SIMULATION MONTE CARLO DU MODÈLE D'ISING
PHQ-404

DEVOIR IV
MÉTHODES STOCHASTIQUES

par

Michael Bédard,
Antoine de Lagrave

Remis à

Martin Schnee

Université de Sherbrooke
Faculté de Sciences
Département de Physique
7 avril 2023

Table des matières

Objectif

L'objectif de ce travail vise l'étude du modèle d'Ising via un algorithme de simulation Monte Carlo dont le hamiltonien est de la forme

$$\mathcal{H} = -J \sum_{\langle i,j \rangle} s_i s_j,$$

où i et j sont des indices de sites dans la grille de spins, $\langle i, j \rangle$ représente les paires de *premiers voisins* et J la contribution énergétique de deux spins voisins s'ils sont antiparallèles. On modélisera ce problème grâce à un système de spins ferromagnétique bidimensionnel avec un couplage entre les premiers voisins. On étudiera l'évolution de l'énergie et l'aimantation moyenne dudit système en fonction de la température de celui-ci et de même pour le temps de corrélation sur les mesures des observables.

1 Arborescence du code

L'arborescence du code est très simple et se décompose en trois parties : la racine, le code et les résultats. La racine du projet, de par son nom contient tout les scripts et fichiers du travail dont la configuration de l'environnement virtuel, le `README` et etc.

Dans le répertoire nommé `./ising/` se trouve les deux autres parties c.-à.-d. le code et les résultats. Le code est constitué de trois scripts : `ising.py` qui contient la définition des objets *Ising* et *Observable* respectivement responsables de la grille bidimensionnelle de spins et des mesures d'énergies et d'aimantation du système. Le second script : `simulate.py` contient une fonction qui permet d'effectuer la simulation pour des paramètres donnés ainsi que la sauvegarde des résultats et une fonction qui permet la mise en graphique desdits résultats. Le troisième script : `main.py` est seulement, par convention, le script permettant à un usager de simuler facilement le système via la fonction *main*.

Finalement, les résultats (figures et fichiers de données bruts) se trouvent dans le répertoire `./ising/data/`. C'est à cet endroit que la fonction *simulate* enregistre les données et les figures par défaut.

2 Modèle d'Ising

C'est dans le script `./ising/ising.py` que se trouve l'implémentation du modèle de spins d'Ising via la classe *Ising*. La création d'un de ces objets nécessite la température du système ainsi qu'une grille de spins. On utilise la fonction *ising_aleatoire* qui prend en argument la température et la taille de la grille (en «nombre de spins») et retourne un objet *Ising* qui est globalement un tableau bidimensionnel constitué de 1 ou -1 répartis aléatoirement [1].

1. La grille est essentiellement un tableau *Numpy* dont les éléments sont générés aléatoirement via la librairie *numpy.random.randint()*.

La classe *Ising* possède un méthode permettant d'effectuer une itération aléatoire suivant l'algorithme de Metropolis. On calcule d'abord la variation d'énergie entre la configuration de spins initiale et une configuration dans laquelle un spin aléatoire a été inversé. Si cette différence minimise l'énergie ou ne l'influence pas, ladite configuration est conservée et si la différence d'énergie est positive, on accepte la nouvelle configuration suivant une probabilité en $e^{-\Delta E/T}$. Pour calculer cette différence d'énergie, il est préférable d'étudier l'influence de l'inversion du spin localement dans la grille, donc de calculer l'énergie du voisinage direct de ce spin avant et après l'inversion. Cette astuce permet de sauver beaucoup d'itérations lors de la simulation et donc d'accélérer le processus. On retrouve ces algorithmes dans les méthodes *difference_energie* et *iteration_aleatoire* de la classe *Ising*.

3 Méthode du binning

On retrouve l'implémentation de la méthode du binning dans le script `./ising/ising.py` dans la classe nommée *Observable*. Le moyennage des mesures ajoutées s'effectuent directement dans la méthode *ajout_mesure* de cette classe. On accède ainsi à la moyenne finale par la méthode *moyenne* de chaque objet. L'erreur associée au moyennage est calculé par la méthode *erreur*, calculé à partir de la variance du niveau -6. C'est d'ailleurs d'une manière similaire que l'on obtient le temps de corrélation par la méthode *temps_correlation* que nous présenterons à la section *5 Visualisation des résultats*.

