



École d'ingénieur Denis Diderot
Université Paris Cité

TP Modélisation MD

Gros Grain

Auteur : Owen GRIERE

Date : 9 octobre 2025

Table des matières

1	Introduction	2
2	Fonctions	3
2.0.1	get_arguments	3
2.1	Fonctions utilisées pour le Mapping	3
2.1.1	check_purine	3
2.1.2	centre_de_masse	3
2.1.3	mapping_RIA	3
2.1.4	formatage	3
2.1.5	MAPPING	4
2.2	Fonctions utilisées pour le Champ de force	4
2.2.1	distance	4
2.2.2	E_Lennard_Jones	4
2.2.3	E_elastique	5
2.2.4	E_electrostatic	5
2.2.5	compute_energy	5
2.3	Fonctions pour la méthode de Monte Carlo	6
2.3.1	MMC	6
2.4	Fonctions de plot et d'analyse diverses	6
2.4.1	analyse_T_pas	6
2.4.2	analyse_profil	7
2.5	Fonctions pour le paramètres R_0	7
2.6	Fonctionnement global (main)	7
3	Le Mapping	8
4	Le champ de Force	10
4.1	Les différents potentiels	10
4.2	Les paramètres associés	10
5	Résultats	11
6	Discussion	13

1 Introduction

La simulation MD est une méthode numérique permettant de suivre le mouvement de particules en résolvant les équations de Newton à partir de forces dérivées de potentiels d'interaction. Dans le cas d'une modélisation gros grain (coarse-grain), plusieurs atomes sont regroupés en une seule entité appelée « beads », afin de simplifier la description du système. Cette approche réduit considérablement le nombre de degrés de liberté et donc le coût computationnel. Elle permet ainsi d'accéder à des échelles de temps et d'espace plus grandes que celles atteignables par les simulations atomistiques classiques. Les interactions entre les billes sont modélisées par des potentiels effectifs, ajustés pour reproduire les propriétés physiques globales du système réel. Ce type de simulation est particulièrement utilisé pour étudier les macromolécules

biologiques comme les protéines où les détails atomiques ne sont pas systématiquement indispensables. La MD gros grain constitue donc un compromis entre réalisme structurel et efficacité numérique. Dans notre cas nous étudions différentes conformation d'un ARN grâce à diverses structures cristallisées (PDB). L'objectif final étant de trouver la conformation la plus stable de l'ARN. Cette recherche dépendra du mapping (regroupement de groupes d'atomes en beads pour obtenir une simulation gros grain) que nous choisirons, d'un champ de force spécifique à notre problème. Une fois les étapes précédentes accomplies nous utiliserons alors une méthode de Monte Carlo pour explorer les diverses conformations jusqu'à obtenir la conformation la plus stable, donc la plus basse en énergie.

MD simulation is a numerical method for tracking particle motion by solving Newton's equations based on forces derived from interaction potentials. In coarse-grain modeling, several atoms are grouped together into a single entity called "beads" in order to simplify the description of the system. This approach significantly reduces the number of degrees of freedom and therefore the computational cost. It thus allows access to larger time and space scales than those achievable by classical atomistic simulations. The interactions between the beads are modeled by effective potentials, adjusted to reproduce the overall physical properties of the real system. This type of simulation is particularly used to study biological

macromolecules such as proteins, where atomic details are not always essential. Coarse-grained MD therefore represents a compromise between structural realism and numerical efficiency. In our case, we are studying different conformations of RNA using various crystallized structures (PDB). The ultimate goal is to find the most stable conformation of the RNA. This research will depend on the mapping (grouping of atoms into beads to obtain a coarse-grained simulation) that we choose, and on a force field specific to our problem. Once the previous steps have been completed, we will then use a Monte Carlo method to explore the various conformations until we obtain the most stable conformation, the one with the lowest energy.

2 Fonctions

2.0.1 `get_arguments`

Cette fonction a pour objectif de récupérer l'argument `'-mapping'` qui peut être placé dans la ligne d'exécution du code afin de réaliser le mapping de tous les fichiers PDB d'un dossier. Si `'-mapping'` n'est pas présent alors le mapping ne sera pas effectué. Le nouveau mapping remplace le précédent.

2.1 Fonctions utilisées pour le Mapping

Dans cette section, nous allons introduire les fonctions permettant d'effectuer le mapping en gros grain des fichiers PDB qui sont à l'origine au niveau atomistiques.

2.1.1 `check_purine`

Cette fonction sert à vérifier si la sous-structure observée dans l'ARN correspond à une purine ou à une pyrimidine. En effet, le mapping implique des atomes des bases azotées ainsi il est important de faire la différence entre les deux.

2.1.2 `centre_de_masse`

Cette fonction a pour objectif de récupérer les coordonnées d'un groupe d'atomes que l'on veut regrouper en une seule bead afin de déterminer la position de cette bead en calculant le centre de masse du groupe d'atomes.

