CY Cergy Paris Université

L3 Physique

Projet Numérique : Monte Carlo

Projet final : Modèle d'Ising ferromagnétique sur le réseau de Lieb

---

Élève :

Mizaan-abbas KATCHERA

Enseignants :

Andréas HONECKER

Mohammed ALKHATEEB

Burak CIVITCIOGLU

31 Mars 2023

Table des matières

Logiciel : Python - Spyder **1**

Introduction **2**

Résumé de l'analyse **2**

Résultats finaux et exploitation **3**

Conclusion **5**

Références **5**

Logiciel : Python - Spyder

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-


import math as m
import numpy as np
import matplotlib.pyplot as plt
import random as r
import time


thermalization = 1000   # number of sweeps to equilibrate at each T
num_mc_steps = 10000    # number of Monte Carlo sweeps for each T

cnt_L = 0

# Tc = 2/ln(1+sqrt(2)) = 2.269185314213 ..

Temperatures = [4, 3.5, 3, 2.7, 2.5, 2.4, 2.35, 2.3, 2.29, 2.28, 2.27, 2.26,
                2.25, 2.24, 2.23, 2.22, 2.21, 2.2, 2.1, 2.0, 1.8, 1.5, 1, 0.5]

# Testing purposes:
# Sizes = [5,10,20]
Sizes = [6,12,22]

M=np.zeros((len(Sizes),len(Temperatures)))      # initialize a matrix where we insert the values of the magnetization |M|
M2=np.zeros((len(Sizes),len(Temperatures)))     # initialize a matrix where we insert the values of M^2
M4=np.zeros((len(Sizes),len(Temperatures)))     # initialize a matrix where we insert the values of M^4
En=np.zeros((len(Sizes),len(Temperatures)))     # initialize a matrix where we insert the values of the energy
En2=np.zeros((len(Sizes),len(Temperatures)))    # initialize a matrix where we insert the values of the energy square

UL=np.zeros((len(Sizes),len(Temperatures)))     # initialize a matrix for Binder cumulant UL
C=np.zeros((len(Sizes),len(Temperatures)))      # initialize a matrix fpr specific heat C

# a seep

def balayage(reseau):
    for i in range(L):
        for j in range(L):
            voisins = []
            if (i % 2 == 0 and j % 2 == 0) or (i % 2 == 1 and j % 2 == 1):  # arretes
                voisins = [(i, (j + 1) % L), ((i + 1) % L, j), (i, (j - 1) % L), ((i - 1) % L, j)]
            else:  # milieux des côtés
                voisins = [((i - 1) % L, (j - 1) % L), ((i + 1) % L, (j + 1) % L), ((i - 1) % L, (j + 1) % L), ((i + 1) % L, (j - 1) % L)]
            dE = 2*reseau[i][j]*sum([reseau[x][y] for x, y in voisins])
            if dE < 0 or r.random() < m.exp(-dE/T):
                reseau[i][j] = -reseau[i][j]
    return reseau

# measure magnetization

def aimantation(reseau):
        return(np.sum(reseau))


# measure energy

def energie(reseau):
    E = 0
    for i in range(L):
        for j in range(L):
            if (i % 2 == 0 and j % 2 == 0) or (i % 2 == 1 and j % 2 == 1):  # arretes
                E += -reseau[i][j]*(reseau[(i+1)%L][j]+reseau[i][(j+1)%L])
    return E

# do simulations

for L in Sizes:
    reseau=[[np.sign(-1 + 2 *r.random()) for i in range(L)] for i in range(L)]   # initialize a with random spins
    cnt_T =0
    for T in Temperatures:
        print("L =", L, ", T =", T)

# Thermalize (required for each T)
        print("thermalization")
        for mc_step in range(thermalization):
            balayage(reseau)

# Start Monte Carlo simulation with calculation of M, U, and C
        print("measuring")
        moy, moy2, moy4 = 0, 0, 0
        moyE, moyE2 = 0, 0
        for mc_step in range(num_mc_steps):
            balayage(reseau)
            Mact = aimantation(reseau)/L/L  #L au carré = N
            moy += abs(Mact)
            moy2 += Mact**2
            moy4 += Mact**4
            Eact = energie(reseau)
            moyE += Eact
            moyE2 += Eact**2
    #if mc_step % 10 ==0:
                #plt.imshow(reseau)
                #plt.pause(0.1)
                #plt.clf()
        moy = moy/num_mc_steps
        moy2 = moy2/num_mc_steps
        moy4 = moy4/num_mc_steps
        moyE = moyE/num_mc_steps
        moyE2 = moyE2/num_mc_steps
        print("|M| =", moy, ", E =", moyE)
        M[cnt_L][cnt_T] = moy #pour chaque taille et chaque temp
        M2[cnt_L][cnt_T] = moy2
        M4[cnt_L][cnt_T] = moy4 #cumulant de blander
        En[cnt_L][cnt_T] = moyE
        En2[cnt_L][cnt_T] = moyE2 #pour la chaleur specifqiue
        print("# =======")

