Compte-rendu de Simulation atomistique des matériaux

- Dynamique moléculaire -

M2 Physique Numérique & Chimie théorique - 2024/2025

Dépôt Github du projet :https://github.com/RemNX/Atomistique

Rémy DUPOND, Khaled MADANI, Adrien TOULOUSE, Manon NOLOT.

Table des matières

1 Introduction

La dynamique moléculaire est une méthode de simulation numérique permettant de modéliser et d'analyser le mouvement et les interactions des atomes et des molécules dans un système. Basée sur les lois de la mécanique classique, en particulier les équations de Newton, elle offre une perspective microscopique précieuse en permettant de suivre l'évolution temporelle d'un système. Dans notre travail, nous nous concentrons sur la simulation de systèmes composés de plusieurs particules interagissant selon le potentiel de Lennard-Jones, en résolvant les équations de mouvement pour chaque particule. Nous analyserons ensuite les résultats obtenus pour calculer des grandeurs physiques essentielles et vérifier la validité de notre code de simulation.

2 Mise en place de la simulation

Afin de mettre en place une simulation de dynamique moléculaire, il est nécessaire de bien mettre en place les paramètres de la simulation comme le domaine de la simulation, le potentiel qui régis les particules et les paramètres de celui-ci et enfin la méthode d'intégration pour la mise à jour des particules.

Avant toute chose, il est important de préciser que, pour éliminer le plus de paramètres superflus possible comme ϵ et σ, nous avons adimensionné le problème ainsi on peut poser la masse des particules $m = 1$ et les paramètres du potentiels de Lennard-Jones $\epsilon = 1$ et $\sigma = 1$.

Cela signifie que, dans un système adimensionné, les unités des différentes grandeurs physiques sont exprimées comme suit :

— La longueur en σ ,
— Le temps en $\tau = \sqrt{m\sigma^2}/\epsilon$,
— La température en ϵ/k_b,
— L'énergie en ϵ,

Et ainsi de suite pour d'autres grandeurs physiques.

Pour commencer, voici les convention que nous avons utiliser pour les constantes de la simulation :

1. Le nombre de particule : N
2. Le nombre de dimension : $d = 2$
3. La taille du domaine : on choisit une boite carré de taille L

Concernant le potentiel nous avons choisit le potentiel de Lennard-Jones tronqué et décalé en énergie. Nous avons choisit ce potentiel car il a un comportement qui nous intéresse. Il est attractif à longue distance mais répulsif à courte distance. Cela nous permet de faire une assez bonne approximation des forces réelles (électromagnétique, de Van de Waals, liaisons covalentes...). Le potentiel de Lennard-Jones est décrit par la relation suivante :

$$\begin{cases} U_{LJ}(r) = 4\left[(\frac{1}{r*})^6 - (\frac{1}{r*})^{12}\right] - U_{LJ}(R_c) & r \leq R_c \\ \\ U_{LJ}(r) = 0 & r > R_c \end{cases} \tag{1}$$

Ici $R_c = 2.5$ (en réalité $R_c = 2.5\sigma$ mais étant en adimensionné, $\sigma = 1$).Ce tronquage du potentiel fausse légèrement l'énergie, nous allons donc devoir ajouter un terme de correction de l'énergie, chose que nous détaillerons plus loin.

Ensuite, nous avons choisit d'utiliser la méthode d'intégration dite de "Velocity-Verlet" cette méthode est bien plus précise que celle d'Euler et n'est pas très chère en terme de rapidité ou complexité de calcul. Voici les équations utilisées :

$$r(t+\delta t) = r(t) + v(t)\delta t + \frac{1}{2}f(t)\delta t^2 \tag{2}$$

$$v(t+\delta t) = v(t) + \frac{1}{2}[f(t+\delta t) + f(t)]\delta t \tag{3}$$

Enfin, pour éviter les effets de bords et afin d'être capable de simuler des environnement "infinis", nous utilisons la convention de l'image périodique la plus proche. C'est à dire que nous considérons des boites fictives exactement identiques à celle qu'on simule et qui l'entoure. Les particules de ces boites fictives sont capable d'interagir avec la boite qu'on simule.