2.1.3 `mapping_RIA`

Cette fonction sert à effectuer le "mapping gros grain" d'un nucléotide d'ARN en regroupant ses atomes en 5 "beads" représentatives. Elle identifie d'abord le type de base (purine ou pyrimidine), puis renomme et réindexe certains atomes clés. Trois atomes de la base seront renommés en Ni1, Ni2 et Ni3. Le groupement phosphate (Pho) reste inchangé simplement renommé, et les atomes constituant le sucre qui lie la base azotée au groupement phosphate est renommé en ribose (RIB). Les coordonnées du ribose sont remplacées par son centre de masse, calculé à partir des atomes du cycle sucre. Chaque "bead" reçoit un identifiant unique dépendant du rang du nucléotide (sT). Enfin, la fonction renvoie une nouvelle liste d'atomes représentatifs (4 ou 5 selon la présence du phosphate) correspondant au modèle gros grain de ce nucléotide.

2.1.4 `formatage`

Cette fonction sert à formater le résultat du mapping dans une structure de fichier de type PDB ou chaque élément n'est pas séparé par une tabulation mais doit impérativement se situer dans un interval de colonnes.

2.1.5 MAPPING

Cette fonction réalise le mapping en parcourant tout le fichier PDB, et en exécutant les fonctions précédentes en itérant sur les nucléotides afin de réécrire un fichier de mapping pour chaque fichier PDB ouvert.

2.2 Fonctions utilisées pour le Champ de force

Dans cette section nous allons décrire les fonctions qui composent la construction du champ de force ainsi que le calcul de l'énergie libre de chaque conformation.

2.2.1 distance

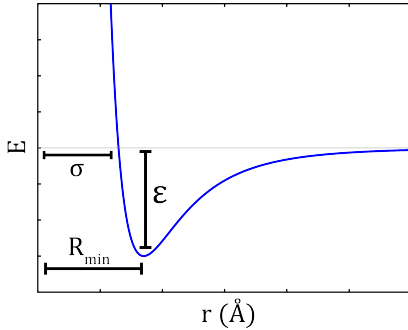
Cette fonction calcul la distance euclidienne entre deux atomes i et j en trois dimensions.

$$r_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2}$$

où :

- x_i, y_i, z_i sont les **coordonnées cartésiennes** de l'atome i ,
- x_j, y_j, z_j sont celles de l'atome j ,
- r_{ij} représente la **distance euclidienne** entre les deux atomes dans l'espace tridimensionnel.

2.2.2 E_Lennard_Jones



Cette fonction calcule le potentiel de Lennard-Jones entre deux atomes non reliés par le squelette de la manière suivante :

$$E_{LJ}(i, j) = 4 \varepsilon_{ij} \left[\left(\frac{\sigma_{ij}}{r_{ij}} \right)^{12} - \left(\frac{\sigma_{ij}}{r_{ij}} \right)^6 \right]$$

FIGURE 2.1 – Potentiel de Lennard-Jones entre deux atomes ou beads non liés

où :

- $\varepsilon_{ij} = \sqrt{\varepsilon_i \varepsilon_j}$ est la **profondeur du puits de potentiel**, représentant l'intensité de l'interaction attractive,
- $\sigma_{ij} = \frac{1}{2}(\sigma_i + \sigma_j)$ est la **distance caractéristique** à laquelle le potentiel est nul,
- r_{ij} est la **distance** entre les atomes i et j ,
- le terme $\left(\frac{\sigma_{ij}}{r_{ij}} \right)^{12}$ modélise la **répulsion** à courte portée (chevauchement électronique),
- tandis que $\left(\frac{\sigma_{ij}}{r_{ij}} \right)^6$ décrit l'**attraction de van der Waals** à longue portée.

2.2.3 E_elastique

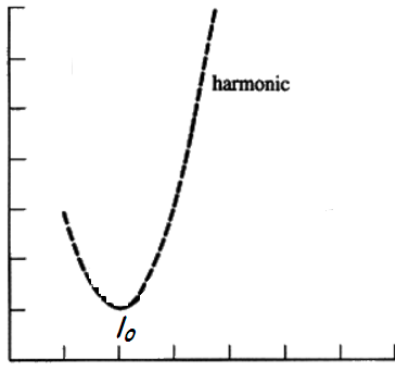


FIGURE 2.2 – Potentiel élastique entre deux atomes ou beads chimiquement liés

Cette fonction calcule le potentiel de Lennard-Jones entre deux atomes reliés par le squelette de la manière suivante :

$$E_{\text{élastique}}(i, j) = \frac{1}{2} k_{ij} (r_{ij} - r_{0,ij})^2$$

où :

- k_{ij} est la **constante de raideur** associée à la paire d'atomes (i, j) , obtenue à partir de la table K_{EL} ,
- $r_{0,ij}$ est la **distance d'équilibre** entre les deux atomes, issue de la table $R0_{\text{EL}}$,
- $(r_{ij} - r_{0,ij})$ représente l'**élongation** du lien,
- le facteur $\frac{1}{2}$ correspond à la **forme harmonique** de l'énergie selon la loi de Hooke.