        cnt_T += 1

    cnt_L += 1

# magnetization per site

plt.subplot(221)
line1, = plt.plot(Temperatures, M[0], label = "L = " + str(Sizes[0]))
line2, = plt.plot(Temperatures, M[1], label = "L = " + str(Sizes[1]))
line3, = plt.plot(Temperatures, M[2], label = "L = " + str(Sizes[2]))
plt.legend(handles = [line1,line2,line3])
plt.xlabel('T')
plt.ylabel('<|M|>')

# Binder cumulant

for L in range(len(Sizes)):
        for T in range(len(Temperatures)):
                UL[L][T] = 1-M4[L][T]/(3*M2[L][T]**2)
plt.subplot(222)
line1, = plt.plot(Temperatures, UL[0], label = "L = " + str(Sizes[0]))
line2, = plt.plot(Temperatures, UL[1], label = "L = " + str(Sizes[1]))
line3, = plt.plot(Temperatures, UL[2], label = "L = " + str(Sizes[2]))
plt.legend(handles = [line1,line2,line3])
plt.xlabel('T')
plt.ylabel('B')

# Specific heat per site

for L in range(len(Sizes)):
        for T in range(len(Temperatures)):
                C[L][T] = (En2[L][T] - En[L][T]**2)/(Temperatures[T]*Sizes[L])**2
plt.subplot(223)
line1, = plt.plot(Temperatures, C[0], label = "L = " + str(Sizes[0]))
line2, = plt.plot(Temperatures, C[1], label = "L = " + str(Sizes[1]))
line3, = plt.plot(Temperatures, C[2], label = "L = " + str(Sizes[2]))
plt.legend(handles = [line1,line2,line3])
plt.xlabel('T')
plt.ylabel('U_L')
plt.show()
```

FIGURE 1 – Python 3.8.2 : Replit (voir programme en pj)

Introduction

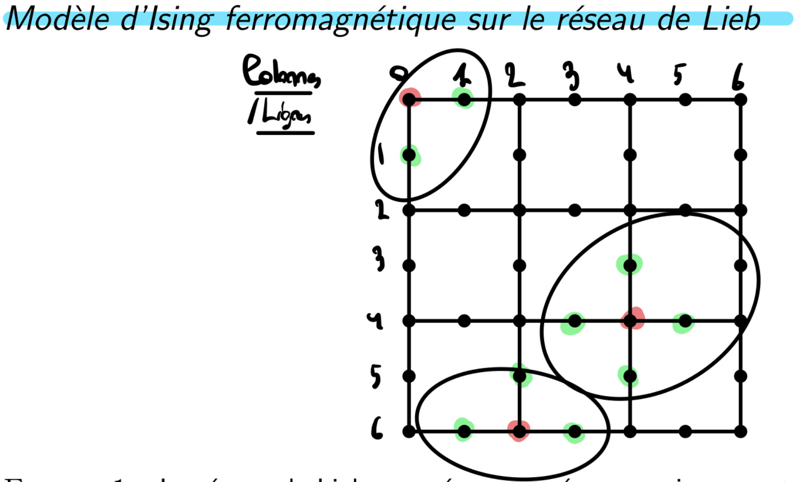

FIGURE 1 – Le réseau de Lieb : un réseau carré appauvri au quart. Les cercles remplis indiquent les sites où se trouvent les spins.

L'objectif de ce projet est d'étudier un modèle d'Ising ferromagnétique (2D) sur le réseau de Lieb qui est appauvri au quart. Dans un premier temps, on implémentera l'algorithme de Métropolis à travers un programme python selon le schéma ci-dessus et dans un second temps on discutera de la transition de phase ainsi que de l'estimation de la température critique (avec la constante de Boltzmann k_B égale à 1 en unité).

Résumé de l'analyse

On reprend la structure du programme du TD5 en implémentant l'algorithme de Metropolis afin de modéliser les spins dans un réseau carré auppauvri au quart en deux dimensions. L'objectif est de simuler le comportement d'un modèle d'Ising ferromagnétique à l'aide d'une réseau carré de taille L x L (3x3 sur le schéma). Les spins peuvent prendre les valeurs +1 ou -1.

La fonction "initialisation" permet d'initialiser le réseau carré en attribuant aléatoirement la valeur de spin aléatoire de +1 ou -1 pour chaque site. La fonction "balayage" nous permet d'implémenter l'algorithme de Metropolis à un nombre suffisant d'itérations que l'on a défini. A chaque itération tous les sites du réseau sont parcourus et pour chaque site, la variation d'énergie est calculée pour une valeur de spin aléatoire. Si la variation d'énergie (deltaE) est négative, le