Intelligence Artificielle pour la physique: rapport final - Dynamique moléculaire

Lorris Giovagnoli Pierre Houzelstein

Janvier 2021

Abstract

Dans ce projet, réalisé dans le cadre de l'UE Intelligence Artificielle pour la physique, sous la supervision de MM. Julien Heu et Timothée Devergne, nous approximons la surface d'énergie potentielle du cation Zundel $H_5O_2^+$ à l'aide de réseaux neuronaux, afin d'en pouvoir estimer l'énergie de façon rapide, pour n'importe quelle configuration spatiale. Le but est de s'en servir pour réaliser une simulation de dynamique moléculaire à l'aide d'un algorithme Monte-Carlo.

1 Contexte

Pour réaliser une simulation de dynamique moléculaire, il est nécessaire de connaître la surface d'énergie potentielle (potential energy surface, ou PES) du système étudié, en général un ensemble d'atomes. Il s'agit de connaître l'énergie de ce système en fonction de paramètres d'intérêt, le plus souvent la position des atomes.

Une première approche pour calculer les PES est l'utilisation de méthodes ab initio, utilisant la théorie de la fonctionnelle de la densité (DFT). Celles-ci nécessitent de calculer l'ensemble des forces exercées sur chaque atome, à chaque pas de la simulation, à partir d'une configuration initiale, en partant des lois fondamentales de la physique quantique.

Cette méthode étant computationnellement très coûteuse, elle ne permet pas de réaliser de simulation sur des plages de temps très longues pour un très grand nombre d'atomes: au plus quelques picosecondes pour quelques milliers d'atomes[7]. Pour cette raison, on peut lui préférer l'utilisation de potentiels empiriques, tels que le potentiel de Lennard-Jones par exemple. On a alors accès à une formule physique permettant de calculer facilement les énergies du système. Cependant, selon la situation étudiée, les potentiels empiriques connus peuvent ne pas être adaptés, ce qui amènerait à des résultats numériquement faux. C'est pourquoi l'utilisation de réseaux neuronaux est une solution intéressante.

Un réseau neuronal est un algorithme au fonctionnement inspiré du système cognitif, particulièrement adapté aux problèmes de classification. Il se constitue d'un ensemble de nodes, assimilables aux neurones du cerveau, prenant une valeur numérique comprise entre 0 et 1 qui représente une intensité d'activation. Ces nodes sont organisées en couches

successives, chaque node d'une couche étant individuellement reliée à toutes les nodes des couches précédente et suivante de façon plus ou moins importante (c'est à dire avec un certain poids). Les différentes intensités d'activation des nodes antérieures affectent ainsi les intensités d'activation des nodes postérieures, et ce successivement jusqu'à la couche de sortie du réseau.

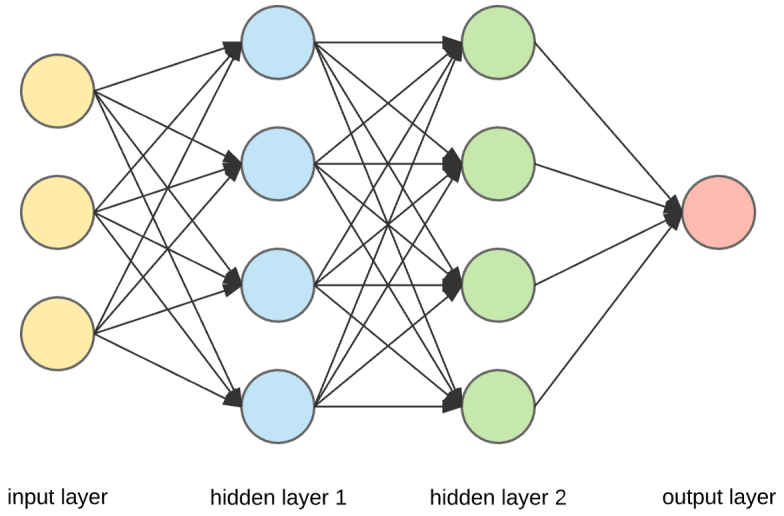

Figure 1: Structure générale d'un réseau neuronal[1]

