# TP Méthode de Métropolis Monte-Carlo

Sous-titre / sujet

Auteur : Owen GRIERE
Date : 2 octobre 2025

# Table des matières

# 1 Introduction

Ce TP a pour but de générer sur Python une méthode de Métropolis Monte Carlo sur un profil de potentiel dans un espace réduit appelé boite.

## 1.1 Les parametres

Voici les parametres a faire varier lors de l'execution du code :

**`L = 5`** *Taille de la boîte.* Le domaine de calcul est le carré $\Omega = [-L, L] \times [-L, L]$. Les positions sont donc bornées par $-L \leq x, y \leq L$.

**`T = 10`** *Température* Contrôle la probabilité d'accepter un déplacement défavorable.

**`pas_aleatoire = 3`** *Amplitude de déplacement proposée* À chaque itération, on tire un décalage $(\delta x, \delta y)$ dans l'intervalle

$$\delta x,\, \delta y \in [-p_{\text{aléa}},\, p_{\text{aléa}}], \quad \text{où } p_{\text{aléa}} = \texttt{pas\_aleatoire}.$$

**`pas = 50`** *Nombre d'itérations.* On effectue $N_{\text{pas}} = \texttt{pas}$ propositions/acceptations de déplacements (boucle Monte Carlo).

**`epsilon =`** $10^{-3}$ *Seuil de minima d'énergie* On considère que l'on a atteint le fond d'un puits de potentiel lorsque la projection du gradient vérifie la condition

$$\left|D_F\right| < \varepsilon,$$

(Autrement dit, la pente locale en descente de gradient est proche de 0)

**`Discret = True`** *Type d'espace*

- **True** : espace discret.
- **False** : espace continu.

**`plotting = '2D'`** *Mode de visualisation*

- `'2D'` : espace 2D dans le plan $(x, y)$, avec coloration suivant la valeur de $f$.
- `'3D'` : surface $z = F(x, y)$ au-dessus du domaine $\Omega$.

**`GRADIENT = False`** *Affichage optionnel du gradient projeté*

- **True** : en plus du profil d'energie, on affiche une carte de $D_F$ pour visualiser les zones de forte pente.
- **False** : on n'affiche que la figure principale.

**`les tables`** $W_i, X_i, Y_i$ *Paramètres de la fonction de potentiel*

- Les couples $(X_i, Y_i)$ représentent les **positions des centres des puits de potentiel**. Chaque point correspond donc à une localisation spatiale où l'énergie est minimale.
- Les coefficients $W_i$ jouent le rôle de **profondeur des puits**. Plus $W_i$ est grand, plus le puits associé est profond, ce qui correspond à une interaction énergétique plus marquée.

# 2 Explication du Code

Ici est présenté toutes les fonctions utilisé dans le code python sachant que le tout est articulé dans la fonction main() qui sera également explicitée.

## 2.1 La fonction de potentiel

Cette fonction a pour but de caluler la valeur de potentiel en chaque point de la grille (X, Y) définit par cette fonction :

$$F(x,y) \;=\; -\sum_i W_i\, e^{\left(-(X_i-x)^2-(Y_i-y)^2\right)}$$

Cette fonction permet d'obtenir un profil énergétique aléatoire.

## 2.2 Les fonctions intermédiaires

### 2.2.1 La fonction `grad_f`

Cette fonction sert à calculer le gradient en point de cette manière :

$$\nabla F(x,y) = \begin{bmatrix} -\sum_i W_i\, 2(X_i-x)\, e^{\left(-(X_i-x)^2-(Y_i-y)^2\right)} \\ -\sum_i W_i\, 2(Y_i-y)\, e^{\left(-(X_i-x)^2-(Y_i-y)^2\right)} \end{bmatrix}$$

Ensuite, on calul son projeté sur la direction de $-\nabla F$ pour obtenir un scalaire :

$$D_F \;=\; \nabla F(x,y) \cdot \frac{-\nabla F(x,y)}{\|-\nabla F(x,y)\|} \;=\; -\|\nabla F(x,y)\|$$

On obtient au final : $D_F \;=\; -\|\nabla F(x,y)\|$ Ainsi on obtient que $D_F$ est l'opposé de la norme du gradient de F.

