Simulation du modèle XY par méthode Monté-Carlo

Rapport de projet pour le cours de Computationnal Statistical Physics

Aurélien Valade

Introduction

La physique statistique est l'étude des comportements systèmes macroscopiques composés de systèmes microscopiques dont les propriétés sont connues. On va donc systématiquement s'intéresser à un grand nombre de particules en interactions, ce qui est complexe à représenter mathématiquement sans faire approximation de continuité, de champ moyen, etc. Ainsi, les comportements de nombreux modèles de physique statistique ne sont pas discriptibles analytiquement pour des cas pourtant pertinents. Si une résolution analytique n'est pas possible, il faut travailler avec des valeurs numériques, comme cela a par exemple était fait pendant la seconde guerre mondiale pour les calculs portant sur la bombe atomique, ou encore en astrophysique pour des calculs de trajectoires. Cependant la quantité de calcul nécessaire pour résoudre des systèmes réalistes est tout bonnement inhumaine.

L'informatique nous offre une capacité de calcul immensément plus grande que celle des calculatrices humaines, puisqu'on passe d'une dizaine d'opérations par minute à des milliards par seconde. Mais ce n'est toujours pas suffisant pour attaquer le problème de front, par exemple, calculer directement la fonction de partition du système reste souvent hors de portée. Il nous faut donc trouver des méthodes plus puissantes pour résoudre le système. Cette formalisation à cependant un prix puisque de telles méthodes sont alors plus spécifiques : une méthode de résolution est adaptée à une certaine question, mais ne permet pas toujours de bien répondre à d'autres. On pourra ainsi rapidement calculer des propriétés d'un système à l'équilibre sans pouvoir décrire la dynamique physique de la relaxation. Réciproquement, une autre méthode peut permettre de décrire la dynamique du système avec précision, mais demande une très grande quantité de calcul pour atteindre l'état d'équilibre.

Nous allons ici nous pencher sur l'étude du modèle XY, qui est un modèle d'interaction sur réseau se rapprochant du modèle d'Ising, à la différence que les états pris par les spins sont continus entre -1 et +1 et non restreints aux valeurs $\{-1, +1\}$. Il serait cependant faux de dire que le modèle XY est une généralisation du modèle d'Ising puisque pour un système équivalent, aucun choix de paramètres du modèle XY ne permet de reproduire des comportements semblables à ceux conjecturés par modèle d'Ising. Le modèle XY est donc adapté à d'autres systèmes que le modèle d'Ising : il permet notamment de décrire des systèmes dont le paramètre d'ordre ne correspond pas à une brisure de symétrie spatiale lors de la transition de phase, mais plutôt à une brisure de la symétrie interne, comme c'est par exemple le cas pour l'Helium II supercritique ou encore les cristaux liquides hexatiques [1].

Nous allons dans un premier temps présenter le problème dans sa formulation physique.

1. La phase hexatique est une phase située entre les phases solide isotropique et liquide isotropique dans les systèmes de particules à deux dimensions (cristal liquide bidimensionnel). Il est caractérisés par deux paramètres d'ordre : un ordre moléculaire de positionnement à courte portée et un ordre orientationnel à longue portée [Wik17].

1 Modèle physique

1.1 Description du modèle

Soit un ensemble de particules arrangées sur une grille carrée de taille $N \times N$. On note la position $\boldsymbol{r}ij$ pour la particule de la ligne $0 \leq i < N$ et la colonne $0 \leq j < N$. On utilise aussi l'indiciation des particules $0 \leq I < N \times N$ où la position spiatiale n'apparaît pas explicitement bien qu'on puisse définir $I = i \times N + j$.