2.2.4 E_electrostatic

Cette fonction calcule le potentiel electrostatique entre deux atomes non reliés par le squelette de la manière suivante :

$$E_{\text{elec}}(i, j) = \frac{1}{4\pi\epsilon_0\epsilon_r} \cdot \frac{q_i q_j}{r_{ij}}$$

où :

- q_i et q_j sont les **charges électriques** des atomes i et j ,
- r_{ij} est la **distance** séparant les deux atomes,
- ϵ_0 est la **perméabilité du vide**,
- ϵ_r est la **constante diélectrique relative** du milieu,
- $\frac{1}{4\pi\epsilon_0\epsilon_r}$ est le **facteur de Coulomb**, qui module l'intensité de l'interaction selon le milieu considéré.

2.2.5 compute_energy

Cette fonction calcule l'énergie libre d'une conformation moléculaire à partir d'un fichier d'entrée (au format type PDB). Elle commence par lire toutes les lignes du fichier et extrait uniquement celles correspondant aux atomes ('ATOM'), qu'elle stocke dans une liste. Ensuite, elle parcourt toutes les paires d'atomes pour évaluer leurs interactions dans une double boucle sans redondance. Si les deux atomes appartiennent à la même sous-structure et appartiennent au squelette (grâce la liste 'bonds'), ou s'il s'agit de deux groupements phosphates voisins le long de la chaîne, la fonction calcule une énergie élastique. Dans tous les autres cas, elle évalue le potentiel electrostatique ainsi que l'interaction de type Lennard-Jones, représentant les forces de van der Waals entre beads non liés. La somme de toutes ces contributions fournit l'énergie libre U du système.

2.3 Fonctions pour la méthode de Monte Carlo

2.3.1 MMC

Comme vu dans le précédent TP cette fonction a simplement pour objectif de parcourir les différentes conformations de l'ARN qui se trouve être les différents fichiers PDB. Cette fonction utilise l'algorithme de Monte Carlo pour être plus efficace et trouver la conformation la plus stable.

2.4 Fonctions de plot et d'analyse diverses

2.4.1 analyse_T_pas

Cette fonction sert à plot une heatmap de la performance de la méthode de Monte Carlo en fonction de diverses valeurs de Température et de nombre de pas, à chaque itération nous réalisons un nombre choisi d'expériences. Dans le cas de la Figure 2.3, nous avons réalisé 1000 expériences de Monte Carlo pour chacune des valeurs de Température et de nombre de pas.

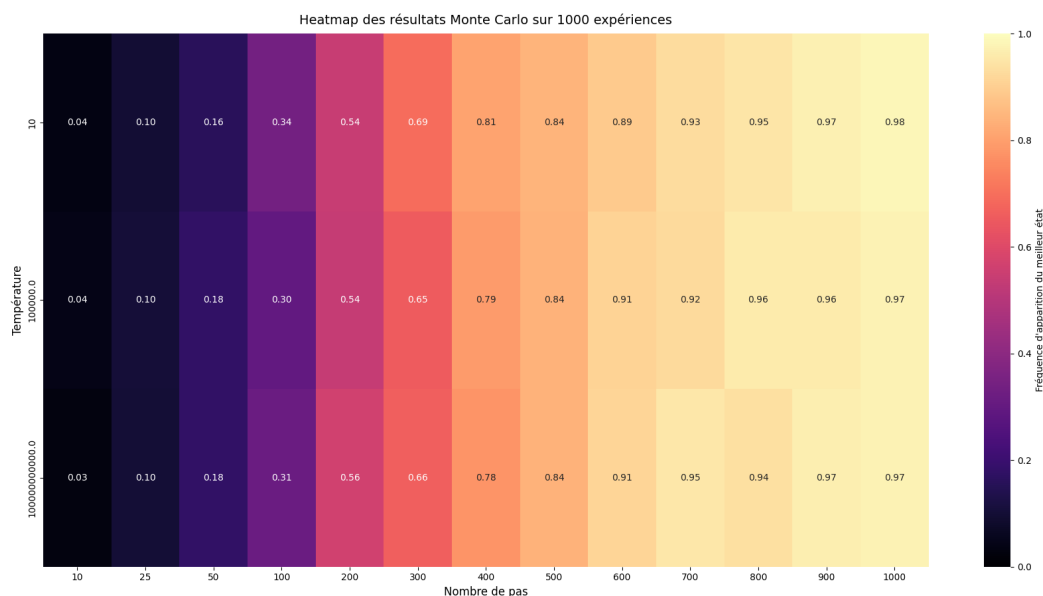


FIGURE 2.3 – Heatmap montrant la performance de la MMC pour différentes valeurs de Température et de nombre de pas par MMC, le nombre d'expérience a été fixé à 1000 pour pouvoir récupérer des statistiques plus fiables

