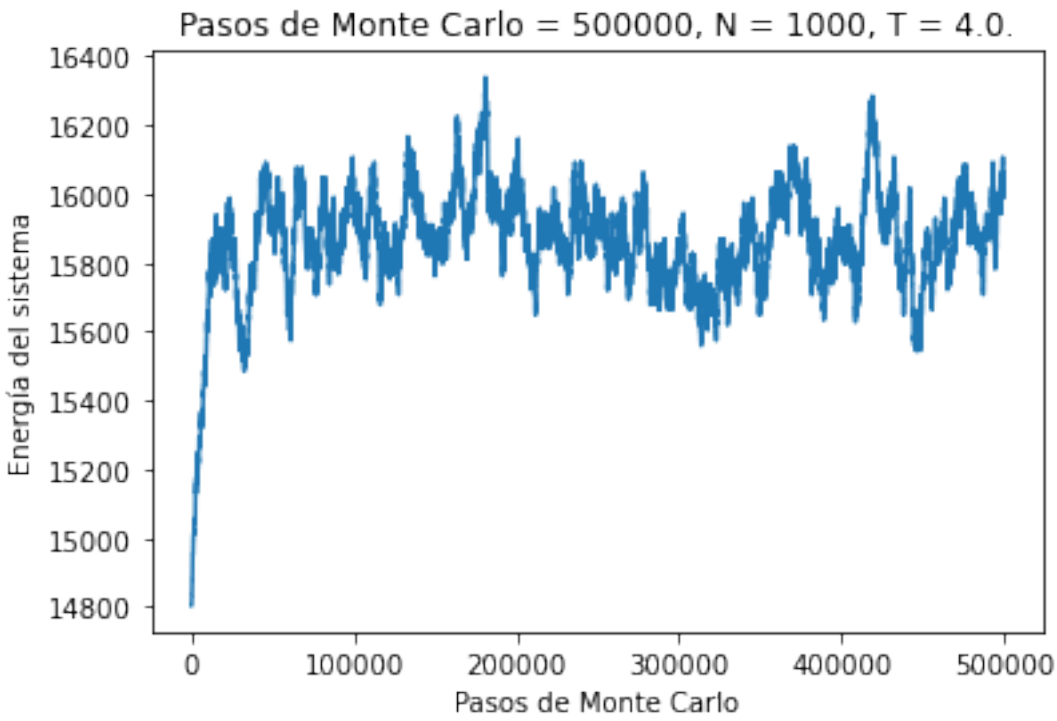
[7]:
```
for i in range(pasos):
  # Elegir una partícula al azar
  particula = np.random.randint(N)
  # Elegir una dirección al azar (0: x, 1: y, 2: z)
  direccion = np.random.randint(3)
  # Elegir un cambio al azar (+1 o -1)
  cambio = np.random.choice([-1, 1])
  # Verificar que el número cuántico no sea menor que 1
  if estado[particula, direccion] + cambio >= 1:
    # Calcular el cambio de energía
    dE = deltaE(estado[particula, 0], estado[particula, 1], estado[particula,␣
↪2], *(np.eye(3)[direccion] * cambio))
    # Generar un número aleatorio entre 0 y 1
    r = np.random.random()
    # Comparar con la probabilidad de aceptación
    if r < p_aceptar(dE, T):
      # Aceptar el cambio
      estado[particula, direccion] += cambio
      energia += dE
  # Guardar la energía en la lista
  lista_energia.append(energia)

# Graficamos la energía en función de los pasos
plt.plot(lista_energia)
plt.xlabel("Pasos de Monte Carlo")
plt.ylabel("Energía del sistema")
plt.title(f'Pasos de Monte Carlo = {pasos}, N = {N}, T = {T}.')
plt.show()
```

**VARIAMOS NÚMERO DE PASOS.** Cuanto mayor sea el número de pasos más 'oportunidades' tendrán las partículas de explorar diferentes estados. Esto también nos proporcionará una mayor precisión en los resultados esperados.

