



Informe Método de Montecarlo

INFORMÁTICA III

SEBASTIÁN POSADA POSADA

COD.: 1015541

PROFESOR:
ABILO ANDRÉS
VELASQUEZ



ING. FÍSICA

ABSTRACT

Se presentará la simulación del método de un sistema magnético Montecarlo, el cual nos mostrará un sistema de determinado número de partículas, de espines y direcciones de los espines en determinados casos los cuales se irán definiendo a lo largo de la simulación. Estas partículas se encontrarán en un sistema de 2 dimensiones y posteriormente de 3 dimensiones, dentro de una caja cúbica con espines en unas posiciones aleatorias y luego determinadas mediante el método de metrópolis de montecarlo. Por último, mediante un número determinado de pasos, se mostrará cómo se estabiliza el sistema en cada uno de los casos.

Palabras Clave: *magnetismo – espín – Metrópolis – Ising – aleatoriedad – estabilización – Simulación-Física – Partícula-Posición*

1. Introducción

El objetivo del presente informe es ilustrar la simulación de un material ferromagnético por el método Montecarlo, simulando cajas cúbicas con espines en posiciones iniciales aleatorias, luego determinadas debido a ciertas condiciones.

Observaremos el comportamiento del sistema simulado cuando apliquemos diferentes métodos como por ejemplo Ising, metrópolis de Montecarlo. Podremos evidenciar a lo largo de la simulación cual es la evolución magnética dentro de un material.

En este informe mostraré paso por paso como se puede ir entendiendo dichos métodos para hallar magnetización de los espines desde 2 dimensiones usando método de ising, montecarlo, entre otros, hasta poder llegar a evidenciarlo en 3 dimensiones como sería en un caso que se aproxime más a la realidad.

2. Procedimiento Metodológico y/o Experimental

Dentro del código Empiezo llamando algunas librerías y definiendo ciertos parámetros como:

- Tamaño del sistema $L \times L$
- Constante de intercambio
- Constante de Boltzmann

En principio podríamos ver un arreglo bidimensional que se vaya ajustando poco a poco al modelo que queremos llegar.

`sites = numpy.random.choice([-1, 1], size=(L, L))`

el cual me arroja una matriz de na organización aleatoria de números -1 y 1 de un tamaño $L \times L$

```

sites
array([[ -1,  -1,  1,  1,  1,  1,  1, -1, -1, -1],
       [-1,  1,  1,  1,  1, -1,  1,  1, -1, -1],
       [-1,  1, -1, -1,  1, -1,  1, -1, -1,  1],
       [ 1,  1,  1,  1,  1, -1, -1,  1, -1,  1],
       [ 1,  1, -1, -1,  1, -1, -1,  1, -1, -1],
       [-1, -1, -1,  1, -1, -1, -1, -1,  1,  1],
       [ 1, -1,  1,  1, -1, -1, -1,  1,  1,  1],
       [ 1, -1,  1,  1,  1,  1, -1,  1, -1, -1],
       [-1, -1, -1, -1, -1,  1, -1, -1, -1, -1],
       [ 1,  1,  1, -1,  1,  1, -1,  1,  1, -1]])

```

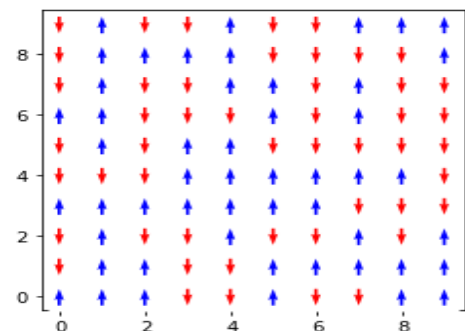
Los cuales se van a traducir luego en el sistema como posiciones de los spines (recordemos que en este caso únicamente tendré 2 opciones: espín arriba, o espín abajo)

Posteriormente defino las siguientes funciones

- `plot_state()`: grafica el estado
- `magnetization()`: calcula la magnetización del estado
- `local_energy(x, y)`: calcula la energía de un solo sitio (x, y)

- `total_energy()`: calcula la energía de todo el estado
- `metropolis(x, y, T)`: realiza el algoritmo de Metrópolis en un sitio (x, y) dada una temperatura T
- `monte_carlo_step(T)`: realiza el algoritmo de Metrópolis $L \times L$ veces, escogiendo un sitio aleatorio cada vez

se simula gráficamente el [resultado de la matriz aleatoria](#)



Prosigo con la función de la magnetización del estado

La cual es la sumatoria de las posiciones de todos los espines del sistema

$$M = \sum_i S_i$$

Luego sigo con la función de la energía local haciendo uso del modelo de ising para este caso del magnetismo, pero con referencia a sus vecinos. En este caso, el hamiltoniano de este modelo de Ising consta únicamente del término de intercambio

$$\mathcal{H} = -J \sum_{i,j} S_i S_j$$

donde J es la constante de intercambio y S_i y S_j son los espines en la posición i y j , respectivamente. Dichos espines pueden tomar únicamente dos posibles configuraciones: \uparrow o \downarrow .