Les fonctions `grad_grid` et `projected_gradient_field` permettent de faire la même chose mais sur une grille Numpy afin de pouvoir plot les valeur de la norme du gradient de F sur l'entiereté de la boite.

### 2.2.2 La fonction `verif_deplacement`

Cette fonction a pour objectif de vérifier si à partir d'une position (x, y), un décalage de cette position de (dx, dy) est possible au sein de la boite. Si le déplacement est possible alors elle retournera True alors que dans le cas contraire False sera retourné.

### 2.2.3 La fonction `init_decalage`

Cette fonction pour objectif de générer un vecteur de déplacement. Cette fonction prend en entrée un boolean 'puit', un pas aléatoire ainsi qu'un scalaire gradient. Le boolean puit a pour objectif de désigner a l'attribution aléatoire que la position de la conformation se trouve dans un puit de potentiel.
On effectue cette comparaison pour definir le fond d'un puit, donc un minima de potentiel.

$$|D_F| < \varepsilon$$

Quand cette condition est validé, on choisit un décalage comme si nous n'étions pas dans un puit, afin d'en sortir. Alors que si nous touchons pas le fond du puit, le décalage définit par le pas aléatoire est multiplié par un facteur $\frac{1}{e^{|D_F|}}$ pour toucher progressivement la fin du puit pour atteindre plus finement le minima de potentiel.

Ici l'objectif est d'introduire un terme qui va venir réduire la fenêtre d'un choix aléatoire d'un pas afin de rester au maximum dans le puit de potentiel et d'etre sur d'atteindre le plus possible le minima de celui-ci.

### 2.2.4 La fonction `vecteur_deplacement`

Cette fonction effectue le déplacement qui a été vérifié au préalable.

## 2.3 La fonction de Métropolis Monte Carlo

Cette fonction est le coeur de l'algorithme de Metropolis Monte Carlo, elle est alors détaillé ci-dessous :
Soit une position $(i, j) \in \mathbb{R}^2$ (correspondant a une conformation spécifique) et la fonction d'energie $f : \mathbb{R}^2 \to \mathbb{R}$. On propose un déplacement $(x_{\text{décal}}, y_{\text{décal}})$ conduisant à la conformation candidate

$$(i', j') \; = \; \big(i + x_{\text{décal}},\, j + y_{\text{décal}}\big),$$

à condition qu'il soit admissible (`verif_deplacement` donne True). On définit la variation d'énergie

$$\Delta f \; = \; F(i', j') - F(i, j).$$

La règle d'acceptation de Metropolis fonctionne de la manière suivante : On accepte le candidat dans 2 cas différents et distincts :

– Si $\Delta f \leq 0$ correspondant a une conformation nécessairement plus stable

– Si en tirant $R \sim \mathbb{U}(0, 1)$ le candidat vérifie $e^{-\Delta f/T} > R$ (acceptation probabiliste d'une dégradation de la stabilité) où $T > 0$ est la "Température".

Puis, l'on met à jour l'état via

$$(i, j) \leftarrow (i', j') \quad (\texttt{vecteur\_deplacement}).$$

Sinon, on refuse le déplacement et on conserve l'état courant $(i, j)$. Dans le code fourni, cela correspond au test

$$\big(F(i, j) > F(i', j')\big) \;\text{ ou }\; \Big(F(i, j) \leq F(i', j') \text{ et } e^{(F(i,j)-F(i',j'))/T} > R\Big),$$

ce qui est exactement $e^{-\Delta F/T} > R$ lorsque $\Delta F > 0$.