Ces particules ont une propriété appelée spin que l'on peut lier à l'angle entre la normale au plan et un axe intrinsèque à la particule. On note cet angle $\theta_{ij} \in [0, \pi]$. On peut alors définir le spin de chaque particule $\sigma_{ij} = \cos(\theta_{ij})$. On considère que ces particules interagissent entre elles à travers leur spin, et qu'il existe un champ exterieur avec lequel ces spin tendent à s'aligner. Le Hamilonien décrivant l'énergie d'un tel système s'écrit alors :

$$\begin{aligned} H &= -k \underbrace{\sum_{I,I'} J_{II'} \cos(\theta_I - \theta_{I'})}_{\text{interactions}} - h \underbrace{\sum_I \cos(\theta_I)}_{\text{champ exterieur}} \\ &= -kQ - hM \end{aligned} \tag{1}$$

avec

$$Q = \sum_{I,I'} J_{II'} \cos(\theta_I - \theta_{I'}) \ , \qquad M = \sum_I \cos(\theta_I) \tag{2}$$

nommées respectivement le couplage et la magnétisation par analogie aux systèmes magnétiques.

La forme de la matrice J est centrale au problème. Le modèle XY préconise de prendre une fonction de la distance entre les deux particules : $J_{II'} = (\boldsymbol{r}_I - \boldsymbol{r}_{I'})^{-\alpha}$. On peut ensuite choisir de n'appliquer cette définition qu'aux n plus proches voisins et de négliger les interactions à longue distance :

$$J_{II'} = \begin{cases} (\boldsymbol{r}_I - \boldsymbol{r}_{I'})^{-\alpha} & \text{si } |i - i'| < n \text{ et } |j - j'| < n, \\ 0 & \text{sinon.} \end{cases} \tag{3}$$

Les paramètres h et k vont nous permettre permettent d'adapter ce système à différents cas concrets. La valeur de h représente l'intensité du champ externe auquel les spins tendent à s'aligner, et k modélise la force du couplage entre les spins. Pour un modèle magnétique, $k > 0$ signifie que le matériau est ferromagnétique : les spins cherchent à pointer dans la même direction et $k > 0$ représente un matériau paramagnétique : les spins cherchent à pointer dans une direction opposée à celle de leurs voisins. Ces quantités ont été préalablement adimensionnées, et dérivent de termes de la forme $h = h'/(\not{k}_B T)$, $k = k'/(\not{k}_B T)$ où $\not{k}_B$ est la constante de Boltzmann et T la température. C'est donc à travers ces paramètres qu'on peut jouer sur la température pour observer les transitions de phase, etc.

1.2 Comportement prédit

A haute température les paramètres h et k du système sont très faibles, le Hamiltonien résultant est donc quasi-nul et n'a pas de minimum significatif. Il n'y a pas d'état qui puisse réellement minimiser l'énergie, tous sont donc équiprobaux, il en résulte que la magnétisation ainsi que le couplage sont nuls en moyenne.

À basse température, le comportement du modèle XY est différent du modèle d'Ising. Tout d'abord, lors de la transition de phase, on n'observe pas de brisure de symétrie spatiale. A champs magnétique externe nul $h = 0$, lors d'une trempe (*i.e.* lors qu'on abaisse progressivement la température du système), la magnétisation reste nulle. En revanche, en présence d'un champ externe, la symmétrie du système étant déjà brisée, et les spins s'alignent effectivement avec ce champ.

On observe à la transition de phase un changement globale de comportement de la fonction de corrélation

$$C_{II'} = \left|\left\langle \sigma_I \sigma_{I'} \right\rangle\right|. \tag{4}$$

En effet, à haute température, la fonction de corrélation décroît exponentiellement :

$$C_{II'}^{HT} \propto e^{\Gamma(\beta)|\boldsymbol{r}_I - \boldsymbol{r}_{I'}|} \tag{5}$$

alors qu'à basse température, on s'attend à trouver une fonction qui décroît moins vite que

$$C_{II'}^{BT} \propto \frac{\Gamma(\beta)}{1 + |\boldsymbol{r}_I - \boldsymbol{r}_{I'}|}. \tag{6}$$

2 Implémentation

Le but de ce travail est de fournir un programme puissant, flexible, et aisément utilisable qui permette de simuler et d'analyser un modèle XY classique par une méthode Monté-Carlo. Il doit notamment utiliser un système de production, de stockage et d'analyse des données cohérent qui permette de reproduire chaque experience. Il faut donc notamment ne pas écraser les données au fur et à mesure qu'elles sont produite, mais il faut en plus garder des traces des paramètres associés à chaque experience produite, ainsi que des conventions de nomages compréhensibles et cohérentes.

