Sistemas de Espínes

Laura Catalina Arboleda Hernández*

*Instituto de Física, Universidad de Antioquia

1. Resumen

Se realizó un conteo directo de los microestados de un sistema de L espines para posteriormente estimar la varianza de la energía del sistema y obtener así curvas de calor específico como función de la temperatura. Se encontró que el método de directo es inviable en témrinos computacionales, debido al tiempo de computo al gasto de memoria para almacenar todos los microestados en un sistema real. Además se implementó el modelo de Ising 2D y se usó un método de montecarlo con probabilidad de aceptación de Metropolis para general microestados a partir del cambió de espín de uno de los átomos de la malla del modelo de Ising. Se comprobó que el uso de un muestreo representativo produce resultados muy cercanos a la solución real en sistemas de tamaños de varios tamaños, hasta 64×64, inalcanzable con el método directo, es decir, que el muestreo reduce el tiempo de computo y la cantidad de memoria requerida. Aún así, el muestreo de monte carlo depende fuertemente del número de pasos o intentos y un número muy grande requiere un tiempo de computo significativo, sobretodo en sistemas mucho más grandes.

Palabras Clave

Modelo de Ising, Monte Carlo, espín, calor específico.

2. Método: Cálculo Directo

Se desarrolló un código en Python para estimar el número de microestados de un sistema $L = N \times N$ átomos, generándolos todos a partir de una configuración incial dada. El código se muestra a continuación

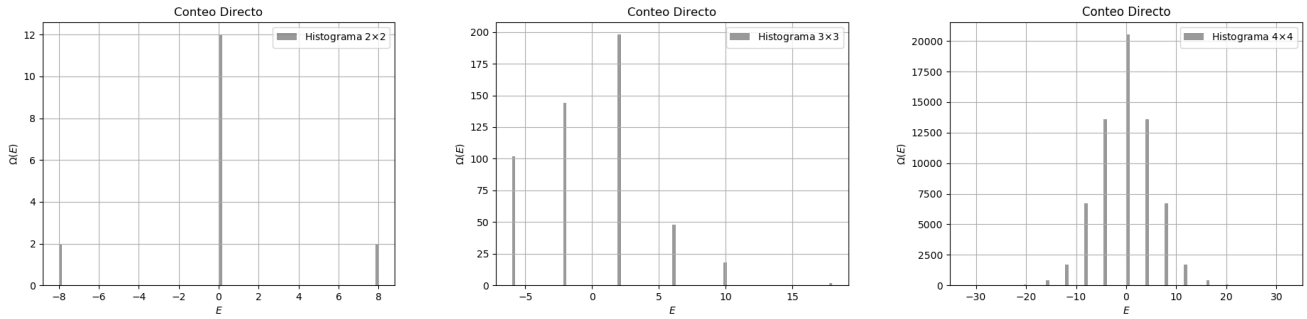
```
#-----------------------------------------
# Enegía de cada microestado y energía total
#-----------------------------------------

def all_microstates_energy(all_microstates,N):

    #tiempo de CPU
    t_init = time.time()

    Etot = E = 0.0
```

```
    Ei = []

    for k in range(2**(N*N)):

        Emicro = 0.0

        micro = all_microstates[k]

        for i in range(N):
            for j in range(N):

                #valor de los espín vecinos del átomo [i,j] en el microestado [k] con

                ngb1 = micro[i,(j+1)%N]
                ngb2 = micro[(i-1)%N,j]
                ngb3 = micro[i,(j-1)%N]
                ngb4 = micro[(i+1)%N,j]

                # dividimos entre 2 porque cada interacción se cuenta dos veces

                Eij =  micro[i,j]*(ngb1 + ngb2 + ngb3 + ngb4)/2
                Emicro += Eij

        #energía de los microestados individuales
        Ei.append(Emicro)

        #energía total del sistema
        Etot += Emicro

    t_end = time.time()
    print("Time: ",t_end-t_init)

    return Ei, Etot


#-------------------------------------
# Generación de todos los microestados
# Posibles para un sistema de L átomos
# L = N x N
#-------------------------------------

def define_microstates(N):

    L = N*N

    print("Defining microstates...")

    #number of configurations
```

```
    n_micro = 2**L

    m_prev = [-1]*L

    all_microstates = [np.reshape(np.array(m_prev),(N,N))]

    for i in range(n_micro-1):

        k = 0

        m = m_prev[:]

        while m[k] == 1:

            m[k] = -1

            k = k + 1

        m[k] = 1

        m_prev = m[:]

        all_microstates.append(np.reshape(np.array(m),(N,N)))

    return(all_microstates)


#----------------------------
#Programa Principal
#----------------------------

if len(sys.argv) < 2:
    N = 4
else:
    N = int(sys.argv[1])

print("Running with %d x %d"%(N,N))

all_microstates = define_microstates(N)

Ei, Etot = all_microstates_energy(all_microstates,N)


#-------------------------
# histograma de texto
#-------------------------

energias, OmegaE = np.unique(Ei,return_counts = True)
```

```
for i in range(len(energias)):
    print(energias[i],"\t",OmegaE[i])

#-----------------------------------------
# Histograma de energias de los microestados
#-----------------------------------------

n_bins = 100

plt.hist(Ei, bins = n_bins)
```

Con este código se generaron los microestados correspondientes a sistemas 2×2, 3×3 y 4×4. Dado que el costo computacional es muy alto, pues el número de microestados es 2^L, en la realidad es impráctico usar el método directo y en este caso no se pudo simular sistemas mayores 4×4. Los resultados se muestran en la figura 1. Se observa que el histograma para 3×3 no es simétrico, lo cuál se debe a que es un número impar de átomo por lo que la malla en la que se disponen no es cuadrada, sino rectángular. En la figura 2 se muestra la densidad de estados, es decir, el número de estados que tienen un valor dado de energía.



