Master 1 CHPS

Rapport

Calcul de densité d'état de particule pour la modélisation de nanoparticule à transition de spin (SCO) selon un modèle d'Ising.

Moursal MAHAMOUD ALI
Ange Bertrand Dizo
Ibra Ndiaye
Amina Glenn Stevy

Encadrants :
Dr. Thomas Dufaud
Dr. Devan Sohier

Mai 2019

Table des matières

1 Introduction

Le modèle Ising [1] donne une description microscopique de la ferromagnétisme qui est causé par l'interaction entre les spins. Un spin est considéré comme une grandeur scalaire pouvant atteindre les valeurs +1 et - 1. Le modèle est simple et statistique. Il montre la transition de phase entre la phase de paramagnétisme à haute température et la phase ferromagnétique à basse température à une température donnée. En fait, la symétrie entre haut et bas est spontanément brisée lorsque la température descend en dessous de la température critique. Cependant, le modèle unidimensionnel d'Ising, qui a été résolu avec précision[2], ne montre aucune transition de phase. Le modèle d'Ising à deux dimensions a été résolue d'une manière déterministe avec la programmation dynamic [3]. Malgré de nombreuses tentatives pour résoudre le modèle 3D Ising, on peut dire que ce modèle n'a jamais été résolu exactement. Tous les résultats du modèle tridimensionnel d'Ising ont été utilisés comme approches d'approximation et des méthodes de Monte Carlo.Les méthodes de Monte Carlo ou les méthodes de simulation statistique sont largement utilisées dans différents domaines scientifiques tels que la physique, la chimie, la biologie, la finance informatique. La simulation peut être effectuée en échantillonnant à partir de la fonction de densité de probabilité et en générant des nombres aléatoires de manière uniforme. La simulation du modèle d'Ising sur un grand réseaux augmente le coût de la simulation. Une façon de réduire les coûts de simulation consistent à concevoir des algorithmes plus rapides. Les algorithmes de Swendsen-Wang et Wolff [4, 5] et les méthodes de codage à spin multiple [6-7] en sont des exemples. Nous proposons une méthode du modèle Ising qui est Monté Carlo Métropolise [8], ensuite nous parallélisons le modèle avec plusieurs processus (ou threads).

Dans ce rapport, nous présentons un algorithme séquentiel dans un premier temps et dans un seconde la partie parallèle pour calculer la densité d'état d'un plan 2D d'Ising à l'aide de la méthode de Monte Carlo Métropolise. Ensuite, nous exécutons l'algorithme sur un ordinateur portable classique et ensuite sur le cluster de la MDLS (Maison de la Simulation) en utilisant C comme langage de programmation et OpenMP. OpenMP est un modèle de programmation utile dans les systèmes HPC dans lequel des Threads (processus léger) partagent les tâches au sein d'un hotspot : la partie du code qui consomme plus de CPU, ce modèle a été conçu pour des architectures à mémoire partagée. Ainsi, nous allons expliquer théoriquement notre projection MPI qui est aussi un modèle de programmation mieux que OpenMP dans les systèmes HPC et conçu pour des architectures à mémoire distribuée qu'on a regorgé des documentations et des teste d'implémentation.

2 Présentation du Modèle

Le modèle d'Ising est un modèle fréquemment utilisé pour tester de nouvelles idées et méthodes en physique statistique. Afin de représenter les atomes qui composent un matériau magnétique, on fixe deux hypothèses :

— chaque atome a un moment magnétique appelé spin. Pour simplifier, les spins n'auront que deux orientations possibles : haut (on dira que le spin est +1) ou bas (-1) de chaque molécule de spin-crossover (SCO) ;
— initialement, ils sont orientés aléatoirement.