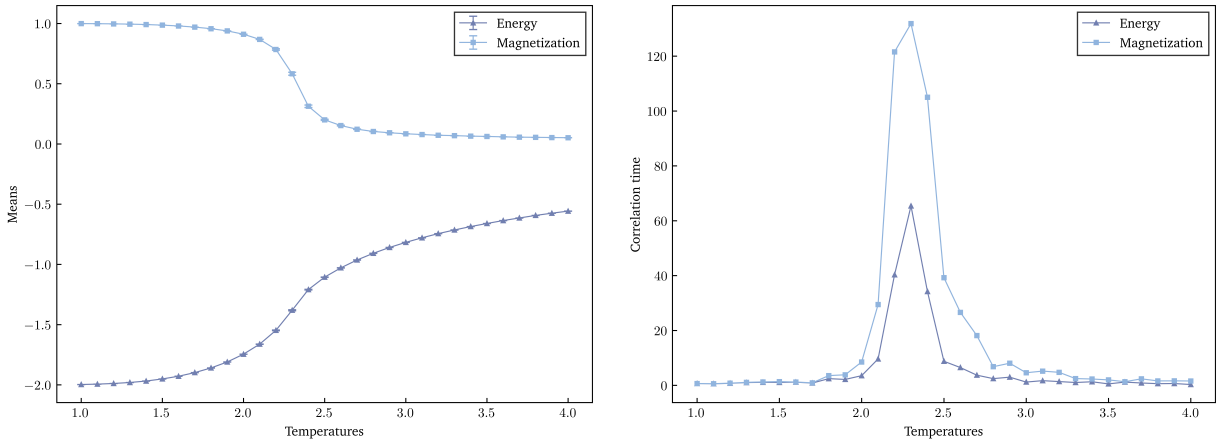
4 Simulation Monte Carlo

La simulation effectuée sur les grilles de spins d'Ising peut être décomposée en plusieurs parties soit : l'initialisation du système et de ses observables, la thermalisation et le moyennage. Le script qui en est responsable est `./ising/simulate.py`. La fonction de simulation s'appelle *simulate* et prend en argument un tableau de températures pour lesquelles simuler des modèle d'Ising (*temperatures*), un intervalle d'itérations pour la prise de mesure de chaque observable (*update*), le nombre de niveaux du moyennage par binning (*levels*), un nombre d'itérations lié au réchauffement du système (*warmup*), la taille d'un des côtés de la grille de spins (*grid_size*) ainsi qu'un paramètre permettant la sauvegarde des résultats (*save*).

On procède donc à la simulation en initialisant une grille aléatoire et des observables (énergie et aimantation) pour chaque température donnée. On effectue une thermalisation du système en effectuant *warmup* itérations. Cela nous assure que la configuration initiale correspond à un état typique pour la température désirer avant de prendre des mesures. Ensuite, on attend que le moyennage se termine et on collecte pendant ce temps des mesures de nos observables à chaque *update* itérations. Il est ensuite possible de sauvegarder toutes ces mesures moyennes, leur erreur, le temps de corrélation pour chaque température et de mettre en graphique ces résultats à l'aide de la fonction *plot_data* présente dans le même script.

5 Visualisation des résultats

Soit la figure 1a, on remarque malgré le nombre relativement élevé de spins dans la grille 32×32 que l'énergie moyenne et l'aimantation sont de l'ordre de l'unité. Il est important de noter à ce sujet qu'il s'agit de l'énergie/aimantation **par spin** et donc on s'attend plutôt à des courbes 1024 fois plus élevées lorsque l'on considère l'énergie/aimantation moyenne **du système**. On note également que le temps de corrélation est bien plus élevé que celui présenté dans l'énoncé du devoir. Apparemment, cela est correct puisque notre graphique représente le temps de corrélation réel alors que le temps de corrélation présenté dans le devoir est un temps de corrélation par spin. Nous remarquons cependant que les barres d'erreurs sont extrêmement petite, ce qui n'est pas ce à quoi on pourrait s'attendre. Nous n'avons toujours pas compris pourquoi cela est le cas.



(a) (b)

FIGURE 1 – Résultats d'une simulation Monte Carlo du modèle d'Ising 2D d'une grille de spins 32×32 pour des températures qui varient entre 1K et 4K (a) Valeurs moyennes et les erreurs d'estimation pour l'énergie et l'aimantation par spin. Les erreurs d'estimation sont obtenues avec la méthode du binning de 16 niveaux. (b) Temps de corrélation pour l'énergie et l'aimantation en fonction de la température.

Nous observions auparavant quelques sauts aléatoires qui apparaissaient fréquemment dans les courbes d'énergies et de d'aimantation. Nous sommes parvenus à les éliminer en refroidissant progressivement le système chaque fois que nous calculions un point à une nouvelle température, en utilisant le dernier états comme point de départ pour la prochaine simulation. Cela empêche notre système de rester pris dans d'autres minimums d'énergie du système, qui généraient les immenses fluctuations.