Entonces para continuar hallando la energía local, hago uso de los vecinos de cada espín, los cuales son:

- Arriba (0, 1)
- Abajo (0, -1)
- Derecha (1, 0)
- Izquierda (-1, 0)


me interesa calcular la energía total del sistema para posteriormente meterme con el método de metrópolis, el cual me ayuda a minimizar la energía del sistema que estoy simulando, de este modo voy a poder llegar al estado de mínima energía que tengo en este momento. Entonces para poder llevar a cabo este modelo, tuve que seguir los siguientes pasos que nombraré a continuación.

+ Escogí aleatoriamente un espín
 + Calculé la energía local
 + generaré un nuevo espín en el sistema
 + calculé la energía resultante local nueva
 + calculé el cambio que obtuve en la energía
 + si el cambio me da menor que cero se acepta el nuevo espín
 + sino, se genera un nuevo número aleatorio
 + si este nuevo número aleatorio es menor o igual a $1/e^{(\text{cambio de energía}/KbT)}$ se acepta el espín; de lo contrario se rechaza

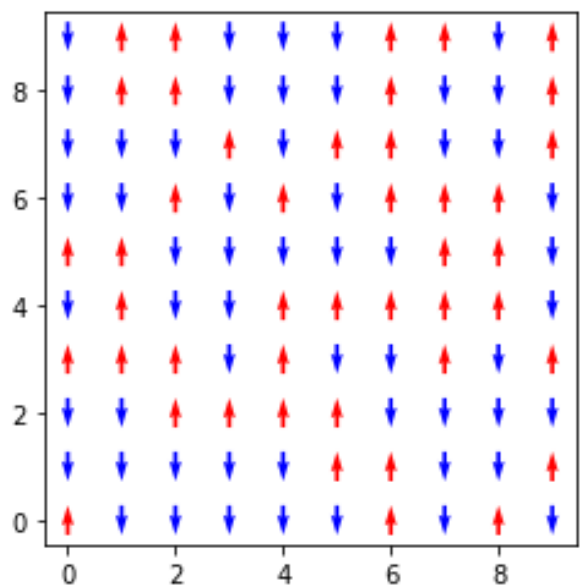
Para de este modo poder llegar hasta el método de montecarlo, el cual me va a ayudar a solucionar el modelo mediante números aleatorios.

Es decir, consiste en realizar N veces el Algoritmo de Metrópolis, donde N es el número total de espines.

Simularé un Sistema a temperatura constante; para ello necesito una nueva matriz con valores aleatorios

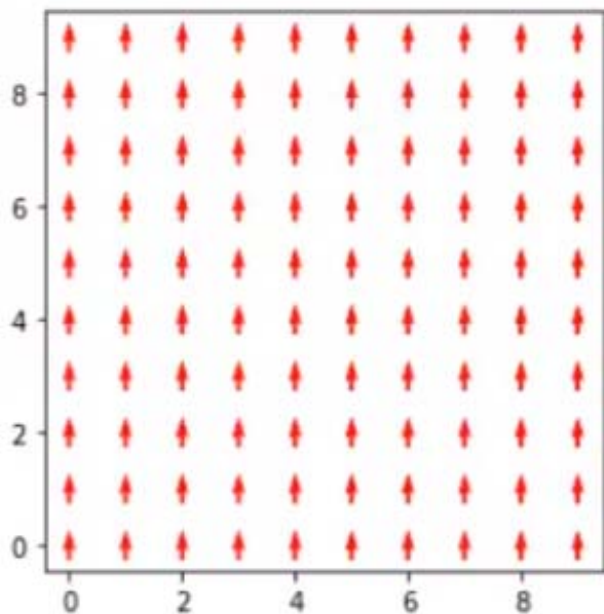
 sites

```
array([[ 1,  1, -1,  1,  1, -1, -1, -1,  1,  1],
       [ 1,  1,  1,  1, -1, -1, -1,  1,  1, -1],
       [-1, -1,  1, -1,  1, -1, -1,  1, -1,  1],
       [ 1, -1,  1, -1,  1, -1, -1,  1, -1, -1],
       [-1,  1, -1, -1, -1, -1,  1,  1, -1, -1],
       [-1, -1, -1,  1, -1, -1,  1,  1,  1,  1],
       [ 1,  1,  1,  1, -1,  1, -1,  1, -1,  1],
       [ 1,  1,  1, -1,  1,  1, -1, -1, -1,  1],
       [-1, -1,  1,  1,  1,  1,  1, -1, -1, -1],
       [ 1, -1, -1,  1,  1, -1, -1, -1,  1, -1]])
```



Puedo decir entonces que este es el estado del que parto o es mi estado inicial del sistema