(a) (b) (c)

Figura 1: Histogramas de energías para sitemas de tamaño: (a) 2×2, (b) 3×3 y (c) 4×4

32	2
24	32
20	64
16	424
12	1728
8	6688
4	13568
0	20524
-4	13568
-8	6688
-12	1728
-16	424
-20	64
-24	32
-32	2

(a)

Figura 2: Densidad de estados para el sistema 4×4.

2.1. Calor Específico

El calor específico representa el incremento en la energía internal del sistema causado por un aumento *infinitesimal* en la temperatura, es decir

$$C_v = \frac{\partial \langle E \rangle}{\partial T} = \frac{\partial \beta}{\partial T} \frac{\partial \langle E \rangle}{\partial \beta} \tag{1}$$

En unidades naturales, $\beta = 1/T$, luego $\partial\beta/\partial T = -1/T^2 = -\beta^2$. Con esto, el calor específico por partícula se puede escribir como

$$\begin{aligned} c_V = \frac{C_V}{N} &= -\frac{\beta^2}{N} \frac{\partial \langle E \rangle}{\partial \beta} \\ &= -\frac{\beta^2}{N} \frac{\partial}{\partial \beta} \left[\frac{1}{\mathcal{Z}} \sum_r E_r e^{-\beta E_r} \right] \\ &= -\frac{\beta^2}{N} \left[\frac{\partial}{\partial \beta} \left(\frac{1}{\mathcal{Z}} \right) \sum_r E_r e^{-\beta E_r} + \frac{1}{\mathcal{Z}} \frac{\partial}{\partial \beta} \left(\sum_r E_r e^{-\beta E_r} \right) \right] \\ &= -\frac{\beta^2}{N} \left[-\frac{1}{\mathcal{Z}^2} \frac{\partial \mathcal{Z}}{\partial \beta} \sum_r E_r e^{-\beta E_r} - \frac{1}{\mathcal{Z}} \sum_r E_r^2 e^{-\beta E_r} \right] \end{aligned} \tag{2}$$

La función partición está dada por

$$\mathcal{Z} = \sum_r e^{-\beta E_r} \Longrightarrow \frac{\partial \mathcal{Z}}{\partial \beta} = -\sum_r E_r e^{-\beta E_r} \tag{3}$$

Así, la ecuación en 2 se reescribe como

$$c_V = \frac{\beta^2}{N} \left[-\frac{1}{\mathcal{Z}^2} \left(\sum_r E_r e^{-\beta E_r} \right)^2 + \frac{1}{\mathcal{Z}} \sum_r E_r^2 e^{-\beta E_r} \right] \tag{4}$$

Además, se tiene que

$$\langle E \rangle = \frac{1}{\mathcal{Z}} \sum_r E_r e^{-\beta E_r} \tag{5}$$

$$\langle E^2 \rangle = \frac{1}{\mathcal{Z}} \sum_r E_r^2 e^{-\beta E_r} \tag{6}$$

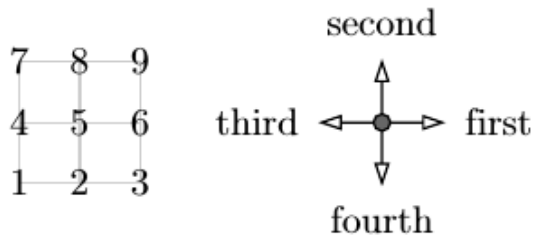
Reemplazando 5 y 6 en la ecuación 4, el calor específico por partícula es entonces

$$\frac{\beta^2}{N} \left(\langle E^2 \rangle - \langle E \rangle^2 \right) \tag{7}$$

3. Modelo de Ising

Se realizó un algorítmo para la implementación del modelo de Ising en 2D. Este modelo describe la interación entre los espines de un conjunto de átomos que se configuran en una malla, en este caso 2D, con condiciones de forntera periódicas. Cada átomo puede tener un espín $+1$ ó -1 y

las interacciones con sus primeros vecinos (espines en la malla) pueden provocar un cambio de espín. Los vecinos de un punto en la malla (átomo) se definen como se muestra en la figura 3



(a)

Figura 3: Esquema de primeros vecinos para el modelo de Ising 2D. Tomado de Krauth [2006].