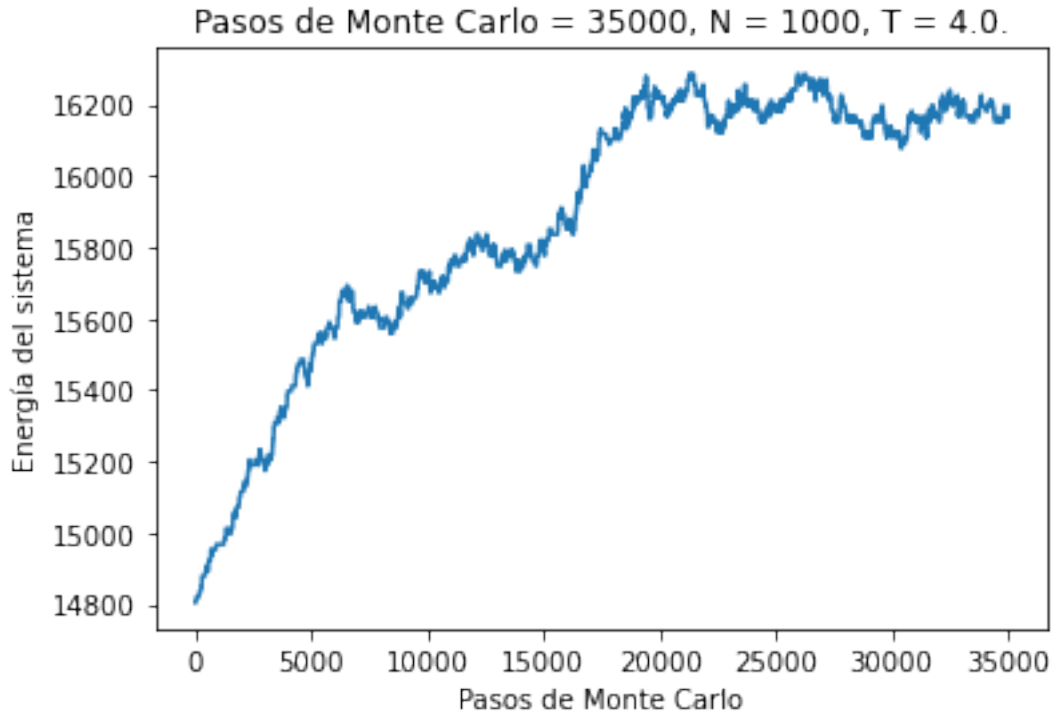
Podemos observar en los resultados gráficos que para los tres casos llegamos a ciertos valores de energía de equilibrio entorno a los cuales el sistema oscila. Para N mayores parece que se cumple lo que mencionamos, que la energía oscila con una mayor amplitud, lo que se puede entender como una mayor variedad de estados en el sistema.