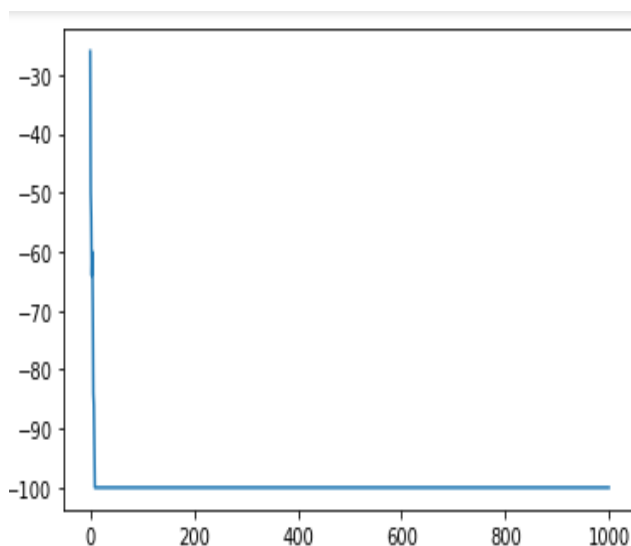
Pero mediante una temperatura constante y unos pasos determinados de montecarlo, me va a resultar lo siguiente



Acá puedo ver que, a una temperatura baja, en el estado de mínima energía los espines apuntan a una misma dirección. Por ejemplo, en este caso utilicé 0.02T

En este caso la magnetización del sistema es máxima y la energía total del sistema es mínima

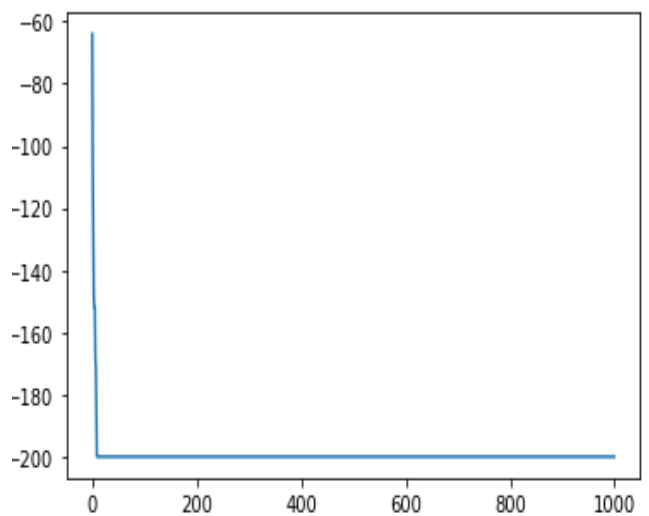
GRÁFICA DE MAGNETIZACIÓN (EJE Y) VS PASOS DE MONTECARLO (EJE X)



Siguiente paso:

Haré la evolución de un sistema con un determinado número de pasos para que el sistema se estabilice, para ello parto de un sistema con una temperatura alta, y posteriormente lo llevaré a una temperatura baja

GRÁFICA DE ENERGÍA (EJE Y) VS PASOS DE MONTECARLO (EJE X)

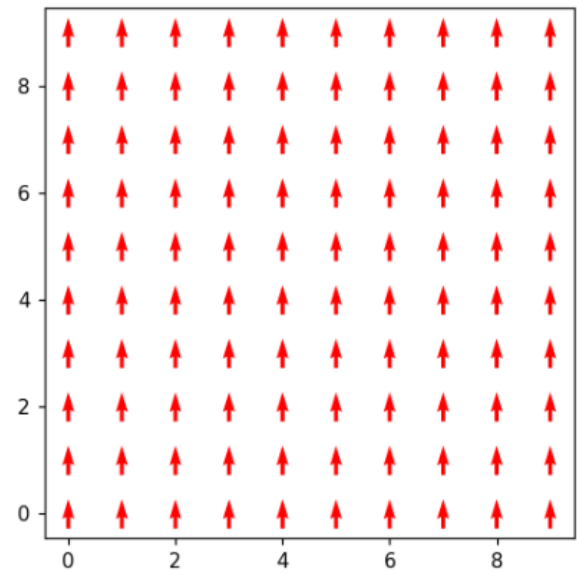


vamos a iniciar con un sistema con espines aleatorios en posición, debido a la alta temperatura, y Mediante unos pasos montecarlo vamos a llevar el sistema a baja temperatura y que de este modo se estabilice

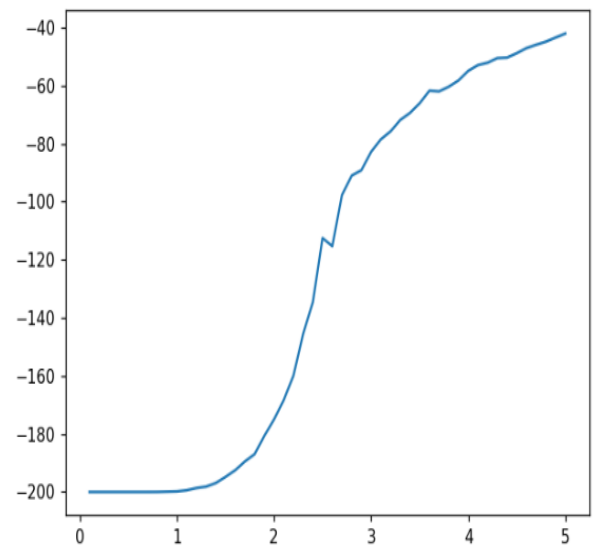
4.9
 4.800000000000001
 4.700000000000001
 4.600000000000001
 4.500000000000002
 4.400000000000002
 4.300000000000025
 4.200000000000003
 4.100000000000003
 4.000000000000036
 3.900000000000004
 3.800000000000043
 3.700000000000046
 3.600000000000005
 3.500000000000053
 3.400000000000057
 3.300000000000006
 3.200000000000064
 3.100000000000068
 3.000000000000007
 2.900000000000075
 2.800000000000008
 2.700000000000008