2.4.2 analyse_profil

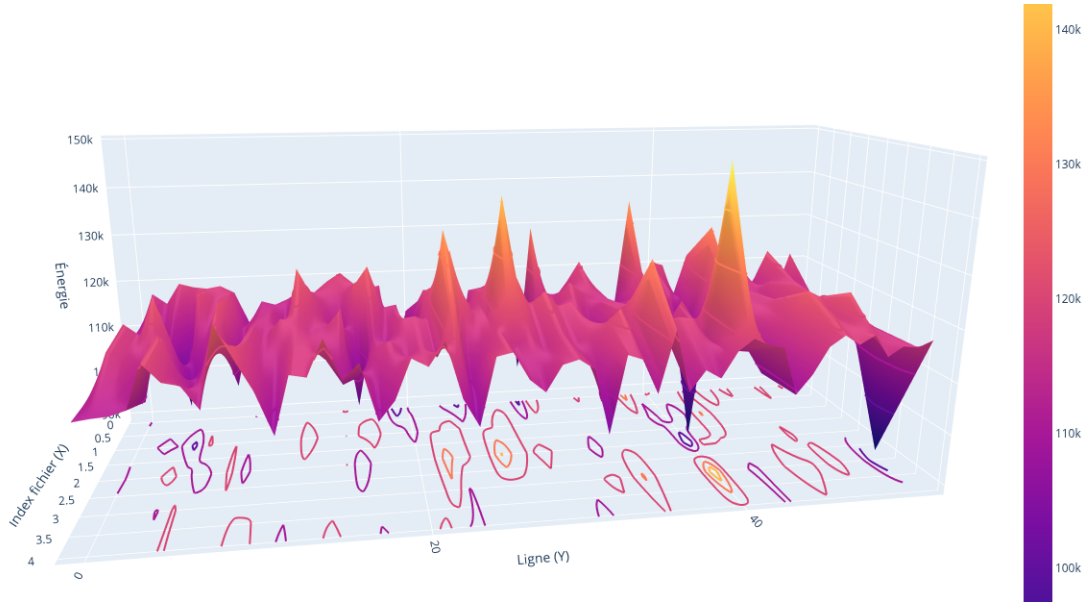


FIGURE 2.4 – Paysage énergétique de toutes conformations disponibles de notre ARN (avec Plotly)

Cette fonction permet de voir comment est distribuer l'énergie libre de chacune des conformations a travers un graphique interactif Plotly où tout les fichiers sont dans le plan $P = (X, Y)$ et l'énergie associé aux conformations en Z. Dans la Figure 2.4 on a du mal a distinguer pour quel fichier est associé une telle energie mais nous pouvons observer le paysage global de l'énergie libre.

2.5 Fonctions pour le paramètrés R_0

Cette fonction sert a partir des fichier mapper et trouver une approximation du r_0 en calculant la médiane de chacune des liaisons pour chaque fichier C'est le role de la fonction `find_parameter` de creer un dictionnaire qui accumule toute les distances entre les liaisons ensuite la fonction `median` calcul la mediane de chacune des liaisons du dictionnaire concaténé par fichier (action executé dans le main)

2.6 Fonctionnement global (main)

Pour articuler tout ces fonctions, j'utilise un main qui va executer le mapping (si besoin), calculer l'énergie libre de chacune des structures et les placer dans un dictionnaire qui a associe une structure a son énergie libre totale. Ensuite, celui-ci execute la méthode de Monte Carlo pour trouver la meilleur conformation pas à pas. Enfin j'effectue l'analyse que je désire en appelant la fonction souhaité.

3 Le Mapping

Un ARN se constitue d'une chaîne très similaire à l'ADN. L'ADN est une double hélice constituée de deux chaînes en opposition composé chacune d'un phosphore, d'un deoxyribose et d'une base azotée tandis que l'ARN est constitué d'un ribose complet.