Pour qu'un tel algorithme soit utile, il faut fixer les poids des connexions entre les neurones. Il est donc nécessaire de passer par une phase d'entraînement du réseau. Pour cela, on utilise un ensemble de données déjà classifiées, ce qui se traduit par un jeu de deux ensembles de valeurs, l'un donnant les entrées, l'autre les sorties. Le réseau commence avec des poids fixés aléatoirement (ou à la main par le développeur); à partir d'une première entrée, il calcule la sortie ainsi obtenue, évalue l'écart de son résultat avec la sortie réelle à l'aide d'une fonction de "loss", et apporte un ajustement aux poids, avant de recommencer avec le couple entrée/sortie suivant. Ainsi, après entraînement avec un grand nombre de données, le réseau peut atteindre une calibration suffisamment fine pour lui permettre de prédire un résultat correct à partir d'une nouvelle entrée.

Par conséquent, à l'aide d'un ensemble de données de départ associant les positions individuelles d'un ensemble d'atomes à l'énergie totale de la configuration, il est possible d'entraîner un réseau neuronal qui fitte le PES de cet ensemble d'atomes. Ainsi, on peut connaître l'énergie totale de n'importe quelle configuration et s'en servir pour une simulation de dynamique moléculaire. Ces données de départ peuvent par exemple provenir d'une simulation de dynamique moléculaire ab initio: de cette façon, il n'y a besoin de réaliser cette simulation longue et coûteuse qu'une seule fois. Il est ensuite possible d'entraîner autant de réseaux neuronaux qu'on le veut, et ceux-ci permettront de calculer rapidement

[1] `https://medium.com/@ksusorokina/image-classification-with-convolutional-neural-networks-496815db12a8`

l'énergie associée à n'importe quelle configuration, ce qui représente donc un gain de temps conséquent.

Dans ce projet, nous avons donc cherché à coder un réseau neuronal qui fitte le PES du cation Zundel $H_5O_2^+$. Cependant, la structure habituelle des réseaux neuronaux telle que décrite plus haut n'est pas entièrement adaptée à notre problème. En effet, nous avons certaines exigences liées à la physique du système. En premier lieu, nous voulons que le réseau conserve les symétries présentes: si on intervertit deux atomes identiques (deux hydrogènes ou deux oxygènes), l'énergie du système reste la même. Or, le réseau décrit plus haut est entraîné avec une structure de système fixe et ne sera pas capable de s'adapter à l'échange.

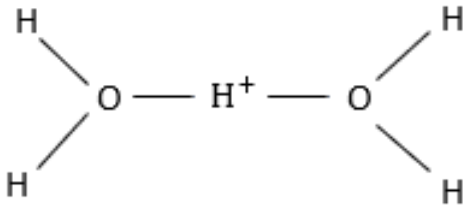

Figure 2: Structure du cation Zundel

De plus, cette structure ne permet pas de déterminer les PES de systèmes de taille différente de celui avec lequel elle a été entraînée. Il serait appréciable de pouvoir réutiliser notre réseau avec des systèmes deux fois, trois fois, n-fois...plus grand que celui de nos données d'entraînement. Pour cela, nous avons utilisé une structure différente et adaptée au problème, que nous décrivons ci-dessous.

2 Traitement des données et génération du réseau

L'ensemble de ce projet a été réalisé en Python. En particulier, le réseau neuronal a été construit grâce à la bibliothèque Keras, suivant la structure exposée dans les graphes ci-dessous.

L'hypothèse fondamentale que nous avons faite est que l'énergie totale de la molécule est décomposable comme une somme des énergies individuelles de chaque atome. Cette hypothèse est très contestable, mais elle permet d'aboutir plus facilement à des résultats utilisables.

Le set de données d'entraînement que nous avons utilisé nous a été fourni par M. Heu. Il se présentait sous la forme de deux fichiers pickle, contenant respectivement les positions des 7 particules, en Ångström, et l'énergie associée, en Hartree, à une température de 100 Kelvin, dans 999990 cas. Ces énergies et positions avaient été obtenues à l'aide de simulations de dynamique moléculaire.

