Rapport

Analyse en C du modèle de gaz sur réseau

Gauthier Rey

Contents

1 Introduction sur le modèle d'Ising et la transition de phase

La transition de phase liquide-gaz est un phénomène fascinant qui a été étudié en profondeur dans la physique statistique.

Pour étudier ce phénomène, on peut utiliser différents modèles, tels que le modèle d'Ising, qui est un modèle simplifié utilisé pour comprendre les transitions de phase dans les systèmes de spins.

Le modèle d'Ising a été résolu exactement pour des cas spécifiques. En 1D, la solution exacte a été obtenue par Lars Onsager en 1944. Il a démontré que le modèle 1D d'Ising n'a pas de transition de phase à une température non nulle, ce qui signifie qu'il ne présente pas de phénomène de transition liquide-gaz ou de magnétisation spontanée.

Pour le modèle d'Ising 2D, la situation est plus complexe et intéressante. Onsager a également réussi à déterminer l'énergie libre d'un système d'Ising en 1944, en utilisant des techniques mathématiques avancées. Dans ce cas, il a été démontré que le modèle présente une transition de phase à une température critique non nulle. La solution exacte en 2D est assez compliquée et lourde en termes de calculs, ce qui la rend difficile à généraliser pour des systèmes de dimensions supérieures.

Il est important de noter que, bien que le modèle d'Ising ne représente pas directement la transition liquide-gaz, il présente des similitudes frappantes avec des modèles de gaz sur réseau, ce qui en fait un outil utile pour étudier les transitions de phase en général. Les solutions exactes en 1D et 2D ont permis d'améliorer notre compréhension des transitions de phase et ont inspiré des approches numériques, telles que la méthode de Monte Carlo, pour étudier des systèmes plus complexes et des dimensions supérieures.

Dans cette étude, nous explorons la transition de phase en utilisant une approche numérique alternative, la méthode de Monte Carlo. Notre modèle se situe sur un réseau discret bidimensionnel à maille carrée, où chaque site peut être occupé ou non par une molécule, formant ainsi un "gaz sur réseau".

Dans ce cadre, nous associons une variable booléenne n_i à chaque site i du réseau, qui indique si le site est occupé ($n_i = 1$) ou vide (ni = 0). Nous travaillons dans l'ensemble canonique, où la température T et le nombre total de molécules N sont fixés. L'énergie d'une configuration des variables n_i est donnée par une fonction hamiltonienne qui prend en compte l'intensité de l'attraction entre les molécules situées sur des sites voisins.

Pour étudier les propriétés du système, nous utilisons des conditions aux bords périodiques. La probabilité canonique de chaque configuration n_i est déterminée par la fonction de partition, qui dépend de la température et de l'énergie de la configuration. Cependant, le calcul exact de ces sommes est généralement impossible pour des valeurs importantes de Ns, rendant difficile l'approche numérique de la limite thermodynamique.

Pour surmonter cette difficulté, nous employons un algorithme de Monte Carlo de type Metropolis. La méthode Monte-Carlo est une technique de simulation numérique largement utilisée en mathématiques, physique, économie, finance, et autres domaines pour résoudre des problèmes complexes en utilisant des échantillons aléatoires. Cette approche est particulièrement utile pour évaluer des situations où il est difficile d'obtenir une solution exacte ou analytique. Les algorithmes Monte-Carlo tirent leur nom du célèbre casino de Monte-Carlo à Monaco, en référence à la nature aléatoire des jeux de hasard.

Pour illustrer l'utilisation de la méthode Monte-Carlo nous pouvons utiliser les échecs, considérons un joueur qui cherche à déterminer le meilleur coup à jouer ou qui cherche à savoir s'il faut échanger, ou bien mener une attaque. On pourrait générer des coups et pour chaque coup effectuer un grand nombre de parties aléatoires (appelées plongées) en jouant des coups choisis au hasard jusqu'à la fin de la partie. En examinant le résultat moyen des plongées on peut alors trouver le meilleur coup.

Ici, notre algorithme consiste à sélectionner un site occupé i au hasard et l'un de ses voisins j, puis à évaluer la différence d'énergie ΔE entre la configuration initiale et celle où la molécule a sauté de i à j. En fonction de cette différence d'énergie, nous décidons si le saut doit être effectué ou non. Cette procédure, répétée suffisamment de fois, conduit à une distribution de configurations qui respecte la probabilité d'équilibre.

2 Études numériques de niveau 1

2.1 Question 1

1. Commencer par prendre un réseau de taille modeste, par exemple Ns = 20×20, rempli de N = 100 particules et par simuler le système à haute température (par exemple kBT = 20J) pour vérifier qu'il est en phase gaz.

Le programme commence par initialiser la grille et placer 100 particules aléatoirement. Il effectue ensuite 50 000 itérations au cours desquelles une particule et un voisin sont choisis aléatoirement. Si le voisin est vide, l'énergie de la configuration actuelle et celle après le saut de la particule sont calculées. La décision de réaliser le saut est basée sur la différence d'énergie (dE) et la température du système (T). Le saut est effectué si dE < 0 ou si un nombre aléatoire tiré entre 0 et 1 est inférieur à la probabilité calculée.

La probabilité étant e^{-dE/k_bT}, il y a une compétition entre la température et l'interaction entre chaque particule.

Après chaque itération, l'énergie moyenne du système est mise à jour et enregistrée dans un fichier. Finalement, une fois toutes les itérations terminées, la grille finale est enregistrée dans un fichier texte et une image PNG de la grille est générée à l'aide de Gnuplot pour visualiser l'état final du système.

Nous allons définir plusieurs fonctions qui vont nous être utiles pour la suite. Voici une fonction qui calcule le nombre de voisins.

```
int calc_nb_voisins(int tab[][Y], int x, int y){
    int nb_vois = 0;
    int voisx1 = ( x + 1 ) % X;
    int voisy1 = ( y + 1 ) % Y;
    int voisx_1 = ( (x-1) + X ) % X;
    int voisy_1 = ( (y-1) + Y ) % Y;

    if(tab[x][voisy1] == 1){
        nb_vois+=1;
    }
    if(tab[x][voisy_1] == 1){
        nb_vois+=1;
    }
    if(tab[voisx1][y] == 1){
        nb_vois+=1;
    }
    if(tab[voisx_1][y] == 1){
        nb_vois+=1;
    }
    return nb_vois;
}
```

Maintenant nous pouvons donc calculer l'énergie d'une case en fonction de son nombre de voisins.

```
double calculer_energie_case(int x, int y, double J, int tab[][Y]){
    double E;
    E = -J * calc_nb_voisins(tab, x, y);
    return E;
}
```

Finalement, en passant dans toutes les cases de notre tableau nous pouvons calucler l'énergie de toute notre configuration.

```
double calculer_energie_configuration(int tab[][Y], double J){
    double Enr = 0;
    int i,j;
```

```
    for(i = 0; i<N ; i++){
        for(j=0; j<N; j++){
            if(tab[i][j] == 1){
                Enr = Enr + calculer_energie_case(i, j, J, tab);
            }
        }
    }
    return Enr;
}
```

C'était les trois fonctions les plus importantes de notre programme. Il y a avec les fichiers envoyés tout le reste du code. Pour que ce document reste propre je ne vais pas recopier le code en entier mais je vais seulement afficher les résultats.

Dans le cadre de notre modèle, plutôt que de calculer systématiquement l'énergie totale de la configuration à chaque déplacement d'une particule, nous adoptons une approche plus efficace qui se concentre sur les particules directement impactées par le changement de position. En effet, seuls les voisins des voisins de la particule déplacée sont affectés par ce mouvement. Ainsi, nous nous limitons à calculer l'énergie des voisins des voisins de la particule concernée, et réitérons cette méthode à chaque déplacement. La différence d'énergie résultant de ces calculs locaux correspond à la différence d'énergie globale entre les deux configurations complètes, ce qui permet de gagner un temps précieux dans le traitement du modèle.

La figure 1 nous montre trois gilles finales.

Il faut noter que j'ai ici pris 50 000 itérations.

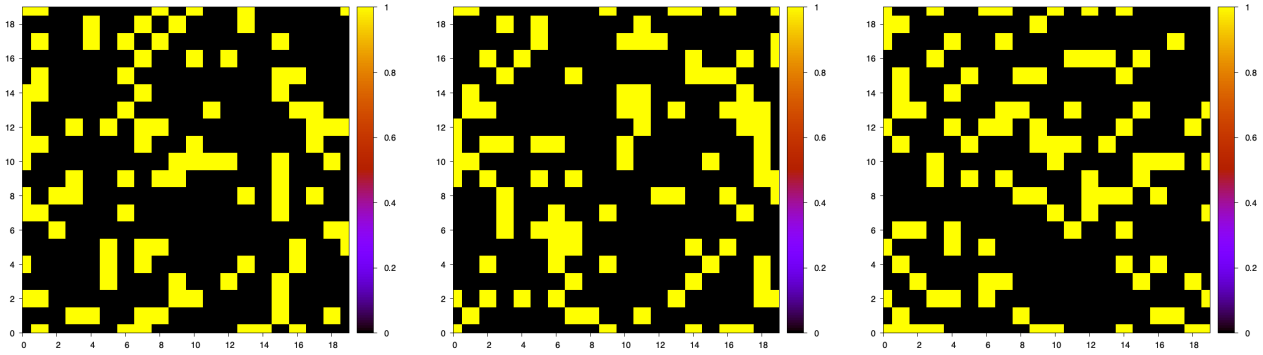

Figure 1: Grille finale pour kbT = 20J

Lors de la simulation du modèle d'Ising à haute température, on observe que le système présente bien un comportement correspondant à une phase gazeuse. Dans cette phase, les particules ont une distribution plus homogène sur la grille et ne forment pas de structures localisées ou de domaines d'alignement. C'est dû à l'augmentation de l'énergie thermique qui permet aux particules de surmonter les interactions attractives et d'occuper plus librement les positions disponibles sur la grille.