spin est inversé. En outre, si la variation d'énergie est positive alors le spin est toujours inversé et il faut calculer w qui correspond à la probabilité de Bolzmann : $w = \exp\left(-\frac{E_2 - E_1}{k_B T}\right)$. Dans ce cas, il faut tirer un nombre aléatoire r qui suit une distribution uniforme (sur [0, 1]). Au cas où r w on accepte l'inversion du spin. Pour r > w, l'inversion du spin doit être abandonné, c'est-'a-dire que nous devons revenir à l'ancien état pour le spin considéré. Enfin l'algorithme continue d'itérer jusqu'à ce qu'un nombre suffisant de configurations soit crées. Dans notre cas, chaque spin est au contact de deux, trois ou quatre spins voisins. (voir schéma) La fonction "energie" calcule l'énergie totale du réseau de par la somme des interactions entre les spins voisins.

On défint cette intéraction entre deux spins avec -J si les spins sont alignés et +J si les spins sont opposés. Dans notre cas, l'intéraction est ferromagnétique donc J est positif. On utilise afin de calculer l'énergie le Hamiltonien suivant : $H = -J_{ij} \sum_{i,j} s_i s_j$ où si, sj sont les spins et i,j voisins entre eux.

La boucle principale de notre programme initialise le réseau et effectue un certain nombre de balayages pour atteindre un état d'équilibre (thermodynamique). Dès lors, l'énergie et la magnétisation sont calculées pour un nombre spécifié de mesures.

Ainsi, on obtiendra 3 graphiques qui correspondent à l'aimantation moyenne en fonction de la température, le cumulant de Binder en fonction de la température et la chaleur spécifique en fonction de la température. : $M = \frac{1}{N} \sum_{i=1}^{n} s_i$ correspond à l'aimantation où les conditions aux bords s'appliquent. Cependant, il est plus judicieux pour obtenir une estimation précise de la température critique d'introduire le cumulant de Binder : $U_L = 1 - \frac{\langle M^4 \rangle_L}{3\langle M^2 \rangle_L^2}$. Enfin, pour confirmer nos résultats on caractérise la chaleur spécifique $c = \frac{1}{L}^2 \frac{1}{(k_B T)^2} (< E^2 > - < E >^2)$ où E est l'énergie, k_B la constante de Boltzmann (= 1 dans notre projet) et L la taille du réseau.

Résultats finaux et exploitation

Tout d'abord on va prend comme différentes valeurs de température (T) à savoir T = [4, 3.5, 3, 2.7, 2.5, 2.4, 2.35, 2.3, 2.29, 2.28, 2.27, 2.2, 2.25, 2.24, 2.23, 2.22, 2.21, 2.2, 2.1, 2.0, 1.8, 1.5, 1, 0.5]. Ci-dessous le tracé l'aimantation, le cumulat de Binder et la chaleur spécifique en fonction de la température pour des valeurs L = 3, 6, 12.

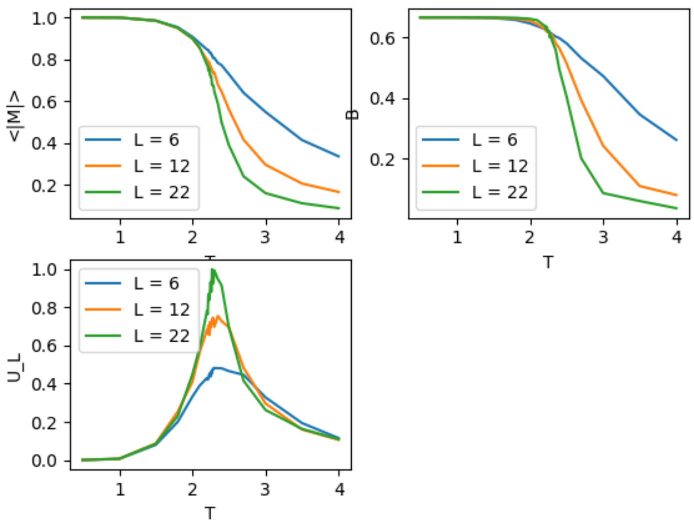

FIGURE 2 – L = 6, 12, 22 - Tracés de U_L, B et <|M|> en fonction de T