3 Considérations théoriques

Comme dit un peu plus haut nous avons fait certaines approximations numériques qui entraînent des erreurs plus ou moins importante.

On peut notamment parler des erreurs qui découlent du tronquage du potentiel de Lennard-Jones. Ces corrections sont dites "de queue" et ont une influence sur l'énergie totale du système et sur la pression. On peut démontrer que pour la température et la pression on a les corrections suivantes à calculer :

$$U_{tail} = \frac{1}{2}N\int_{R_c}^{\infty} U_{LJ}(r)d^2r \tag{4}$$

$$P_{tail} = \frac{1}{d}\frac{1}{2}(\frac{N^2}{L^2})\int_{R_c}^{\infty} \overrightarrow{r}.\overrightarrow{f}(\overrightarrow{r})d^2r \tag{5}$$

$$\tag{6}$$

Après intégration en deux dimensions on obtient :

$$U_{tail} = \pi\epsilon\frac{N^2}{L^2}(\frac{2}{5}\frac{1}{R_c^{10}} - \frac{1}{R_c^4}) \tag{7}$$

$$P_{tail} = 6\pi\epsilon\frac{N^2}{L^2}(\frac{2}{5}\frac{1}{R_c^{10}} - \frac{1}{2}\frac{1}{R_c^4}) \tag{8}$$

Conditions aux bords periodiques

Les problèmes de conditions aux bords jouent un rôle essentiel dans les simulations de systèmes à N particules, et leur traitement est crucial pour obtenir des résultats précis et réalistes Pour gérer cela, nous devons établir des conditions sur les particules et leurs interactions afin d'obtenir un résultat valide. Dans ce travail, nous utilisons les conditions aux bords périodiques, qui consistent à considérer un système fini se répétant à l'infini(Figure 1).

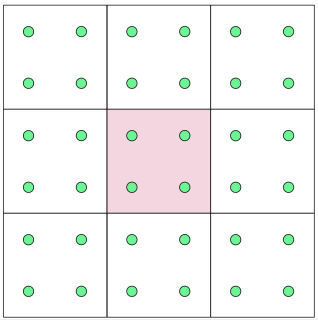

FIGURE 1 – Schéma pour les copies periodiques montrant le système en rose et les copies autour

Chaque réplique se comporte exactement de la même manière que le système défini initialement, avec les mêmes interactions et les mêmes mouvements.

En pratique, cela signifie que si une particule se déplace vers la droite selon l'axe x, toutes les copies font de même. Au moment où la particule atteint le bord du système, elle sort et sa copie entre du côté opposé. Cette idée s'applique de la même manière, quelle que soit la direction.

Ce qui est aussi intéressant, c'est que nous allons également poser des conditions sur les interactions. Cela signifie que nous ne calculerons pas les forces qu'entre les particules du système, mais nous allons vérifier si une réplique est plus proche que celle-ci de notre particule de référence. Nous avons comparé les distances selon x et y par rapport à $L/2$, L étant l'arête de la boîte. Ainsi, si nous prenons une particule i et une particule j, nous vérifions que $x_j - x_i < L/2$. Sinon, nous prenons la particule j d'une autre réplique, et il en va de même pour y.

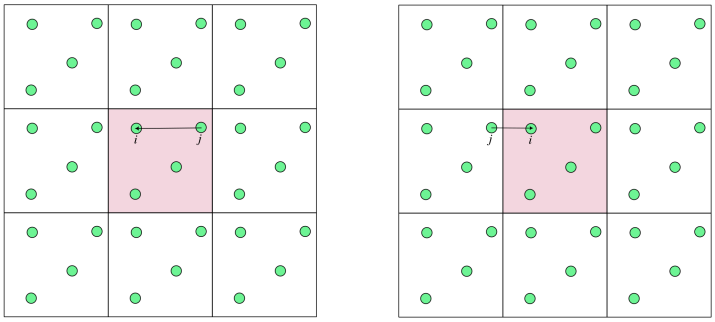

FIGURE 2 – Figure montrant l'interaction entre une particule i du système et une particule j d'une réplique à la place de celle du système.