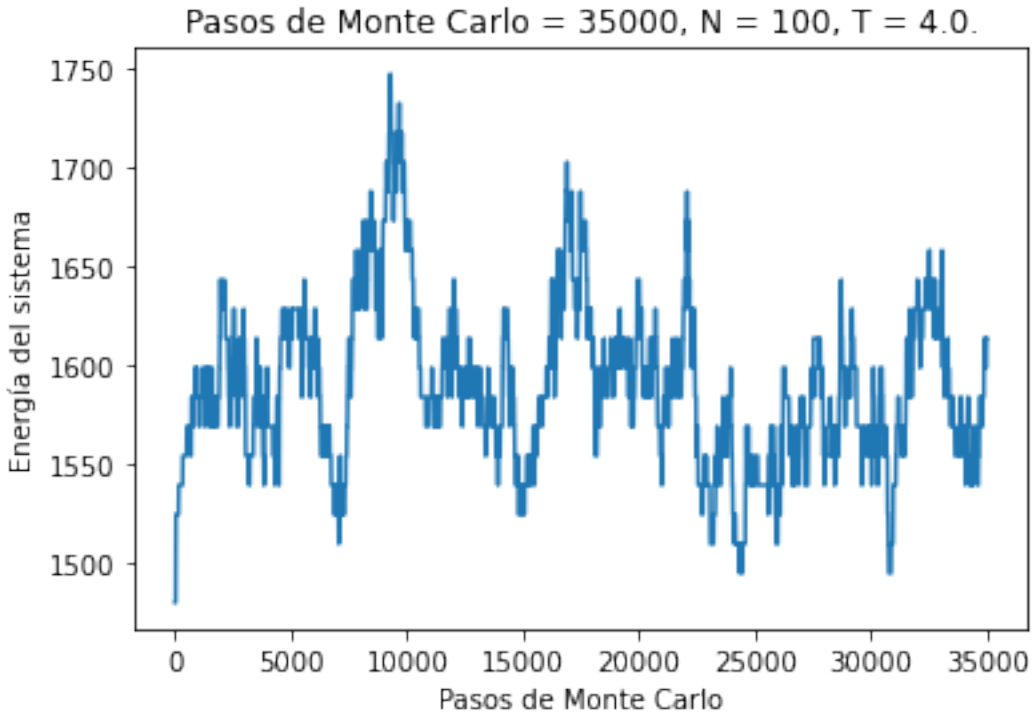

Pasos de Monte Carlo = 50000, N = 1000, T = 4.0.
Energía del sistema
16000
15800
15600
15400
15200
15000
14800
0
10000
20000
30000
40000
50000
Pasos de Monte Carlo

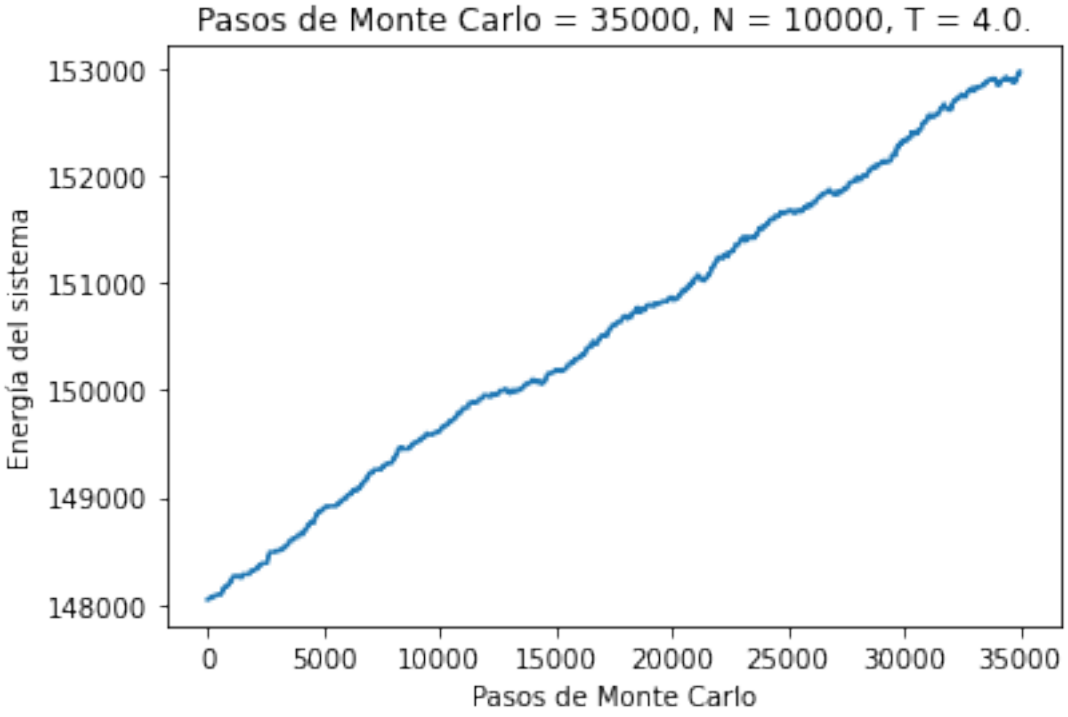

Pasos de Monte Carlo = 25000, N = 1000, T = 4.0.
Energía del sistema
15800
15600
15400
15200
15000
14800
0
5000
10000
15000
20000
25000
Pasos de Monte Carlo




