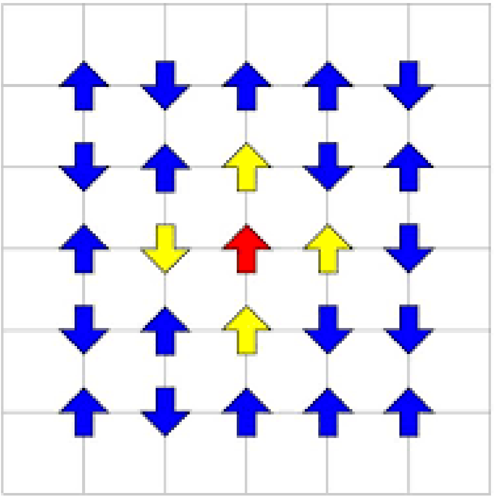
Modelo Ising 2D

Actividad en clase

Física Computacional I 2021-2



Grupos: 1
Horario: M-J de 12:00-2:00 p.m
Aula: 6-303

El sistema que se desea simular es un Ising 2D, y su Hamiltoniano consiste únicamente del término de interacción del espín σ_i con sus primeros vecinos

$$H = -J \sum_{<i,j>} \sigma_i \sigma_j$$

J es la constante de intercambio.
σ_i y σ_j son los espines en la posición *i* y *j*, respectivamente.

Para calcular la magnetización del sistema se emplea:

$$M = \frac{1}{N} \sum_{i}^{N} \sigma_i$$

Y para el cálculo de la susceptibilidad magnética y el calor específico:

$$\chi = \frac{(\langle M^2 \rangle - \langle M \rangle^2)}{k_B T}, \qquad c_v = \frac{(\langle E^2 \rangle - \langle E \rangle^2)}{k_B T^2}$$

Algoritmo de metropolis

El siguiente algoritmo describe el modelo de Ising 2D.

- ❑ Si ejecuta el programa varias veces pero cambiando en cada una de ellas el número total de pasos de Monte Carlo (nsteps), ¿Qué puede notar?

```
import random, math
import matplotlib.pyplot as plt
import numpy as np

def energy (S, N, nbr):
  E = 0.0
  for k in range(N):
    E = S[k] * sum(S[nn] for nn in nbr[k])
  return 0.5 * E

L = 10
N = L * L
nbr = {i : ((i // L) * L + (i + 1) % L, (i + L) % N,
            (i // L) * L + (i - 1) % L, (i - L) % N) \
                                        for i in range(N)}

T = 2
beta = 1.0 / T
S = [np.random.choice([1, -1]) for k in range(N)]
nsteps = N * 100
Energy = energy(S, N, nbr)

E = []
for step in range(nsteps):
  k = np.random.randint(0, N - 1)
  delta_E = 2.0 * S[k] * sum(S[nn] for nn in nbr[k])
  if np.random.uniform(0.0, 1.0) < np.exp(-beta * delta_E):
    S[k] *= -1
    Energy += delta_E
  E.append(Energy)
print('mean energy per spin:', np.sum(E) / float(len(E) * N))
```

- Para la graficación de los datos, copie el código que se muestra al lado y ejecútelo.

- Cambie L = 128 y T = 3. A partir de una configuración inicial aleatoria, obtenga una gráfica para la configuración final.

- Para obtener una mejor visualización de cómo cambian los datos, con un subplot, grafique la configuración inicial y la configuración final del sistema.

- Ahora disminuya el valor de la temperatura, T = 1. Si realiza esta parte con L = 32 y luego con L = 128, ¿Qué observa? Haga los respectivos análisis y comentarios de lo que observa.

```python
import random, math
import matplotlib.pyplot as plt
import numpy as np

def x_y(k, L):
  y = k // L
  x = k - y * L
  return x, y

conf = [[0 for x in range(L)] for y in range(L)]
for k in range(N):
  x, y =x_y(k, L)
  conf[x][y] = S[k]

plt.imshow(conf, extent = [0, L, 0, L], interpolation = 'nearest')
plt.set_cmap('jet')
plt.title('T = %0.2f, L = %d' %(T, L))
plt.show()
```

- Modifique el código para comenzar desde una configuración ordenada, este es el caso de un material **ferromagnético**.

Partiendo desde esta configuración y con L=32, obtenga las siguientes gráficas:

- $\langle E \rangle \ vs \ T$
- $M \ vs \ T$
- $\chi \ vs \ T$
- $c \ vs \ T$

Haga los respectivos análisis.

- Modifique el código para comenzar desde una configuración desordenada, este es el caso de un material **paramagnético**.
- Partiendo desde esta configuración y con L=32, obtenga las misma gráficas obtenidas en el punto anterior. Haga los respectivos análisis.

- Investigue cómo son las expresiones analíticas de la energía, magnetización, susceptibilidad y calor específico. Realice un gráfico de cada una de ellas en función de la temperatura y compárelo con la solución numérica obtenida en los apartes anteriores.