Práctica 7-
Modelo de Ising bidimensional

Asignatura: *Computación Avanzada*

Profesor: *David Martín y Marero*

Luis Sánchez Izquierdo | domingo 22 de abril de 2018

Contents

Introducción

Teoría

En esta práctica vamos a estudiar el modelo de Ising bidimensional. Se trata de un método físico para estudiar el comportamiento de un material bidimensional en que tiene espines que pueden tomar los valores de $s = +\frac{1}{2}$ o bien $s = -\frac{1}{2}$

La forma de abordar el problema es considerar el colectivo NHT, es decir, tenemos un número fijo de partículas N sometidas a un campo magnético H y bajo una temperatura T.

Computacionalmente se usa el método de Montecarlo. Este método consiste básicamente en crear un gran número de microestados sobre los que haremos un promedio estadístico. Ara generar los microestados los vamos variando aleatoriamente en función de una función de probabilidad que determina si un cambio es aceptado o no.

Objetivos

La práctica consiste en:

- Desarrollar el algoritmo computacional para realizar la simulación de Montecarlo del modelo de Ising bidimensional
- Hacer un primer análisis a $H = 0$ de la evolución de la magnetización M en función de los pasos de Montecarlo $t_{Montecarlo}$
- Ver la dependencia de la energía media $< E >$ y de la magnetización media $< M >$ en función de la temperatura del colectivo T
- Estudiar el valor del calor específico C_V
- Variando el campo magnético H estudiar el ciclo de histéresis para la magnetización media $< M >$

Además, realizar un apartado extra en el que considero la interacción a segundos vecinos.

Desarrollo del algoritmo computacional para la simulación de Montecarlo

- ## Definiciones iniciales

La idea es crear un mallado del espacio sobre la que se van a situar las partículas con espín. Este mallado es una matriz llamada "$espines$" de dimensión $N \cdot N$ en la que guardaré variables de tipo *int*. Si la variable es $+1$ la partícula tendrá espín $+\frac{1}{2}$ y si la variable es -1 entonces el espín es $-\frac{1}{2}$. Como la matriz tiene dimensiones $N \cdot N$ entonces hay un total de $N_T = N^2$ partículas. Normalmente tomaré como valor $N = 10$. A parte de esto habrá que definir la temperatura T y la magnetización H

```
N = 10          # numero de espines por lado
Nm1 =N -1       # es útil
NT = N * N      # numero total de espines

# inicializar espines
espin = zeros([N,N],int)
espin_ini = zeros([N,N],int)

# asignar valores aleatorios
for i in range(N):
    for j in range(N):
        espin_ini[i,j]=choice([int(-1),int(1)])
```

- ## Calculo de la energía y la magnetización del microestado

La magnetización viene dada por la suma de los espines individuales

$$M = \sum_i s_i$$

Esto se implementa de forma directa:

```
# Función para calcular la magnetización por espin
def Magnetizacion():
    Maux = 0
    for i in range(N):
        for j in range(N):
            Maux += espin[i,j]
    return Maux / NT
```

Por otro lado, en el modelo de Ising la energía surge de la interacción a primeros vecinos entre espines.

$$E = -\sum_{\substack{primeros \\ vecinos}} J \cdot s_i \cdot s_j + H \cdot M$$

En donde tomamos $J = 1$

Normalmente tomaremos $H = 0$ excepto cuando calculemos el ciclo de histéresis en el último apartado.

Como el sistema es bidimensional lo que se hace es considerar que las partículas de los bordes interaccionar con las del borde opuesto. Es decir, estamos tomando unas condiciones de contorno en la que consideramos que el sistema está abierto.

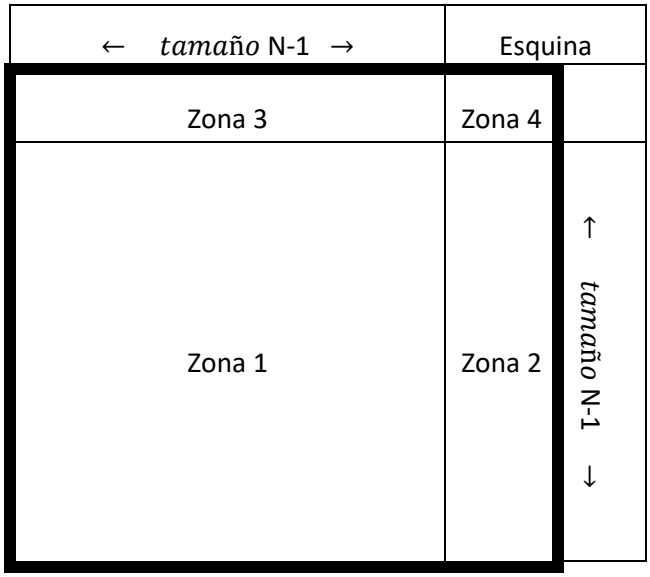
Esto lo he implementado de forma que computacionalmente sea lo más efectivo posible. Para ello lo que hago es definir una función en la que no hace falta usar condicionales *if*.

```
# Función para calcular la energía total
def EnergiaTotal():
    energy = 0
    # Paso 1
    # i!=Nm1 and j!=Nm1
    for i in range(Nm1):
        for j in range(Nm1):
            energy += espin[i,j] * ( espin[i,j+1] + espin[i+1,j] )
    # Paso 2
    # i=!Nm1 and j=Nm1
    for i in range(Nm1):
        energy += espin[i,Nm1] * ( espin[i,0] + espin[i+1,Nm1] )
    # Paso 3
    # i=Nm1 and j!=Nm1
    for j in range(Nm1):
        energy += espin[Nm1,j] * ( espin[Nm1,j+1] + espin[0,j] )
    # Paso 4
    # j=Nm1 and i=Nm1
    energy += espin[Nm1,Nm1] * ( espin[Nm1,0] + espin[0,Nm1] )
    return -energy
```

El algoritmo consiste en recorrer todas las partículas y e ir añadiendo la interacción con su vecino superior y derecho (de modo que así sólo contamos la interacción entre 2 partículas una única vez y no tenemos que dividir entre 2 al final, ahorrando la mitad del tiempo).

Además, para aplicar las condiciones de contorno no hay condicionales if. Para evitar usar condicionales lo que hago es dividir todo el sistema en 4 zonas y a cada zona dedicarle un bucle. Como en Python si hacemos $matriz[-1]$ lo entiende, no habrá problema con las partículas que estén en el borde inferior e izquierdo. Sin embargo, para las que están en los bordes derechos e izquierdo sí que hay problemas al considerar los vecinos superiores.