1.900000000000011
 1.8000000000000114
 1.7000000000000117
 1.600000000000012
 1.5000000000000124
 1.4000000000000128
 1.3000000000000131
 1.2000000000000135
 1.1000000000000139
 1.0000000000000142
 0.9000000000000146
 0.8000000000000149
 0.7000000000000153
 0.6000000000000156
 0.500000000000016
 0.4000000000000163
 0.3000000000000167
 0.2

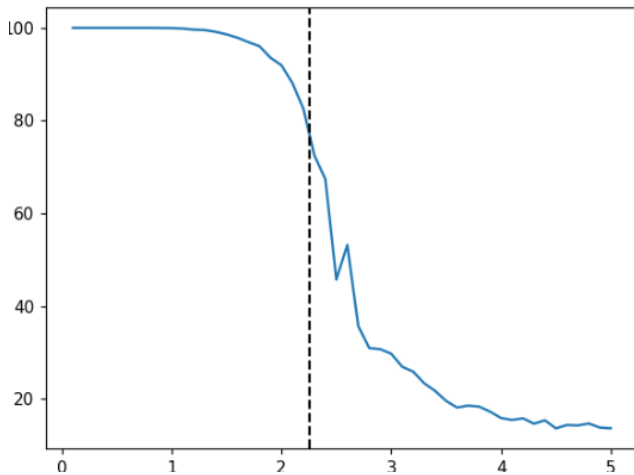
Estado final



GRÁFICA DE ENERGÍA DEL SISTEMA (EJE Y)
 VS TEMPERATURA (EJE X)



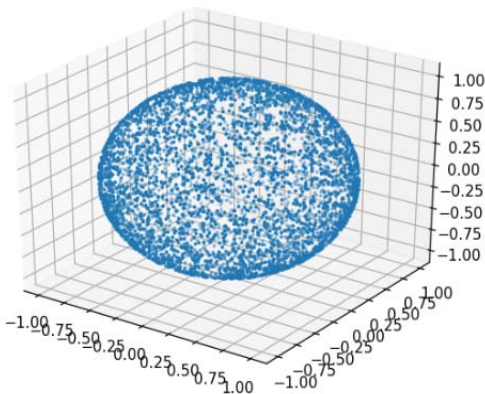
GRÁFICA MAGNETIZACIÓN (EJE Y) VS TEMPERATURA (EJE X)



Ahora ya no tendré espines de -1 y +1, es decir, ya no tendré únicamente espines que apunten hacia abajo y hacia arriba, sino que tendré espines que me apunten en cualquier dirección de una esfera unitaria

Empiezo generando vectores aleatorios en dirección infinitas dentro del espacio tridimensional

En este caso utilizando el modelo de Heidelberg, el cual me permite estudiar la interacción entre diferentes espines, y en este caso lo puedo analizar en 3 dimensiones, teniendo en cuenta que los espines, en este caso me puedan apuntar hacia cualquier dirección del espacio tridimensional



Acá puedo ver que tengo una serie de valores aleatorios uniformemente distribuidos dentro de una esfera.

El nuevo Hamiltoniano para este tipo de simulaciones, es muy parecido al que usé para el caso anterior (de 2 dimensiones), pero ahora tendré vectores unitarios operados por producto punto, de la siguiente forma, (este es el de 3 dimensiones)

$$\mathcal{H} = - \sum_{i,j} J_{i,j} \vec{S}_i \cdot \vec{S}_j + \mathcal{H}_{anis} + \mathcal{H}_{field} + \dots$$

Además, vemos que al hamiltoniano en este caso se le puede sumar algunos otros términos por ejemplo la interacción de intercambio, la interacción de la isotropía y la interacción del campo, entre otros.