2.2 Question 2

2. Le simuler à basse température (par exemple kB T = 0.5 J) pour vérifier l'apparition d'une phase condensée en coexistence avec une phase gaz.

Nous allons maintenant le simuler à "basse" température et observer la phase. En théorie, il y a moins d'énergie disponible donc nos particules devraient avoir moins de mal à se regrouper. La seule chose qui change dans le code précédent est la valeur que nous prenons pour kbT. Ici, nous avons choisi 0.5J. La seule ligne qui change est donc la ligne numéro 83 :

p0 = exp(-dE/(0.5) (J vaut 1 ici)

La figure 2 représente cette fois-ci trois grilles obtenues à basse température.

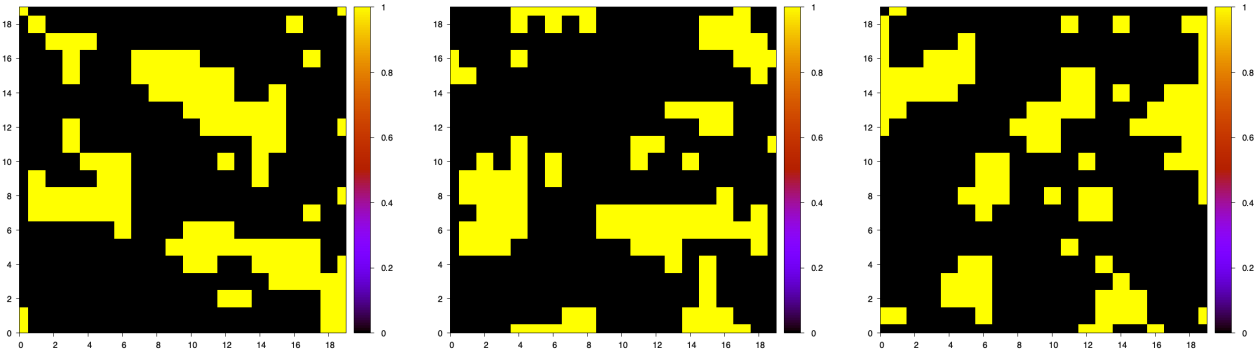

Figure 2: Grille finale pour kbT = 0.5J

Dans cette simulation à température intermédiaire, il a été observé que le système présente à la fois une phase condensée et une phase gazeuse. Cette coexistence des deux phases est caractéristique d'un point de transition entre les états d'ordre et de désordre du système. La simulation à cette température spécifique démontre la capacité du modèle d'Ising à capturer le comportement complexe de ces systèmes et révèle l'importance de l'effet de la température sur la nature des interactions entre les particules.

2.3 Question 3

3. Étudier cette la transition de phase et rechercher numériquement la température critique Tc en fonction de la densité $\rho = N/Ns$. On pourra augmenter la taille du système à densité ρ constante pour vérifier que Tc n'en dépend que faiblement.

Pour cela, l'algorithme devient un peu plus complexe, nous devons désormais utiliser la température et non prendre des valeurs arbitraires et la faire varier. Nous devons ensuite calculer l'énergie d'équilibre pour chaque température. Pour calculer l'énergie d'équilibre, nous sommons seulement l'énergie de chaque case. La fonction utilisée calcule l'énergie de la configuration et l'enregistre dans un fichier en .txt externe. A la fin du programme, nous allons pouvoir observer notre fichier final grâce à Gnuplot.

Etant donné que notre figure n'est pas très lisse pour $N = 20$, nous aimerions le faire tourner sur une grille plus grande et avec plus de points mais je me suis vite rendu compte que notre algorithme n'était pas assez rapide. Il était temps de faire quelques modifications.

- Tout d'abord, la méthode expliquée dans la question 1 (ne considérer que les voisins des voisins de la particule) nous permet de gagner beaucoup de temps.
- Ensuite pour rendre initialiser notre tableau aléatoirement à chaque nouvelle température, je devais faire deux boucles pour remplir le tableau entièrement de 0 et encore refaire deux boucles pour placer aléatoirement des 1 dans notre tableau. Cela prenait évidemment bien trop de temps. Pour réduire cela, nous pouvons créer une liste qui va contenir tous les indices du tableau en une dimension.

  Tout d'abord, je crée un tableau indices qui contient les indices de chaque case du tableau principal (de taille X * Y). Ensuite, je mélange aléatoirement les indices avec la fonction shuffle. Je parcours ensuite chaque indice mélangé et je remplis le tableau principal tab avec des 1 et des 0 en fonction de la position de l'indice dans le tableau mélangé. Si l'indice est inférieur à Ns le nombre de particule, la case du tableau principal reçoit la valeur 1 ; sinon, elle reçoit la valeur 0. Cette version de l'algorithme permet de gagner du temps sur la façon de remplir aléatoirement mon tableau étant donné que je n'ai pas besoin de passer deux fois dedans.
- De plus, j'ai mis en place un autre tableau, nommé tab_1, qui permet de conserver les positions des particules ayant la valeur 1 dans le tableau principal. Cette approche s'avère extrêmement utile pour sélectionner rapidement une particule de manière aléatoire. Auparavant, je choisissais ma particule en sélectionnant un nombre inférieur à Ns, puis en parcourant le tableau principal jusqu'à trouver la particule désirée. Toutefois, cette méthode s'avérait très consommatrice de temps, en particulier pour les grands tableaux et un nombre important de particules.

Dorénavant, pour choisir une particule, je me contente simplement de générer un nombre aléatoire et d'accéder directement à sa position grâce au tableau tab_1. Cette optimisation me permet de gagner un temps précieux lors de la sélection des particules.

Voici la fonction qui enregistre l'énergie dans le fichier voulu:

```
void udpate_E_moy(int tab [][Y], double J){
    double Enr = calculer_energie_configuration(tab, J);
    FILE *fichier;
    fichier = fopen ("nrjmpy.txt", "a");
    fprintf(fichier, "%f\n", Enr);
    fclose(fichier);
}
```

Nous pouvons ensuite afficher le fichier texte :

```
void afficher_energie_equilibre_gnuplot() {
    system("gnuplot -e \"set terminal pngcairo size 800,600; set output 'energie_equilibre.png'; set xlabel
        \\\"Temprature\\\"; set ylabel \\\"nergie d''quilibre\\\"; set grid; plot 'energie_equilibre.txt'
        using 1:2 with linespoints title \\\"nergie d''quilibre\\\"\"");
}
```

Cette fonction crée un fichier PNG dans le dossier de travail qui va nous afficher l'énergie d'équilibre en utilisant Gnuplot.

Ensuite, lorsque nous calculons l'énergie pour une température, nous allons pour être plus précis prendre comme "énergie d'équilibre" la temprétaure moyenne sur les 50 dernières itérations. Une fois que nous avons cette énergie d'équilibre associée à une température, nous pouvons l'enregistrer dans un fichier texte. Lorsque nous avons calculé cette énergie d'équilibre pour toutes ces températures, nous pouvons afficher notre courbe encore avec gnuplot. Nous allons chercher pour quelle température notre système change d'énergie d'équilibre ce qui nous permet de déterminer la température critique.

La figure représente la courbe affichée si je fais varier la température de 0.1 à 5 en pas de 0.01 (il y a donc 490 points).

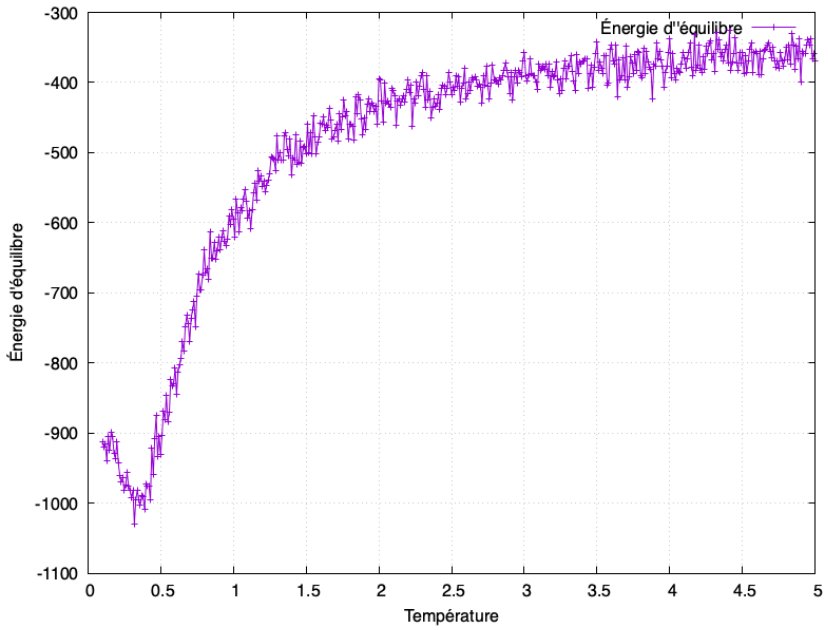

Figure 3: Energie d'équilibre en fonction de la température

Nous pouvons voir qu'il y a une certaine température pour laquelle notre énergie arrête d'augmenter, c'est la température pour laquelle on change de phase. Pour essayer de la calculer on peut faire deux régression linéaire, une sur la première pente et la seconde sur le plateau des derniers points. Voici un schéma qui nous permettra de trouver notre

température d'équilibre 4. Pour obtenir ce schéma j'ai fais tourner mon algorithme sur une grille de 80*80 avec 1000 points, ce qui représente une densité de 0.15.