La energía está dada por

$$E = -J \sum_{\langle k,l \rangle} \sigma_k \sigma_l \tag{8}$$

En donde $\sigma_k \pm 1$, con $k = 1, ..., N$ los átomos en la malla. La suma es sobre todos los pares de vecinos, pero considerando una sola vez la interacción entre un par, es decir ya sea la interacción $\langle k, l \rangle$ ó $\langle l, k \rangle$. Se desarrolló un código en python que usa una técnica de monte carlo para realizar un muestreo representativo sobre un microestado inicial, considerando posibilidad de cambio de espín de uno de los átomos en la malla, con probabilidad de aceptación de Metropolis. De esta forma, al aceptar un cambió de espín se genera un nuevo microestado. La idea es estimar la energía del sistema sin tener que considerar el número total de microestados ($\Omega = 2^L, L = N \times N$), sino sólo una cantidad representativa, que suponemos exhibira bien las propiedades del sistema.

```
#-----------------------------------------------------------
# Función para generar un estado de la malla aleatorio:
# cada átomo en un estado de espín aleatorio
#-----------------------------------------------------------

def initial_microstate(N):
    #microestado aleatorio de L átomos. L = N x N
    state = 2*np.random.randint(2, size=(N,N))-1
    return state

#-----------------------------------------------------------
# Movidas de monte carlos usando el algoritmo Metropolis
# para generar o no un microestado nuevo
#-----------------------------------------------------------

def monte_carlo_realization(microstate, N, beta):

    for i in range(N):
        for j in range(N):
```

```
            #escogemos aleatoriamente la posición del
            #átomo (s) en la malla: (fila(a),columna(b))

            a = np.random.randint(0, N) #0 y N-1
            b = np.random.randint(0, N)

            s =  microstate[a, b]

            #vecinos del átomo s:
            ngb1 = microstate[a,(b+1)%N]
            ngb2 = microstate[(a-1)%(N),b]
            ngb3 = microstate[a,(b-1)%N]
            ngb4 = microstate[(a+1)%(N),b]

            deltaE = 2*s*(ngb1 + ngb2 + ngb3 + ngb4)

            if deltaE < 0: #acepte con probabilidad igual 1 el cambio de espín
                s *= -1
            elif random.uniform(0.0,1.0) < np.exp(-deltaE*beta):
                s *= -1

            #cambie el espín del átomo y guardelo en su posición,
            #esto genera un nuevo microestado
            microstate[a, b] = s

    return microstate

#----------------------------
# Energía de un microestado
#----------------------------

def microstate_energy(microstate, N):
    energy = 0
    for i in range(len(microstate)):
        for j in range(len(microstate)):
            #S: espín del átomo en la fila i columna j
            S = microstate[i,j]
            ngb1 = microstate[i,(j+1)%N]
            ngb2 = microstate[(i-1)%N, j]
            ngb3 = microstate[i,(j-1)%N]
            ngb4 = microstate[(i+1)%N, j]
            energy += -S*(ngb1 + ngb2 + ngb3 + ngb4)/2
    return energy

#--------------------------------
# Magnetización de un microestado
#--------------------------------
```

```
def magnetization(microstate):
    mag = np.sum(microstate)
    return mag

#------------------------------
# parámetros para las pruebas
#------------------------------

#-----------------------------
#tamaño de la malla: L = N x N
#-----------------------------

N = [6,32,128]

#-----------------------------------
# Temperatura fija
#-----------------------------------

T = [0.1,1.0,2.269,5.0]

#--------------------
#pasos de monte carlo
#--------------------

n_steps = 10000

for k in N:

    #-------------------------------------------
    #fijar configuración inicial para una T fija
    #-------------------------------------------

    Microstate = initial_microstate(k)

    for j in T:

         E = []
         M = []
         t_steps = []
         E_cumulative = []
         E_cum = 0.0

         beta = 1/j

         for i in range(n_steps):

            monte_carlo_realization(Microstate, k, beta)
```

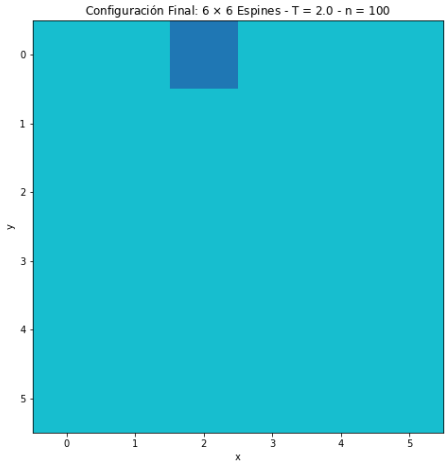
```
    #energía
    Ene = microstate_energy(Microstate, k)

    #magnetización
    Mag = magnetization(Microstate)

    #guardamos solo los valores del muestreo
    #respresentativo correspondientes
    #a la relajación del sistema: equilibrio

    if i>(n_steps/3):
        t_steps.append(i)
        E.append(Ene)
        M.append(Mag)
        E_cum =  E_cum + Ene
        E_cumulative.append(E_cum/(i+1))

Emean = sum(E)/len(E)
print("Energía media \t = %f"%Emean)

#-----------------
#archivo de datos
#-----------------

f = open('datos_tfix_nfix/%dx%d_T%0.1f_nsteps%d.dat'%(k,k,j,n_steps),'w')

for p in range (len(E)):
    f.write("%d %.8f %d %.8f\n"%(t_steps[p], E[p], M[p], E_cumulative[p]))
f.close()

#-----------------------
#graficas snap final
#-----------------------

plt.imshow(Microstate)

#-----------------------
#graficas E vs t_steps
#-----------------------

plt.plot(t_steps,E)

#-------------------------
#graficas E_cum vs t_steps
#-------------------------

plt.plot(t_steps,E_cumulative)
```