Las zonas son:



- ## Variación de los microestados y probabilidad de aceptación

Aquí es donde reside la esencia del método de Montecarlo. Para variar un microestado lo que se hace es:

- Elegir una partícula al azar
- Calcular la variación de energía que produce al cambiar el espín

$$\Delta E = E_{final} - E_{inicial}$$

- Si la energía es $\Delta E < 0$ se acepta el paso
- Sino entonces se haya la probabilidad de cambio $P = e^{-\frac{\Delta E}{T}}$ y se genera un número aleatorio entre $random \in (0,1)$
- Si $r < P$ entonces se acepta el cambio

```
124         for i in range(NT):     # realizar un total de NT cambios (1 paso Montecarlo)
125             # escoger una partícula al azar
126             indx = randint(N)
127             indy = randint(N)
128 
129             DE = DEnergiaSpin(indx,indy)    # hallar la variación en energía
130             DM = -2 * espin[indx,indy] / NT # variacion en magnetización
131 
132             if DE <= 0:                     # se acepta el cambio
133                 espin[indx,indy] *= (-1)
134                 M.append(M[-1]+DM)
135                 Maux += M[-1]
136 
137             elif random() < exp (-DE/T):    # también se acpeta el cambio
138                 espin[indx,indy] *= (-1)
139                 M.append(M[-1]+DM)
140                 Maux += M[-1]
141 
142             else:                           # no se acepta el cambio
143                 M.append(M[-1])
144                 Maux += M[-1]
```

Nota: realizar un paso de Montecarlo significa intentar mover las N_T partículas

- ## Optimización del código: cálculo de la variación de energía

La clave para realizar una optimización del código reside en el cálculo de la variación de energía. Para realizar este cálculo se podría usar la función de energía total y expresar la variación de energía como

$$\Delta E = Energía Total()_{microestado\ final} - Energía Total()_{microestado\ inicial}$$

Sin embargo, este cálculo es muy pesado e ineficiente si se realiza muchas veces ya que tienes que recorrer las N_T partículas. La solución radica en darse cuenta que la variación de energía solo depende de la partícula que estás considerando y sus 4 vecinos. Así pues, la variación de energía se puede calcular con:

```python
# Función para calcular la variación de energía al cambiar un spin
def DEnergiaSpin(i,j):      # con CC
    # Posibilidad 1: que esté en la zona 1
    # i!=Nm1 and j!=Nm1
    if (i != Nm1 and j != Nm1):
        return (-2) * ( - espin[i,j]) * ( espin[i,j+1] + espin[i+1,j] + espin[i,j-1] + espin[i-1,j] )
    # Posibilidad 2: que esté en la zona 2
    # i!=Nm1 and j=Nm1
    elif (i != Nm1):
        return (-2) * ( - espin[i,Nm1]) * ( espin[i,0] + espin[i+1,Nm1] + espin[i,Nm1-1] + espin[i-1,Nm1] )
    # Posibilidad 3: que esté en la zona 3
    # i=Nm1 and j!=Nm1
    elif (j != Nm1):
        return (-2) * ( - espin[Nm1,j]) * ( espin[Nm1,j+1] + espin[0,j] + espin[Nm1,j-1] + espin[Nm1-1,j] )
    # Posibilidad 4: que esté en la zona 4
    # j=Nm1 and i=Nm1
    return (-2) * ( - espin[Nm1,Nm1]) * ( espin[Nm1,0] + espin[0,Nm1] + espin[Nm1-1,Nm1] + espin[Nm1,Nm1-1] )
```

En donde la variación de energía será

$$\Delta E = -2 \cdot EnergíaSpin(i,j)$$

$$\Delta E = 2 * s_{ij} * (s_{i+1,j} + s_{i,j+1} + s_{i-1,j} + s_{i,j-1})$$

Como se puede ver, de nuevo se tiene en cuenta la condición de contorno al dividir el espacio en las cuatro zonas. A partir de esta cantidad se puede calcular la energía tras el cambio como

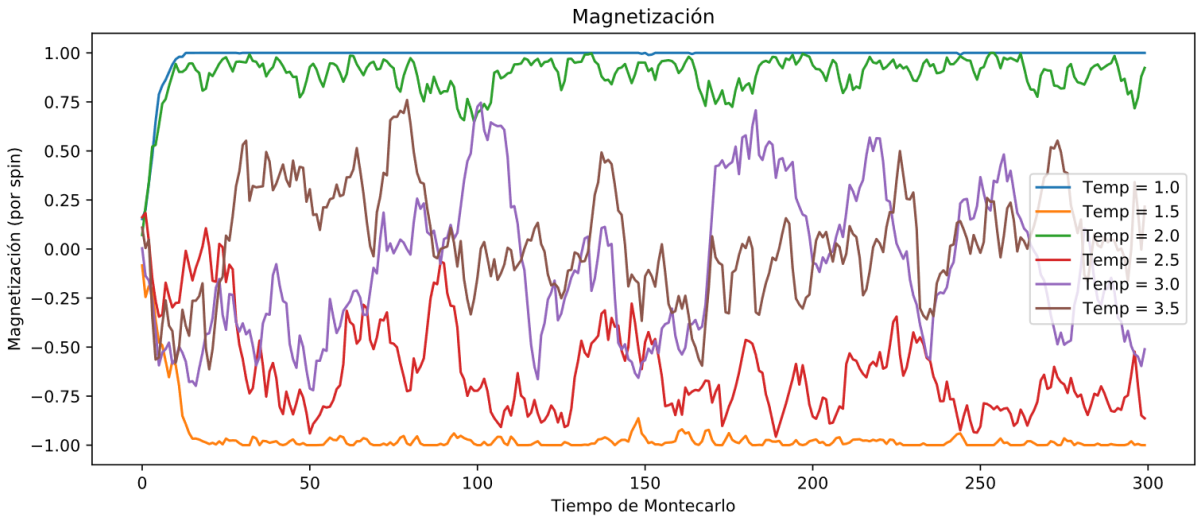
$$E_{microestado\ final} = E_{microestado\ inicial} + \Delta E$$

De modo que no hay que usar la función de la energía total que consume mucho tiempo computacional.

Resultados I: magnetización en función del paso de Montecarlo

Teniendo el algoritmo desarrollado lo que queda es realizar diversos cálculos y explicarlos. Primeramente, tenemos el cálculo de magnetización en función del paso de Montecarlo. Lo que hago aquí es hacer la media de la magnetización para cada paso de Montecarlo. Realizo este proceso para varias temperaturas diferentes.