Par la suite, on prendra des valeurs de températures et de taille de réseau similaire au TD5 i.e avec L = 22, 50, 70 car plus la taille du réseau est grande plus notre estimation sera précise. On remarque sur ces tracés que l'aimantation, le cumulat de Binder décroit brusquement lorsque Tc/(J) est entre 2 et 3 et que ces derniers tendent vers 0 en dehors de cette intervalle. Concernant la chaleur spécifique celle-ci possède un pic sur ce dernier. De ce fait, afin d'affiner nos résultats on recommence l'opération en ciblant sur ces valeurs de T tout en conservant la même configuration avec T = [3, 2.7, 2.6, 2.5, 2.4, 2.3, 2.2, 2.1, 2.0]. (kB = 1)

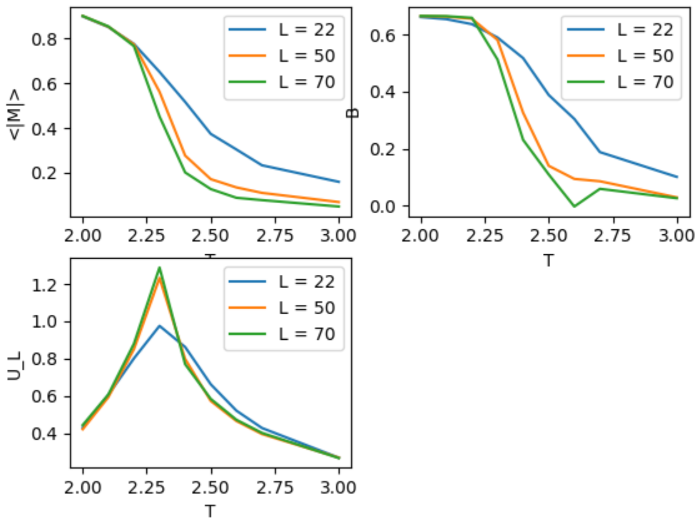

FIGURE 3 – L = 22, 50, 70 - Tracés de U_L, B et <|M|> en fonction de T

Pour plus de précisions il est préférable d'effectuer plusieurs graphiques pour chaque configuration. De plus, le tracé de T en fonction de T-Tc nous permettrai d'obtenir la solution exacte.

Ainsi, on estime de par la même analyse que précédemment que la température critique Tc/(J) se situe autour de 2,25 ce qui coïncide avec la valeur dans le cours pour un réseau carré similaire à notre cas.

On peut donc affirmer que la transition de phase se déroule autour de la température critique ou encore température de Curie. Une telle transition se produit lorsque ce paramètre externe atteint une valeur seuil. La transformation traduit généralement un changement des propriétés de symétrie du système c'est-à-dire que l'on passe d'un état ordonné à l'état désordonné de par les spins qui s'inversent ou non comme énoncé dans le résumé de l'analyse. (M = 0 ou 1). Cela peut se visualiser en représentant l'évolution globale du réseau au cours du temps en fonction de la température globalement sur les graphes ci-dessous avec L = 226 et T = 2,25 on obtient : (long à générer, voir page suivante)

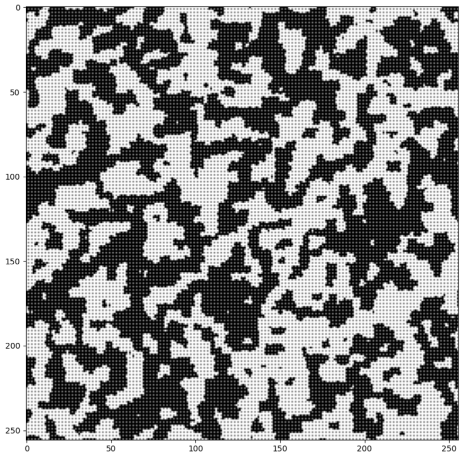

FIGURE 4 – Heat map - L = 226 et T = 2,22

Conclusion

Pour conclure, nous avons implémenté l'algorithme de Métropolis à travers un programme python afin de simuler le comportement d'un tel système. Nous avons discuté de la transition de phase ainsi que de l'estimation de la température critique dans le cas d'un modèle d'Ising ferromagnétique sur un réseau carré de Lieb appauvri au quart avec des spins sur chaque arrête sur chaque carré ainsi que des spins sur chaque médiane de chaque côté de chaque carré. Ce projet nous donc a permis de mieux comprendre la physique statistique pour simuler des systèmes complexes similaires pour d'autres applications à l'avenir. (notamment sur des matériaux)

Références

[Andréas HONECKER] *Cours Magistraux - Monte Carlo, 2023*

[Mohammed ALKHATEEB] *Travaux Dirigés - programmes python TD5 Moodle- Monte Carlo*

[Burak CIVITCIOGLU] *Travaux Dirigés - programmes python TD4,2,1 - Monte Carlo* (algorithmes de métropolis, percolations,..)

[Python.org] *Bibliothèques Python, 2006* (utilisation des bibliothèques)

[Wikipédia] *Transitions de phases physique statistiques, Ising ferromagnétique* (définitions, ...)