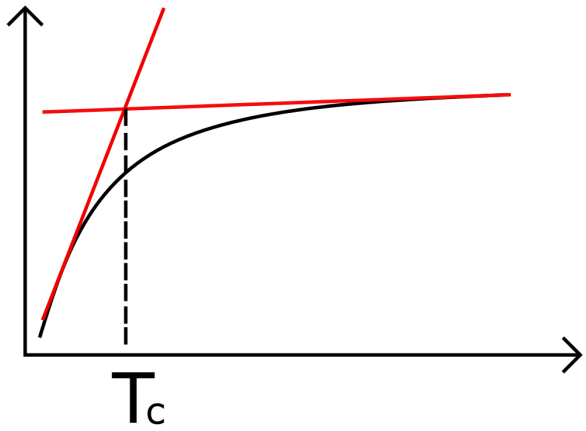

Figure 4: Détermination graphique de la température critique

Pour cela, nous pouvons écrire un petit algorithme en C qui va faire les deux régressions linéaires puis qui va trouver où les deux courbes se croisent. Pour faire la régression linéaire, on va choisir arbitrairement quels points utiliser. Pour la première droite, je vais utiliser les points 0 à 100 et pour la seconde les points 150 à 220 sachant qu'il y a 250 points en tout étant donné que ma température varie de 0,5 à 3 en pas de 0,01.

Voici le programme utilisé pour réaliser la régression linéaire:

```c

#include <stdio.h>
#include <stdlib.h>

typedef struct {
    double a;
    double b;
} LinearRegression;

LinearRegression linear_regression(double *x, double *y, int start, int end) {
    int n = end - start;
    double sum_x = 0, sum_y = 0, sum_xy = 0, sum_x2 = 0;
    for (int i = start; i < end; i++) {
        sum_x += x[i];
        sum_y += y[i];
        sum_xy += x[i] * y[i];
        sum_x2 += x[i] * x[i];
    }

    double a = (n * sum_xy - sum_x * sum_y) / (n * sum_x2 - sum_x * sum_x);
    double b = (sum_y - a * sum_x) / n;

    LinearRegression result = {a, b};
    return result;
}

int main() {
    // Charger les donnes depuis le fichier texte
    FILE *file = fopen("energie_equilibre.txt", "r");
    if (file == NULL) {
        printf("Erreur d'ouverture du fichier.\n");
        return 1;
    }

```

```
    double x[300], y[300];
    int i = 0;
    while (fscanf(file, "%lf %lf", &x[i], &y[i]) == 2 && i < 300) {
        i++;
    }
    fclose(file);

    // Effectuer les rgressions linaires
    LinearRegression reg1 = linear_regression(x, y, 0, 100);
    LinearRegression reg2 = linear_regression(x, y, 150, 220);

    // Afficher les coefficients
    printf("Rgression linaire 1 (points 0  100) : a = %lf, b = %lf\n", reg1.a, reg1.b);
    printf("Rgression linaire 2 (points 150  220) : a = %lf, b = %lf\n", reg2.a, reg2.b);

    // Calculer le point d'intersection Tc
    double Tc_x = (reg2.b - reg1.b) / (reg1.a - reg2.a);
    double Tc_y = reg1.a * Tc_x + reg1.b;

    // Afficher le point d'intersection Tc
    printf("Point d'intersection Tc : x = %lf, y = %lf\n", Tc_x, Tc_y);

    return 0;

}
```

Après avoir fait tourner notre programme nous trouvons une température critique T_c de l'ordre de $1.149453 \pm 0.5°K$. On rappelle que notre grille est de 20*20 et que nous avons 100 points. C'est-à-dire que notre densité est de 0.25.

La figure 5 est une image qui représente l'énergie en fonction de la température, en mettant les deux droites pour visualiser la température critique.

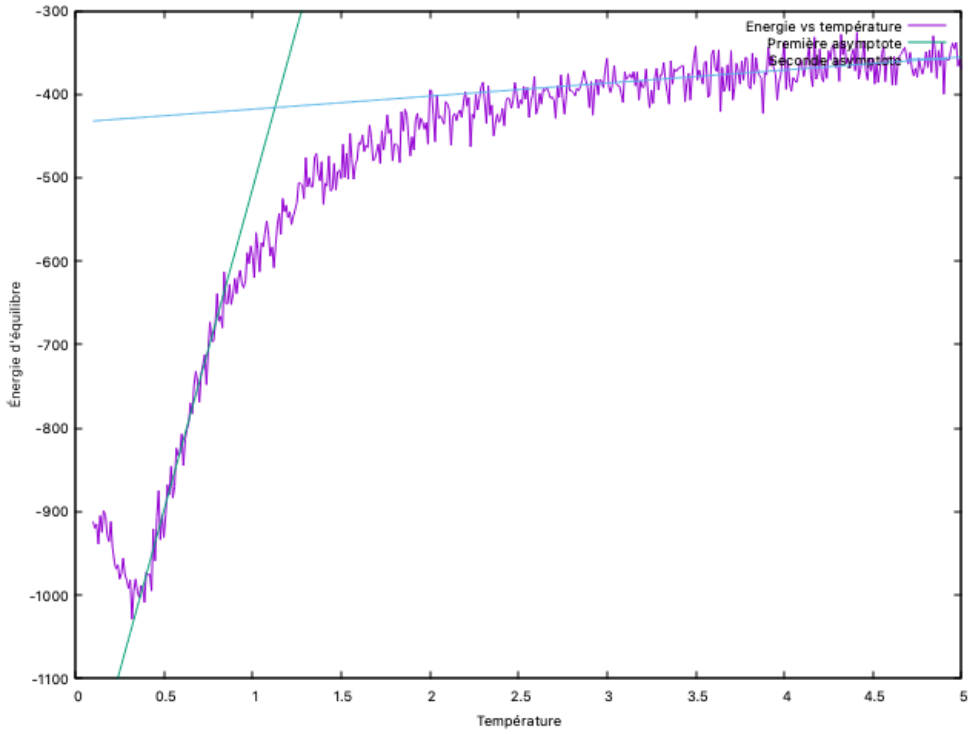

Figure 5: Energie d'équilibre en fonction de la température pour $Ns = 1000$

Nous pouvons nous intéresser à la température critique en changeant la densité.

Ici nous pouvons essayer de prendre différentes valeurs du nombre de particules et regarder comment est affecté la température critique. Pour cela nous allons prendre comme valeurs de N 80 et partir d'un nombre de particules de 300 pour aller jusqu'à 6000. On aura donc une densité qui va varier de 0.05 jusqu'à 0.9375. La figure 6 représente les valeurs que nous avons pour la température critique en fonction du nombre de particules.

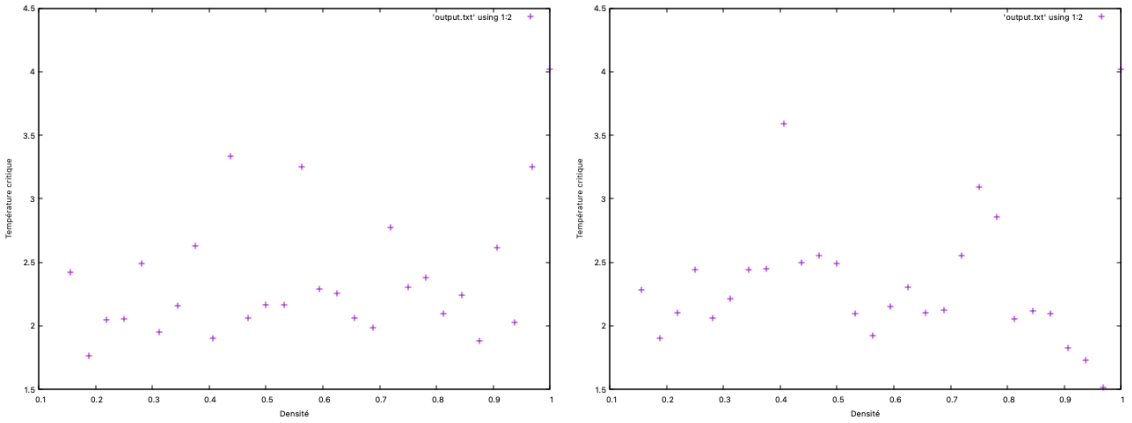

Figure 6: *Tc en fonction de la densité à N=80*

Nous voyons une certaine tendance des points à augmenter puis diminuer comme une sorte de parabole mais cela n'augmente pas énormément, même en changeant la densité de 0 à presque 1 la température critique reste comprise dans l'intervalle $[1, 3]$.

Nous pouvons essayer de changer la taille de la grille avec la même densité.

La figure représente la température critique par rapport au nombre de particules pour $N = 100$ et la figure représente la température critique pour $N = 50$.

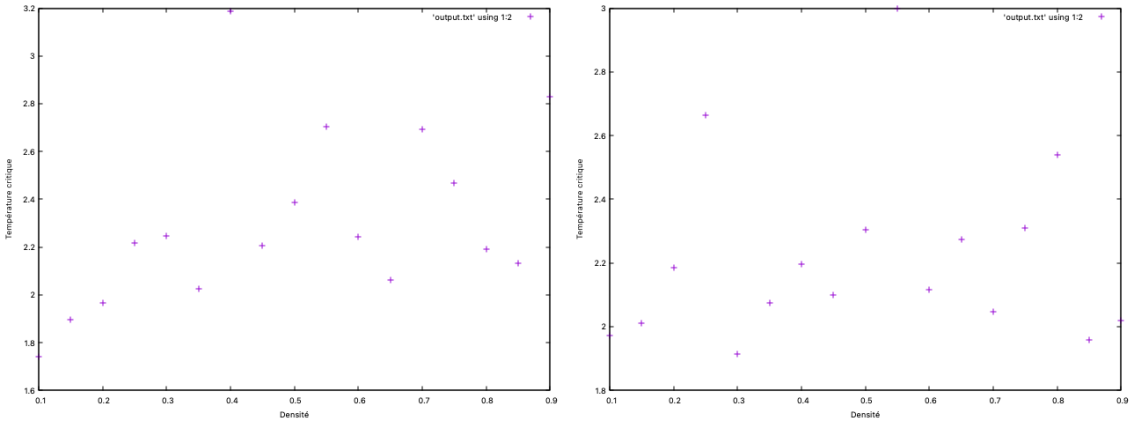

Figure 7: *Tc en fonction de la densité à N=80*

Les figures 6, 7 et 8 sont toutes les trois très similaires, nous remarquons que les points ne changent que très peu si ce n'est pas du tout pour la même densité. On remarque aussi que la moyenne des points est aux alentours de la température calculée par notre modèle ($T_c = 2.269 \frac{J}{k_b}$).