En este paso hago una lista con las posiciones atómicas, (mostraré algunas)

```
((0, 0, 0), (0, 0, 1), (0, 0, 2), (0, 0, 3), (0, 0, 4), (0, 1, 0),
(0, 1, 1), (0, 1, 2), (0, 1, 3), (0, 1, 4), (0, 2, 0), (0, 2, 1), (0, 2, 2), (0, 2, 3),
(0, 2, 4), (0, 3, 0), (0, 3, 1), (0, 3, 2), (0, 3, 3), (0, 3, 4), (0, 4, 0), (0, 4, 1),
(0, 4, 2), (0, 4, 3), (0, 4, 4), (1, 0, 0), (1, 0, 1), (1, 0, 2), (1, 0, 3), (1, 0, 4),
(1, 1, 0), (1, 1, 1), (1, 1, 2), (1, 1, 3), (1, 1, 4), (1, 2, 0), (1, 2, 1), (1, 2, 2),
(1, 2, 3), (1, 2, 4), (1, 3, 0), (1, 3, 1), (1, 3, 2), (1, 3, 3), (1, 3, 4), (1, 4, 0),
(1, 4, 1), (1, 4, 2), (1, 4, 3), (1, 4, 4), (2, 0, 0), (2, 0, 1), (2, 0, 2), (2, 0, 3),
(2, 0, 4), (2, 1, 0), (2, 1, 1), (2, 1, 2), (2, 1, 3), (2, 1, 4), (2, 2, 0), (2, 2, 1),
(2, 2, 2), (2, 2, 3), (2, 2, 4), (2, 3, 0), (2, 3, 1), (2, 3, 2), (2, 3, 3), (2, 3, 4),
(2, 4, 0), (2, 4, 1), (2, 4, 2), (2, 4, 3), (2, 4, 4), (3, 0, 0), (3, 0, 1), (3, 0, 2),
(3, 0, 3), (3, 0, 4), (3, 1, 0), (3, 1, 1), (3, 1, 2), (3, 1, 3), (3, 1, 4), (3, 2, 0),
(3, 2, 1), (3, 2, 2), (3, 2, 3), (3, 2, 4), (3, 3, 0), (3, 3, 1), (3, 3, 2), (3, 3, 3),
(3, 3, 4), (3, 4, 0), (3, 4, 1), (3, 4, 2), (3, 4, 3), (3, 4, 4), (4, 0, 0), (4, 0, 1),
```

Posteriormente debo crear vectores unitarios tridimensionales aleatorios para ciertas posiciones

```
((0, 0, 0): array([ 0.92027403, -0.39019778, 0.02900681]),
(0, 0, 1): array([-0.81315669, 0.57572094, -0.08556634]),
(0, 0, 2): array([ 0.23576709, -0.28923613, -0.92776955]),
(0, 0, 3): array([-0.67383041, -0.71690912, -0.17886836]),
(0, 0, 4): array([-0.21700546, -0.32306241, 0.92116194]),
(0, 1, 0): array([0.39422615, 0.90436932, 0.16340707]),
(0, 1, 1): array([ 0.70597257, 0.6838815, -0.18414347]),
(0, 1, 2): array([ 0.12958131, -0.92100346, -0.36737081]),
(0, 1, 3): array([-0.11264188, 0.97488729, -0.19211088]),
(0, 1, 4): array([-0.62009525, -0.14862327, 0.77032007]),
(0, 2, 0): array([-0.0857457, 0.04249646, 0.99541033]),
(0, 2, 1): array([ 0.37098132, -0.5644616, 0.7373981]),
(0, 2, 2): array([ 0.52538305, 0.30036849, -0.79608506]),
(0, 2, 3): array([ 0.13679685, 0.55816339, -0.81837659]),
(0, 2, 4): array([-0.51012142, 0.67134425, 0.53765512]),
(0, 3, 0): array([ 0.00322897, 0.72634804, -0.6873195]),
(0, 3, 1): array([-0.93666447, -0.29326167, -0.19146086]),
(0, 3, 2): array([-0.23628051, 0.32138792, -0.91699582]),
(0, 3, 3): array([0.78620581, 0.6073904, 0.11383026]),
(0, 3, 4): array([ 0.52160624, -0.54033869, -0.66027345]),
(0, 4, 0): array([-0.69827384, -0.45114531, -0.55577113]),
(0, 4, 1): array([-0.07011338, -0.07835475, 0.99445696]),
(0, 4, 2): array([ 0.9315192, -0.33231263, -0.14778461]),
(0, 4, 3): array([-0.40888029, 0.6853801, -0.60255375]),
(0, 4, 4): array([-0.18325337, -0.59131665, 0.78534248]),
(1, 0, 0): array([-0.19224505, -0.35158555, -0.91620382]),
```

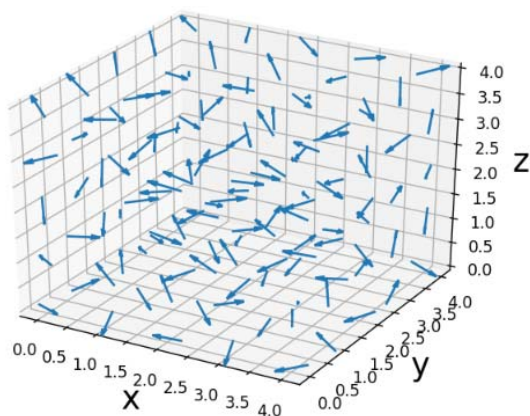

Esto es similar a la matriz de ising de -1 y de +1, pero en este caso ya tenemos infinitos valores que me señalan a cualquier parte del espacio en 3 dimensiones

infinidad de direcciones dentro de las 3 dimensiones y que mediante el método utilizado (usando pivote) podemos verlos distribuidos dentro de la caja cúbica uniformemente.