4 Mise en place du code

Avant de commencer le développement du programme, nous avons élaboré un schéma principal(Figure 3) afin d'organiser notre travail. Nous avons veillé à maintenir une structure à la fois

simple, compréhensible et flexible. Cela signifie que, si des modifications s'avèrent nécessaires, il suffira d'ajuster certaines sections sans avoir à réviser l'intégralité du programme.

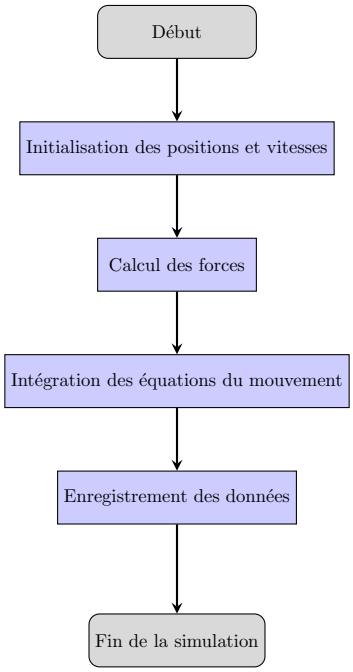

FIGURE 3 – Schéma de l'algorithme de simulation de dynamique moléculaire, montrant les étapes principales

Ensuite, nous avons développé un prototype en Python basé sur ce schéma, ce qui nous a permis de vérifier les premiers résultats et la structure de notre modèle. Par la suite, nous avons transformé cette première version en C.

4.1 Définition de constantes

Pour optimiser le code et gagner en temps de calcul, nous avons évité de recalculer des termes constants ou de redéfinir des valeurs fixes. Ainsi, nous avons utilisé la directive de préprocesseur `#define`, qui remplace du texte avant la compilation. Cela nous aide à avoir des définitions à portée globale dans le programme, afin d'éviter les erreurs et les redéfinitions, comme mentionné précédemment.

Nous avons également utilisé `const double`, qui nous sert à définir des termes constants, mais nous l'avons réservé pour des définitions plus complexes, comme le calcul de `U_tail`, par exemple.

4.2 Programmation orienté Objet

Nous avons convenu dès le départ qu'il était plus structuré et organisé d'utiliser des classes. Nous avons donc mis en place une classe `Particule`, qui possède comme attributs les positions, les

vitesses et les forces appliquées. Nous avons également créé un constructeur permettant d'initialiser les positions et vitesses des particules. Ainsi, il est possible de créer un tableau de particules et de les initialiser selon nos besoins, que ce soit de manière aléatoire ou sous forme cristalline.

Nous avons ajouté les méthodes principales : `calcul_forces()` qui calcule les forces appliquées sur chaque particule et modifie les attributs associés, et `resolution()` qui effectue l'intégration numérique à l'aide de la méthode de Velocity-Verlet ou d'Euler, apportant ainsi des modifications aux attributs de positions et de vitesses.

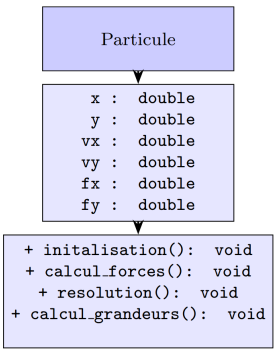

FIGURE 4 – Diagramme UML représentant la classe 'Particule', incluant ses attributs (position, vitesse, force) et ses méthodes .