Los resultados se muestran a continuación:

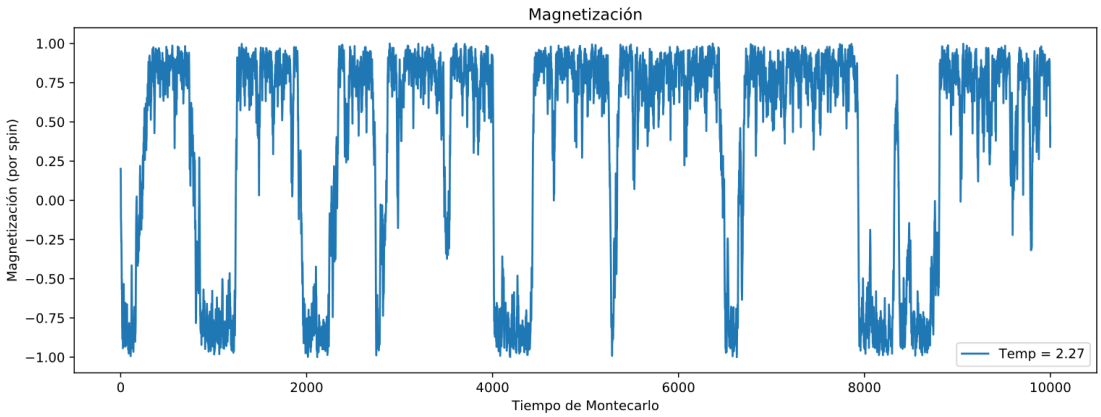


Lo que se representa es la magnetización por espín (los límites $+1$ y -1 indican espín $+\frac{1}{2}$ y $-\frac{1}{2}$ respectivamente)

Se puede ver que para una temperatura de $T = 1$ o de $T = 1{,}5$ e incluso para $T = 2$ como estamos por debajo lo espines se alinean para estar en la configuración de menor energía posible. El hecho de que los espines espines estén alineados significa que tienen una magnetización tanto $+1$ como -1.

Por otro lado, cuando superamos la temperatura crítica $T = 2.27$ (las gráficas rojas, marrones y moradas) se puede ver que el comportamiento es aleatorio y varía en torno al centro $M = 0$

Además he realizado una representación para cuando estamos en la temperatura crítica y se puede ver cómo fluctúa de $+1$ a -1

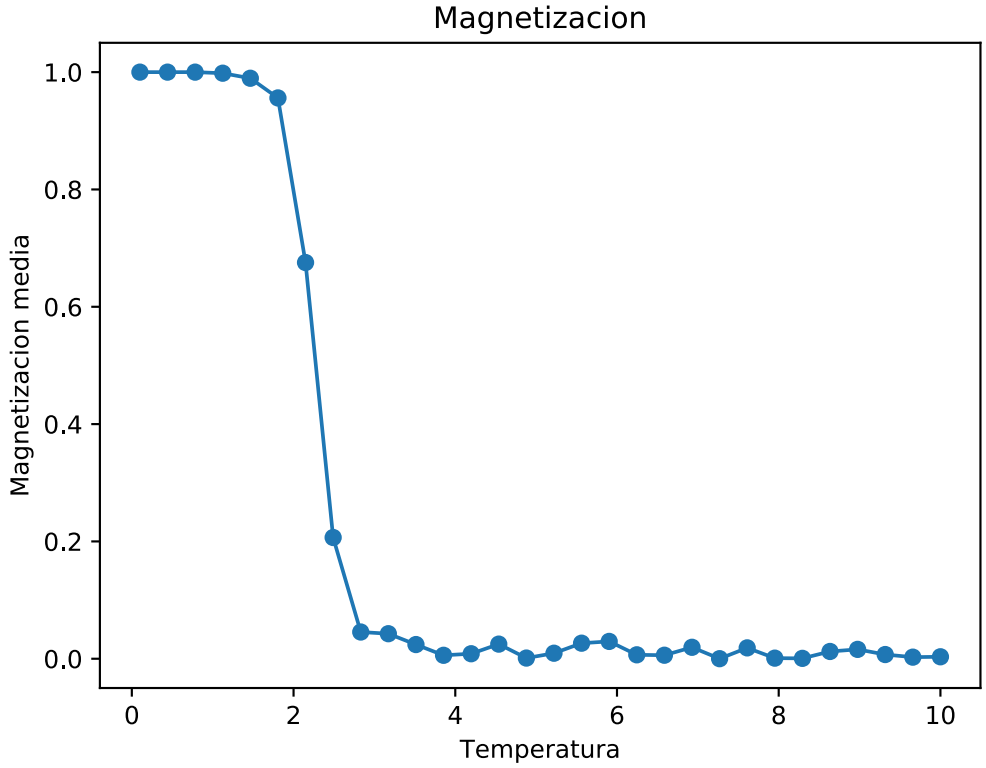


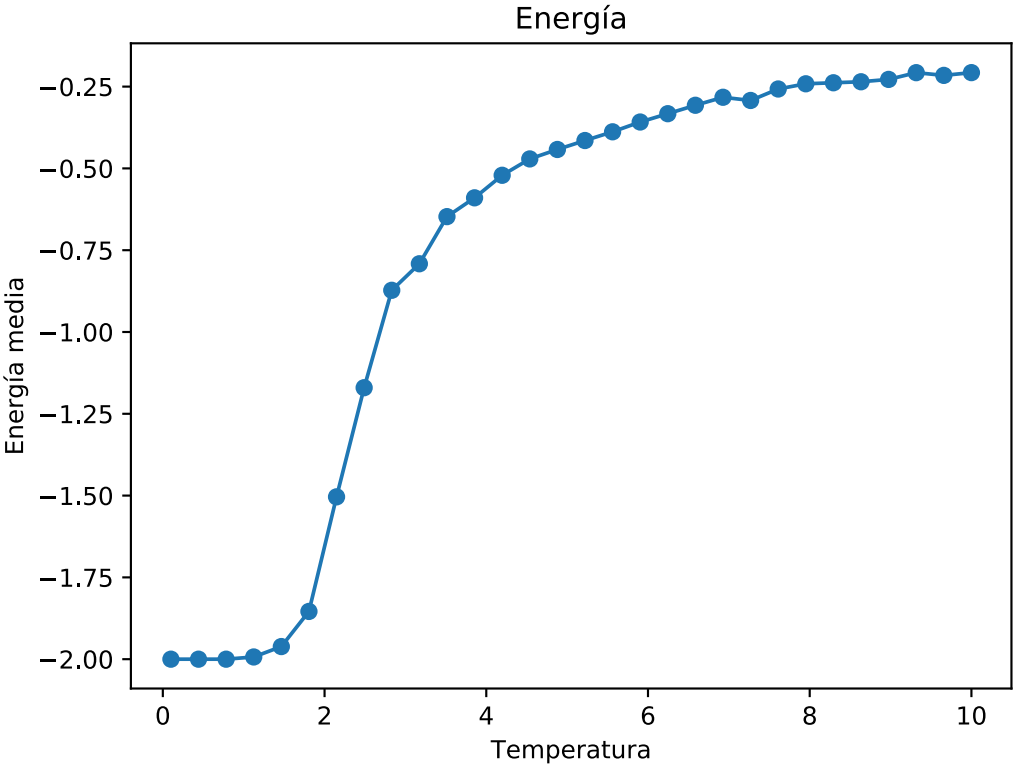
Resultados II: energía y magnetización en función de la temperatura