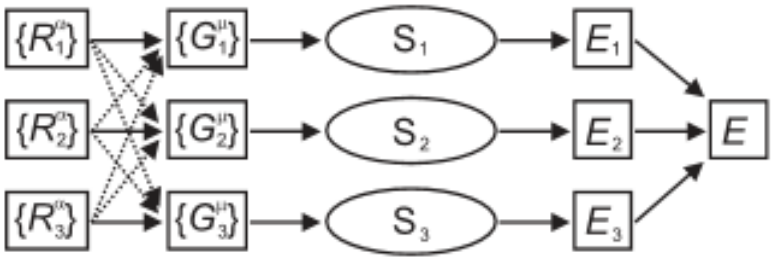

Figure 3: Structure du réseau neuronal utilisé dans un cas à trois particules (d'après Jörg Behler et Michele Parrinello)

Les coordonnées sont utilisées pour calculer les fonctions de symétrie G_i^{μ} décrivant l'environnement local de chacun des atomes, appelées descripteurs. La fonction de transformation utilisée est la fonction SOAP (Smooth Overlap of Atomic Positions)[2], implémentée à l'aide de la bibliothèque DScribe. Ces descripteurs sont composés d'un ensemble beaucoup plus grand de composantes que les simples trois coordonnées initiales de l'espace. Parmi elles, certaines contiennent beaucoup moins d'information pertinente que les autres. On cherche à réduire leur nombre de dimensions, de sorte à faciliter l'entraînement du réseau, par l'analyse en composantes principales ou *Principal Component Analysis* (PCA) en anglais. La PCA permet non seulement de réduire la dimensionnalité des espaces que l'on manipule, mais également et surtout de convertir un ensemble de variables corrélées en un ensemble moins grand de variables décorrélées sur le plan statistique. C'est une transformation qui facilite en général l'entraînement d'un réseau. En effet, l'espace ainsi réduit requiert un nombre de poids à ajuster moindre. Les dimensions conservées étant celles qui ont l'impact le plus grand sur la variance de l'ensemble, on se débarrasse de toute information redondante qui viendrait parasiter l'entraînement du réseau.

Une fois la PCA appliquée il est nécessaire de redimensionner les données nouvellement générées, là encore pour maximiser l'efficacité de l'ajustement. Les réseaux neuronaux se comportent généralement mieux lorsqu'ils travaillent avec un ensemble de données distribuées suivant une loi normale uniformisée.

Il est à noter que lorsque l'on travaille avec des ensembles de données au sein desquelles les différentes dimensions de l'espace sont tout à fait différentes (une taille en mètre par exemple regroupée avec une masse en kilogramme), il est absolument nécessaire de redimensionner les données avant même d'envisager la PCA. Cette dernière s'attache à ne conserver que les dimensions qui ont un impact signifiant sur la variance de l'ensemble, et ces considérations sont faussées dès lors que les différentes dimensions ne sont pas comparables. Dans notre cas néanmoins, toutes nos dimensions sont exprimées dans la même unité et sont donc comparables entre elles. La PCA est alors effectuée sans pré-redimensionnement des descripteurs.

Il est également à mentionner ici une spécificité liée au problème traité. On rappelle que

[2] `https://singroup.github.io/dscribe/latest/tutorials/soap.html`

l'on dispose d'un certain nombre de configurations, et que chacune d'entre elles est constituée des positions individuelles de chacun des atomes. Or, dans le cas du Zundel, il y a deux types d'atomes différents: les atomes d'oxygène et ceux d'hydrogène. Si l'on souhaite conserver une symétrie par inversion des différents éléments, il est nécessaire de les traiter comme similaires lors du redimensionnement et de la PCA. Ainsi, les mêmes coefficients sont utilisés pour redimensionner les deux oxygènes et les cinq hydrogènes respectivement.

Le scaling[3] et la Principal Component Analysis (PCA)[4] de ces données ont été réalisés à l'aide de la bibliothèque sklearn.

Afin de déterminer le nombre idéal de composants à conserver post PCA, la PCA est appliquée une première fois sans réduction de dimension. On regarde dès lors à partir de quelle nombre de composants la variance expliquée dépasse le taux de 99.9999%. C'est alors cette valeur dont on se sert pour réduire la dimension de nos ensembles.