**VARIAMOS NÚMERO DE PARTÍCULAS.**



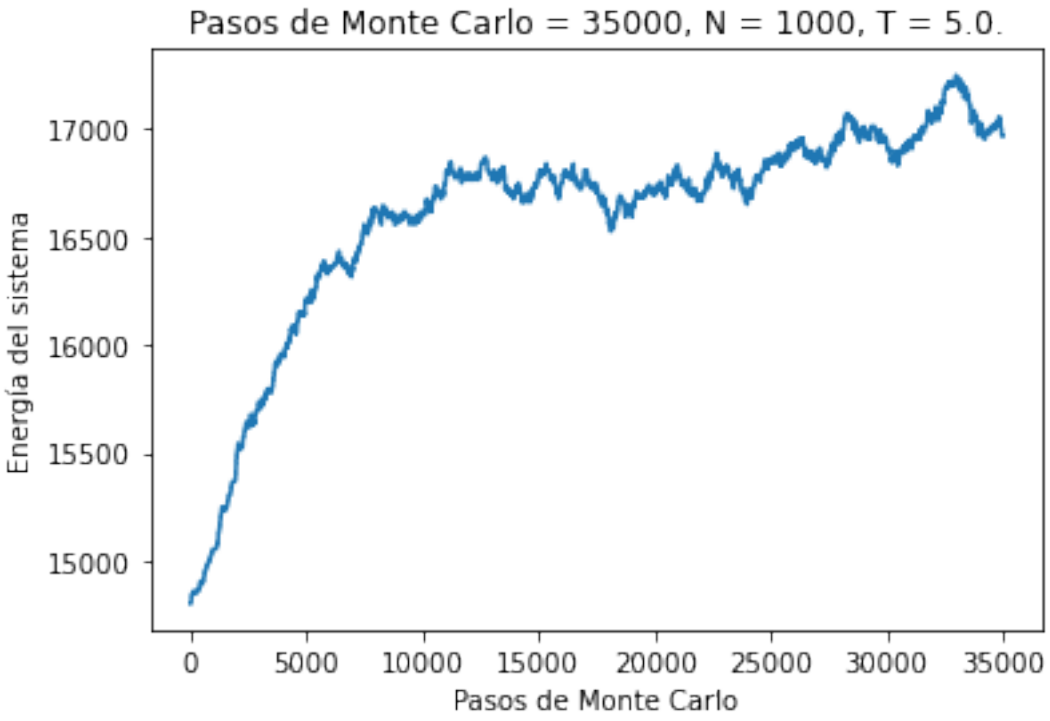

Pasos de Monte Carlo = 35000, N = 100, T = 4.0.
Energía del sistema
1750
1700
1650
1600
1550
1500
0
5000
10000
15000
20000
25000
30000
35000
Pasos de Monte Carlo