En langage C, nous traitons les classes à l'aide de **pointeurs**, qui permettent d'accéder directement aux attributs. Il est important de noter que le C n'est pas un langage orienté objet comme Java ou C++. Par conséquent, la définition des constructeurs et des méthodes ne se fait pas de la manière habituelle. En effet, nous devons passer l'instance de la "classe" en argument pour pouvoir appliquer une méthode, ce qui diffère des langages orientés objet où cela se fait implicitement.

En C, nous définissons ce que l'on appelle des **structures** plutôt que des classes. Bien que cela entraîne une différence de syntaxe, la conception orientée objet peut toujours être mise en œuvre. Ainsi, même si le langage C n'offre pas les mêmes fonctionnalités qu'un langage orienté objet, il permet néanmoins d'organiser le code de manière modulaire et structurée.

4.3 Conditions aux bords periodiques

Pour implémenter les conditions aux bords périodiques pour le mouvement, nous avons modifié la partie du code qui résout les équations du mouvement. Nous avons ajouté une condition qui vérifie si une particule dépasse les limites de la boîte elle réapparaisse de l'autre côté. En d'autres termes :

— Si `Particule.x > L/2` alors `Particule.x -= L`

— Si `Particule.x < - L/2` alors `Particule.x += L`

Et on applique le même raisonnement selon y.

Pour les interactions, nous avons modifié la partie qui calcule les forces mouvements à fin vérifier que :

Les différences de coordonnées entre deux particules i et j :

$$\Delta x = x_j - x_i$$

$$\Delta y = y_j - y_i$$

Si $|\Delta x| > L/2$ alors :

$$\Delta x = -\frac{\Delta x}{|\Delta x|} \cdot (L - |\Delta x|)$$

De même, si $|\Delta y| > L/2$:

$$\Delta y = -\frac{\Delta y}{|\Delta y|} \cdot (L - |\Delta y|)$$

Ainsi nous n'avons pas besoin de créer des particules supplémentaires ou quoi que ce soit mais juste de changer les distances et de calculer les forces en utilisant Δx et Δy.

4.4 Optimisation

Nous avons laissé cette section en dernier pour éviter les erreurs et les bugs inattendus. Les changements apportés étaient principalement basés sur les éléments que nous avons appris dans la partie avancée de ce module. Voici une petite liste des modifications effectuées :

- Définition des termes constants une seule fois.
- Éviter l'appel de fonctions multiples.
- Remplacement des calculs de type `pow(rij, 6)` $\longrightarrow$ `rij2 * rij2 * rij2`.
- Création de fonctions plus générales, comme celles pour calculer toutes les grandeurs en même temps.
- Choix d'un Δt adapté (que nous allons discuter dans la section suivante).

5 Vérifications effectuées

Pour vérifier si notre simulation produisait des résultats acceptables, nous avons envisagé plusieurs cas de test différents. Cependant, cette étape reste difficile, car certaines vérifications peuvent indiquer des comportements cohérents sans garantir pour autant que le code soit entièrement correct.

5.1 Système simple

Nous savons qu'avec le potentiel de Lennard-Jones, deux particules proches s'attirent en raison de l'interaction attractive à longue portée. Cependant, à mesure que les particules se rapprochent, cette attraction atteint un seuil où la répulsion devient dominante. Pour valider ce phénomène de manière simple et visuelle, une vérification préliminaire peut être réalisée à l'aide d'une animation. Cela permettrait d'observer en temps réel l'évolution des forces entre les particules et de confirmer l'interaction attractivo-répulsive, tout en offrant une vue claire du comportement des particules au fur et à mesure de leur mouvement.