En este apartado lo que se hace es realizar simulaciones para sistemas con diversas temperaturas para calcular el promedio de energía y magnetización y ver así las dependencias con T

En mi código el tiempo de ejecución para un total de $N_{temperaturas} = 40$ es de tan solo $t_{computacional} = 17.5s$, lo que muestra que el código verdaderamente está optimizado.

Los resultados se muestran a continuación:





Como se puede ver, la magnetización sigue el patrón de modelo de Ising. A temperaturas mayores que $T_{crítica}$ el valor es $< M > = 0$

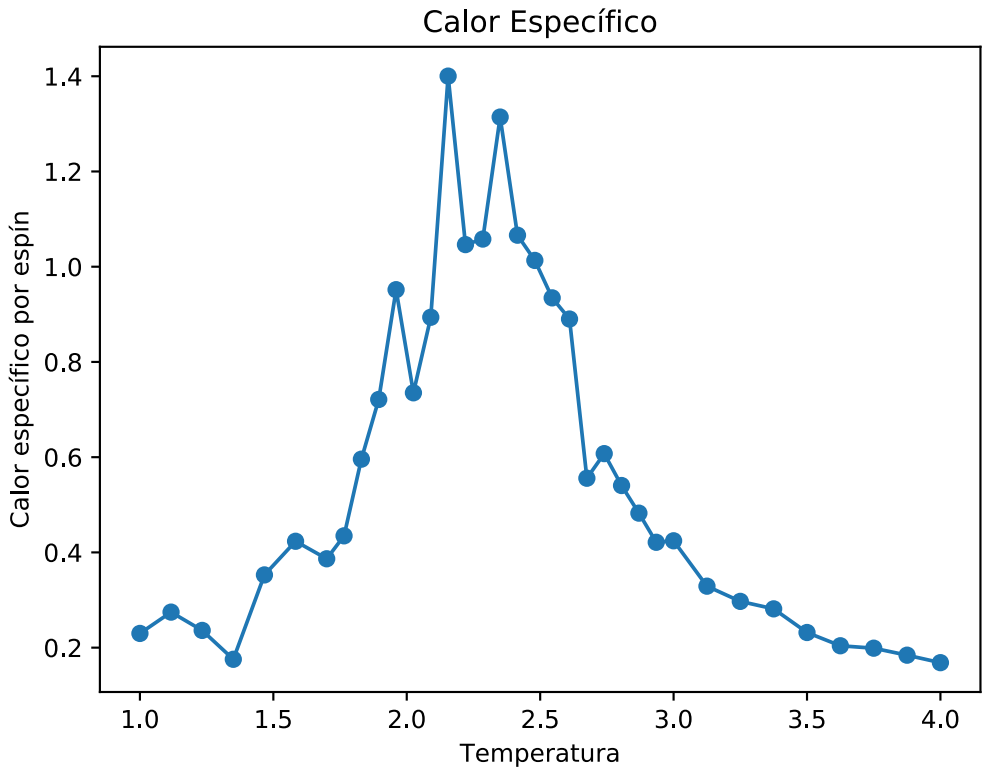
Por otro lado, la energía media se puede ver cómo pasa de estar en el mínimo $< E > \approx -2$ a $T < T_{crítica}$ para luego tender a $< E > \approx 0$ para $T > T_{crítica}$ pero sin llegar a alcanzar ese límite.

Resultado III: el calor específico

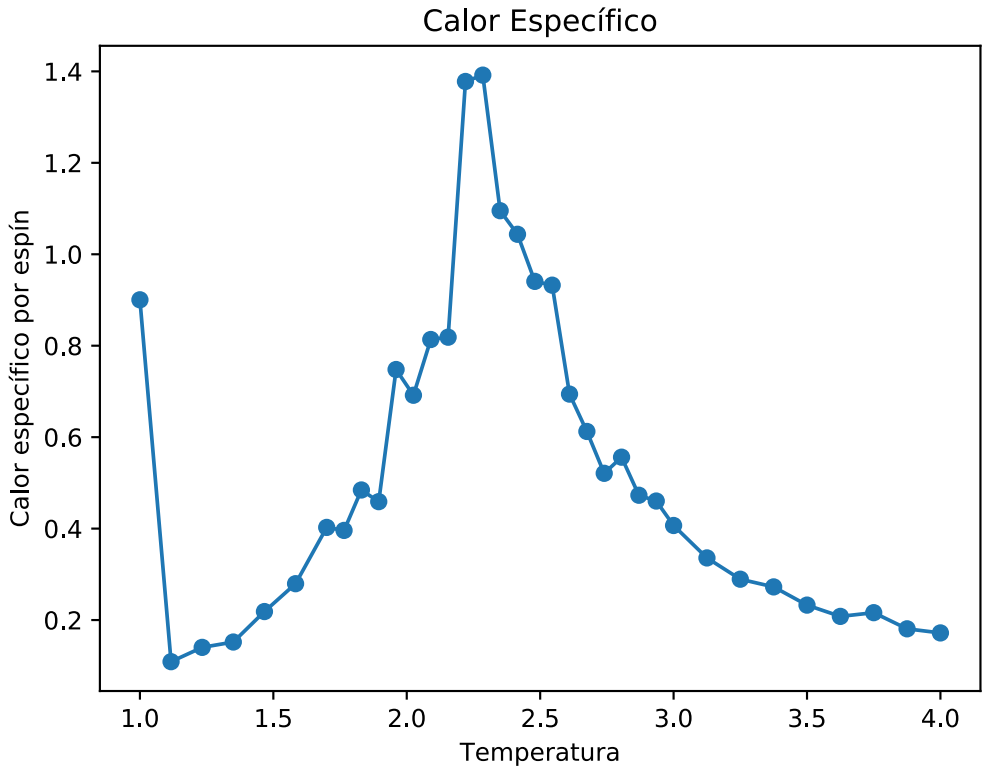
En este apartado lo que se calcula es el calor específico a partir de la ecuación

$$C_V = \frac{\Delta E^2}{KT^2}$$

Lo que se debe obtener es un pico en la altura de $T = T_{crítica}$ relacionado con la transición de estado. Como se observa a continuación se puede ver que los resultados son correctos:



Y para un sistema de $12\ x\ 12$ partículas el resultado es:

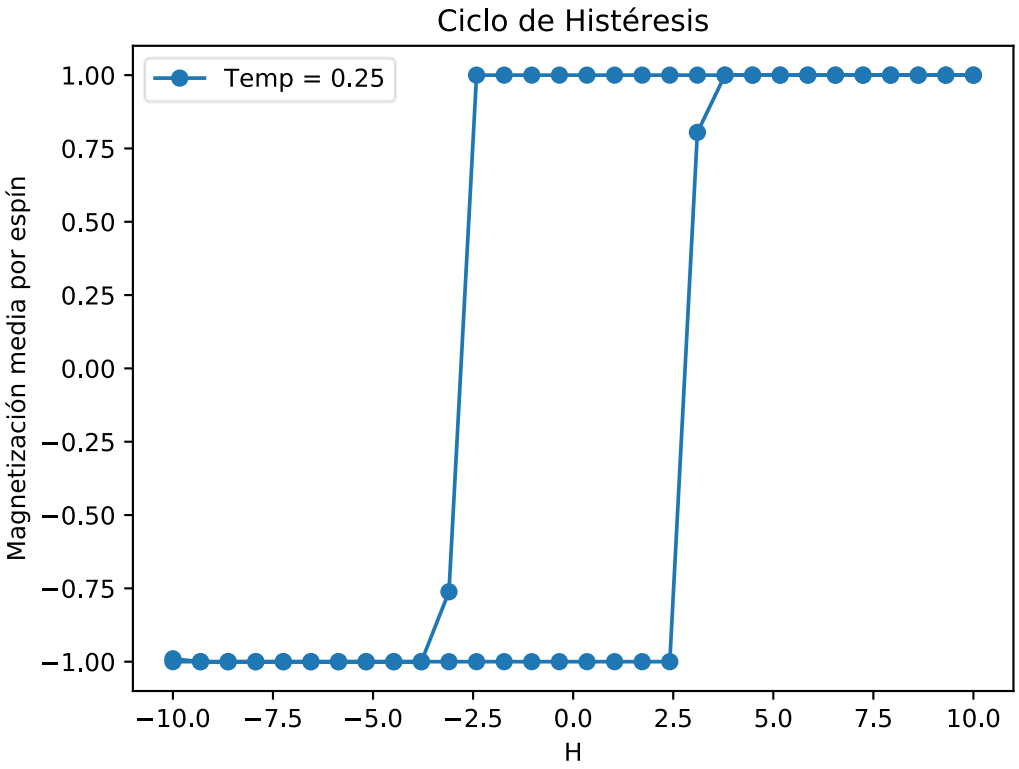


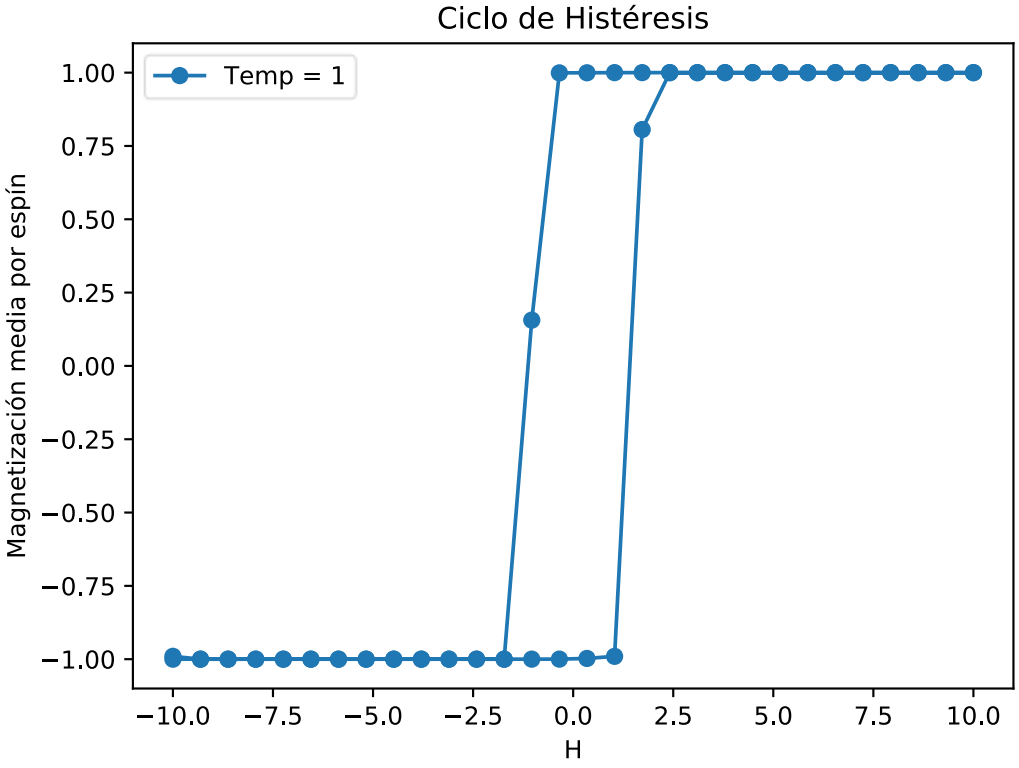
A medida que aumenta el número de partículas el pico cada vez es mayor y más estrecho.