Definición de funciones

- `plot_state()`: grafica el estado
- `magnetization()`: calcula la magnetización del estado
- `get_exchange_energy(site)`: calcula la energía de intercambio de un sitio (x, y, z)
- `get_anisotropy_energy(site)`: calcula la energía de anisotropía de un sitio (x, y, z)
- `get_zeeman_energy(site, H)`: calcula la energía de zeeman de un sitio (x, y, z) debido a un campo magnético externo H
- `local_energy(site, H)`: calcula la energía de un solo sitio (x, y, z) con un campo externo H
- `total_energy(H)`: calcula la energía de todo el estado con un campo externo H
- `metropolis(site, T, H)`: realiza el algoritmo de Metrópolis en un sitio (x, y, z) dada una temperatura T y un campo magnético externo H
- `monte_carlo_step(T, H)`: realiza el algoritmo de Metrópolis L x L x L veces, escogiendo un sitio aleatorio cada vez
-

empiezo con el `plot state`



El cual me grafica los vectores aleatorios en posición como lo había mencionado anteriormente. Notemos que los vectores pueden señalar en

#

Prosigo con Asignación de vecinos

Los vecinos serán almacenados en un diccionario donde tendremos la posición del átomo, y una lista de las posiciones de sus vecinos.

```
[(0, 0, 1),
(0, 0, -1),
(0, 1, 0),
(0, -1, 0),
(1, 0, 0),
(-1, 0, 0)]:
```

En este caso estamos trabajando con una estructura cristalina cúbica simple, y podemos decir que, de este modo aseguramos que cada sitio va a tener exactamente 6 vecinos.

Y continúo definiendo las otras funciones que propuse, como, por ejemplo:

* Energía local, pero en este caso es la energía local de un sitio

* Energía total del sistema

* Desarrollo el método de metrópolis, en este caso relacionado el sitio con la temperatura como ya lo veremos a continuación.

* Pasos para desarrollar el método de metrópolis en este caso

+ Escogí aleatoriamente un espín

+ Calculé la energía local

+ generaré un nuevo espín en el sistema

+ calculé la energía resultante local nueva

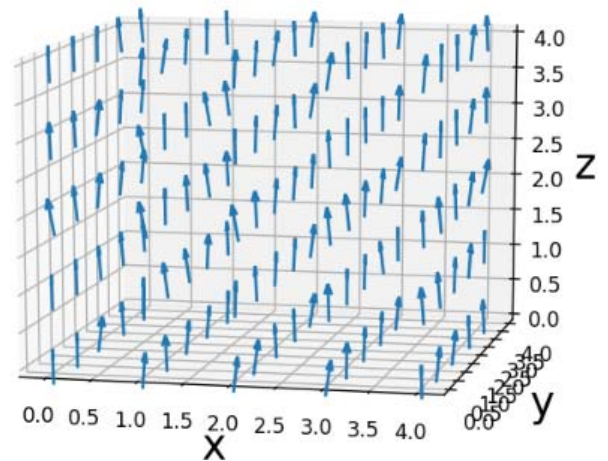
+ calculé el cambio que obtuve en la energía

+ si el cambio me da menor que cero se acepta el nuevo espín

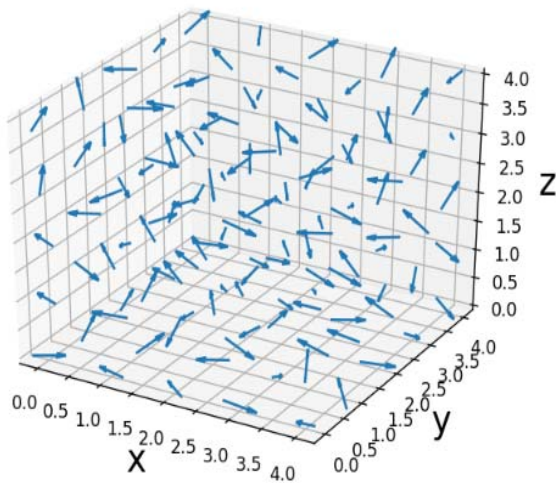
+ sino, se genera un nuevo número aleatorio

+ si este nuevo número aleatorio es menor o igual a $1/e^{(\text{cambio de energía}/k_B T)}$ se acepta el espín; de lo contrario se rechaza