## 2.4 La fonction `main`

Cette fonction execute quand a elle la méthode de Metropolis Monte Carlo pour chaque pas choisit dans les parametres sachant quelle calule a chaque itération la norme du gradient a chaque nouveau point afin de définir la présence d'un puit et de réduire le pas dans le cas d'un détection d'un puit.

les variables `puit` et `flag_puit` permettent comme une bascule de définir un changement d'etat de la présence d'un puit au niveau de la conformation actuelle, `origin_puit` sert a récupérer la valeur du potentiel au niveau maximum du potentiel puit afin de verifier quand la conformation suivante en sort. J'utilise de manière arbitraire $\varepsilon$ pour définir ce critére de sortie d'un puit :

$$\big|F(\texttt{origin\_puit}) - F(\text{conformation actuelle})\big| \;\leq\; \varepsilon \;\texttt{AND}\; ETAT = \textit{dans un puits} \quad\Longrightarrow\quad ETAT = \textit{sortie du puits}$$

la condition $-\|\nabla F(x, y)\| < -0.2$ a été définit empiriquement en imprimant les moyenne et écart-type de $-\|\nabla F(x, y)\|$ pour plusieurs simulation. Il n'est pas fin du tout et ne semble pas etre stringeant pour le moment.

## 2.5 Les fonctions de visualisation graphique

### 2.5.1 La fonction `plot_2D`

Cette fonction plot en 2D le profil énergétique ainsi que la trajectoire de la conformation suivant X et Y. Il est possible de plot également le gradient sur l'entiereté de la boite avec la trajectoire

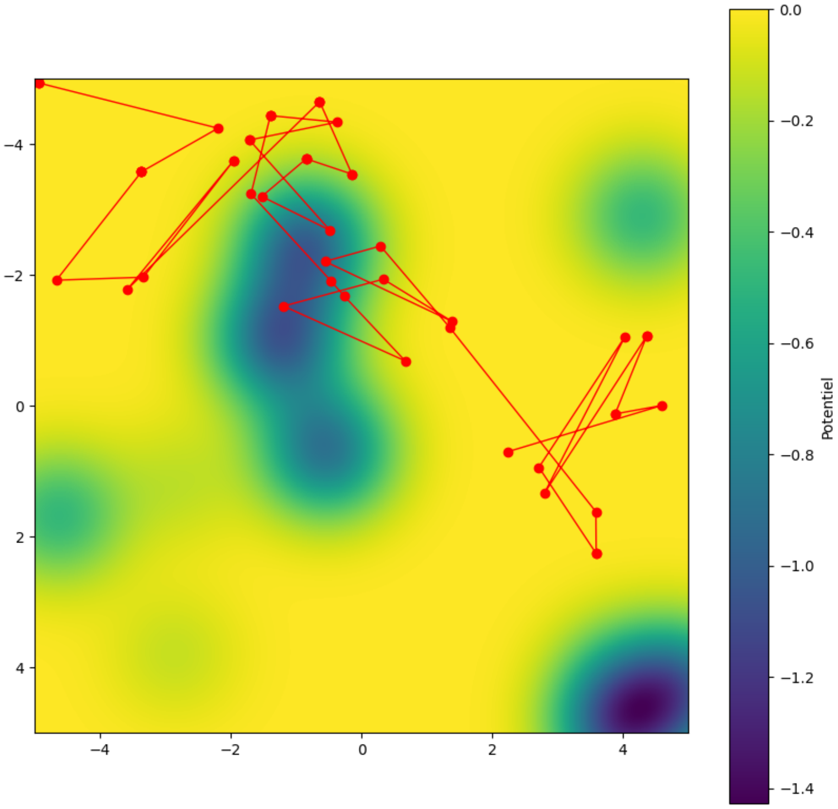


*FIGURE 2.1 – Visualisation 2D du profil énergétique avec la trajectoire obtenu grace a MMC pour 50 pas*

### 2.5.2 La fonction `plot_3D`

Cette fonction plot en 2D le profil énergétique ainsi que la trajectoire de la conformation suivant X, Y et du potentiel de la conformation. Il est possible de plot également le gradient sur l'entiereté de la boite avec la trajectoire