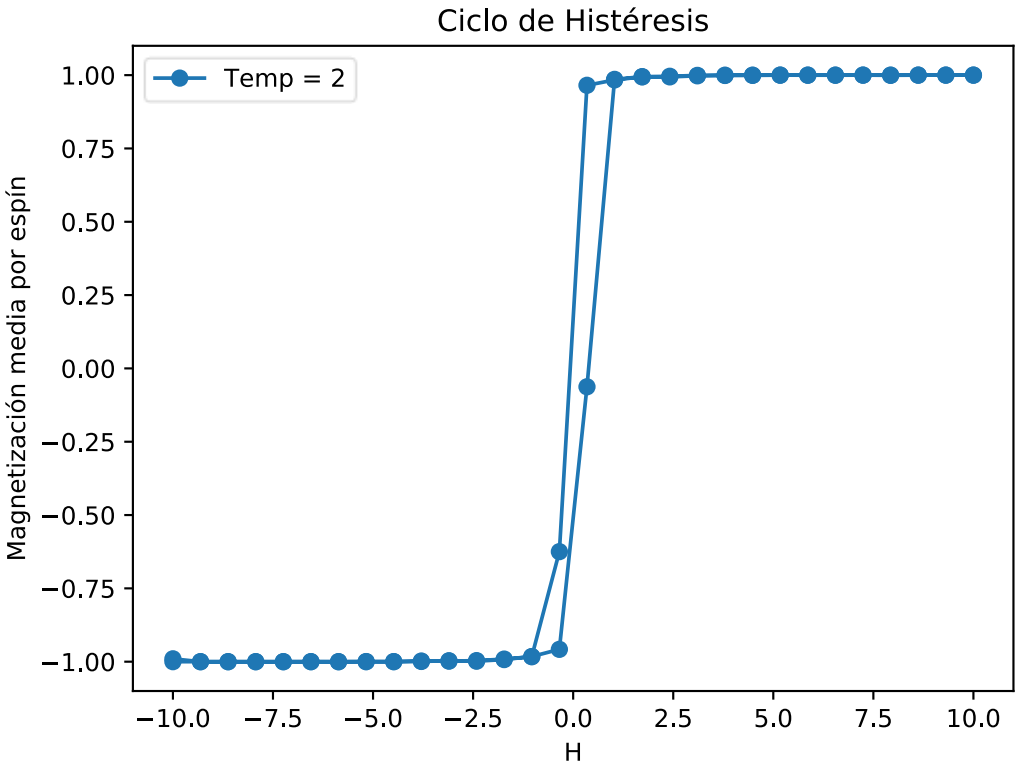
Resultado IV: el ciclo de histéresis

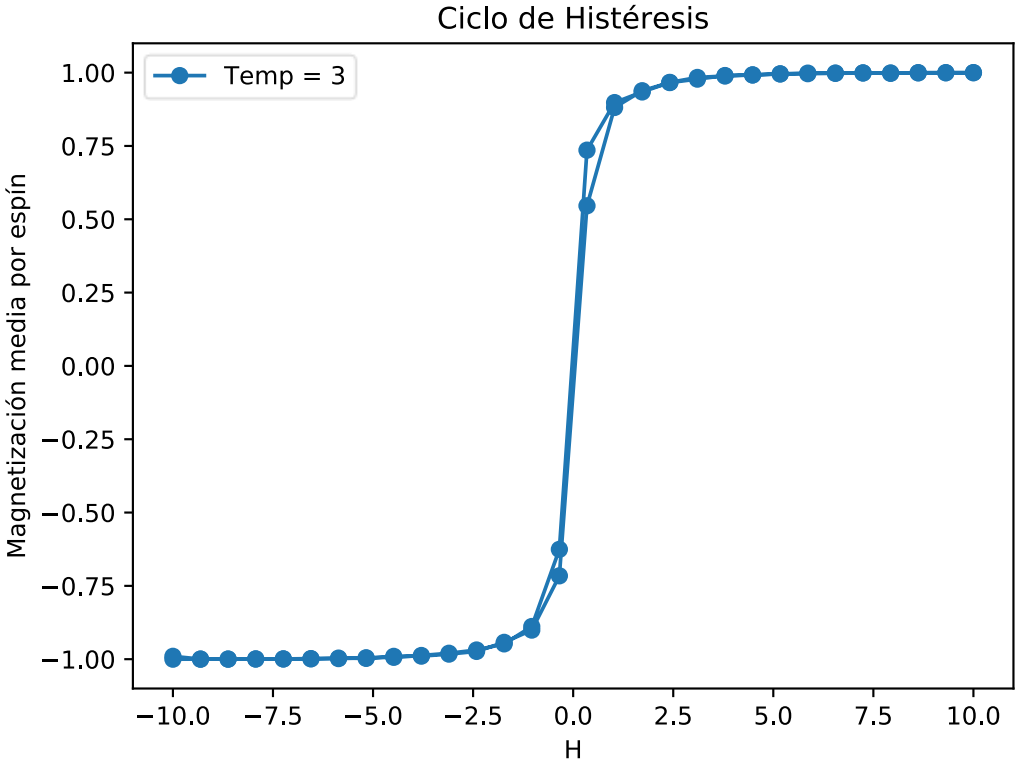
En este último apartado lo que se hace es meter un campo magnético distinto de cero H. A medida que variamos el campo magnético la magnetización varía. Representando los resultados se obtiene lo que se conoce como ciclo de histéresis: la magnetización depende de la historia del objeto.

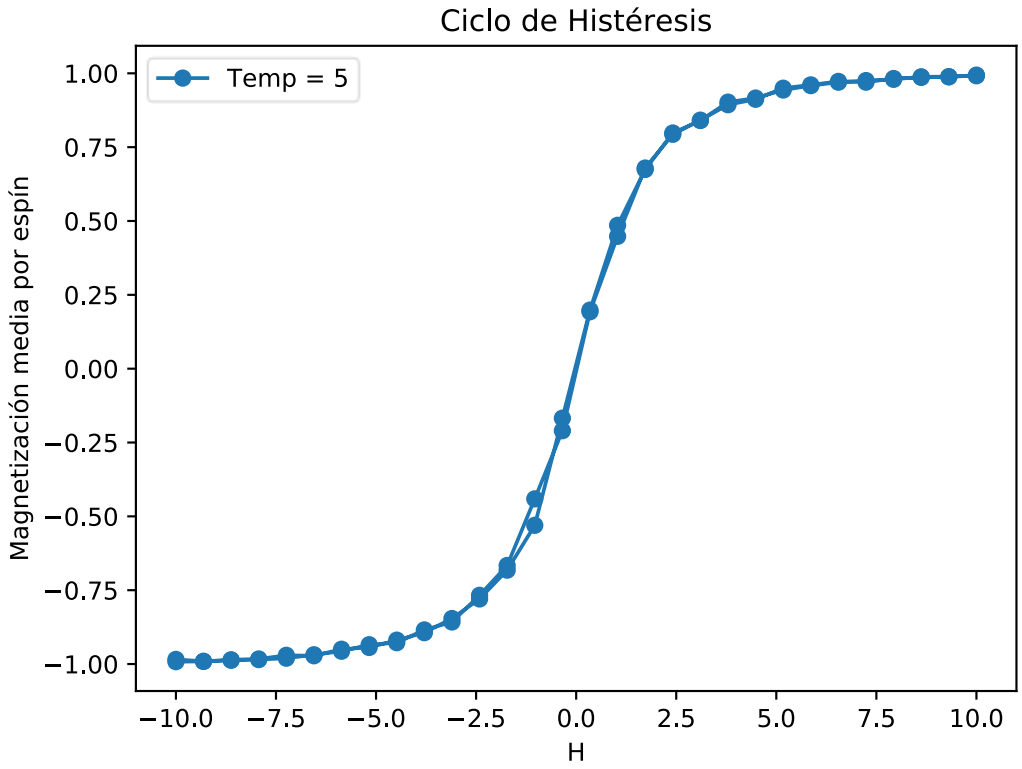
Represento el resultado para distintos valores de la temperatura $T = 0.25,\ 2,\ 3\ y\ 5$