3.1. Temperatura Fija

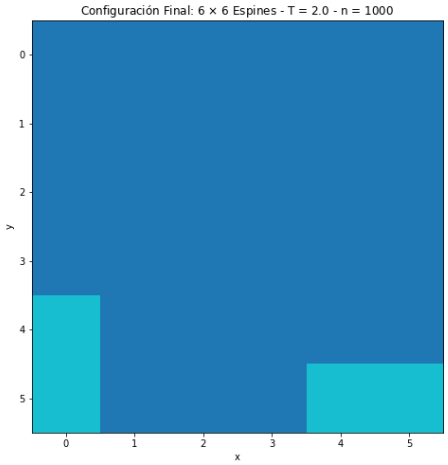
Se realizó la prueba del código para una temperatura fija y varios tamaños del arreglo de espines. En la figura 4 se muestra la configuración final dada por el muestreo de monte carlo para distinto número de pasos (o variaciones del microestado inicial) para un sistema de 36 espines ($N = 6$) a $T = 2{,}0K$. En la tabla 1 se concluye que al aumentar el número de pasos de montecarlo, el resultado se acerca más al valor obtenido con enumeración exacta y la energía media por espín se acerca más al valor $\langle E \rangle / L = -1{,}7473$.

n	$\langle E \rangle / L$
100	-1.73737375
1000	-1.72856188889
10000	-1.74627463889
100000	-1.74591913889

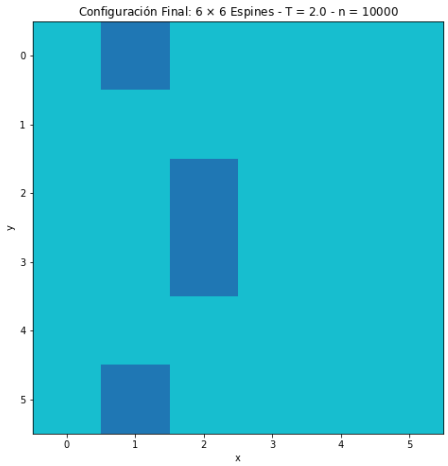
Cuadro 1: Valores de energía media por espín para el sistema 6×6 ($L = 36$), variando el número de pasos de monte carlo (n).



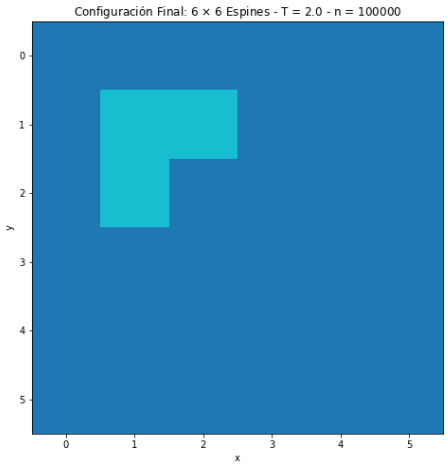
(a)



(b)



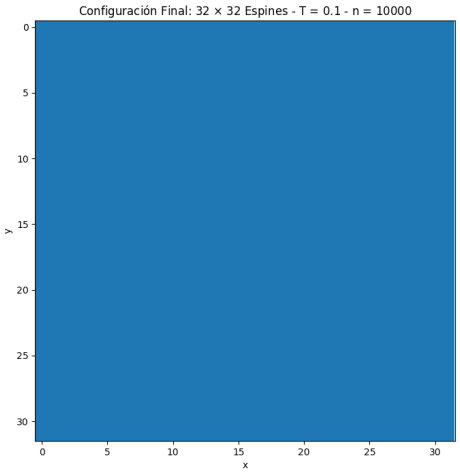
(c)



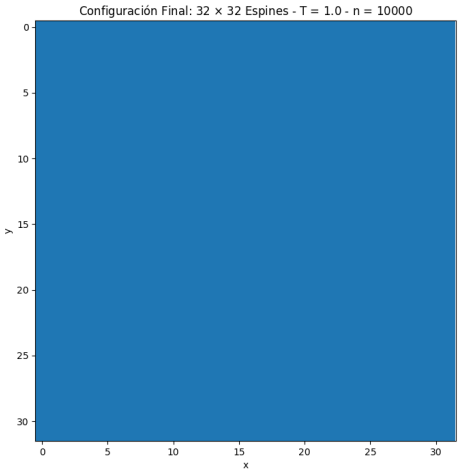
(d)

Figura 4: Configuración final típica para un sistema de 6×6 átomos para $T = 2{,}0K$ diferente número de pasos de montecarlo.