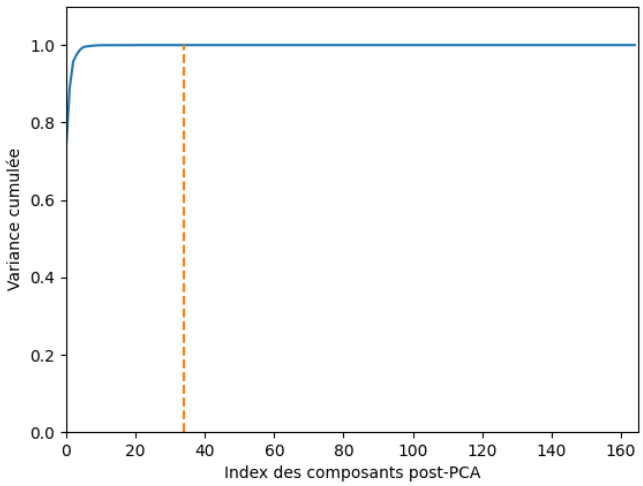

Figure 4: Variance cumulée post-PCA avec en orange l'indice pour lequel on dépasse une valeur de 0.999999

On voit bien sur la figure (4) l'intérêt de la PCA qui dans ce cas permet de réduire le nombre de dimensions d'un facteur 4 environ.

Les énergies sont elles aussi redimensionnées mais de façon beaucoup plus classique. On utilise la classe *MinMaxScaler*, fournit par scikit-learn, appropriée pour les ensembles qui présentent un écart-type faible et qui renvoie des valeurs comprises entre 0 et 1. La formule mathématique utilisée se résume comme suit :

$$X_{scaled} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

[3] `https://scikit-learn.org/stable/modules/preprocessing.html`
[4] `https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html`

Chaque descripteur est transmis à un "sous-réseau". Il y a un type sous-réseau par type de particule. Les sous-réseaux du même type sont calibrés de la même manière. Chaque sous-réseau prend en entrée un descripteur associé à une particule du système, et retourne l'énergie individuelle de cette particule. Dans notre cas, nous avons donc 7 sous-réseaux de 2 types différents (oxygène et hydrogène).

Nous avons utilisé la bibliothèque keras pour la création du réseau. Les deux sous-réseaux mentionnés ci-dessus sont créés à partir de la classe *Sequential* qui comme sont nom l'indique applique ses différentes couches de manière séquentielle et prennent en entrée les descripteurs des différentes particules. Ils disposent chacun de deux couches cachées de 30 neurones et d'une couche de sortie d'un neurone unique qui renvoie ce que l'on considère être la contribution à l'énergie totale d'un élement. Cette valeur n'a néanmoins aucun sens physique réel. Toutes ces valeurs sont enfin additionnées pour renvoyer l'énergie totale associée à la molécule en sortie.

La molécule Zundel étant constituée de deux types d'atomes différents (Hydrogène et Oxygène, en approximant le proton comme un Hydrogène), chacun des types d'éléments passent à travers l'un des deux sous-réseaux.

La figure (5) présente schématiquement la structure décrite.

Nous avons ensuite optimisé le réseau neuronal en ajustant les paramètres par recherche bayésienne à l'aide d'hyperopt.

3 Résultats

Lors de la création d'un réseau neuronal, on sépare les données en ensemble d'entraînement et de validation d'une part, et en ensemble d'entraînement d'autre part. Les premier servent respectivement à entraîner le réseau et à en optimiser les différents paramètres, le second à évaluer l'efficacité du réseau mis face à des données inconnues.

Le premier résultat que nous avons cherché à obtenir est donc naturellement une comparaison des résultats réels et prédits par notre réseau sur les train et test sets. En entraînant notre réseau sur l'ensemble des 999990 configurations disponibles à 100K, nous obtenons les résultats exposés (6) et (7).

La fonction de calcul de la loss utilisée est la fonction Mean Squared Error (MSE):

$$MSE = \frac{1}{n}\sum_{i=1}^{n}[E_i - \hat{E}_i]^2, \tag{1}$$

On constate que notre réseau semble capable de classifier efficacement les configurations avec une énergie totale comprise entre 0 et 0.008 Hartree, et termine avec une loss tendant vers 0. Les hypothèses que nous avons faites (l'énergie totale égale à une somme d'énergie individuelle et le proton centrale équivalent aux hydrogènes) paraissent alors justifiées, ou tout du moins permettent d'obtenir des résultats corrects.