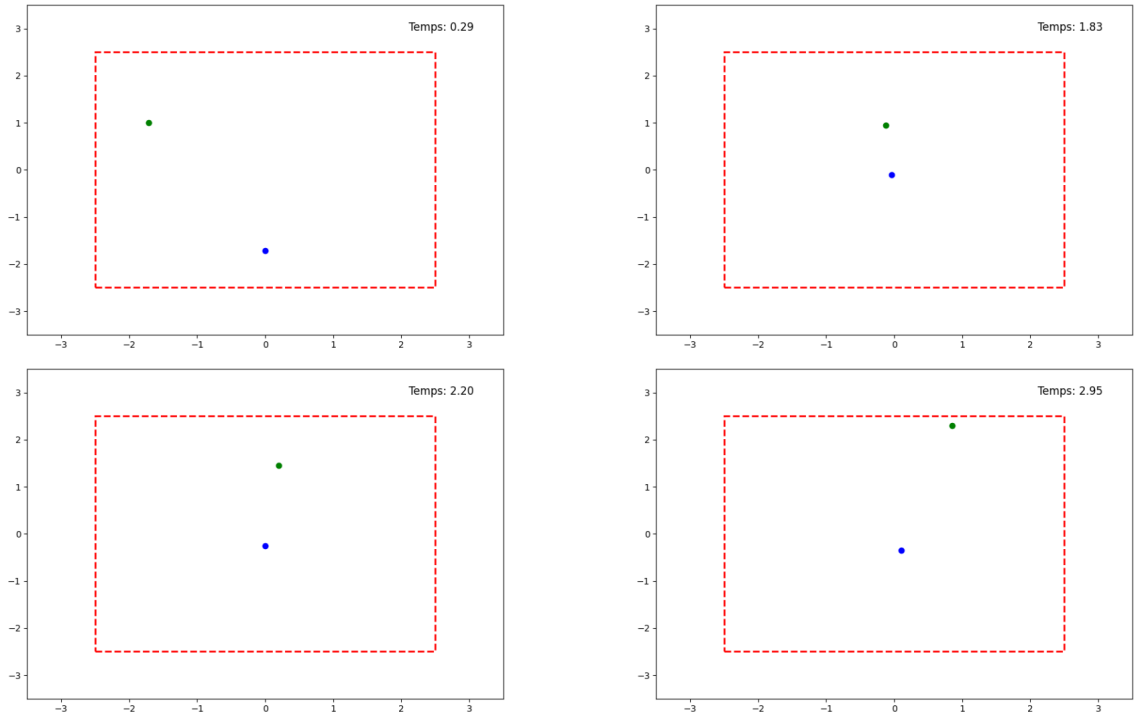

FIGURE 5 – Figure montrant l'interaction en Lennard Jones entre deux particules.

Nous avons observé qu'au début, les deux particules se sont rapprochées en raison de l'attraction, suivie d'une répulsion qui a modifié leurs trajectoires respectives, l'attraction devenant alors moins prononcée. Cette évolution confirme que le comportement attendu du potentiel de Lennard-Jones se manifeste bien dans notre simulation. Cependant, bien que cette vérification constitue un bon point de départ, nous sommes conscients qu'il ne s'agit que d'une validation préliminaire. Ce test visuel permet de s'assurer du comportement de base des particules, mais ne représente pas une validation rigoureuse et complète du modèle.

5.2 Conditions aux bords périodiques

Dans cette partie, nous allons simplifier le modèle en n'utilisant que deux particules. Cela nous permettra de mieux observer et analyser les interactions attendues dans un cadre plus épuré. En limitant le nombre de particules, nous pourrons vérifier plus facilement si les conditions aux bords périodiques fonctionnent comme prévu.

Positions

Pour vérifier le bon fonctionnement des conditions aux bords périodiques — c'est-à-dire que lorsqu'une particule sort de la boîte, elle réapparaît de l'autre côté — nous pouvons effectuer une expérience simple en simulant le mouvement de deux particules. Une particule se déplaçant selon x

reçoit une vitesse initiale (v_x,0) tandis qu'une autre se déplaçant selon y reçoit une vitesse initiale (0,v_y).