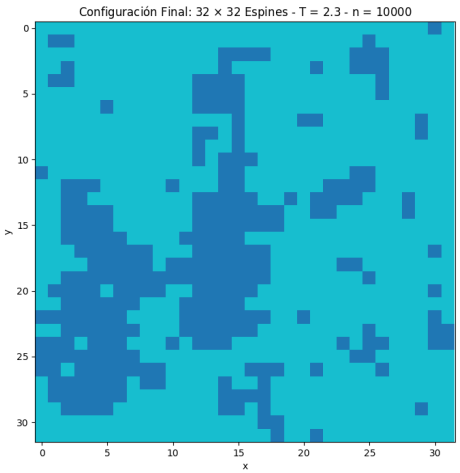
Se dejó fijo el número de pasos de monte carlo en $n = 10000$ y se varió la temperatura para el sistema 128×128 y 32×32 pasando por la temperatura crítica, en donde el sistema pasa del régimen ferromagnético al paramagnético. Los resultados se muestran en las figuras 5 y 6. Se observa que a bajar temperaturas ($T = 0{,}1K$) el sistema está dominado ya sea por átomos con espín arriba o por átomos con espín abajo. A medida que aumenta la temperatura ($T = 5{,}0K$) se puede observar que la configuración espín arriba o espín abajo, ambas, son igualmente probables, luego, la cantidad de puntos azul claro y oscuro es aproximadamente la misma.



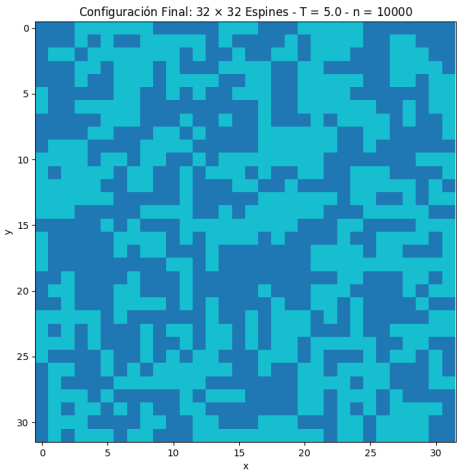
(a)



(b)

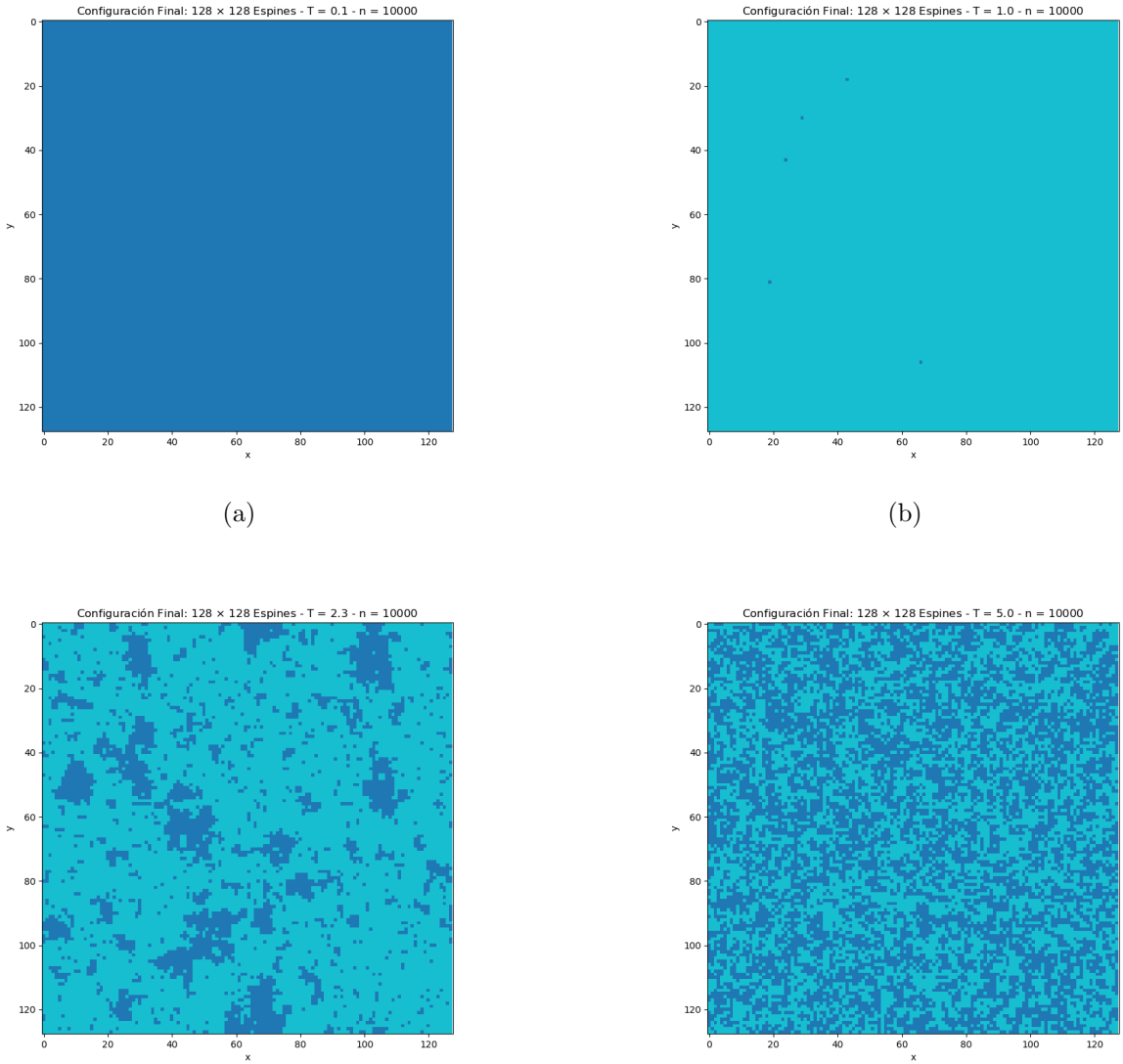


(c)