2.1 Structure du code et des scipts

Le code produit se décompose en deux parties distinctes :
— le code principal d'environs 500 lignes, écrit en C : il produit les données pour une expérience et les stocke dans un dossier ;
— et un code de scirpting écrit en Python d'environs 600 lignes : celui ci permet d'appeler le code C avec les bons paramètres puis d'exploiter les données qui en résultent.

2.2 Fonctionnement du code principal

2.2.1 Entrés et sorties

Chaque expérience est lancée avec la connaissance des paramêtres suivant :
— N la taille de grille ;
— n le nombre de plus proches voisins à considérer pour les intéractions ;
— a l'exposant de la fonction de décroissance pour les intéractions ;
— h la force du champ exterieur ;
— k la force de couplage entre spins ;
— le type d'initialisation :
  — `UP` : $\theta_I = 0$ pour tout I ;
  — `DOWN` : $\theta_I = \pi$ pour tout I ;
  — `CHECKERBOARD` : $\theta_{ij} = \pi|\mathrm{mod}(i,2) - \mathrm{mod}(j,2)|$ pour tout i, j ;
  — `RANDOM` : $\theta_I = U([0,\pi])$ pour tout I avec U la loi uniforme.
— le type de Monte-Carlo : `METROPOLIS` ou `GLAUBER`.
— Ainsi que le nombre de pas, la fréquence de sauvegarde des différentes données et le répertoire où les stocker.

En sortie, le programme fournit les données suivantes :
— Un fichier où sont stockées les valeurs des quantités M, Q et E toutes les `n_sca` itérations ;
— Un fichier toutes les `n_lat` itérations, représentant l'ensemble des angles de la grille ;
— Un fichier toutes les `n_lat` itérations, représentant l'ensemble des interactions Q_I de la grille.

2.2.2 Definition des quantités numériques

Les quantités physiques introduites prédemment sont définies numériquement comme suit :

Magnétisation. La définition de la magnétisation est similaire à sa définition physique :

$$M = \sum_I \cos(\theta_I).$$

On a donc $M \in [-N \times N, N \times N]$.

Couplage. On introduit la matrice de couplage normalisée telle que

$$\hat{J}_{II'} = \frac{J_{II'}}{\sum_{KK'} J_{KK'}}.$$

Cette définition permet de garder un couplage indépendant du nombre de voisins considérés. En effet, en absence de normalisation, pour une configuration donnée, l'énergie de couplage de chaque particule grandit fortement avec le nombre de plus proches voisins considérés, ce qui peut s'avérer gênant pour comparer des simulations à n différent. On introduit ensuite le couplage local

$$Q_I = \sum_{I'} \hat{J}_{II'}(\cos\theta_I \cos\theta_{I'} - \sin\theta_I \sin\theta_{I'})$$

tel que $Q = \sum_I Q_I$, et dont les valeurs qui sont stockées sur la grille des interactions introduite plus haut. Développer le cosinus en une différence de produits de cosinus/sinus permet de stocker les valeurs de cosinus et de sinus pour chaque angle et de calculer un minimum de fonctions trigonométriques à chaque mise à jour. Avec la normalisation de la matrice de couplage, on a aussi $Q \in [-N \times N, N \times N]$.