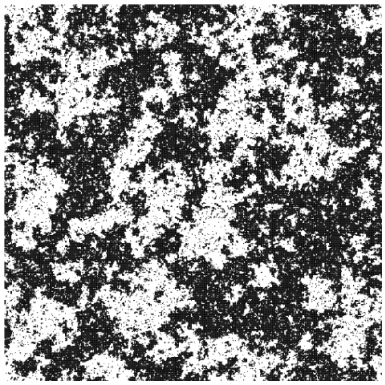

FIGURE 1: Configuration 2D. En noir les spins +1 et en blanc les -1. (src[])

Sur la base de chaque état de configuration, le système SCO macroscopique est décrit par les variables suivantes : m, s, et c avec « m » le nombre d'atomes de spin du domaine, « s » le nombre de paires d'atomes adjacents ayant différentes spins et « c » le nombre d'atomes se trouvant sur le bord, respectivement pour l'aimantation totale, les corrélations à courte portée et l'aimantation de surface. De ce fait, nous déterminons la densité des macrostats d (m, s, c), en donnant le nombre de configurations microscopiques ayant les mêmes valeurs m, s et c.

L'Hamiltonien associée à cette modèle est indiqué à l'équation (1) :

$$\hat{\mathcal{H}} = \frac{\Delta - K_B T\ lng}{2} \sum_{i=1}^{n} \sigma_i \ - G \sum_{i=1}^{n} \sigma_i < \sigma > \ - J \sum_{<i,j>} \sigma_i \sigma_j \ - L \sum_{k=1}^{M} \quad (1)$$

« Cet hamiltonien permet de simuler le comportement d'un matériau dans un environnement caractérisé par ses interactions non seulement au cœur du matériau, mais également à ses bords. Dans l'expression utilisée pour l'hamiltonien, σ est un opérateur des spins fictifs pouvant prendre les valeurs ˇ1 (état de spin faible) et +1 (état de spin élevé), Δ est l'écart entre les états LS et HS , et g = g HS / g LS représente le rapport entre les dégénérescences respectives g HS et g LS des états HS et LS. J et G représentent les paramètres des interactions à court et à long terme, respectivement, et L est un paramètre lié à la contribution des interactions entre les molécules situées sur le bord (surface) et l'environnement local. L'ajout de la constante L est nécessaire, car les molécules à la surface ont des propriétés spécifiques. » Selon l'article [1].

Les variables m, s et c sont définies par les équations (2), (3) et (4) :

$$m = \sum_{i=1}^{n} \sigma_i \ (2) \quad s = \sum_{<i,j>} \sigma_i \sigma_j \ (3) \quad c = \sum_{k=1}^{M} \sigma_k^{'} \ (4)$$

l'équation (1) peut s'écrire alors :

$$\hat{\mathcal{H}} = (\frac{\Delta - K_B T \ ln \ g}{2} \ - G < \sigma >) m \ - Js \ - Lc \ (1)$$

Pour chaque m, s et c designe une configuration donné de densité d(m,s,c). Ainsi, on procède plusieurs itération de monté carlo pour cumuler plus de configuration et on calcule ainsi la densité des macrostats. Pour commencer on s'est poser des questions, comment accepter ou rejeter une configuration ? Selon quelle condition ? Quel est la condition d'arrêt de l'algorithme ?

2.1 Méthode de Monte-Carlo

Les méthodes de Monte-Carlo sont particulièrement utilisées pour calculer des intégrales en dimensions plus grandes que 1 (en particulier, pour calculer des surfaces et des volumes). Elles sont également couramment utilisées en physique des particules où des simulation probabilistes permettent d'estimer la forme d'un signal ou la sensibilité d'un détecteur. La comparaison des données mesurées à ces simulations peut permettre de mettre en évidence des caractéristiques inattendues, par exemple de nouvelles particules [2].

2.2 Choix de la méthode

Dans cette large famille de méthode monté carlo, dans un premier lieux il nous a été demander dans le cadre de se projet d'explorer la méthode **Entropic sampling** pour approcher à la densité d'état. Malgré qu'on a régorgé de documentation sur le sujet on a pas trouver une condition auquel on pourrait se baser pour accepter ou rejeter une configuration donnée. De ce fait, on a opter **Monté Carlo Métropolis** pour tirer parti d'une meilleur approche de la densité d'état selon la variation de l'energie, la constante de Boltzmann et le température.