(d)

Figura 5: Configuración final típica para un sistema de 32×32 átomos para distintas temperaturas y $n = 10000$ pasos de montecarlo.

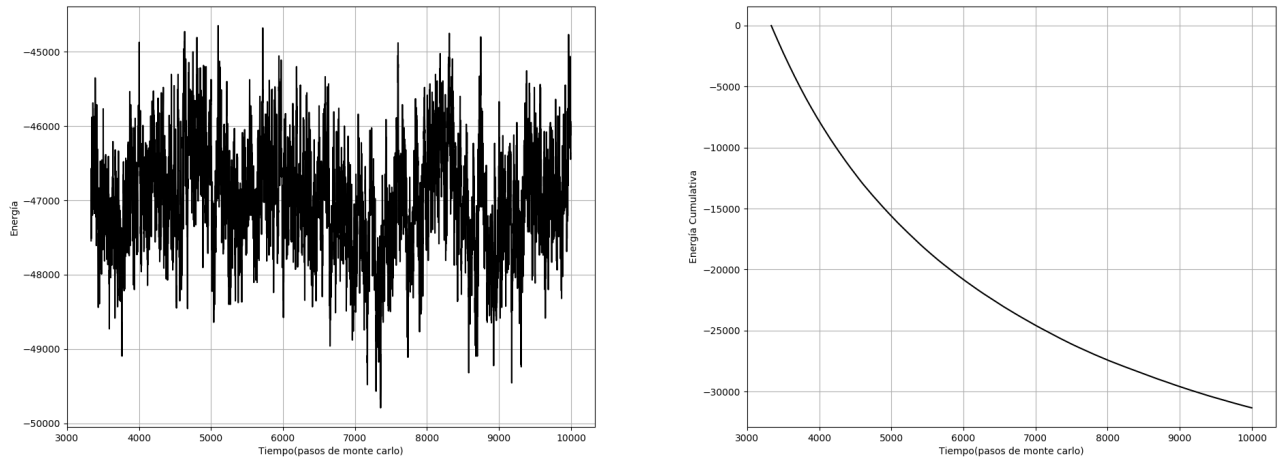


(a) (b)

(c) (d)

Figura 6: Configuración final típica para un sistema de 128×128 átomos para distintas temperaturas y $n = 10000$ pasos de montecarlo.

Se graficó la energía en cada configuración muestreada con monte carlo (metropolis) y la energía acumulada en cada paso del muestreo para el sistema 128×128 a $T = T_c \approx 2{,}3$



(a) (b)

Figura 7: Sistema de 128×128 átomos a $T_c = 2{,}269K$. (a) Energía de los microestados muestreados con 10000 pasos de montecarlo (b) Energía cumulativa de los microestados para los mismos 10000 pasos.

3.2. Energía Media y Calor Específico

Con base en el teorema de fluctuación-disipación se realizaron muestreos de montecarlo para diferentes valores de temperatura, y se estimó el valor medio de la energía para cada temperatura como

$$\langle E \rangle = \sum_{i=1}^{S} E_i \tag{9}$$

Donde E_i es el valor de energía de una configuración del sistema de espines dada con el muestreo de montecarlo. El índice i representa un paso de montecarlo y S indica que sólo sumaremos los M pasos que corresponden al sistema ya en equilibrio, es decir, la cantidad de puntos del muestreo representativo, es decir, no todas la configuraciones generadas representan el sistema en el equilibrio, solo después de un tiempo suficientemente largo (un número alto de pasos de monte carlo), el sistema estará en un estado de relajación y cada configuración representará a el sistema en un estado de equilibrio. Para un sistema ferromagnético, la energía de una configuración dada es

$$E_i = \sum_{<k,l>} \sigma_k \sigma_l \tag{10}$$

El valor esperado de la magnetización se obtuvousando el mismo principio del muestreo representativo como

$$\langle M \rangle = \sum_{i=1}^{S} M_i \tag{11}$$

En este caso la magnetización de una configuración o microestado dado por un paso de monte carlo es