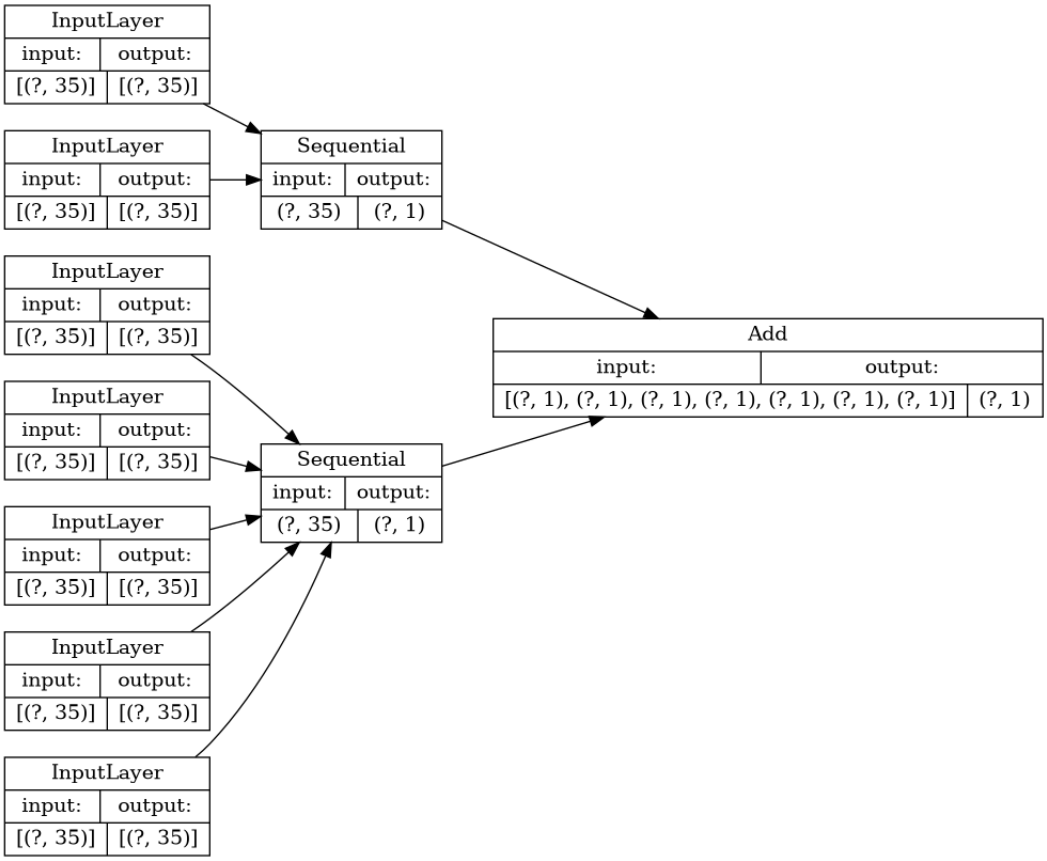

Figure 5: Structure de notre réseau: "?" correspond au nombre de configurations transmises au réseau en entrée et 35 est la dimension des descripteurs SOAP réduite par PCA (dans ce cas précis)

Notre premier objectif est donc atteint: nous avons un réseau capable d'assigner correctement une énergie à une configuration Zundel. Il reste alors à l'utiliser pour des simulations de dynamique moléculaire, et voir s'il est capable de donner des résultats.

Pour tester l'efficacité de notre réseau, nous l'avons utilisé pour réaliser une simulation Monte-Carlo. Notre algorithme Monte-Carlo cherche à reproduire la distribution de Boltzmann: partant d'une configuration initiale prise aléatoirement parmi les configurations connues, l'algorithme propose une nouvelle configuration spatiale de façon aléatoire et incrémentale à l'intérieur d'une boîte de côté delta défini par l'utilisateur; il se sert ensuite du réseau pour calculer l'énergie de cette configuration. Si elle est compatible avec la distribution de Boltzmann, la nouvelle configuration est acceptée, et on considère que le système l'adopte. Sinon, elle est refusée, et le système reste dans sa configuration initiale. On enregistre la position et l'énergie et on réitère, l'idée étant qu'après un grand nombre d'étapes on s'attend à retrouver une distribution semblable à celle d'une simulation de dynamique moléculaire ab initio, selon le principe d'ergodicité. Si c'est bien le cas, on voit l'intérêt de notre travail: grâce au réseau neuronal, on a une méthode de simulation de dynamique