Ciclo de Histéresis
Temp = 0.25
Magnetización media por espín
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0
H


Ciclo de Histéresis
Temp = 1
Magnetización media por espín
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0
H


Ciclo de Histéresis
Temp = 2
Magnetización media por espín
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0
H


Ciclo de Histéresis
Temp = 3
Magnetización media por espín
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0
H



Como se puede ver, el ciclo de histéresis tiene más relevancia a temperaturas bajas en las que se ve cómo la magnetización depende de la historia del material, ya que pasa de -1 a $+1$ cuando $H = 2.5$ y a la vuelta cae de $+1$ a -1 en $H = -2.5$

Extra: interacción con segundos vecinos

En este apartado consideramos el caso en el que la interacción también sea a segundos vecinos. La forma de implementar esto es tan solo cambiar las definiciones de la energía:

```python
# Función para calcular la energía total
def EnergiaTotal():
    energy = 0
    # Paso 1
    # i!=Nm1 and j!=Nm1
    for i in range(Nm1):
        for j in range(Nm1):
            energy += espin[i,j] * ( espin[i,j+1] + espin[i+1,j] + JJ * (espin[i+1,j+1] + espin[i+1,j-1]) )
    # Paso 2
    # i=!Nm1 and j=Nm1
    for i in range(Nm1):
        energy += espin[i,Nm1] * ( espin[i,0] + espin[i+1,Nm1]  +  JJ * (espin[i+1,0] + espin[i+1,Nm1-1]) )
    # Paso 3
    # i=Nm1 and j!=Nm1
    for j in range(Nm1):
        energy += espin[Nm1,j] * ( espin[Nm1,j+1] + espin[0,j] +  JJ * (espin[0,j+1] + espin[0,j-1]) )
    # Paso 4
    # j=Nm1 and i=Nm1
    energy += espin[Nm1,Nm1] * ( espin[Nm1,0] + espin[0,Nm1] +  JJ * (espin[0,0] + espin[0,Nm1-1]) )
    return -energy
```

Y también la variación de energía:

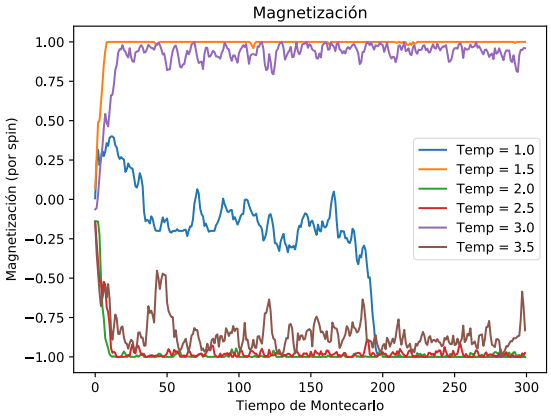
```python
# Función para calcular la variación de energía al cambiar un spin
def DEnergiaSpin(i,j):      # con CC
    # Posibilidad 1: que esté en la zona 1
    # i!=Nm1 and j!=Nm1
    if (i != Nm1 and j != Nm1):
        Eprimeros = ( - espin[i,j]) * ( espin[i,j+1] + espin[i+1,j] + espin[i,j-1] + espin[i-1,j] )
        Esegundos = JJ* ( - espin[i,j]) * ( espin[i+1,j+1] + espin[i+1,j-1] + espin[i-1,j+1] + espin[i-1,j-1] )
        return (-2) * (Eprimeros + Esegundos)
    # Posibilidad 2: que esté en la zona 2
    # i!=Nm1 and j=Nm1
    elif (i != Nm1):
        Eprimeros = ( - espin[i,Nm1]) * ( espin[i,0] + espin[i+1,Nm1] + espin[i,Nm1-1] + espin[i-1,Nm1] )
        Esegundos = JJ* ( - espin[i,Nm1]) * ( espin[i+1,0] + espin[i+1,Nm1-1] + espin[i-1,0] + espin[i-1,Nm1-1] )
        return (-2) * (Eprimeros + Esegundos)
    # Posibilidad 3: que esté en la zona 3
    # i=Nm1 and j!=Nm1
    elif (j != Nm1):
        Eprimeros = ( - espin[Nm1,j]) * ( espin[Nm1,j+1] + espin[0,j] + espin[Nm1,j-1] + espin[Nm1-1,j] )
        Esegundos = JJ* ( - espin[Nm1,j]) * ( espin[0,j+1] + espin[0,j-1] + espin[Nm1-1,j+1] + espin[Nm1-1,j-1] )
        return (-2) * (Eprimeros + Esegundos)
    # Posibilidad 4: que esté en la zona 4
    # j=Nm1 and i=Nm1
    Eprimeros = ( - espin[Nm1,Nm1]) * ( espin[Nm1,0] + espin[0,Nm1] + espin[Nm1-1,Nm1] + espin[Nm1,Nm1-1] )
    Esegundos = JJ* ( - espin[Nm1,Nm1]) * ( espin[0,0] + espin[0,Nm1-1] + espin[Nm1-1,0] + espin[Nm1-1,Nm1-1] )
    return (-2) * (Eprimeros + Esegundos)
```

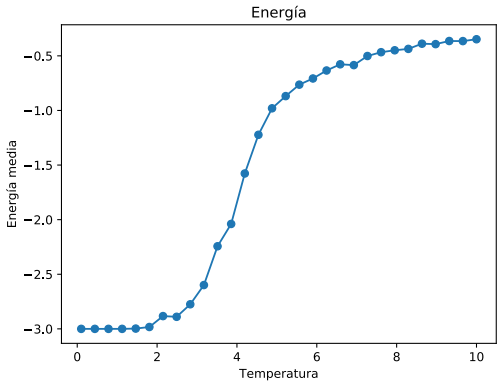
He introducido el parámetro JJ como la intensidad con la que interaccionan a segundos vecinos.