Donc, notre ARN est une chaîne de nucléotide que l'on peut réduire dans une modélisation gros grain. Je choisis alors le modèle XIA, visible dans la Figure 3.1 qui propose de conserver le groupe phosphate comme maillon de la chaîne, le cycle sucre est remplacé par une seule bead. La base azotée est remplacé par 3 beads formant un triangle. L'objectif du mapping de la base azotée est de conserver sa forme qui est systématiquement plane. La géométrie du triangle permet de conserver la structure plane de la base azotée. Le mapping que j'utilise est le suivant :

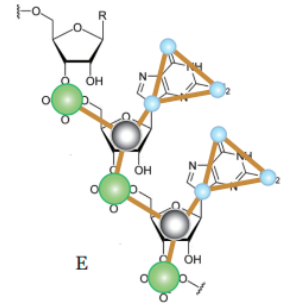


FIGURE 3.1 – Modèle XIA pour les simulations gros grain

- **Ribose** cycle sucre transformé en une bead
- **Pho** groupe phosphate transformé en une bead
- **Base azotée de type purine** trois atomes sont conservés
- **Base azotée de type pyrimidine** trois atomes sont conservés

Ainsi, j'utilise des listes pour récupérer les atomes de chaque structure du nucléotide. Les atomes récupéré pour le cycle sucre sont : O3' C3' HO2' O2' C2' C1' O4' C4' C5' O5' H05'. Je fais la différences entre les purines et pyrimidines dans mon mapping car elles n'ont pas la même structure et donc pas les mêmes propriétés mécanistiques. La purine est composé de deux cycle contrairement a la pyrimidine qui n'en possède qu'un. Tout d'abord nous allons regarder le mapping pour une pyrimidine grace au logiciel PyMol. Pour une base azotée de ce type les atomes récupéré sont : N1, N3 et C5.

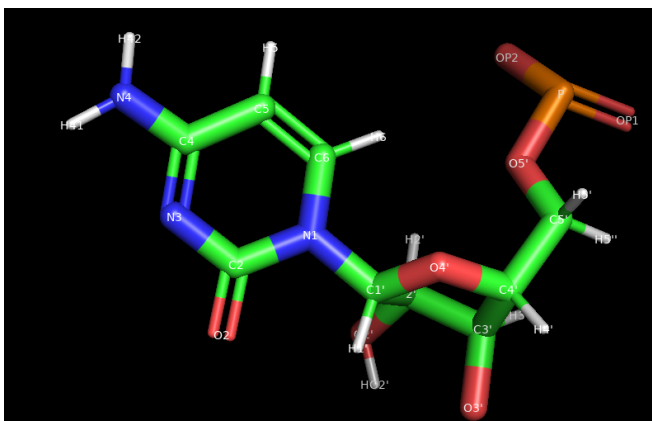


FIGURE 3.2 – Visualisation en 3D d'une pyrimidine au sein d'un nucléotide

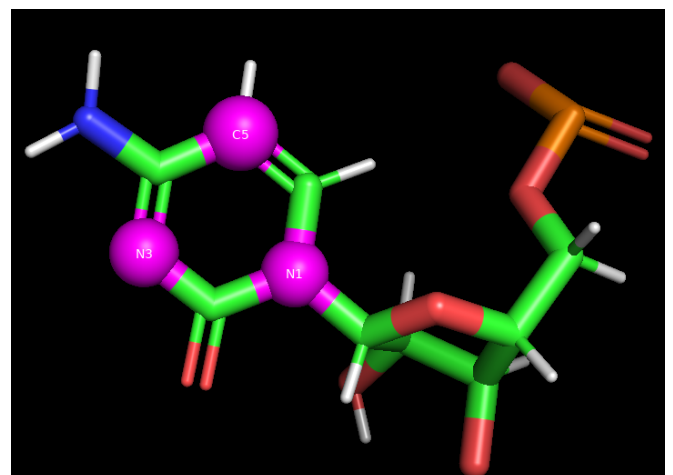


FIGURE 3.3 – Visualisation en 3D d'une pyrimidine au sein d'un nucléotide où les atomes conservés pour le mapping sont modélisés en sphères magenta

Ensuite, nous allons regarder le mapping d'une purine grâce au logiciel PyMol. Pour une base azotée de ce type les atomes récupérés sont : N9, C2 et C6.