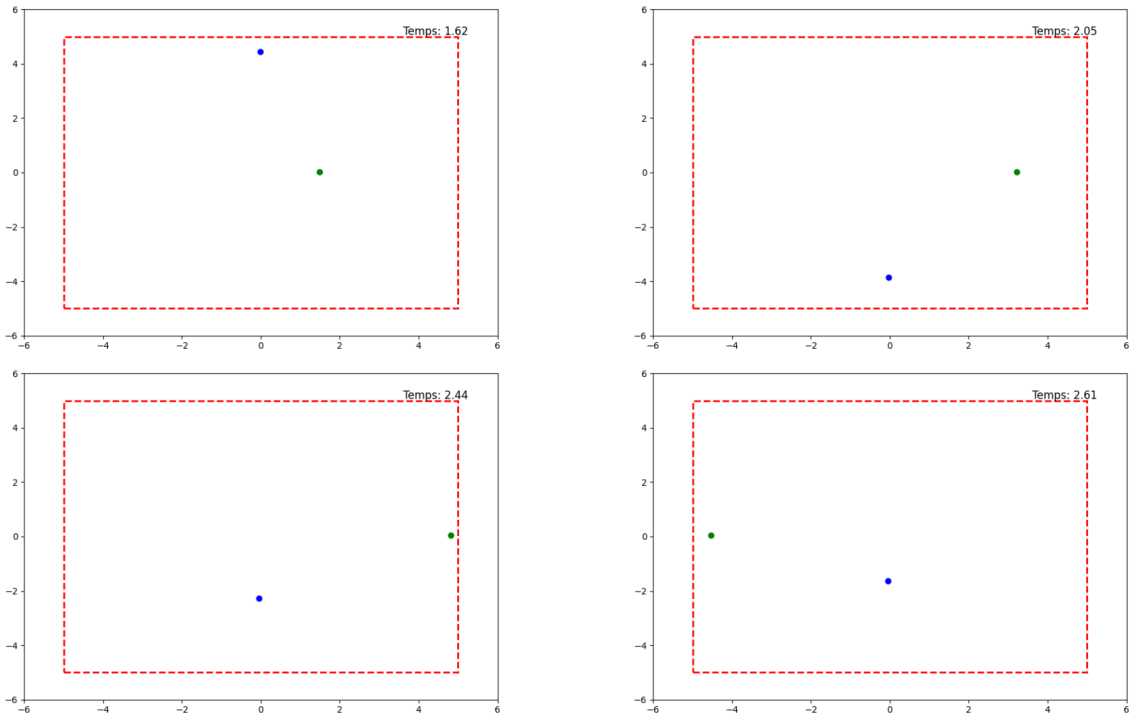

FIGURE 6 – Figure montrant l'effet des conditions aux bords sur les positions.

Comme le montrent les figures ci-dessus, nous pouvons observer que les conditions aux bords périodiques sont correctement implémentées. En effet, lorsque les particules sortent d'un côté de la boîte, elles réapparaissent instantanément de l'autre côté. Cela garantit que les particules se déplacent de manière continue à travers l'espace simulé sans interruption, validant ainsi le bon fonctionnement des conditions aux bords.

Interactions

Lorsque deux particules s'éloignent suffisamment l'une de l'autre, elles ne se "voient" plus directement. Cependant, en raison des conditions périodiques aux bords, chaque particule devrait ressentir l'influence de la force exercée par la réplique de l'autre particule, située de l'autre côté du domaine. Cela signifie que, bien qu'une particule soit éloignée de la particule réelle, elle sera toujours influencée par sa réplique au travers des bords périodiques.

Pour vérifier notre hypothèse, nous choisissons un scénario dans lequel les deux particules interagissent non seulement entre elles, mais également avec leurs répliques périodiques dans l'espace. Nous plaçons initialement les deux particules aux coordonnées (-1,0) et (1,0) avec des vitesses respectives de (-1,0) et (1,0), c'est-à-dire que chaque particule se dirige vers l'extérieur, l'une vers la

gauche et l'autre vers la droite.