3 Etudes numériques de niveau 2

3.1 Question 1

1. Chercher dans la littérature et comprendre les méthodes de résolution analytiques (exactes) du modèle d'Ising sur réseau carré en dimension 2 pour $\rho = 1/2$ et comparer la température critique de la solution analytique avec la solution numérique (attention, les modèles sont équivalents mais ne sont pas strictement identiques).

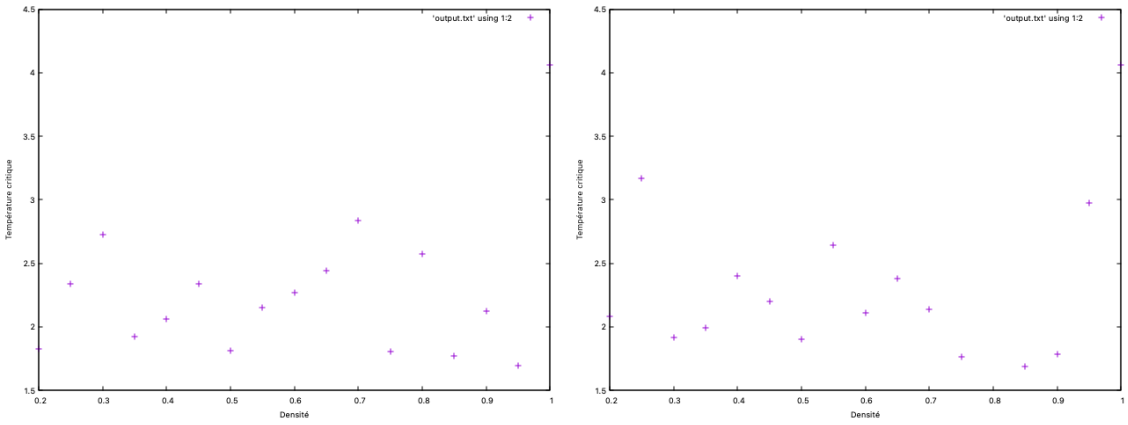

Figure 8: *Tc en fonction de la densité à N=50*

Ising, en 1925 lui-même a effectué une analyse combinatoire du modèle unidimensionnel et a découvert qu'il n'y avait pas de transition de phase à une température T finie. Cela l'a conduit, à tort, à conclure que son modèle ne présenterait pas non plus de transition de phase dans des dimensions supérieures. En fait, c'est cet "échec supposé" du modèle d'Ising qui a motivé Heisenberg à développer, en 1928, la théorie du ferromagnétisme basée sur une interaction plus sophistiquée entre les spins. Ce n'est qu'après une certaine exploitation du modèle de Heisenberg que les gens sont revenus étudier les propriétés du modèle d'Ising.

Le premier résultat exact et quantitatif pour le modèle d'Ising bidimensionnel a été obtenu par Kramers et Wannier (1941) qui ont réussi à localiser la température critique du système. Ils ont été suivis par Onsager (1944 [3]) qui a dérivé une expression explicite pour l'énergie libre en champ nul et a ainsi établi la nature précise de la singularité de la chaleur spécifique. Les auteurs ont utilisé la méthode de la matrice de transfert pour résoudre le problème unidimensionnel correspondant ; son application au modèle bidimensionnel, même en l'absence de champ, s'est avérée être une tâche extrêmement difficile. Bien que certaines de ces difficultés aient été atténuées par les traitements ultérieurs de Kaufman (1949) et de Kaufman et Onsager (1949), il semblait très naturel de rechercher d'autres approches plus simples.

Il faut savoir que dans toutes ces méthodes le calcul reste assez complexe et lourd à mettre dans un rapport. Pour ces raisons nous n'allons pas décrire toute la résolution analytique mais regarder les points les plus important.

Il faut aussi souligner que depuis le début de ce rapport nous parlons de particules sur un réseau qui peuvent se déplacer et de modèle d'ising, mais comment relier les deux ?

Nous pouvons faire beaucoup de liens entre les deux modèles [1]. En rappel, le modèle d'Ising est constitué de spins sur un réseau, ici bidimensionnel et ces spins sont susceptibles de se retourner, valant une valeur binaire de ± 1. La dynamique de ce système est décrite par le Hamiltonien suivant :

$$H\left(\{S_i\}\right) = -J \sum_{\langle i,j \rangle} S_i S_j - B \sum_{i=1}^{N} S_i$$

Nous pouvons souvent limiter l'interaction i,j aux premiers voisins. Nous remarquons plusieurs constantes que nous pouvons décrire et leur affecter des liens avec notre modèle. La constante J représente l'interaction entre chaque spin, cette interaction va tendre à aligner les spins entre eux. On peut aussi prendre un $J < 0$ où l'on rencontrerait des spins qui s'inversent. Dans notre modèle cependant, J représente une force qui va tendre à rapprocher les particules entre elles, on peut par exemple la considérer comme une force de Van der Waals. Ensuite, le champ B va aussi jouer sur chaque spin, il va essayer de les retourner. Si notre champ B est très puissant, nous pouvons nous retrouver avec tous nos spins alignés au champ. Dans notre modèle nous pouvons relier cela à un potentiel chimique qui va fixer le nombre de particules moyen dans notre grille, comme si un gaz était présent au-dessus de notre réseau et que les particules venaient s'y piéger. Dans notre cas nous nous plaçons dans l'ensemble micro-canonique, c'est-à-dire que nous n'échangeons pas de particule à un certain réservoir et que notre énergie et de ce fait notre température reste constante. Nous allons donc nous intéresser à la résolution du modèle d'Ising avec un champ extérieur B nul, mais avant ça il faut fair le lien entre notre modèle et le modèle d'Ising.

Considérons le hamiltonien de notre modèle, le gaz sur réseau :

$$H = -J \sum_{\langle i,j \rangle} n_i n_j$$

En sachant que $n_i = 0$ pour une case vide et $n_i = 1$ pour une case occupée. En faisant le changement de variable $n_i = \frac{1+\sigma_i}{2}$ avec $\sigma_i = \pm 1$:

$$\begin{aligned} H &= -J \sum_{\langle i,j \rangle} \frac{(1+\sigma_i)(1+\sigma_j)}{4} \\ &= \frac{-J}{4} \sum_{\langle i,j \rangle} \sigma_i \sigma_j - \frac{J}{2} \sum_{\langle i,j \rangle} \sigma_i - \frac{J}{4} \sum_{\langle i,j \rangle} 1 \end{aligned}$$

Si on ne somme que sur les voisins proches et qu'on appelle ce nombre ν, qui dépend de la configuration dans laquelle on est, on peut décomposer la somme suivante :

$$\sum_{\langle i,j \rangle} = \frac{1}{2} \sum_{i}^{N} \sum_{1^{ers} voisins}$$

Alors on peut renommer notre Hamiltonien:

$$H_{LG} = -\frac{J}{4} \sum_{\langle i,j \rangle} \sigma_i \sigma_j - \frac{J}{4} \nu \sum_i \sigma_i - \frac{N \nu J}{8}$$

Or, on est ici placé dans l'ensemble canonique, où notre nombre de particules serait fixe, il faut donc se placer dans l'ensemble grand canonique et introduire le potentiel chimique μ.

$$\widetilde{H}_{LG} = H_{LG} - \mu N = -\frac{J}{4} \sum_{\langle i,j \rangle} \sigma_i \sigma_j - \left(\frac{qJ}{4} + \frac{\mu}{2} \right) \sum_i \sigma_i - \frac{N}{2} \left(\frac{qJ}{4} + \mu \right)$$

Donc nous trouvons finalement

$$\widetilde{H}_{LG} = H_{ising} - \frac{N}{2} \left(\frac{qJ}{4} + \mu \right)$$

En posant $J_{ising} = \frac{J_{LG}}{4}$ et $B = \frac{qJ}{4} + \frac{\mu}{2}$
On peut alors calculer la fonction de partition :

$$\Xi_{LG}(\mu) = \sum_N Z_{LG}(N) e^{\beta \mu N} = \sum_{\{n_i\}} e^{-\beta (H_{LG} - \mu N)} = \sum_{\{n_i\}} e^{-\beta \widetilde{H}(\{n_i\})}$$

Ou bien même avoir une relation simple entre avoir une relation simple avec la fonction de partition canonique du modèle d'Ising.

$$\Xi_{LG}(T, \mu, N) = Z_{ising} e^{\frac{BN}{2} \left(q \frac{J}{4} + \mu \right)}$$

On voit donc que la fonction de partition grand canonique du gaz sur réseau est isomorphe à la fonction de partition canonique du modèle d'ising sous champ magnétique et que nous pouvons créer des liens entre les différentes constantes. Par exemple, l'aimantation m peut être reliée à $2\rho - 1$, J_{ising} à $\frac{J_{LG}}{4}$, B à $q\frac{J}{4} + \frac{\mu}{2}$. De ce fait, nous pouvons assumer que la température critique T_c est la même dans le modèle d'ising et dans le modèle du gaz sur réseau. De plus, dans la question on nous stipule de trouver la valeur analytique de T_c pour $\rho = 1/2$.

Nous pouvons donc nous intéresser à la résolution analytique du modèle d'ising en 2D, il existe plusieurs versions, toutes difficiles. Une de ces approches a été développée par Kac et Ward (1952), puis affinée par Potts et Ward (1955), dans laquelle des arguments combinatoires ont été utilisés pour exprimer la fonction de partition du système en tant que déterminant d'une certaine matrice A.

Une autre solution combinatoire, généralement considérée comme la plus simple, a été obtenue par Vdovichenko (1965) et Glasser (1970). Pour un compte rendu exhaustif du modèle d'Ising bidimensionnel, voir McCoy et Wu (1973).