3 Monté Carlo Métropolise

3.1 Principe

Le principe consiste de partir d'un état actuel pseudo-alétiore et de générez un état d'essai aléatoire qui est «proche» de l'état actuel du système. "Proche" signifie ici que l'état de l'essai doit être presque identique à l'état actuel, à l'exception d'un petit changement aléatoire, effectué généralement sur une seule particule ou spin. Ce qui est notre cas de choisir un ou plusieurs spin (cas parrallèle) de façon alétiore. L'état d'essai d'un système de spin implique généralement un retournement ou une rotation aléatoire d'un spin unique ou plusieurs dans le cas parrallèle.

3.2 Démarche Algorithmique

1. Sélectionnez un spin de manière aléatoire et calculez l'énergie d'interaction entre ce spin et ses voisins les plus proches (E).
2. Retournez le spin s (i, j) et calculez à nouveau l'interaction de l'énergie (E ').
3. On fait la différence d'énergie (E') - (E) si cette différence est inférieur ou égale à 0, le spin est accepté, sinon le spin est accepté selon une probabilité : si un nombre réel aléatoire entre [0,1] est inférieur à $exp\frac{-\Delta E}{K*T}$ où K est la constante de Boltzmann et T une température donnée.
4. Répétez les étapes 1 à 3 jusqu'à ce que nous soyons sûrs que chaque rotation a été inversée.
5. Calculez l'énergie totale du réseau pour tout les itérations (E^i_{total})

   Les étapes ci-dessus forme une itération de Monte Carlo. Nous effectuons assez d'itération (N_{MC} nombre de fois de MC) et finalement en moyenne sur (E^i_{total}) pour obtenir Etotal :

   $E_{total} = \frac{1}{N_{MC}}(\sum E^i_{total})$

   Puisque cette énergie totale E_{total} de tout les itérations n'influence pas sur la condition de garder une configuration ou pas donc on va pas le calculer. Par contre l'énergie pour chaque itération est le socle de notre algorithme.

3.3 Peudo-Code MC Métropolise

Algorithm 1 MC Métropolise

$Mat[N][N]$; domaine d'étude 2D
Nb_iter_{MC} : Nombre d'itération de MC
for $0 : Nb\_iter_{MC}$ **do**
  $SelectRandoom(Mat, ai, aj)$
  $Energy_{actuel} \leftarrow CalculEnergy(Mat)$
  $Mat[i][a] \leftarrow -Mat[i][a]$ //retournement de spin
  $Energy_{essai} \leftarrow CalculEnergy(Mat)$
  $\Delta E \leftarrow Energy_{essai} - Energy_{actuel}$
  if $\Delta E \leq 0$ **then**
    $Accept(Mat)$; // on accepte la nouvelle configuration
  else
    $x \leftarrow RANDOOM\_UNIFORM()$; // un nombre réel aléatoire entre [0.0,1.0]
    if $x < e^{\Delta E*\beta}$ **then**
      // avec $\beta = 1/(K_B$*T), K_B constante de Boltzmann et T température
      $Accept(Mat)$;
    end if
  else
    $Mat[i][a] \leftarrow -Mat[i][a]$ // on annule le retournement
  end if
end for

4 Implémentation séquentiel du modèle

4.1 Principe

On ce qui concerne l'implémentation de ce modèle, dans un premier temps on a implémenté en séquentiel avec le langage de programmation *standard* **C** et par la suite on verra un peu plus loin la parallélisation avec l'API OpenMP.