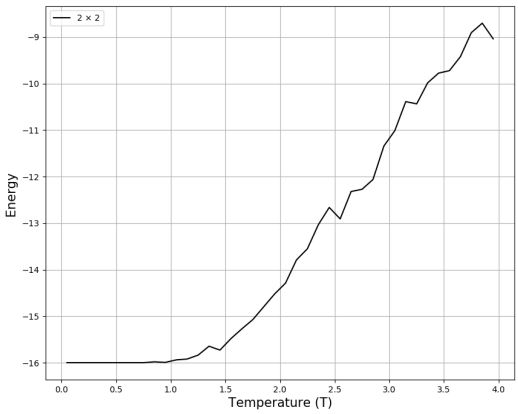
$$M_i = \sum_i \sigma_i \tag{12}$$

Se usó el mismo código descrito en la sección anterior para el muestreo de monte carlo con probabilidad de aceptación de Metropolis, pero esta vez estimando la magnetización y la energía media, como se muestra a continuación

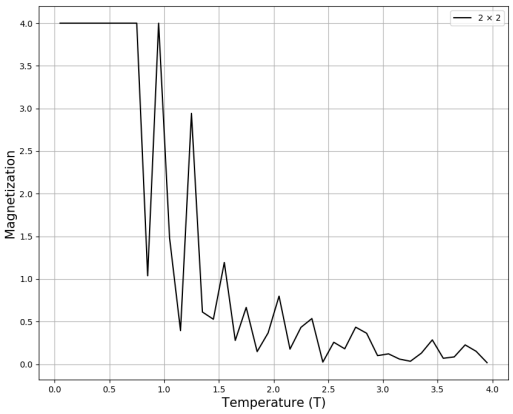
```
#-----------------------------
#tamaño de la malla: L = N x N
#-----------------------------

N = [2,4,8,16,64]

#-------------------------------------
# Temperatura fija
#-------------------------------------

Tc = 2.269
Betac = 0.4407

n_steps = 4000

T =np.arange(0.05,4,0.1)

nt = np.size(T)

n_bins = 100

E_mean          = np.zeros(nt)
E2_mean         = np.zeros(nt)
Magnetization   = np.zeros(nt)
SpecificHeat    = np.zeros(nt)

SpecificHeat_arr = []
plt.figure(figsize=(10,8))
for j in N:

    Microstate = initial_microstate(j)

    for k in range(len(T)):

        E = []
        M = []
        all_E = []

        beta=1.0/T[k]; beta2=beta**2
```

```
        for i in range(n_steps):


            monte_carlo_realization(Microstate,j, beta)

            #energía
            Ene = microstate_energy(Microstate,j)

            #magnetización
            Mag = magnetization(Microstate)

            all_E.append(Ene)

            if i>(n_steps/3):
                E.append(Ene)
                M.append(Mag)

        E_mean[k]             = sum(E)/len(E)
        E2_mean[k]            = sum(p**2 for p in E)/len(E)
        SpecificHeat[k]       = (beta2/j)*(E2_mean[k] - (E_mean[k]**2))
        Magnetization[k]      = sum(M)/len(M)


    SpecificHeat_arr.append(SpecificHeat.copy())

    #------------------------
    #graficas individuales
    #------------------------

    #energía vs T

    plt.plot(T, E_mean)

    #magnetización vs T

    plt.plot(T, abs(Magnetization))


#grafica calor específico: todos los tamaños

for j in range(len(N)):
    plt.plot(T, SpecificHeat_arr[j], '-', label=r"%d $\times$ %d"%(N[j],N[j]))
```

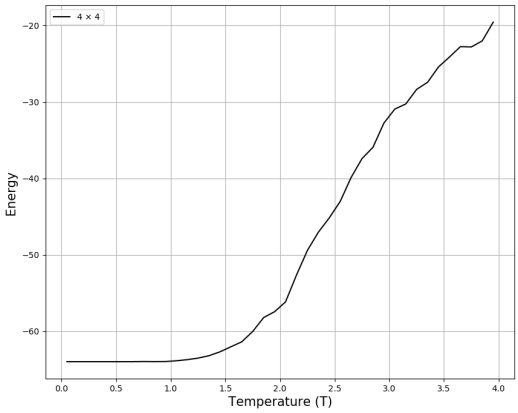
La figura



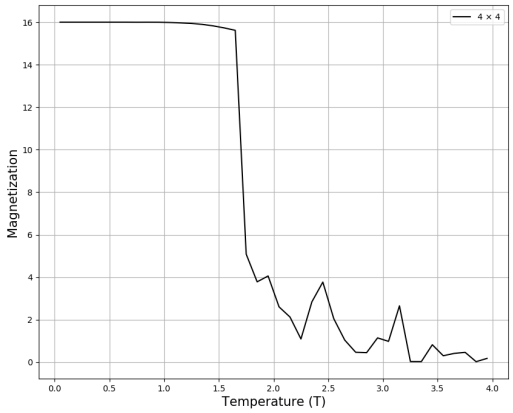
(a)



(b)



(c)



(d)

Figura 8: Valor esperado de la energía y la magnetización para un sistema de espines (a) 2×2 y (b) 4×4 con 5000 pasos de monte carlo.