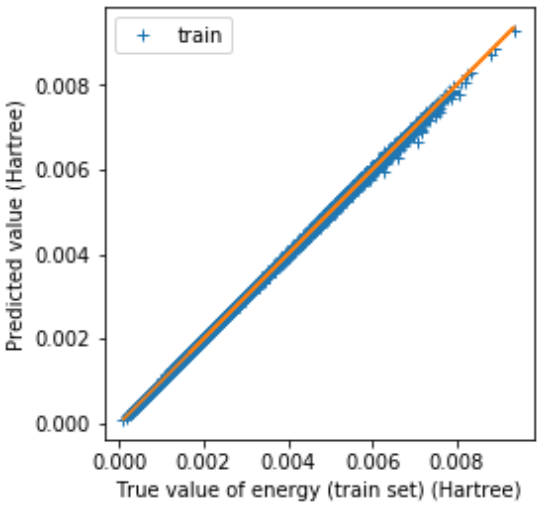

Figure 6: Prédictions sur le train set

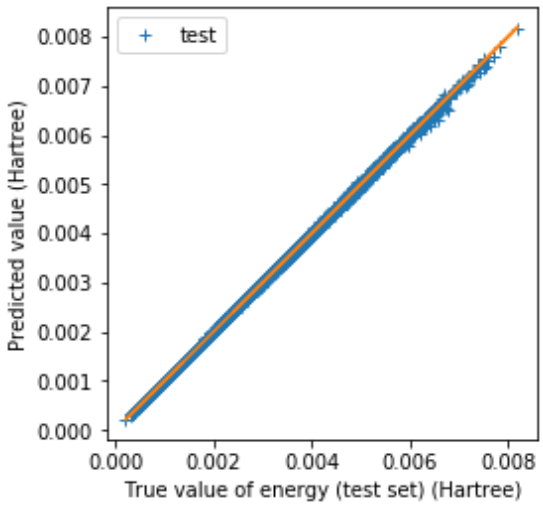

Figure 7: Prédictions sur le test set

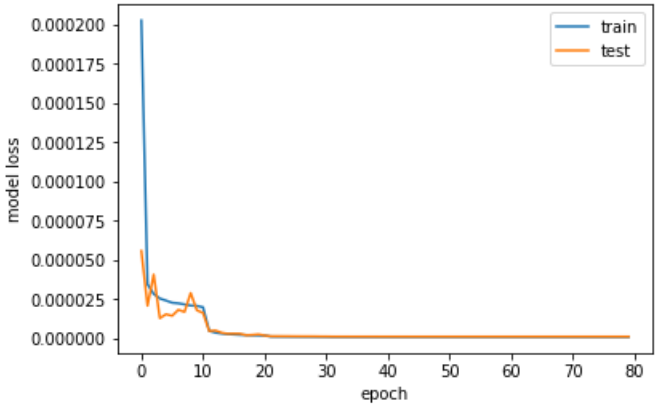

Figure 8: Évolution de la loss du réseau au cours de l'entraînement

moléculaire rapide et efficace.

Malheureusement, nous ne sommes pas parvenus à reproduire les distributions telles qu'obtenues par DFT dans les fichiers qui nous ont été fournis. Puisque notre réseau semble capable de prédire les énergies de façon efficace, nous pensons que l'erreur se trouve plutôt au niveau de l'exécution de notre Monte-Carlo. Il est possible également qu'il y ait une erreur de programmation dans les fonctions de redimensionnement ou chargées de faire la PCA sur les énergies à chaque pas du Monte-Carlo. Cependant, après vérification nous n'avons pas su localiser le problème. Cependant, même si le code ne donne pas de résultats physiques, nous avons pu constater que sa vitesse d'exécution était très rapide (environ deux heures sont nécessaires pour tester 100000 configurations sur un ordinateur personnel). Si le programme arrive à être corrigé pour donner des résultats utilisables, il permettra alors de gagner du temps lors de la réalisation de simulations de dynamique moléculaire