Énergie. On définie l'énergie du système

$$E = -kQ - hM.$$

Quantitées réduites. Pour pouvoir comparer des experiences dont la taille de la grille diffère, on utilise les valeurs réduites :

$$m = M/N^2, \quad q = Q/N^2, \quad e = E/N^2.$$

Ces valeurs sont particulièrement stables puisqu'on a $m, q \in [-1, 1]$ et $e \in [-|h| - |k|, |h| + |k|]$ pour toute expérience, quelque soient ses paramètres.

2.2.3 Algorithmes utilisés

Après initialisation des différentes parties du code, on procède à l'exploration de l'espace des phases par méthode de Monté-Carlo comme vue en cours. À chaque itération, une cellule de la grille est choisie au hasard à laquelle on assigne un angle aléatoire. On recalcule l'énergie de la nouvelle configuration. On propose ensuite deux méthodes

- `METROPOLIS` : la modification est acceptée avec une probabilité de $\min(1, \exp(-\Delta E))$, sinon elle est rejetée ;
- `GLAUBER` : la modification est acceptée avec une probabilité de $1/(1 + \exp(\Delta E)$, sinon elle est rejetée.

La procédure est décrite en pseudo-code dans la figure 1.

Une fois le spin d'une particule mis à jour, il faut mettre à jour toutes les quantitées, a savoir M, Q et E. Il s'agit de la partie du code qui demande le plus de calcul. Plus précisément, c'est le calcul du couplage qui est le plus lourd, et l'est d'autant plus que l'on considère les actions à longue distance.

Une approche naive serait de changer recalculer la magnétisation et du couplage après chaque modification sur toute la grille ; une telle approche serait de complexité $\mathcal{O}(n^2 \times N^2)$. Une meilleur algorithme, celui-ci en $\mathcal{O}(n^2)$ est décrit en figure 2. L'idée consiste à ne recalculer que la partie de grille affectée par le changement et de reporter directement les modifications sur les scalaires M et Q. Le poids final de chaque opération est reporté en figure 3.

```
i ← int_uniform_random(0, N − 1)                          ▷ ligne à modifier
j ← int_uniform_random(0, N − 1)                          ▷ colonne à modifier
a ← uniform_random(0, π)                                  ▷ nouvel angle
p ← uniform_random(0, 1)                                  ▷ probabilité d'acceptation

b ← get_lattice_value(lattice, i, j)                      ▷ sauvegarde de l'angle
E ← get_energy(lattice)                                   ▷ sauvegarde de l'énergie

set_lattice_value(lattice, i, j, a)                       ▷ assignation du nouvel angle
compute_energy(lattice)                                   ▷ calcul de l'énergie dans la nouvelle configuration
ΔE ← get_energy(lattice) −E

if monte_carlo=METROPOLIS then
    if exp(−ΔE) < p then                                  ▷ si le pas est rejeté
        set_lattice_value(lattice, i, j, b)               ▷ réassignation de l'ancienne valeur d'angle
        set_energy(lattice, E)                            ▷ réassignation de l'ancienne valeur d'énergie
    end if
else if monte_carlo=GLAUBER then
    if 1/(1 + exp(ΔE)) < p then                           ▷ si le pas est rejeté
        set_lattice_value(lattice, i, j, b)               ▷ réassignation de l'ancienne valeur d'angle
        set_energy(lattice, E)                            ▷ réassignation de l'ancienne valeur d'énergie
    end if
end if
```

FIGURE 1 – Procédure de mise à jour de la grille en pseudo code.

2.3 Fonctionnement des scripts

2.3.1 Script de lancement du code principal

Ce script est un "wrapper" du code principal. Il simplifie l'appel par le nommage des variables, par le passage des valeurs par défaut, etc [2].

Il permet aussi de lancer plusieurs types d'experiences :

— **Simple sampling.** Le script permet alors de nommer automatiquement les dossiers de stockage, lance bien sûr automatiquement les N_{simple} experiences et compiles tous les fichiers de scalaires en un seul pour une analyse postérieur. Le programme principal est appelé avec un nombre d'itérations à faire nul. Il s'initialise donc avec une grille aléatoire, calcul son énergie, enregistre la grille ainsi que ses valeurs scalaire et quitte.