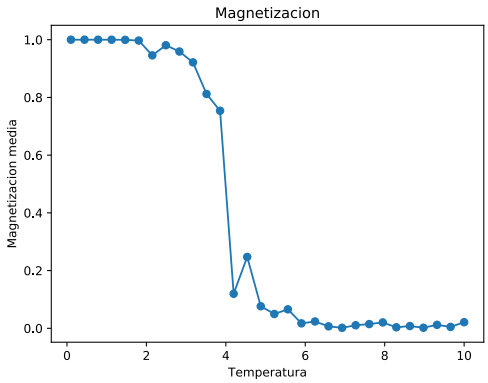
Los resultados los muestra a continuación:

- Caso 1: interacción atractiva

Para el caso en que la interacción sea con $JJ = 0.5$, el comportamiento es análogo al caso que estábamos viendo: los espines tienden a alinearse. Pero sin embargo se notan ciertas diferencias:







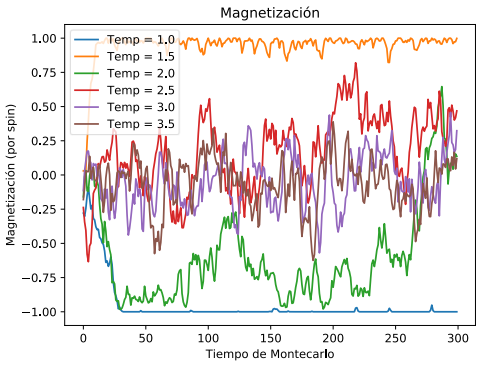
Se puede ver que se comporta como si la temperatura crítica hubiese aumentado a $T_{crítica} \approx 4.0$. Es decir, la interacción atractiva entre segundos vecinos influye también en la temperatura crítica aumentándola.

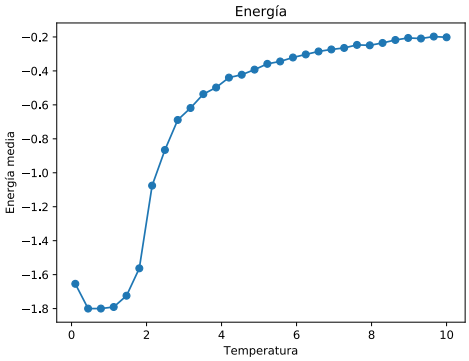
- ## Caso 2: interacción repulsiva

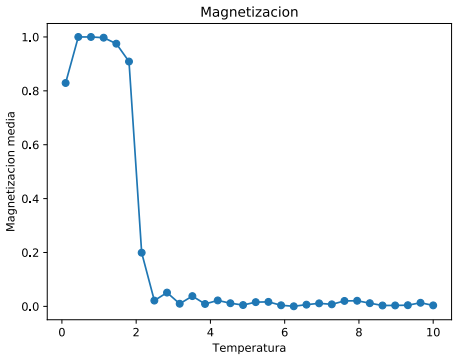
Sin embargo, podemos también considerar el hecho de que la interacción sea repulsiva. Esto es posible que ocurra en algunos sistemas, como por ejemplo en los sólidos en los que los electrones se mueven en un potencial en el que la interacción entre ellos es atractiva (superconductores)

El resultado es el que se muestra a continuación:

Si $JJ = -0.1$

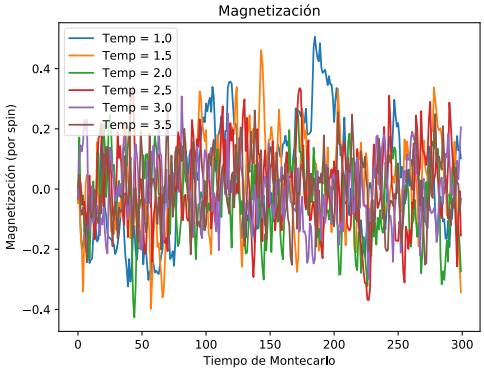


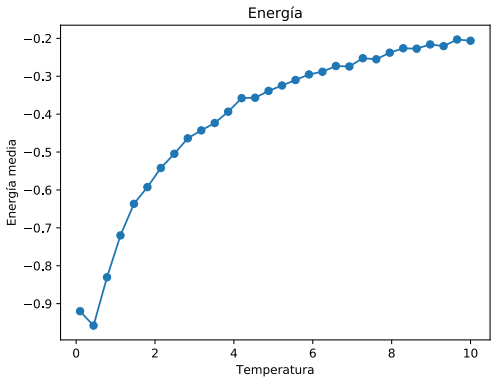


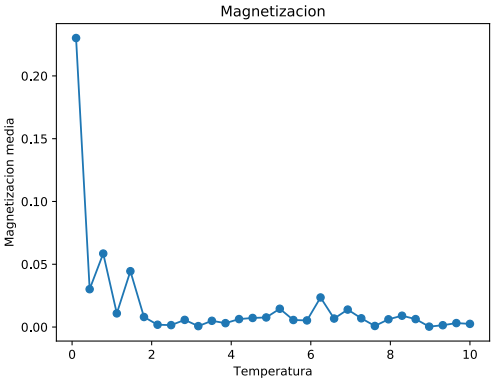


Se ve cómo la temperatura crítica disminuye hasta $T_{crítica} = 2.0$

Si $JJ = -0.5$







Lo que ocurre es que la temperatura crítica disminuyes tanto que $T_{crítica} = 0$ de modo que la magnetización es prácticamente nula $< M > = 0$

Conclusiones

En esta práctica he realizado el algoritmo para estudiar el modelo de Ising. Para ello he optimizado al máximo el código, obteniendo resultados excelentes.

He representado la relación de la magnetización media y la energía media en función de la temperatura obteniendo los resultados típicos del modelo de Ising. A su vez he calculado el calor específico, un cálculo más pesado aún pero que he obtenido satisfactoriamente. Por último, he realizado los ciclos de histéresis para diferentes valores del campo magnético H

Y ya fuera de los límites de esta práctica he querido investigar qué ocurriría al considerar la interacción a segundos vecinos. Los resultados obtenidos son que si esta interacción es atractiva (igual que en el caso de interacción a primeros vecinos) entonces lo que ocurre es que $T_{crítica}$ aumenta, mientras que sin consideramos una atracción repulsiva $T_{crítica}$ disminuye.