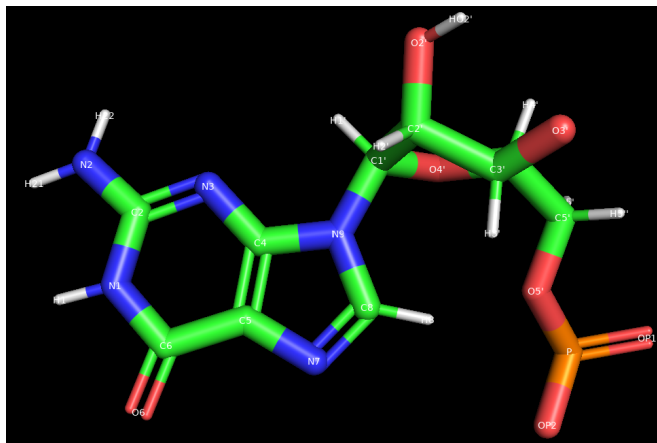


FIGURE 3.4 – Visualisation en 3D d'une purine au sein d'un nucléotide

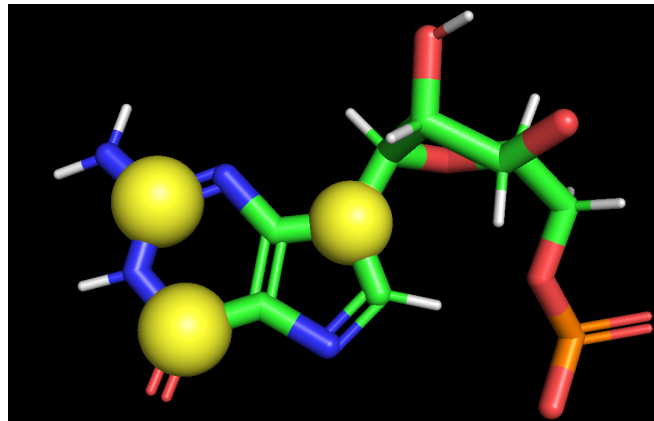


FIGURE 3.5 – Visualisation en 3D d'une purine au sein d'un nucléotide où les atomes conservés pour le mapping sont modélisés en sphères jaunes

Le mapping complet est visible dans la Figure 3.6. Dans cette figure, on peut voir la forme de l'ARN d'origine en cartoon quand les sphères représentent le mapping.

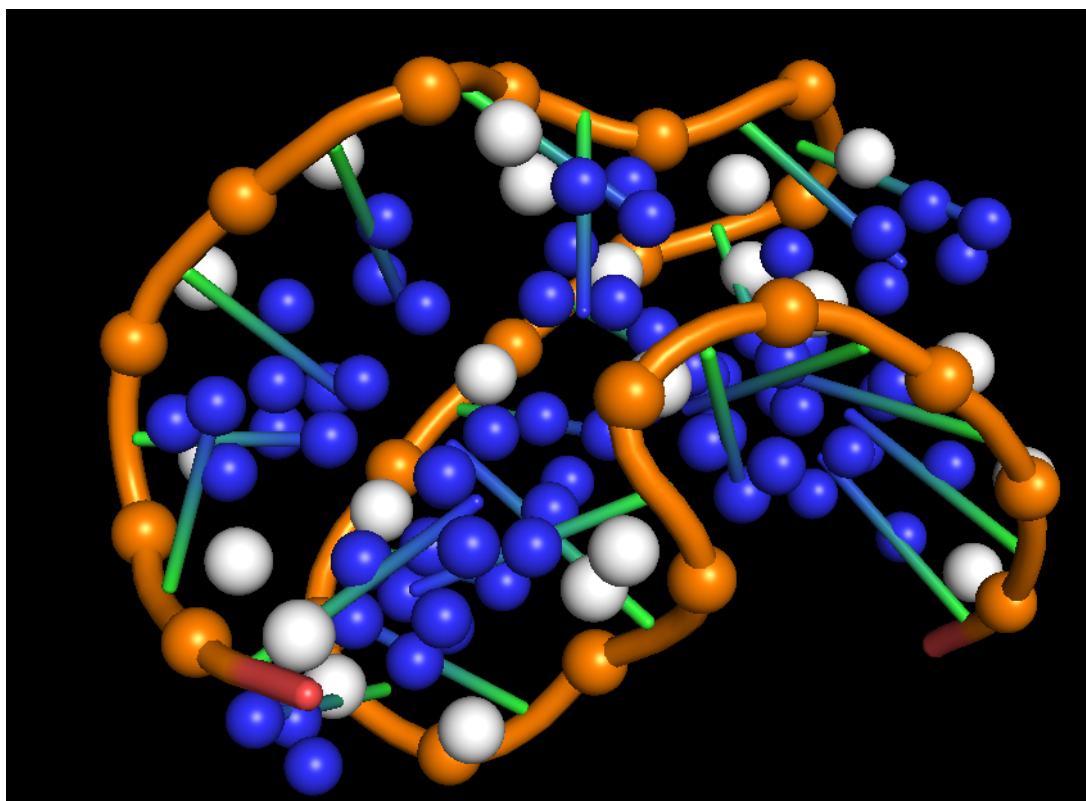


FIGURE 3.6 – Mapping complet de l'ARN ou les bead sont visibles sur une vue 'cartoon' de l'ARN depuis PyMol

J'ai décidé de ce mapping car c'est celui, selon moi, qui propose la vision la plus réaliste de la macromolécule et des degrés de liberté du système biologique, tout en étant en gros grain et relativement facile à mettre en place.

4 Le champ de Force

Tout d'abord, pour le champ de force nous utiliserons deux termes d'énergie correspondant à des interactions entre deux bead non liés chimiquement et un terme de liaisons chimiques.

- Terme de non-liaison chimique
- Terme de liaison chimique

Les termes sont les suivants et la somme de chacun d'eux représente l'énergie libre de la conformation étudiée.