— **Sweep.** Il s'agit de parcourir un ensemble de paramètres et de lancer une simulation pour chaque valeur. Contrairement à ce qui était prévu initialement, on se concentre sur des ensembles de couples (h, k) : on peut donc lancer automatiquement $n_h \times n_k$ simulation avec $\{(h_i, k_j)\}_{h_0<\ldots<h_{n_h}, k_0<\ldots<k_{n_k}}$. En plus de lancer les experiences automatiquement, le script nomme automatiquement les dossiers de stockages.

— **Parallel sweep.** Même chose pour que pour le sweep, mais les intances du code principale s'executent en parallèle. Autant de processus qu'il y a de simulations à faire sont créés, et on laisse l'OS partager le calcul, ce qui n'est pas optimal mais qui est plus rapide à mettre en place.

2.3.2 Scipts d'analyse des données

Ces scripts prennent des dossiers de données formattés par les scripts de lancement présentés précedemment. Pour une seule experience, plusieurs visualisations sont possibles :

2. On utilise le package python `argparse` pour le passage des arguments, plus simple à gérer que les fonctions POSIX en C.

Require: a, b la nouvelle/ancienne valeur de l'angle modifié
Require: i, j la position de l'angle modifié
Require: M, Q, E la magnétisation, le couplage et l'énergie

$M \leftarrow M - \cos(b) + \cos(a)$ ▷ Mise à jour de la magnétisation

$p \leftarrow 0,\ q \leftarrow 0$ ▷ Ancienne/nouvelle valeur du couplage local
for i', j' in the neighborhood of i, j **do** ▷ Boucle sur les particules en interaction
    $p \leftarrow p +$ get_interaction_value(i', j') ▷ Ancienne valeur
    compute_interaction_one_particule(i', j') ▷ Mise à jour couplage local
    $q \leftarrow q +$ get_interaction_value(i', j') ▷ Nouvelle valeur
end for
$Q \leftarrow Q - p + q$ ▷ Mise à jour du couplage
$E \leftarrow -kQ - hM$ ▷ Calcul de l'energie

FIGURE 2 – Procédure de mise à jour de l'énergie en pseudo-code.

```
Children        Self        ...       Symbol
...
+   81.17%     0.41%        ...       update
+   75.87%     2.28%        ...       calc_all_one_change
+   67.41%    22.76%        ...       calc_inter_one_spin
+   14.56%    14.56%        ...       get_cos
+   12.19%    12.19%        ...       get_sin
+   11.88%    11.88%        ...       add_inter
+    8.29%     8.29%        ...       periodic_conditions
+    7.77%     7.77%        ...       get_dist
...
```

FIGURE 3 – Resultats de `perf` sur le surcoût de calcul pour le programme principal. La fonction `update` est décrite en figure 1 et la fonction `calc_all_one_change` est décrite en 2. Les fonctions `get_*` ne servent qu'à lire les valeurs des différents tableaux, `add_inter` ajoute une valeur donnée sur une partie de la grille des couplages locaux, et `periodic_conditions` fait la transformation d'indice qui simule les conditions periodiques.