Pour commencer, on part d'une matrice 2D de taille N*N, chaque case (i,j) de la matrice représente le comportement d'un molécule c'est-à-dire un spin qui a pour valeur ± 1. Et ces valeurs sont générées de façon aléatoire avec la fonction rand() du système mais pour avoir un bon générateur nécessite un peu de technique et nous verrons dans la partie démarche.
Ensuite on tire de façon aléatoire avec une probabilité de $\frac{1}{N}$ de case (i,j) et on calcule son énergie par rapport à ses voisins dans la grille avec la fonction ith_energy qui retourne un type double, ainsi on le multiplie par -1 pour lui faire subir un retournement, après avoir fais cela on recalcule la nouvelle énergie. Alors on effectue la différence d'énergie avant le retournement du la molécule et après, si cette énergie est négatif ou nul, on accepte la configuration. Sinon on compare avec un nombre réel x aléatoire uniformément distribuer entre [0.0,1.0], si x < exp(-$\Delta E * \beta$) : avec $\beta = \frac{1}{K*T}$ et T la température est à rentrer en paramètre du programme, alors on accepte la configuration sinon on rejet la configuration et on procède à l'itération suivantes de *MC*. Chaque configuration est composer de 4 variables m,s, c et z ; les trois premiers ont été détaillés précédemment, mais la quatrième z indique la densité associée.

4.2 Démarche

1. La fonction **updown** effectue l'initialisation de la matrice 2D avec ± 1 selon une graine aléatoire de $rand()\%2$, si la graine est inférieure à 0.5 elle retourne (+1) sinon (-1).
2. Pour tirer un élément de la matrice aléatoirement, les indices i et j vont varier selon une probabilité de $\frac{1}{N}$ avec N(n1*n2) la taille de la matrice, pour ce faire on utilise la fonction *srand*48(5321) avec une graine assez grande pour la génération d'une nouvelle séquence de nombres pseudo-aléatoires, qui seront fournis par rand(), ainsi i prend comme valeur $rand()\%N$ et j de même.
3. On calcule l'énergie avec la fonction **ith_energy**, différent cas s'impose selon la position où se trouve l'élément tirer puisque le calcule se fait selon son voisin le plus proches, si l'élément (i,j) est sur le bord il posède soit 2 voisins ou 3 et si il se trouve sur la surface il en a 4. Soit E_{old} l'énergie calculer avant le retournement et E_{essai} l'énergie après retournement de l'élément.
4. La différence énergétique est $\Delta E = E_{essai} \breve{} E_{old}$, si $\Delta E \leq 0$ on stock la configuration : on fait appelle à la fonction **Calcul_conf** permettant de calculer une configuration d(m,s,c) et qui retourne un pointeur contenant cette configuration et on le stock dans un tableau (*). Sinon on prend un nombre pseudo-aléatoire **x** avec la fonction *drand*48() qui renvoi des valeurs réelles en virgule flottante uniformément distribuées dans l'intervalle [0.0, 1.0], si **x** < exp(-$\Delta E * \beta$) on effectue la même tache (*) sinon on annule le retournement de la case en le ré-multipliant par -1 et on ne stock pas.
   (*) On stock en bloc de 4 à l'aide de la fonction **add_config** tout ces configuration dans un tableau de taille 4 fois le nombre d'itération de *MC*, les variables m, s et c sont stocker au trois premier bloc et la quatrième bloc indique la densité de cette configuration c'est-à-dire le nombre de fois de tour de *MC* où on tombe sur la même configuration.

Le stockage d'une configuration s'effectue par comparaison des configurations déjà enregistrer dans le tableau, si elle existe donc on injecte pas celle-ci dans le tableau mais sa valeur de z est mis à jour.

4.3 Objectif

- Bien choisir une graine aléatoire pour avoir des résultat significatif
- Trouver une condition d'acceptation ou de rejet d'une configuration
- Gérer le stockage des configurations (injection de quatre cases ou incrément d'une case)

Références

[1] F. Varret I. Sheto, J. Linares. Monte carlo entropic sampling for the study of metastable states and relaxation paths. *Physical review E*, page 56, 1997.

[2] Nicholas Metropolis. méthode de monte-carlo, 1949. https ://fr.wikipedia.org/wiki/.