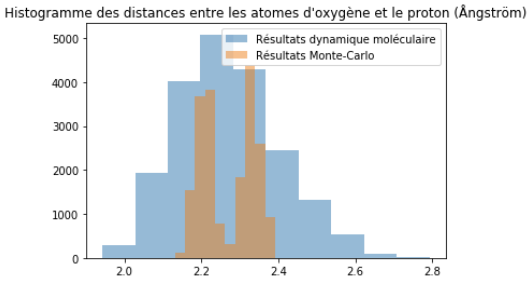

Figure 9: Résultats d'une simulation Monte-Carlo: distances entre les oxygènes et le proton (10000 itérations, delta = 0.03 Hartree)

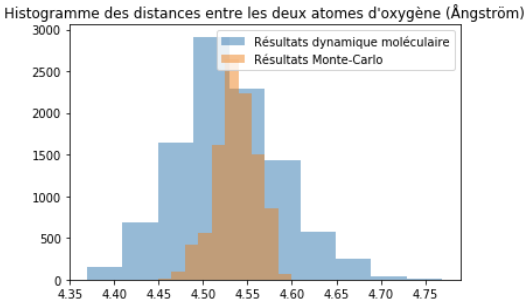

Figure 10: Résultats d'une simulation Monte-Carlo: distances entre les oxygènes et le proton (10000 itérations, delta = 0.03 Hartree)

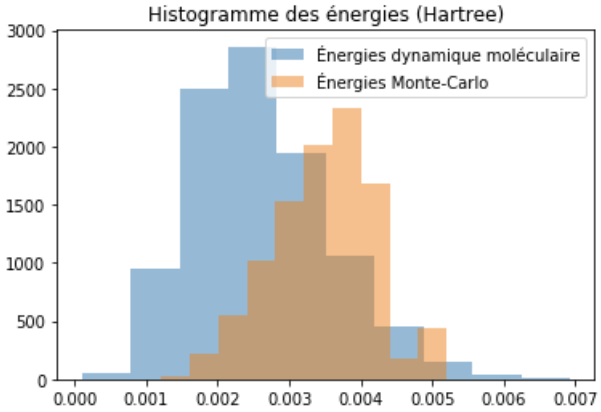

Figure 11: Résultats d'une simulation Monte-Carlo: distribution des énergies (10000 itérations, delta = 0.03 Hartree)

4 Conclusion

En commençant ce projet, nous avions deux objectifs: coder un réseau neuronal capable de prédire les énergies d'un Zundel à partir de sa configuration spatiale, et s'en servir pour réaliser une simulation efficace de dynamique moléculaire. Le premier objectif est atteint, le second reste pour l'instant en suspens. Nous souhaitons aussi noter que le réseau réalisé a pour vocation d'être généralisable à un nombre et une combinaison arbitraires d'atomes, tant que les données nécessaires sont disponibles pour l'entraîner. Cependant, nous n'avons pas eu le temps de le tester avec d'autres ensembles de données, nous ne savons donc pas encore s'il en est capable en l'état.

References

[1] Pankaj Mehta et al. *A high-bias, low-variance introduction to Machine Learning for physicists*. May 2019.

[2] *Documentation Atomic Simulation Environment (ASE)*. URL: `https://wiki.fysik.dtu.dk/ase/index.html` (visited on 01/06/2021).

[3] *Documentation DScribe*. URL: `https://singroup.github.io/dscribe/latest/` (visited on 01/06/2021).

[4] *Documentation Hyperopt*. URL: `https://hyperopt.github.io/hyperopt/` (visited on 01/06/2021).

[5] *Documentation Keras*. URL: `https://keras.io/` (visited on 01/06/2021).

[6] *Documentation Scikit-learn*. URL: `https://scikit-learn.org/stable/index.html` (visited on 01/06/2021).

[7] Jörg Behler et Michele Parrinelol. "Generalized Neural-Network Representation of High-Dimensional Potential-Energy Surfaces". In: *Physical Review Letters* 98 (Apr. 2007).