DESARROLLO LA SIMULACIÓN DEL SISTEMA A UNA TEMPERATURA CONSTANTE

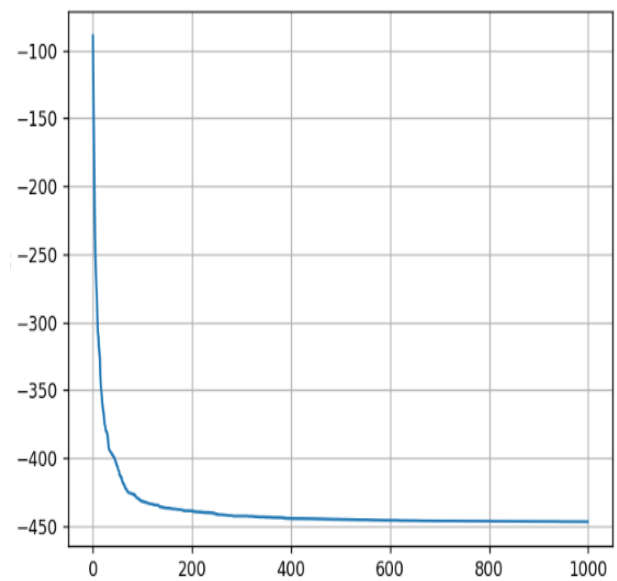


Estado inicial del sistema

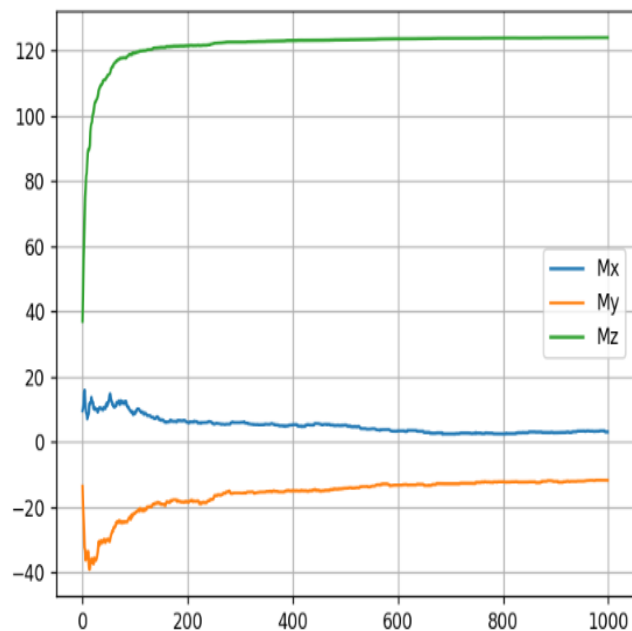


Desde una temperatura inicial alta, llevando el sistema a una temperatura final muy baja, mediante unos pasos determinados de montecarlo, vamos a obtener lo siguiente:

GRÁFICA ENERGÍA (EJE Y) VS PASOS DE MONTECARLO (EJE X)

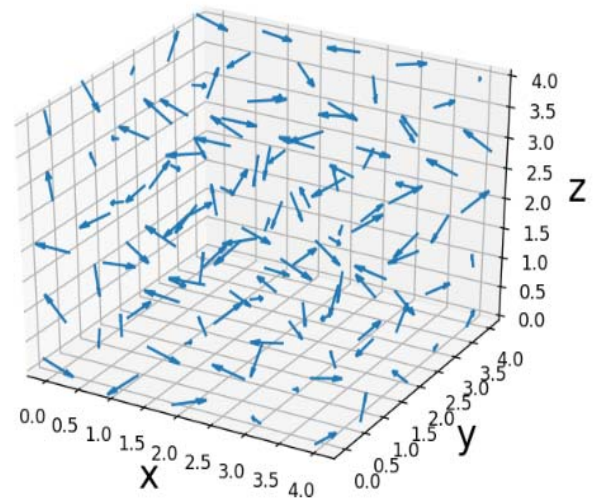


GRÁFICA PARA LA MAGNETIZACIÓN X, MAGNETIZACIÓN Y, Y MAGNETIZACIÓN Z

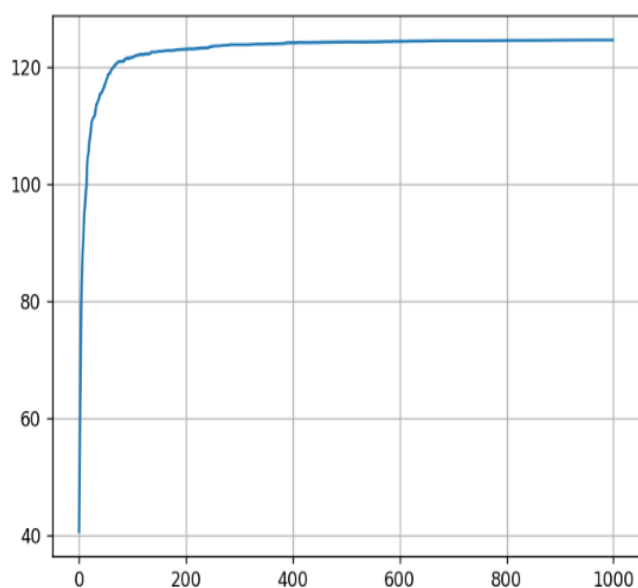


CONTINUÓ CON EL CICLO DE TEMPERATURA

ESTADO INICIAL

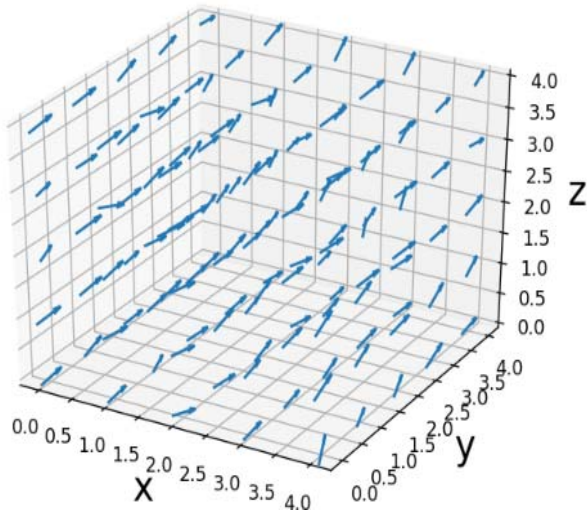


GRÁFICA DE LA MAGNETIZACIÓN ABSOLUTA (EJE Y) VS PASOS DE MONTECARLO (EJE X)