La résolution exacte du modèle d'Ising en 2D nous donne comme température critique la valeur $T_c = 2.269 J/k_b$. Dans les simulations au-dessus, nous avons pris comme valeurs J=1 et kb=1 et nous retrouvons bien environ 2.2 comme valeur de température critique.

3.2 Question 2

2. Étudier les fonctions de corrélations temporelles et en déduire le temps typique nécessaire pour atteindre l'équilibre thermodynamique.

3.2.1 Première approche

Pour étudier les corrélations temporelles, nous avons plusieurs choix. Tout d'abord, nous allons étudier les corrélations d'une case avec elle même. Pour cela nous allons étudier la formule suivante.

$$g_{ij}^{dt} = \sum^{t} tab_{i,j}^{t} * tab_{i,j}^{t+dt}$$

Dans notre cas, il est plus simple de considérer t fixé à 0 et de moyenner sur un grand nombre de particules, cela nous fera gagner un temps considérable pour faire tourner notre programme. On rappelle que pour un processus ergodique, les moyennes statistiques et temporelles sont les mêmes.

Notre nouvelle fonction de corrélation est donc:

$$g_{ij}^{dt} = \sum^{N_s} tab_{i,j}^{0} * tab_{i,j}^{dt}$$

Nous pouvons maintenant faire deux choix, considérer soit les valeur de $tab_{i,j}$ comme étant 0 et 1 ou bien 1 et -1. Le fait de considérer -1 au lieu de 0 nous permet de prendre une case vide en tant que $tab_{i,j}^{0}$, en effet, si cette valeur est 0 alors notre fonction de corrélation nous donnera 0 pour chaque valeur de dt. Si nous avons -1 en revanche, on pourra voir qu'une case vide est corrélée avec elle-même (nous donne $-1 * -1 = 1$ et anti-corrélée avec une case remplie (nous donne $-1 * 1$).

Les résultats vont être très similaires, voici ce qu'on a lorsqu'on prend comme particule 0 ou 1.

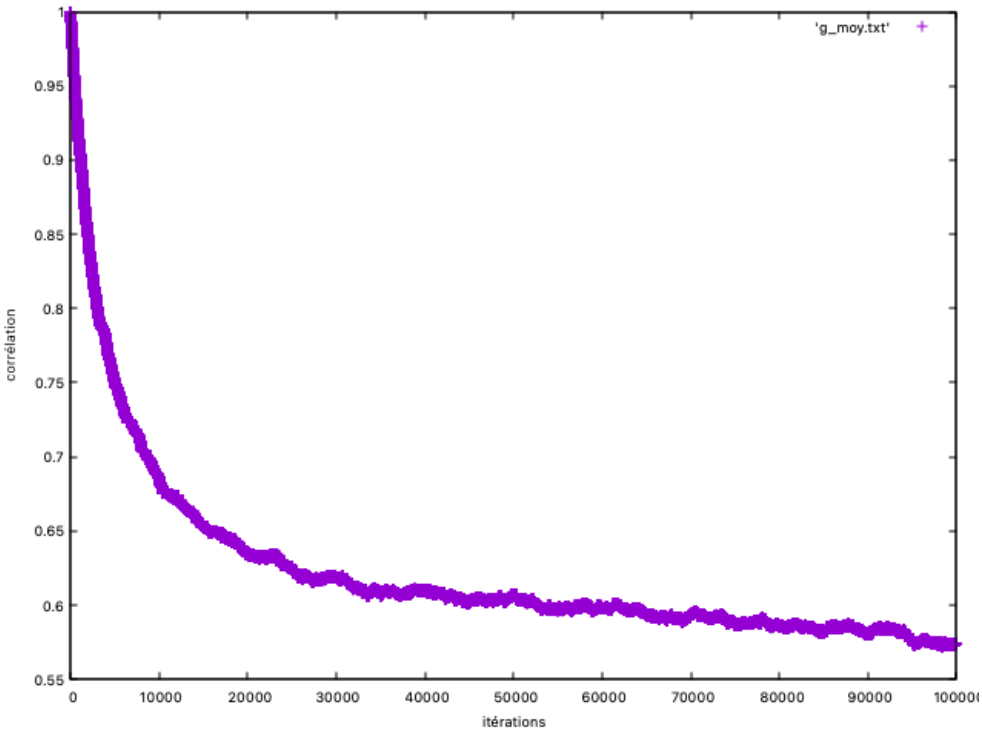

Figure 9: *Corrélation temporelle en fonction du nombre d'itérations sur une grille de 80*80 avec* $\rho = 1/2$.

Nous voyons sur la figure 9 que la fonction de corrélation se stabilise petit à petit autour de 0.5. Cela signifie qu'au bout d'un certain nombre d'itérations, la case a quasiment autant de probabilité d'être un 1 qu'un 0 même si elle était un 1 au départ. Regardons maintenant la corrélation lorsque l'on prend 1 et -1.

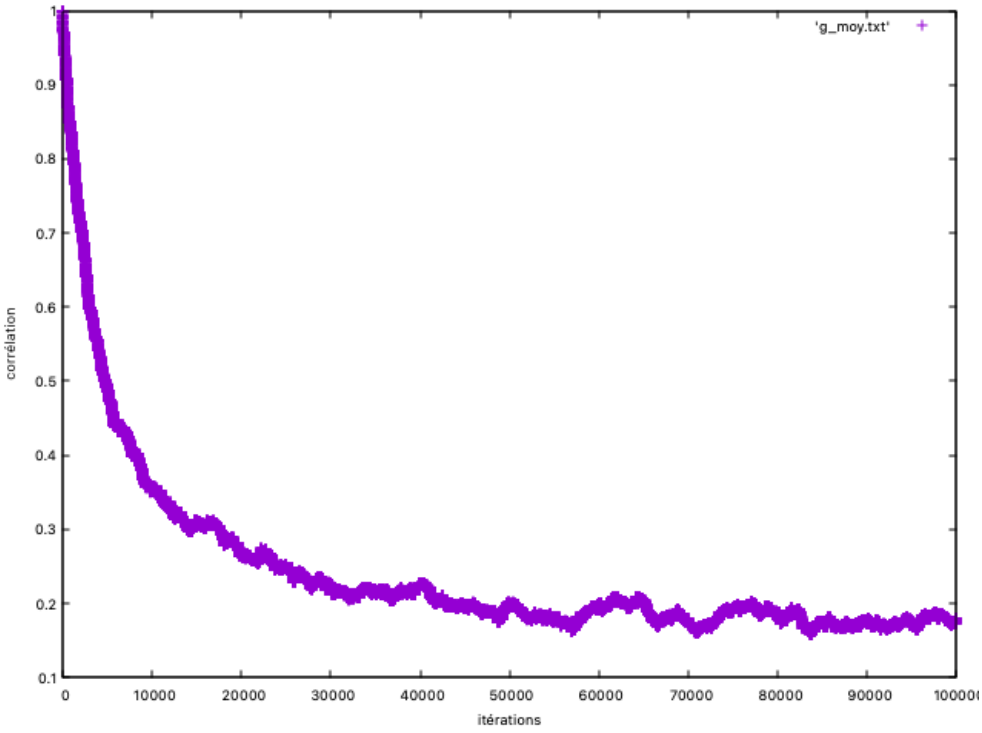

Figure 10: *Corrélation temporelle en fonction du nombre d'itérations sur une grille de 80*80 avec* $\rho = 1/2$ *et en prenant les valeurs 1 et -1.*

Nous voyons effectivement sur la figure 10 la même courbe. Cette fois-ci, l'équiprobabilité entre un 1 et un 0 se passe lorsque nous sommes à une valeur de 0 sur la corrélation. Il faut noter que nous avons initialement pris des cases soit libres soit remplies.

Nous pouvons donc tirer de ces graphiques la durée qu'il nous faut pour être à l'équilibre c'est-à-dire pour que le système se soit stabilisé et que la plupart des cases aient été changées au moins une fois. Forcément, cette durée dépend en partie de la densité car si on a une densité de 1, ou de 0, aucune case ne va changer (soit toutes remplies soit toutes vides) et on va avoir une courbe de corrélation parfaitement plate. La valeur de la densité va aussi affecter la hauteur du plateau que nous allons atteindre.

Le plus gros problème que j'ai eu était pour arriver à analyser la fonction de corrélation finale. Pour cela, je devais essayer de fiter la fonction et en récupérer ses constantes. J'ai alors trouvé sur internet une bibliothèque, la bibliothèque GSL, dont on a besoin pour faire tourner mon code. Je ne sais pas si les bibliothèques sont interdites pour cet UE mais je ne pouvais pas avancer sans.

L'algorithme que j'utilise pour fiter se base sur la méthode Levenberg-Marquardt avec accélération géodésique. J'ai d'abord récupéré le code du fit sur internet que j'ai ensuite remanié pour coller à ma fonction de corrélation. En effet, au début le code était fait pour fiter une gaussienne, ce que j'ai essayé de faire mais comme nous pouvons le voir notre fonction n'arrive jamais à 0 et donc je n'ai pas pu avoir un beau fit. Pareil pour une exponentielle décroissante, la courbe n'arrivait pas à 0. Mon premier réflèxe a été de soustraire le seuil où arrive ma fonction de corrélation, c'est-à-dire la densité $N_s/(X * Y)$. Le problème étant que maintenant ma fonction ne valait pas 1 en 0 donc mon fit d'exponentielle décroissante ne marche pas non plus. J'ai donc dû reprendre leur code depuis le début et j'ai changé le fit pour pouvoir jouer sur 3 paramètres a, b et c. En assumant que ma fonction est de la forme $a * e^{-bx} + c$ [2].

Après avoir fait tourner mon code, je peux enfin récupérer la valeur de b qui m'indique comment diminue ma fonction et donc me donne une idée du temps de relaxation et la valeur de c me donne seulement le seuil qu'atteint ma fonction.

Sur la figure 11, nous pouvons voir que la fonction de fit ne colle pas exactement à notre fonction de corrélation mais c'est suffisant pour récupérer les informations dont nous avons besoin. Je vais maintenant tracer la valeur de b, qui peut correspondre à la durée à attendre pour atteindre l'équilibre en fonction de ρ.