- Potentiel de Lennard-Jones
- Potentiel Électrostatique
- Potentiel Élastique

4.1 Les différents potentiels

Comme on l'a vu dans la Section 2.2, voici la formulation mathématique du champ de force appliqué au système :

$$E_{\text{total}} = \left[\sum_{i=1}^N \sum_{j>i}^N E_{\text{LJ}}(i, j) + \sum_{i=1}^N \sum_{j>i}^N E_{\text{elec}}(i, j) \right]_{\text{non liés}} + \left[\sum_{i=1}^N \sum_{j>i}^N E_{\text{élastique}}(i, j) \right]_{\text{liés}}$$
$$\Leftrightarrow E_{\text{total}} = \left[\sum_{i=1}^N \sum_{j>i}^N 4\epsilon_{ij} \left(\left(\frac{\sigma_{ij}}{r_{ij}} \right)^{12} - \left(\frac{\sigma_{ij}}{r_{ij}} \right)^6 \right) + \sum_{i=1}^N \sum_{j>i}^N \frac{1}{4\pi\epsilon_0\epsilon_r} \frac{q_i q_j}{r_{ij}} \right]_{\text{non liés}} + \left[\sum_{i=1}^N \sum_{j>i}^N \frac{1}{2} k_{ij} (r_{ij} - r_{0,ij})^2 \right]_{\text{liés}}$$

où i et j sont des atomes.

4.2 Les paramètres associés

Les paramètres sont placés dans un fichier JSON à côté. Ce qui permet de modifier les paramètres sans toucher au cœur du programme. Voici les paramètres qui composent le champ de force :

Potentiel élastique $K_{\text{EL}} = 50\text{--}100 \text{ kcal} \cdot \text{mol}^{-1} \cdot \text{\AA}^{-2}$, $R_0 = 1.4\text{--}1.6 \text{ \AA}$ selon les liaisons (Ni–Ni, Ni–RIB, RIB–Pho).

Potentiel de Lennard–Jones $\epsilon = 0.15\text{--}0.25 \text{ kcal} \cdot \text{mol}^{-1}$, $\sigma = 3.5\text{--}4.5 \text{ \AA}$.

Potentiel électrostatique

5 Résultats

Pour commencer on peut prêter attention au profil énergétique de nos conformation. La Figure 5.2 montre dans le plan (X, Y) les différentes conformations de notre ARN, l'énergie associé à ces conformations se trouve dans la dimension Z ensemble les trois dimensions vont créer la surface correspondant au profil énergétique.

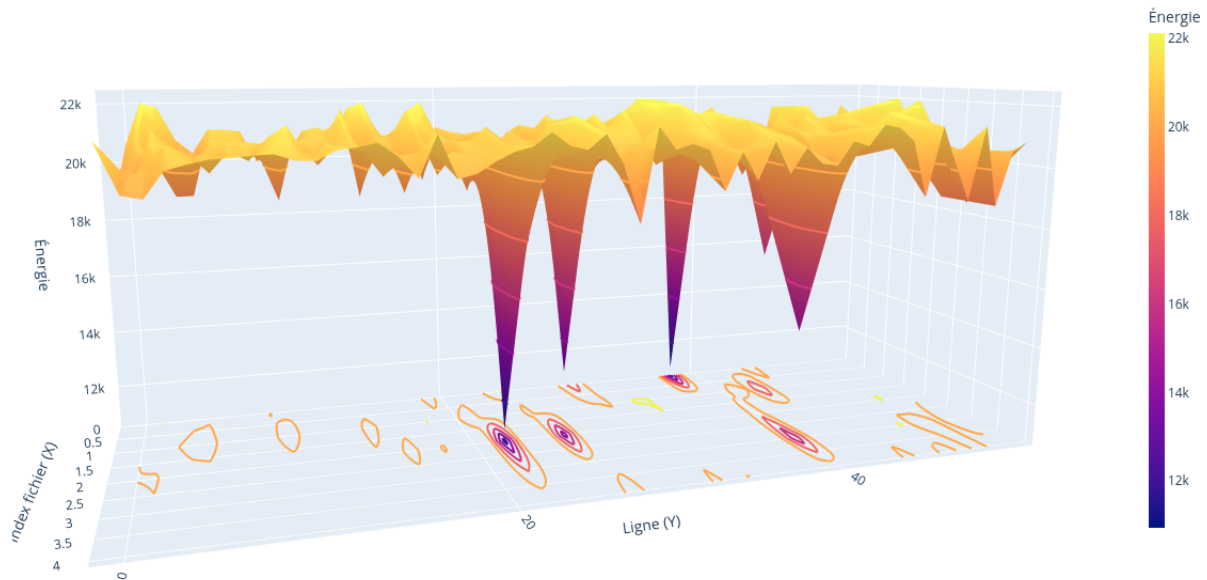


FIGURE 5.1 – Profil énergétique calculer par notre champ de force sur tout les conformations PDB

Cette figure montre que certaines conformation en particulier sont très stables. Les voici dans ce tableau :

Conformation	Energie libre associée
32	10944
201	11010
21	12873
44	14395
56	16002
236	16734

TABLE 5.1 – Tableau des meilleurs conformations de l'ARN et de leur énergie libre associées

Ainsi pour ce champ de force et ce mapping nous allons lancé une méthode de Monte Carlo pour voir comment trouver la meilleure conformation en fonction de la Température et du nombre de pas dans chacune des expériences de Monte Carlo.