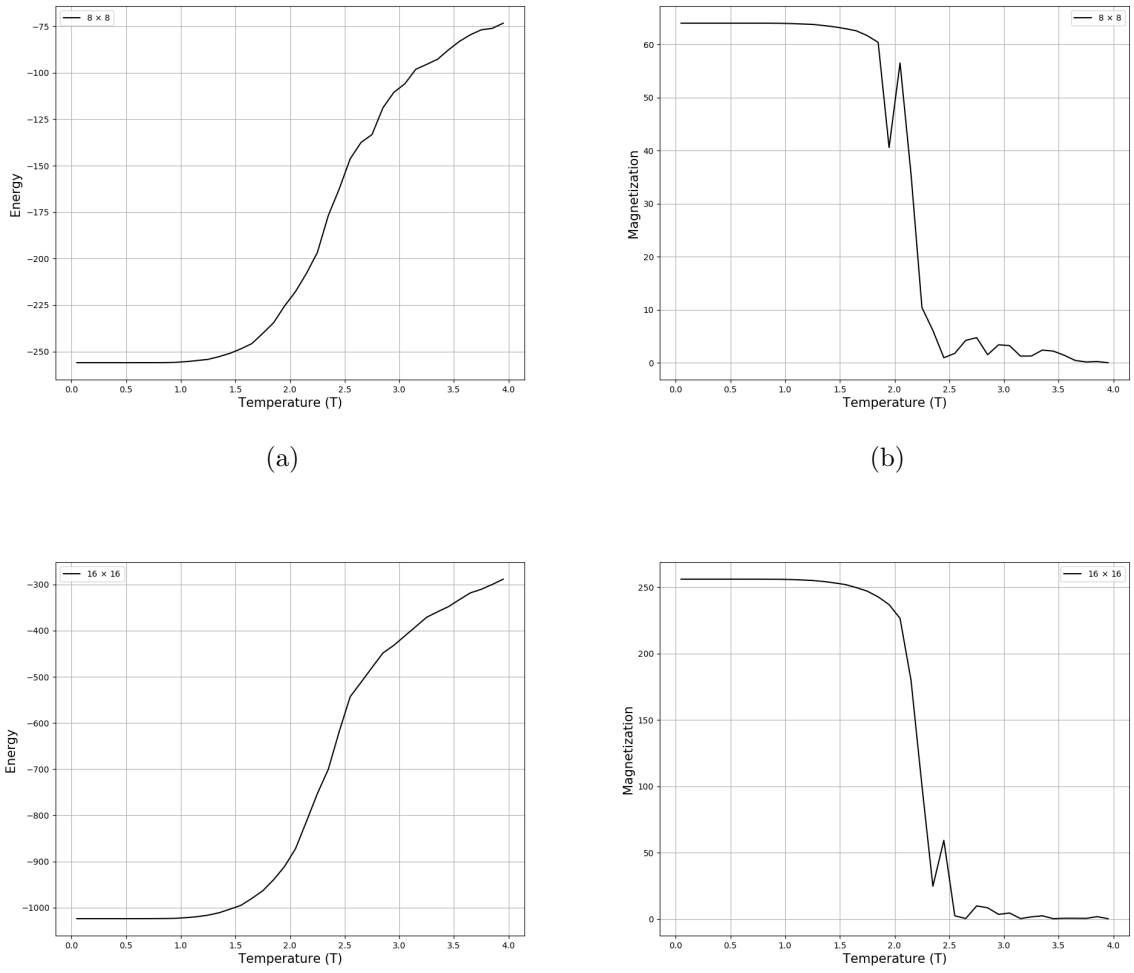
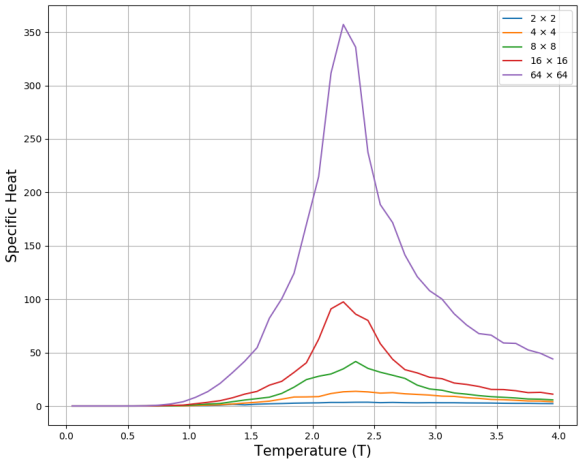


Figura 9: Valor esperado de la energía y la magnetización para un sistema de espines (a) 8×8 y (b) 16×16 con 5000 pasos de monte carlo.

En la figura 10 se muestran las curvas de calor específica para sistemas de espines de diferente tamaño. Se observa que el calor específico decae a cero a medida que la temperatura se acerca al cero absoluto. Así, se comprueba la tercera ley de la termodinámica, el cero absoluto es inalcanzable. Por otro lado, se puede observar que el pico del calor específico (máximo) se corre hacia la izquierda a medida que el tamaño de la distribución aumenta. El calor específico diverge en $T = T_c = 2{,}269$. Por encima de esta temperatura, a temperaturas altas, el sistema es paramagnético, lo que puede observarse por ejemplo en la figura en donde vemos que por encima de T_c la magnetización es cero. Por debajo de T_c el sistema es ferromagnético.



(a)

Figura 10: Curvas de calor específico para sistemas de espines de diferente tamaño.

Referencias

W. Krauth. *Statistical mechanics: algorithms and computations*, volume 13. OUP Oxford, 2006.