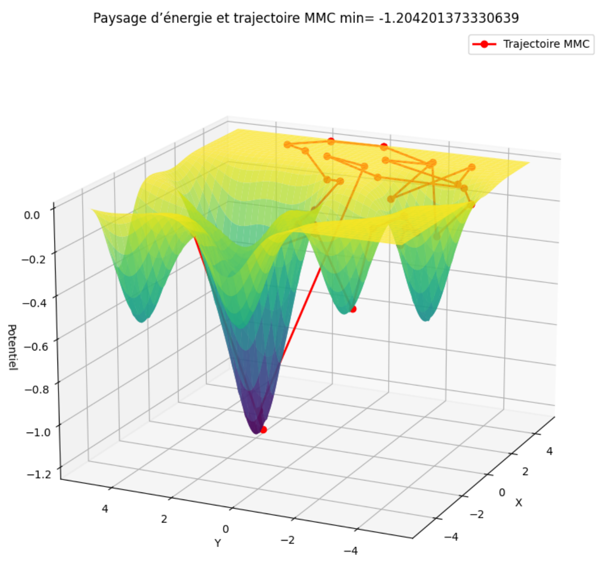


*FIGURE 2.2 – Visualisation en 3D du profil énergétique avec la trajectoire obtenue grâce à MMC pour 50 pas*

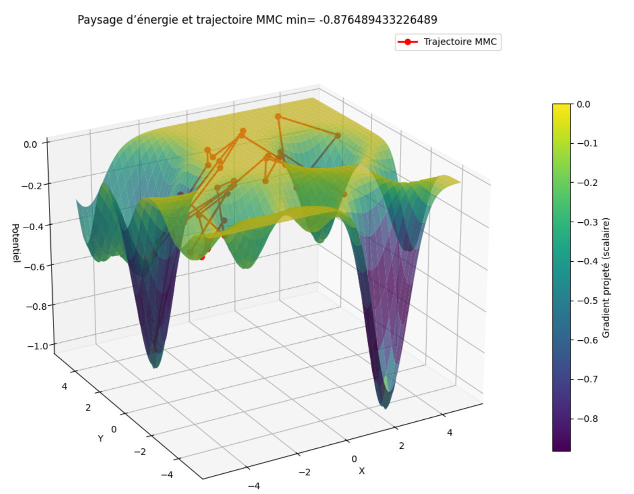


*FIGURE 2.3 – Visualisation en 3D du gradient projeté avec la trajectoire obtenue grâce à MMC pour 50 pas*

# 3 Résultats

Dans cette section je vais présenter mes différents résultat atteint avec la méthode de Metropolis Monte Carlo. On à choisit dans cette Figure 3.2 $\varepsilon = 10^{-3}$ ainsi que 200 itération avec une température égale à 2. L'objectif étant d'augmenter la température pour voir l'évolution de la méthode en fonction de celle-ci

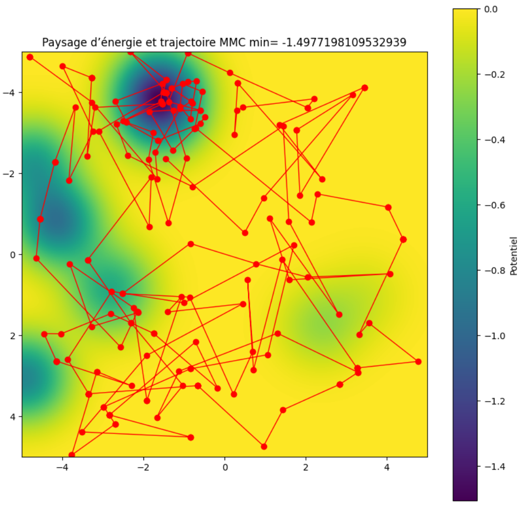


FIGURE 3.1 – *Visualisation en 2D du profil énergétique avec la trajectoire obtenue grâce à MMC pour 200 pas*