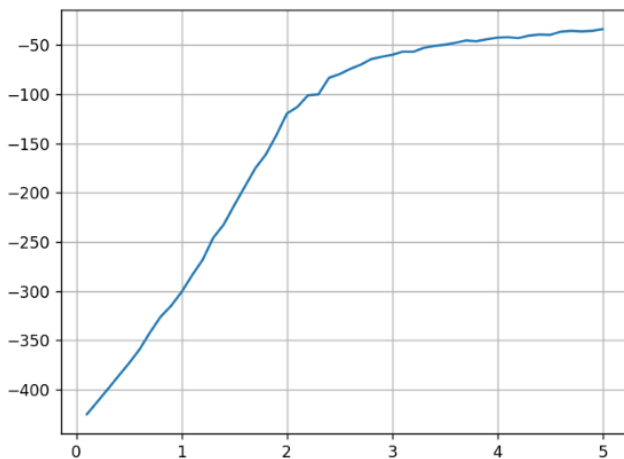
Ti = 5.0 # Temperatura inicial (alta)
Tf = 0.01 # Temperatura final (baja)
dT = -0.1 # Paso de temperatura

Desde una temperatura inicial alta, como se evidencia en la gráfica anterior, apreciamos los espines aleatorios en su posición, y mediante unos métodos de montecarlo, llegando a una temperatura bajita, podemos apreciar lo siguiente:

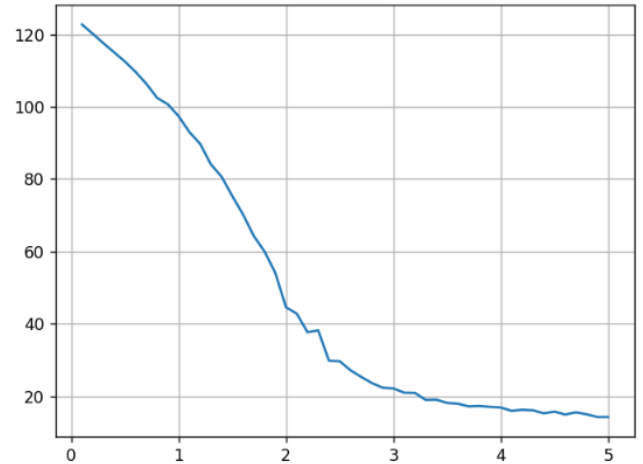


Los espines se alinean, prácticamente homogéneamente.

GRÁFICA ENERGÍA DEL SISTEMA (EJE Y) VS TEMPERATURA DEL SISTEMA (EJE X)



GRÁFICA MAGNETIZACIÓN DEL SISTEMA (EJE Y) VS TEMPERATURA DEL SISTEMA (EJE X)



CONCLUSIONES:

- Podemos ver varias analogías entre el modelo de ising y el modelo de metrópolis para heisenberg, Igual que en el modelo de ising vemos que a una temperatura constante y un número determinado de pasos de montecarlo, el sistema, va a tender a una magnetización homogénea
- La temperatura en el material incide sobre el momento magnético del mismo, haciendo que el material se polarice o que tome cierto valor magnético.
- A temperaturas altas, el material tiende a tener una magnetización total cercana a cero
- Mientras que el material sometido a temperaturas muy bajas puede manipularse más fácilmente para llegar a una magnetización cercana a 100
- Los métodos utilizados en esta simulación los podemos emplear en las aplicaciones de materiales ferromagnéticos, y de este modo entender el comportamiento del momento angular de los mismos, dando lugar a predicciones de fenómenos de este tipo

BIBLIOGRAFÍA

Efecto de la región interracial en el comportamiento magnético de nanoestructuras core/Shell – TESIS DE MAESTRÍA DE Juan David Alzate Cardona

<https://iopscience.iop.org/article/10.1088/0953-8984/26/10/103202>

<https://aip.scitation.org/doi/10.1063/1.5101006>

Hugo, K., De Gree, M., Pessoa, J., & Lucero, P. (2011). Estudio del campo magnético de un solenoide con experimento remoto y simulación. *Revista de Enseñanza de la Física*, 24(2), 55-64.

Rojas, J., & Rojas, C. (2007). Procesos de coalescencia de dos nanopartículas de cobre: Simulación con el método de dinámica molecular.

Illana, J. I. (2013). Métodos monte carlo. *De Departamento de Física Teórica y del Cosmos, Universidad de Granada*, 26(01).