Nous remarquons sur la figure 12 que la valeur de $\frac{1}{b}$ augmente en fonction de la densité. En effet, plus on augmente la densité, plus il faut du temps pour que chaque particule ait changé de valeur assez de fois. A une densité trop élevée en revanche je ne suis pas sûr qu'on peut associer l'équilibre thermodynamique à cette valeur étant donné que à une densité trop élevée on peut considérer avoir des zones denses sans avoir fait tourner l'algorithme.

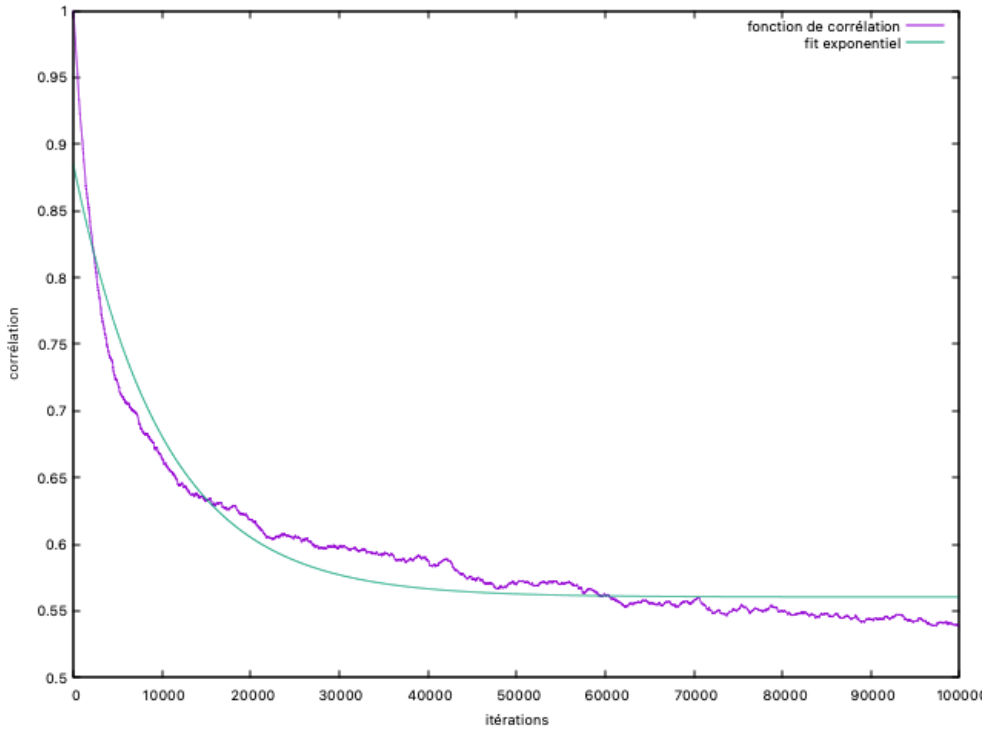

Figure 11: *Fit en exponentielle décroissante de la corrélation temporelle, pour Ns = 3000, N=80.*

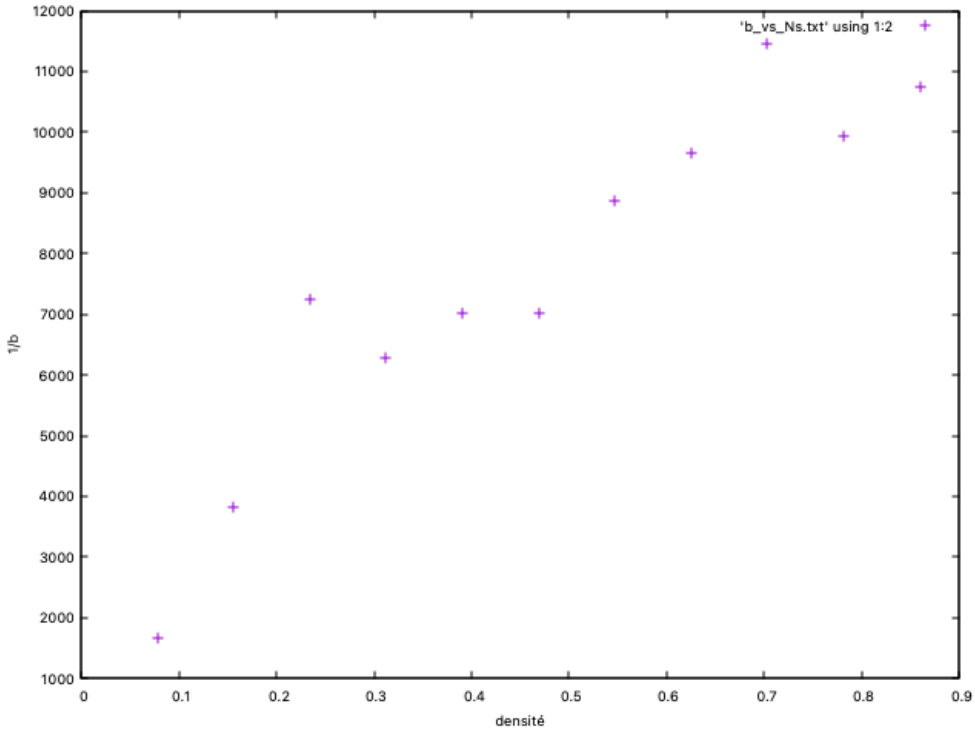

Figure 12: $\frac{1}{b}$ *en fonction de la densité* $\rho = \frac{Ns}{X*Y}$.

3.2.2 Idée de seconde approche

Une seconde approche pourrait être de regarder la valeur de la fonction de corrélation sur la courbe d'énergie de notre système.

3.3 Question 3

3. Étudier les fonctions de corrélations spatiales, notamment lorsqu'on s'approche de Tc.

La fonction de corrélation spatiale va nous renseigner plusieurs choses, elle peut être vue comme la probabilité de trouver une autre particule à une certaine distance d'une particule. On peut la définir comme suivant:

$$g^r_{k,l} = \sum_{n=1}^{N_s} tab_{i,j} * tab_{i+k,j+l} * \frac{1}{Ns}$$

Avec i et j les coordonnées de la particule n. On voit que j'ai seulement multiplié la valeur de la case (i,j) par la valeur de la case (i+k,j+l).

On peut essayer de faire varier la température pour voir ce que cette fonction peut nous donner, en prenant une température en dessous de la température critique, une égale à T_c et une plus grande que T_c.

On se place encore à $N = 80$ et avec 1000 particules. Le nombre d'itérations est ici de 100 000.

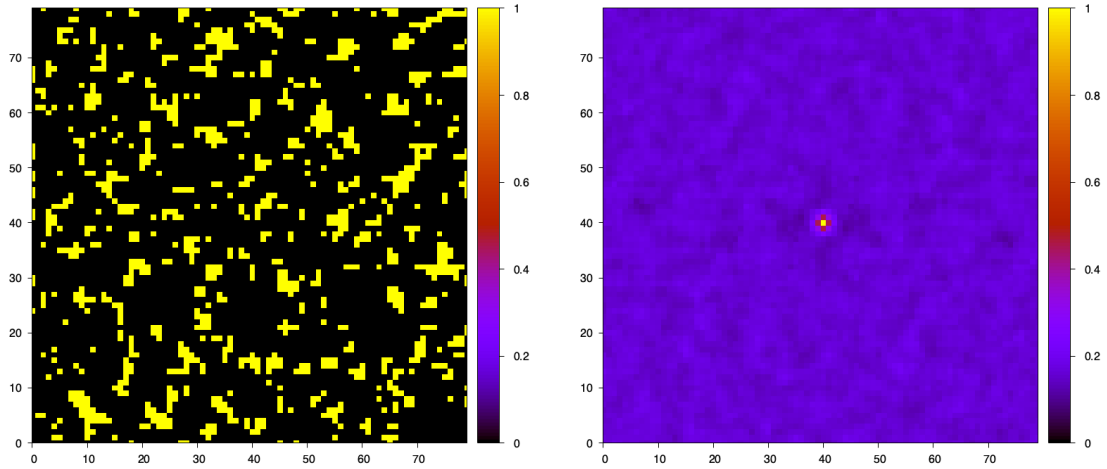

Figure 13: *Grille et fonction de corrélation spatiale pour N = 80; Ns = 1000 et 100 000 itérations à une température de 1*

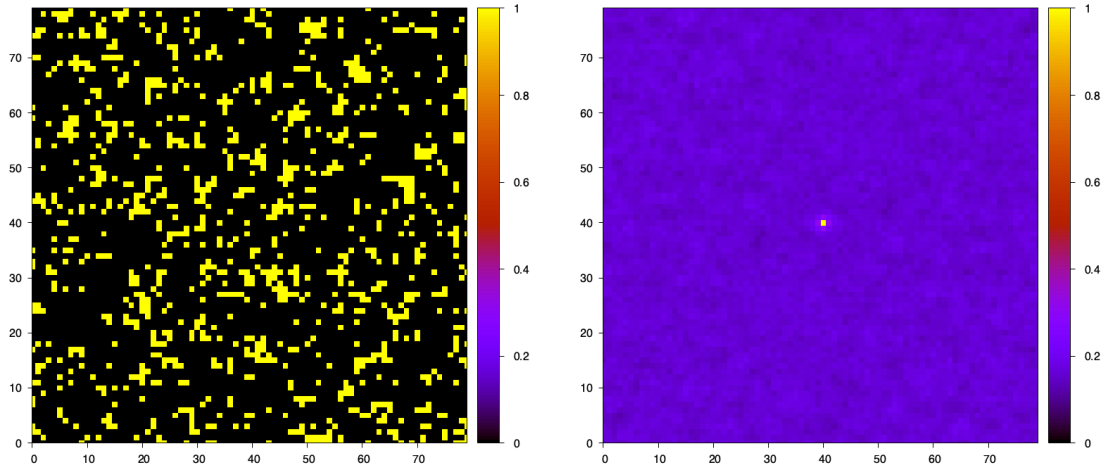

Figure 14: *Grille et fonction de corrélation spatiale pour N = 80; Ns = 1000 et 100 000 itérations à une température de 2*