Require: prod_shift(lattice, i, j) qui retourne un tableau tel que $A_{kl} = B_{kl}B_{(k+i)(j+l)}$ avec des conditions periodiques
Require: sum(lattice) qui ajoute toutes les valeurs d'une grille
Require: abs_sort(X, Y) tri les valeurs des listes X et Y rapport à X
$X \leftarrow \emptyset,\ Y \leftarrow \emptyset$
for i, j in $[0, N/4] \times [0, N/4]$ **do** ▷ Boucle sur le voisinage à $N/4$ dans une direction
push_item$(X, \sqrt{i^2 + j^2})$ ▷ Calcul du rayon
$S^{(++)}$ = prod_shift(lattice,i, j) ▷ Correlations par direction
$S^{(+-)}$ = prod_shift(lattice,$i, -j$)
$S^{(-+)}$ = prod_shift(lattice,$-i, j$)
$S^{(--)}$ = prod_shift(lattice,$-i, -j$)
push_item$\left(Y, \text{sum}\left(\frac{S^{(++)}+S^{(+-)}+S^{(-+)}+S^{(--)}}{4N^2}\right)\right)$ ▷ Moyenne des correlations directionnelles
end for
$X, Y \leftarrow$ abs_sort(X, Y)

FIGURE 4 – Procédure de calcul de la fonction d'autocorrelation.

— Tracé des scalaires m, q et e en fonction du nombre d'itération ;
— Tracé de q en fonction de m ;
— Tracé de grilles θ_{ij} (une seule, un ensemble, toutes, la dernière), si plusieurs grilles sont demandées, elles sont affichées sous forme d'une film à 5 fps par défault ;
— Tracé de la fonction de d'auto-corrélation radiale $C(r)$.

Fonction d'autocorrelation. La fonction de d'auto-correlation radiale peut être vue comme la moyenne des correlations

$$C(r) = \langle \sigma_i \sigma_j \rangle_{i^2+j^2=r^2} \quad = \sum_{\substack{i,j \\ i^2+j^2=r^2}} \sigma_i \sigma_j \tag{7}$$

Le processus utilisé pour la calculer est décrit en figure 4. Le calcul de cette fonction est relativement lourd, il est cependant fondamental pour observer la transition de phase. D'autres méthodes, plus rapides, basées sur des transformées de Fourier ont été essayées, sans succès cependant.

On rappelle qu'on s'attend à trouver une fonction de la forme

$$C_h(r) = be^{-ar} \iff \log(C_h(r)) = -ar + b \tag{8}$$

pour des températures au delà de la transition de phase, et

$$C_l(r) = \frac{1}{b + ar} \iff \frac{1}{C_l(r)} = ar + b \tag{9}$$

pour des températures inférieure à la transition de phase. Comme on peut le voir, il est facile de transformer ces fonctions en équations linéaires, ce qui peut s'avérer utile pour reconnaitre l'un ou l'autre des régimes. On va donc chercher à *fitter* ces deux droites affines et noter les R^2 pour essayer d'observer une transition de comportement.

Stabilisation. Une première méthode qui permet de détecter la stabilisation est mise en place. Il s'agit de moyenner les m derniers points de $E(t)$, et de noter la première fois que cette valeur est atteinte. Cette heuristique se base sur le fait que l'énergie est globalement décroissante si (h, k), et qu'elle converge donc vers sa valeur finale, moyennant fluctuations. Pour éviter les problèmes d'égalités entre nombres flottants et pour tenir compte desdites fluctuations, on considère :

$$t_{\text{stab}} = \min t \text{ t.q. } \left(\frac{\langle E \rangle_{\text{end}} - E(t)}{E(0) - \langle E \rangle_{\text{end}}}\right) < 1.$$

Require: $L_M,\ L_Q,\ L_E$ les listes des valeurs de $M,\ Q,\ E$ après stabilisation
Require: average_2(A) une fonction qui renvoie une liste de taille $N/2$ dont chaque point vaut

$$B_i = 1/2\,(A_{2i} + A_{2i+1})$$

Require: standard(A) une fonction qui calcule l'écart type
  $S \leftarrow (\emptyset, \emptyset, \emptyset)$
  for $i \in \{M, Q, E\}$ **do**
    $L_n = \emptyset$
    $n \leftarrow \text{len}(L_i)$
    while $n > 1$ **do**
      $L_i = \text{average\_2}(L_i)$
      append_value(get_sublist(S, i), standard(L_i))
      $n \leftarrow \text{len}(L_i)$
    end while
  end for
  $n_M \leftarrow$ argmax get_sublist$(S, 1)$
  $n_Q \leftarrow$ argmax get_sublist$(S, 2)$
  $n_E \leftarrow$ argmax get_sublist$(S, 3)$
  $n \leftarrow \min(n_M, n_Q, n_E)$
  $e_M \leftarrow$ get_value($S(1)$, n)
  $e_Q \leftarrow$ get_value($S(2)$, n)
  $e_E \leftarrow$ get_value($S(3)$, n)