Pour ce faire on va généré une heatmap contenant les resultats des 1000 expériences de Monte Carlo pour chaque tuple (Température, nombre de pas) le nombre réel contenue dans chacune des cases de la heatmap correspond a la fréquence de réussite de l'expérience de Monte Carlo. Sachant qu'une expérience réussite correspond a une tentative qui s'est soldé par la trouvaille de la meilleur conformation.

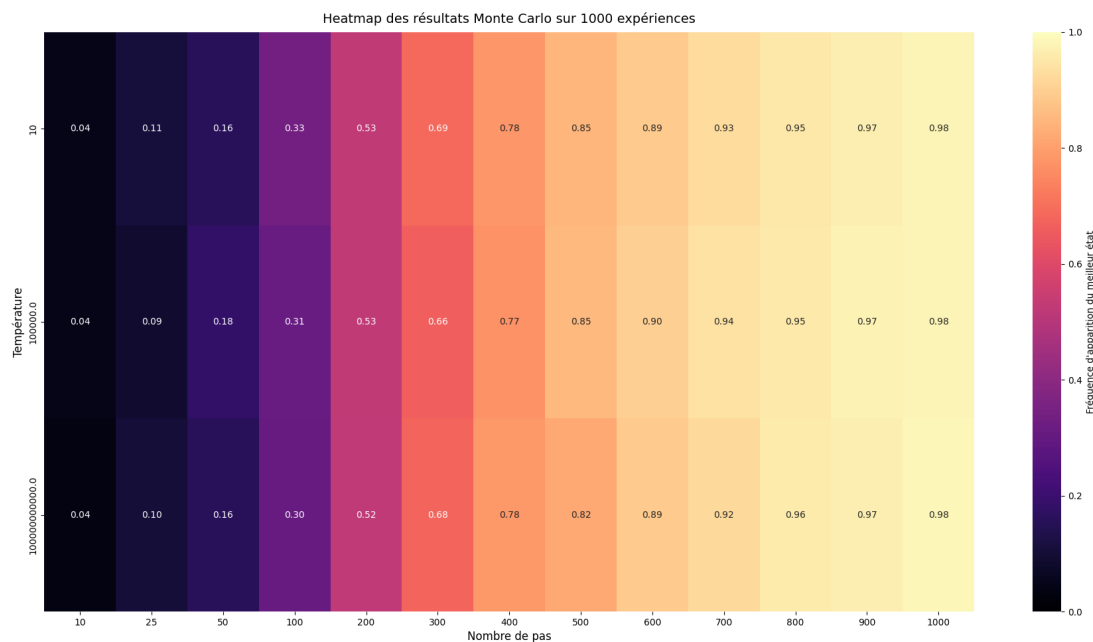


FIGURE 5.2 – Heatmap des résultats des expériences de Monte Carlo pour des Températures et nombre de pas donnés

Premièrement, cette figure montre que la Température n'a aucun impact sur la fréquence de réussite de l'expérience. De plus, on remarque qu'en augmentant le nombre de pas la fréquence augmente drastiquement ce qui est parfaitement logique mais qui indique que l'algorithme fonctionne. Ensuite, le fait que l'expérience de Monte Carlo ne fonctionne pas avec la température est normal car pour cela il aurait fallu générer une grille des conformations et pas à pas chercher les puits de potentiel. Le problème ici est que nous ne pouvons pas mieux réduire nos conformations en un quadrilatère orthonormés que en passant par le rectangle 5x53 ce qui rend la recherche moins complexe de plus les puits sont souvent définis par 1 ou 3 confoamtions maximum ce qui n'aide pas du tout à faire de la recherche par Monte Carlo autrement que par le hasard. En effet, c'est actuellement comme cela que fonctionne mon 'Monte Carlo', il est entièrement basé sur le hasard à cause de la fonction choice.

Peut-être qu'il possible de rendre la méthode de monte carlo plus réaliste en générant une grille des conformations et de définir une boite autour du premier élément cette boite plus petite que le rectangle 5x53. C'est pour cela que j'ai en plus développé les deux fonctions `box` qui définit une boite 1x10 autour d'un indice central et `analyse_T_pas_improved` pour exectuer l'expérience de Monte Carlo de manière plus fonctionnelle mais sans amélioration. Ces améliorations pourront être d'intégrer un scalaire en fonction du gradient pour augmenter ou diminuer la taille de la boite.

6 Discussion