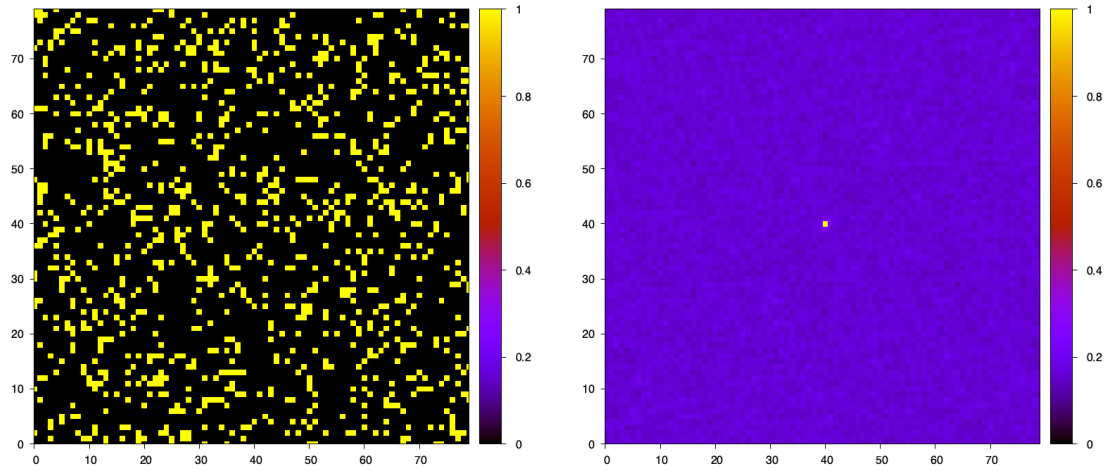

Figure 15: *Grille et fonction de corrélation spatiale pour N = 80; Ns = 1000 et 100 000 itérations à une température de 5*

Nous remarquons que plus on augmente la température et plus les valeurs centrales sont faibles. Il faut noter que la particule centrale vaudra toujours 1 car on la multiplie avec elle-même. Cependant, plus on augmente la température plus on retombe vite dans du "bruit". On peut remarquer une tâche centrale qui se démarque, qui est assez étalée dans la première image, peu dans la seconde et quasiment inexistante dans la dernière. La taille de cette tâche nous informe sur la taille moyenne des clusters, on peut essayer d'analyser la taille de cette tâche en fonction de la température.

On peut donc jouer avec notre fonction de corrélation et la première chose que j'ai effectué est de passer notre corrélation en 1D, je prends simplement tous les points de notre tableau, je détermine leur distance au centre puis j'ajoute leur valeur à un tableau $g_{1D}(r)$, une fois tout le tableau complété je divise chaque valeur de r par le nombre de points qui a contribué dans la somme pour avoir des valeurs normalisés. Nous nous retrouvons finalement avec un tableau 1D que nous pouvons stocker dans un fichier. La figure 16 représente la carte 2D ainsi que sa version 1D en fonction de la distance.

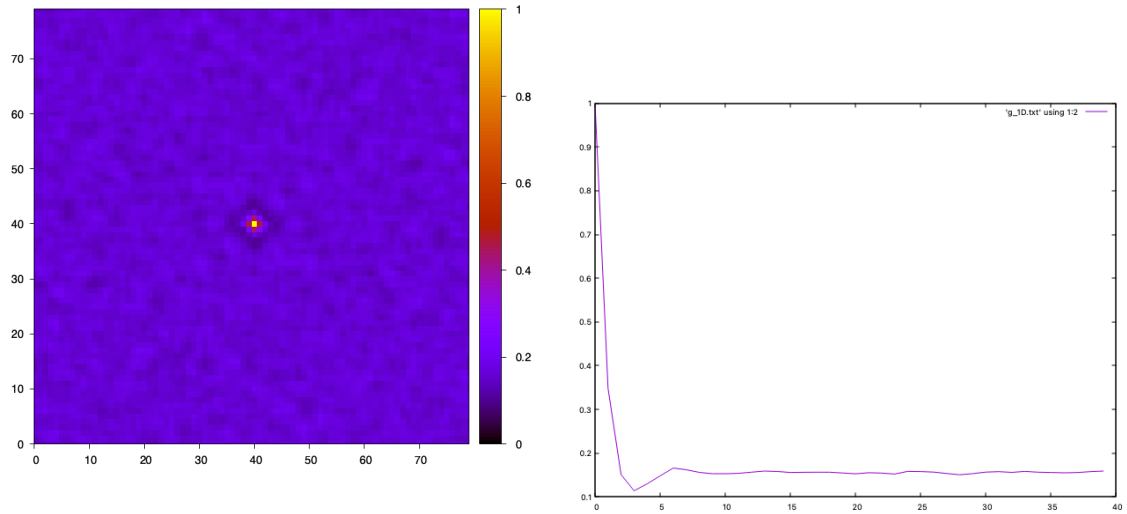

Figure 16: *Fonction de corrélation en 2D et en 1D pour T=0.001.*

Nous pouvons fiter notre fonction 1D pour mieux voir apparaître les spécifications de notre fonction, j'ai essayé de la fiter avec deux fonctions différentes. D'abord, j'ai voulu la fiter avec une gaussienne, mais dans la documentation j'ai trouvé que la fonction associée était plutôt du genre exponentielle décroissante $e^{-r/L}$, ce qui nous donnerait directement accès à la longueur de corrélation L. La figure représente des images de $g_{1D}(r)$ fitées soit avec une gaussienne soit avec

une exponentielle décroissante.

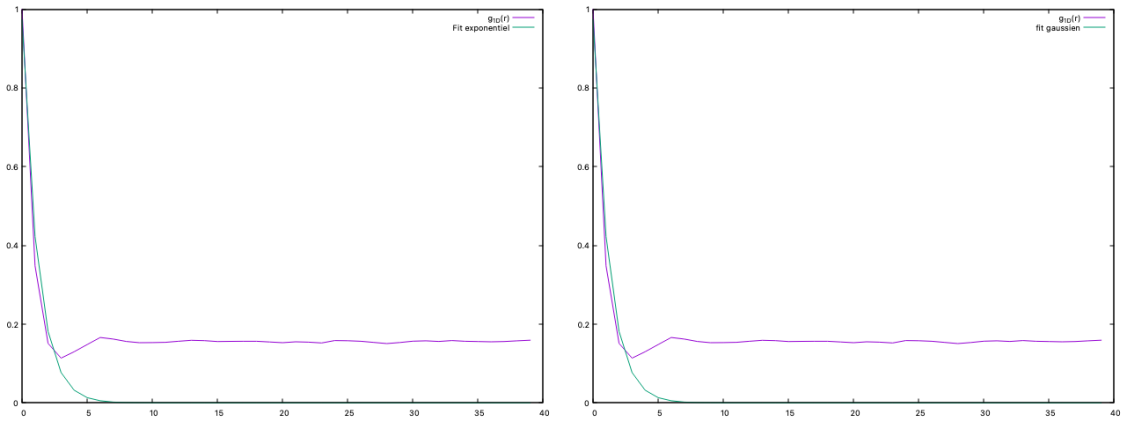

Figure 17: *Fit par une exponentielle décroissante (à gauche) et par une gaussienne (à droite) de la fonction de corrélation 1D*

Nous remarquons que les deux fit sont quasiment aussi prècis, en effet, nous pouvons avoir la fonction moyenne de coût, qui représente à quel points nous sommes loin de notre modèle.

$$C(x) = \frac{1}{2}\sum_i f_i^2$$

Avec

$$f_i = y_i - Y(a, b, c, r)$$

Où les y_i sont tous nos points de notre fonction (ici la fonction g_{1D}) et les points Y sont le modèle, c'est à dire notre exponentielle ou notre gaussienne. Il faut noter que Y prend ici 4 arguments qui seraient la moyenne, l'amplitude et la variance pour une fonction gaussienne, et bien sûr la position, et on aurait seulement 2 arguments pour la fonction de l'exponentielle décroissante (L).

Notre algorithme utilise la méthode Levenberg-Marquardt avec une accélération géodésique. Le programme a une certaine fonction coût minimale à atteindre, lorsqu'il l'atteint, il s'arrête. S'il ne l'atteint pas il peut aussi décider de s'arrêter lorsque la fonction de coût n'a pas beaucoup varié ou lorsqu'on dépasse un certain nombre d'itérations. Dans notre cas, l'algorithme converge assez rapidement mais n'arrive pas à atteindre la fonction de coût minimale et s'arrête donc car la fonction coût s'est stabilisée. Cela provient probablement du fait que g_{1D} ne représente pas exactement une exponentielle décroissante ou même un gaussienne. En effet, nous pouvons voir un plateau atteint lorsque nous sommes à plus de 10 cases. De ce fait nous n'allons utiliser que les 10 plus proches voisins lorsque nous allons essayer de fiter notre fonction.

Nous pouvons remarquer que notre fonction coût finale pour notre exponentielle décroissante (1.372205715077e-01) et pour notre gaussienne (1.349438117700e-01) sont très similaires. Cependant lorsque nous considérons seulement les 10 premiers points, l'algorithme utilisant la fonction exponentielle décroissante converge bien plus vite que celui utilisant la gaussienne. Par soucis de rapidité nous allons donc considérer l'utilisation de l'exponentielle décroissante et utiliser L comme longueur de cohérence.

Voici pour différentes températures la longueur de cohérence obtenue T.

Nous pouvons voir que la largeur à mi-hauteur de notre longueur de cohérence se fait environ à la température critique, c'est-à-dire $2.269 J/kb$. Nous remarquons donc que, la longeur de cohérence étant reliée à la taille des clusters de notre modèle de gaz sur réseau, passé la température critique, la taille des clusters change brutalement pour atteindre un équilibre où nous n'avons plus du tout de clusters. Cette formation serait le moment où la longueur de cohérence tend vers 0.