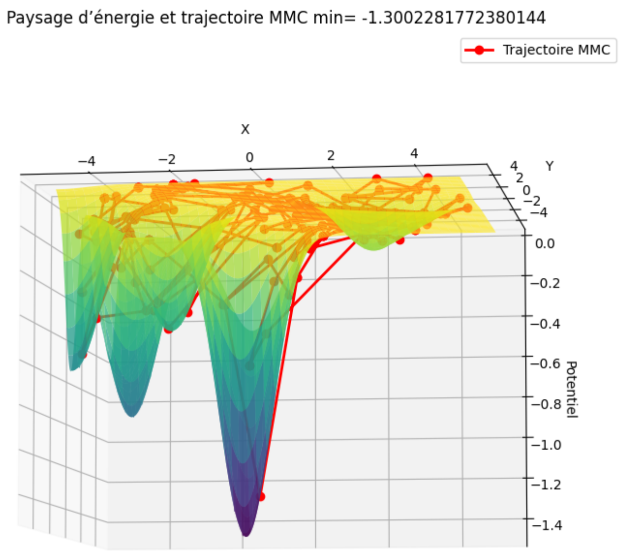


FIGURE 3.2 – *Visualisation en 3D du profil énergétique avec la trajectoire obtenue grâce à MMC pour 200 pas*

Désormais il s'agit d'augmenter uniquement la température et de voir comment la simulation augmente. J'ai utiliser la seed '42' afin de reproduire le même profil énergétique afin de d'isoler l'impact de la température. Dans cette Figure 3.4, la température a été fixé à 200.

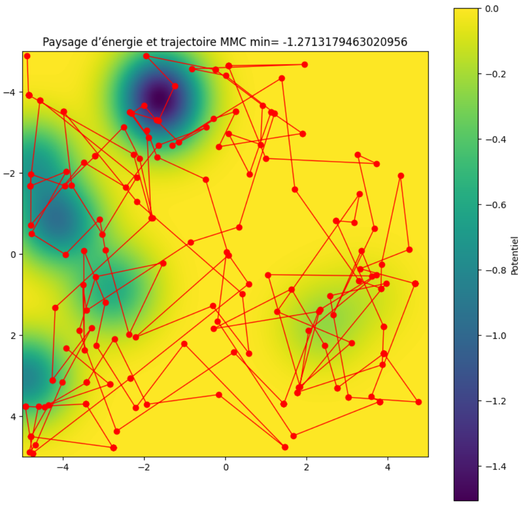


FIGURE 3.3 – *Visualisation en 2D du profil énergétique avec la trajectoire obtenue grâce à MMC pour 200 pas*

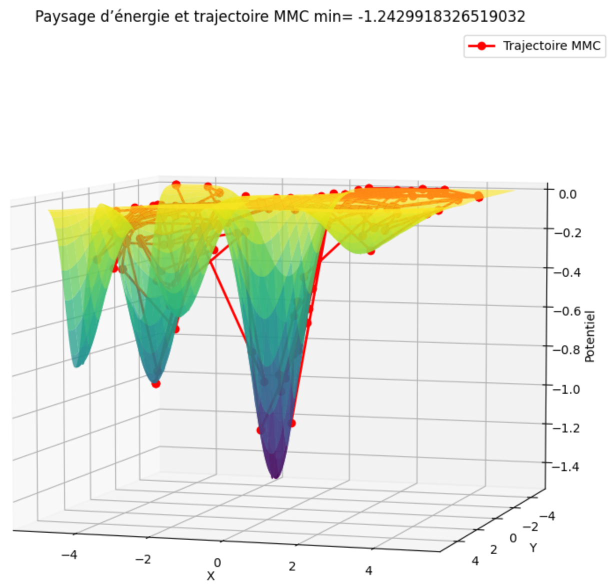


FIGURE 3.4 – *Visualisation en 3D du profil énergétique avec la trajectoire obtenue grâce à MMC pour 200 pas*

On remarque que plus la température augmente plus il est compliqué pour la simulation d'atteindre les extremums d'énergie, ce qui relativement logique compte tenu du modèle d'exclusion du critère de Metropolis.

# 4 Discussion