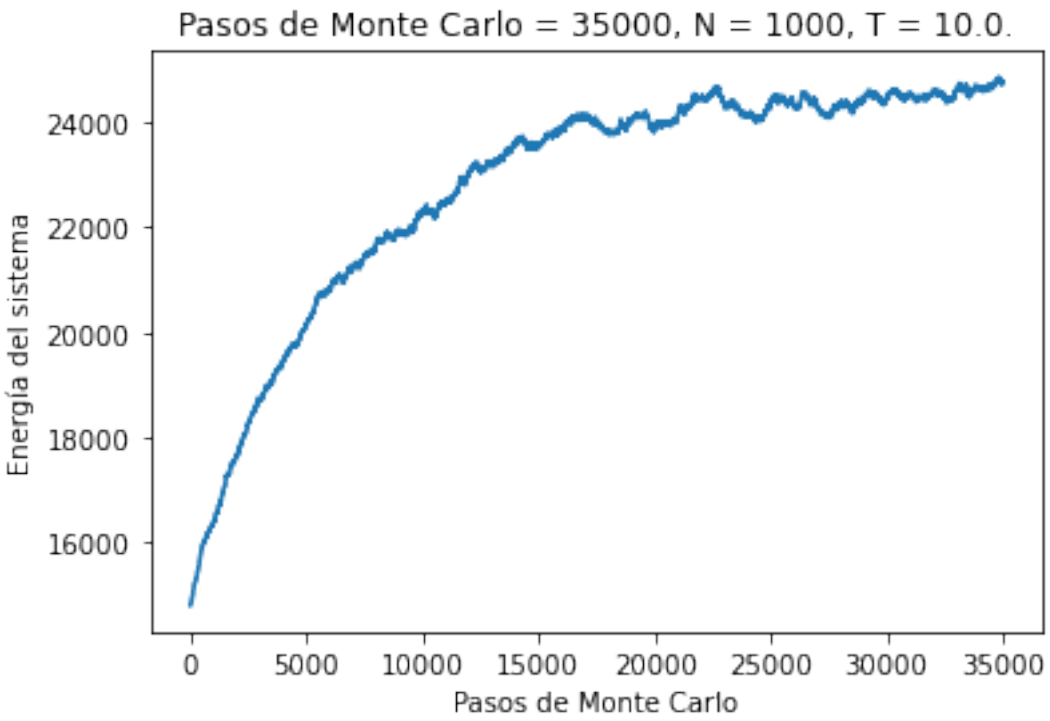

Pasos de Monte Carlo = 35000, N = 10000, T = 4.0.
Energía del sistema
153000
152000
151000
150000
149000
148000
0
5000
10000
15000
20000
25000
30000
35000
Pasos de Monte Carlo

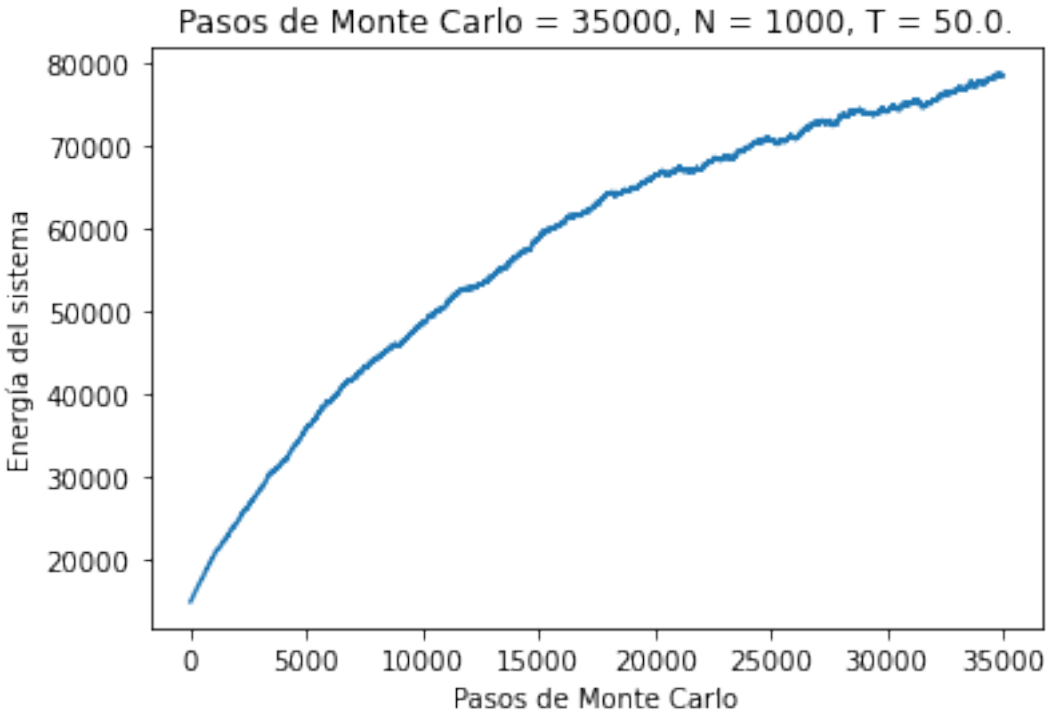
**VARIAMOS TEMPERATURA.** Este caso es el más significativo, a mi parecer. Es por ello que he añadido un caso más, respecto a los demás.