3.4 Bonus : J négatif

Nous pouvons aussi essayer de regarder à quoi correspond notre corrélation spatiale lorsque nous prenons un J négatif. Lorsque J est négatif, les particules essaient de s'éloigner, pour diminuer leur énergie. Si l'on pense au modèle d'ising, un J positif représente un cristal anti-ferromagnétique. Voici quelques figures caractéristiques d'une constante J positive.

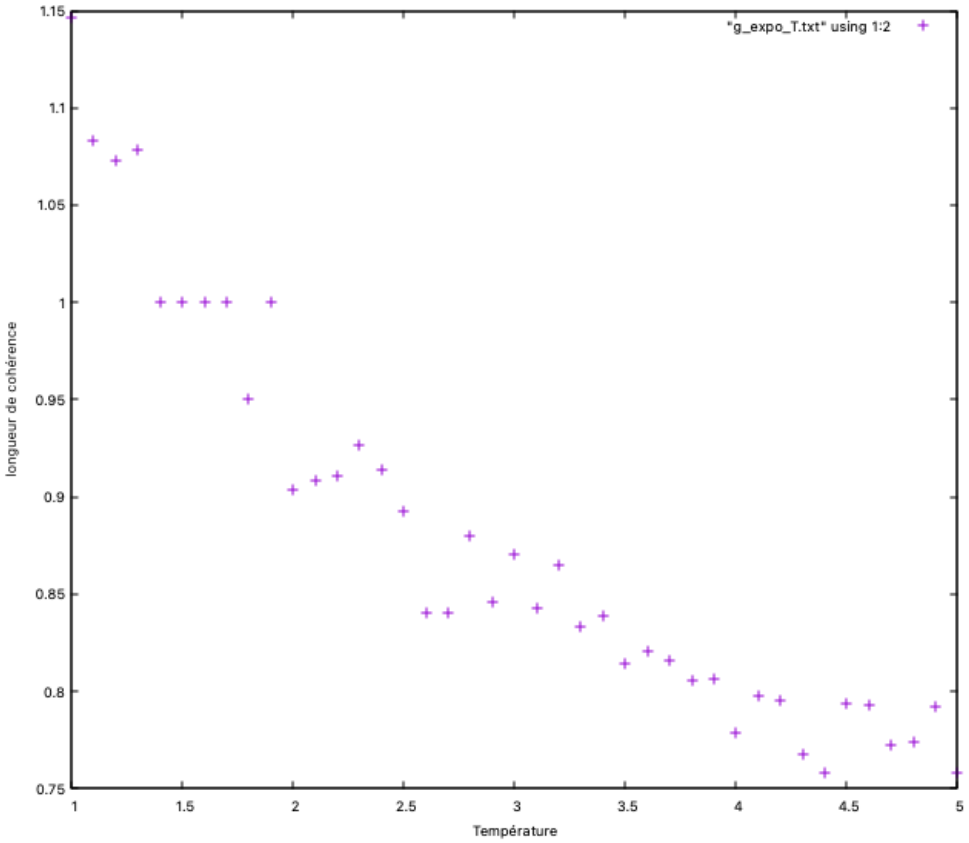

Figure 18: *Longueur de cohérence en fonction de la température*

Nous pouvons remarquer que le tableau (19) comme un échiquier, avec les particules qui refusent de se coller à leurs 4 voisins principaux. Et bien sûr, quand on rajoute de l'énergie en augmentant la température, on peut observer des zones condensées. On peut aussi regarder la fonction de corrélation spatiale 20.

Nous remarquons encore une fois une espèce d'échiquier qui se dessine dans la fonction de corrélation spatiale, en effet à une case sur deux, la probabilité d'avoir une particule est faible et à une case sur deux cette probabilité est forte.

A une certaine distance cependant, on a presque autant de probabilité d'avoir une particule ou une case vide. Ce qui fait que cette image est particulièrement marquée est le choix de $\rho = 1/2$ qui "oblige" une case éloignée de deux d'être occupée.

References

[1] DocPlayer. *Correspondance entre modèle de gaz sur réseau et modèle d'ising*. 2023. URL: https://docplayer.fr/189638413-Exercice-9-2-chaine-d-ising.html.

[2] GNU. *Bibliothèque pour faire un fit avec GSL*. 2022. URL: https://www.gnu.org/software/gsl/doc/html/nls.html#geodesic-acceleration-example-2.

[3] Lars Onsager. "Crystal Statistics. I. A Two-Dimensional Model with an Order-Disorder Transition". In: *Phys. Rev.* 65.3-4 (Feb. 1944), pp. 117–149. DOI: 10.1103/PhysRev.65.117. URL: http://link.aps.org/doi/10.1103/PhysRev.65.117.

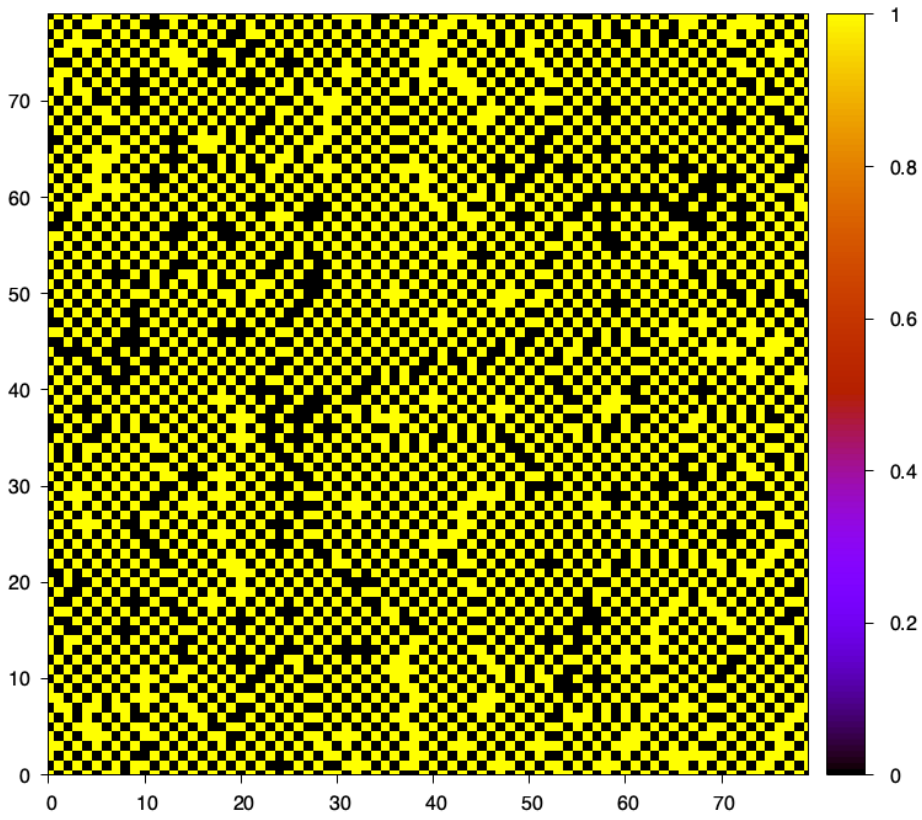

Figure 19: *Tableau principal avec J = -1, avec* $\rho = 1/2$.

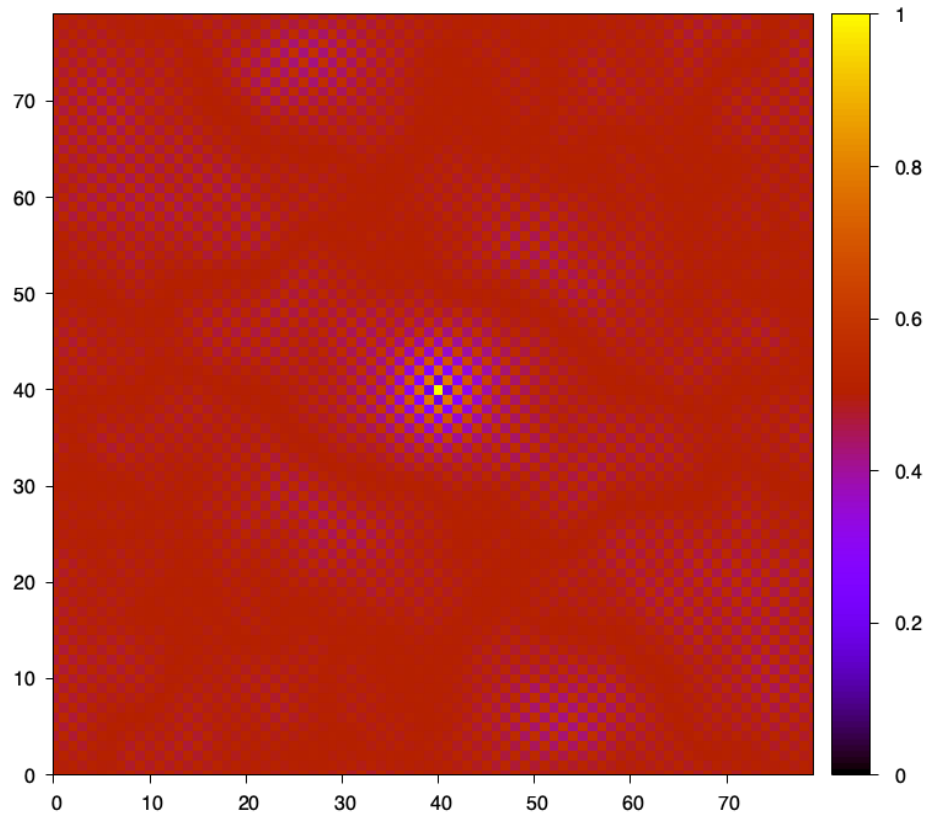

Figure 20: *Fonction de corrélation 2D avec J = -1 et* $\rho = 1/2$ *et à basse température.*