FIGURE 5 – Procédure de calcul de l'erreur statistique sur une longue expérience.

On peut ensuite se poser la question de la corrélation temporelle : une longue expérience doit avoir un poids statistique plus fort qu'une expérience plus courte, le lier à l'erreur statistique est plus difficile. Une méthode pour estimer l'erreur est présentée en figure 5. Elle consiste à découper récursivement une expérience en plusieurs sous-expériences de taille un demi, et de choisir l'ensemble de sous-expériences pour laquelle l'erreur statistique

$$\Sigma_N = \frac{\sigma}{\sqrt{N}}$$

est la plus grande. Cela nous permet de savoir combien de sous-expériences "indépendantes" sont présentes dans chaque longue expérience.

3 Résultats

3.1 Experience à température infinie

Pour simuler une température infinie, on prend $(h, k) = (0, 0)$. Comme le montre la figure 6, l'énergie relative reste donc nulle tout au long de l'expérience, toute modification est acceptée. Ce comportement est cohérent avec la physique du problème : à température infinie, chaque particule a une énergie infinie, et donc est indifférente à sa configuration.

Références

[Wik17] Wikipédia. Phase hexatique — wikipédia, l'encyclopédie libre, 2017. [En ligne ; Page disponible le 7-avril-2017].

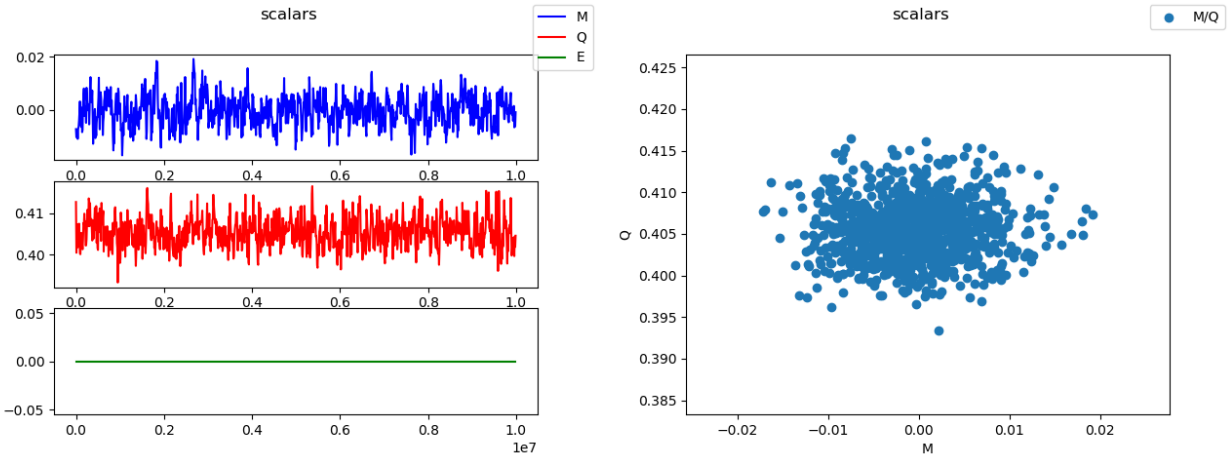

FIGURE 6 – Scalaires une experience à température infinie $(h, k) = (0, 0)$ pour une grille de $N = 128$, avec prise en compte des $n = 1$ plus proches voisins et une décroissance en $a = 1$.