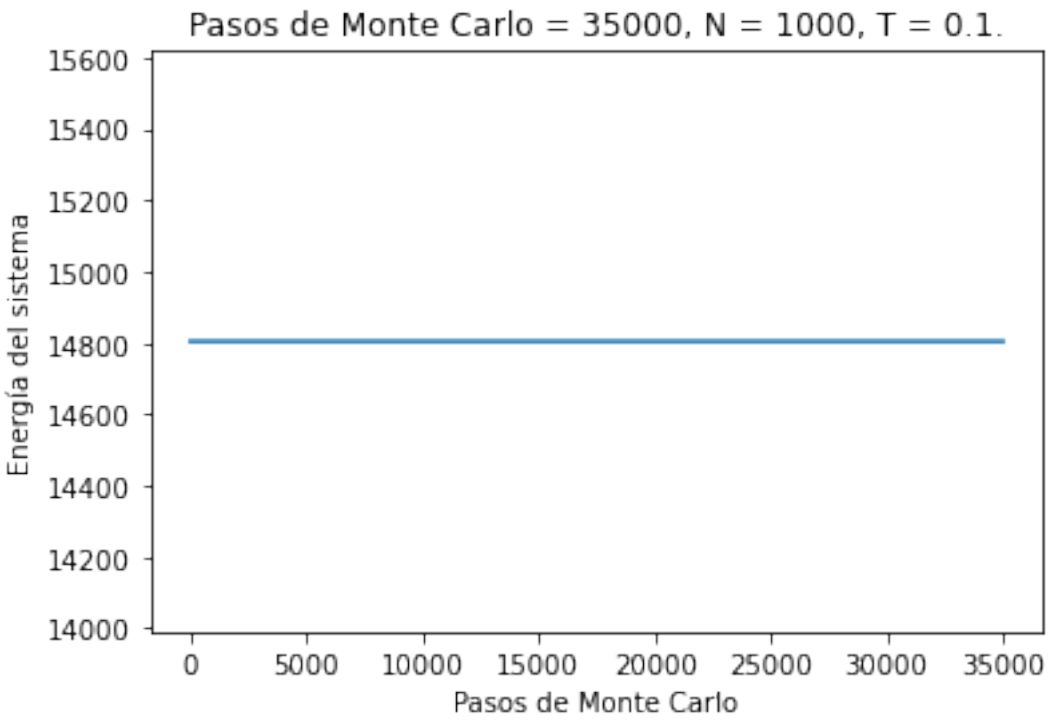
La temperatura nos afecta de forma directa a la cantidad de pasos que se aceptan en la iteración, es decir, como el parámetro de aceptación utilizado es el factor de proporcionalidad de Boltzman, vemos que la temperatura afecta de tal manera que si esta se hace menor la fracción energía/T se hará mayor y en general, 1/exp(E/T) se hará más restrictiva. En pocas palabras, que temperaturas bajas nos hace que se acepten menos cambios en el sistema. Esto lo observamos en la última gráfica, donde dando $T = 0.1$ observamos una linea recta, el sistema está congelado, energéticamente hablando.

Se puede observar también, en todas ellas, una tendencia a una energía de equilibrio, entorno a la cual el sistema oscila. Además, de forma significativa se puede observar que una mayor temperatura implica valores mayores de energía.




Pasos de Monte Carlo = 35000, N = 1000, T = 10.0.
Energía del sistema
24000
22000
20000
18000
16000
0
5000
10000
15000
20000
25000
30000
35000
Pasos de Monte Carlo


Pasos de Monte Carlo = 35000, N = 1000, T = 50.0.
Energía del sistema
80000
70000
60000
50000
40000
30000
20000
0
5000
10000
15000
20000
25000
30000
35000
Pasos de Monte Carlo


Pasos de Monte Carlo = 35000, N = 1000, T = 0.1.
Energía del sistema
15600
15400
15200
15000
14800
14600
14400
14200
14000
0
5000
10000
15000
20000
25000
30000
35000
Pasos de Monte Carlo