Ainsi, dans notre configuration, la particule se dirigeant vers la gauche devrait, à un certain moment, ressentir une force qui la repousse vers la droite, en réponse à l'interaction avec sa réplique. Inversement, la particule se dirigeant vers la droite devrait également ressentir une force qui la repousse vers la gauche, sous l'influence de la réplique de la particule gauche.

En lançant la simulation, nous obtenons les résultats suivants :

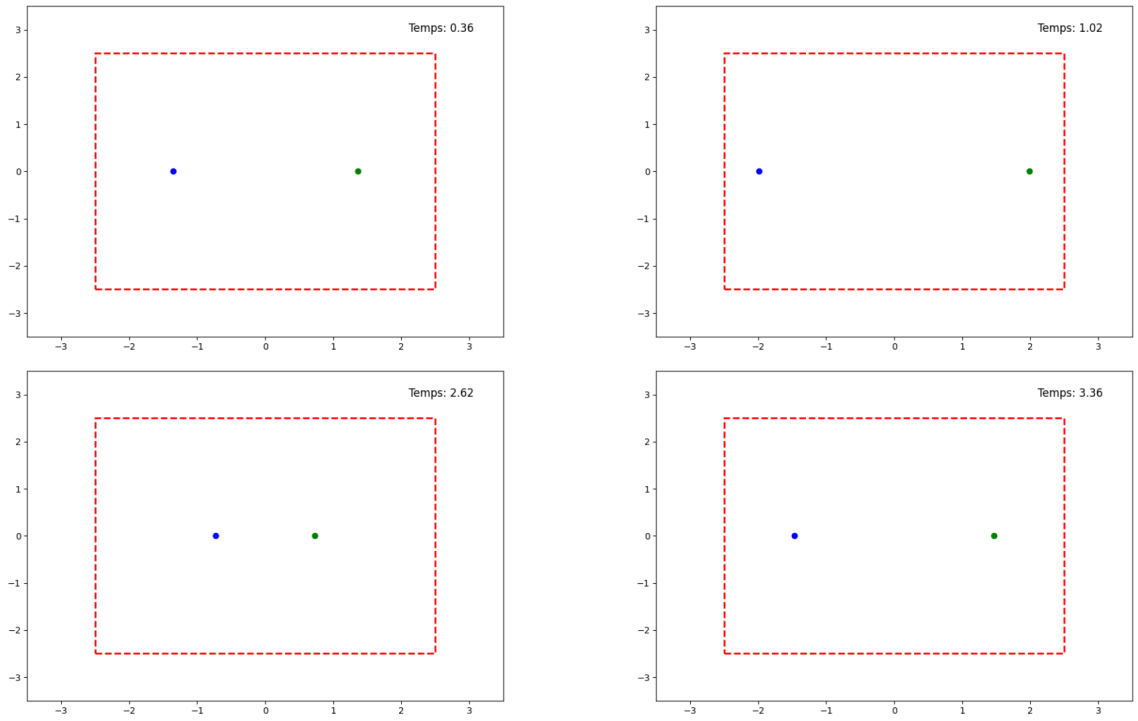

FIGURE 7 – Figure montrant l'effet des conditions aux bords sur les interactions.

Comme le montrent les figures ci-dessus, nous observons bien une interaction entre les particules et leurs répliques, indiquant que les conditions aux bords périodiques sont correctement appliquées pour les interactions.

5.3 Conservation de l'énergie mécanique

Nous disposons également d'une bonne méthode pour vérifier que le code ne produit pas d'erreur, en utilisant la conservation de l'énergie totale, qui est un principe fondamental en physique.

Exemple 1

Dans un système de 100 particules initialement disposées dans une configuration cristalline et sans vitesses initiales, on s'attend à ce que l'énergie totale reste constante au cours du temps. La disposition initiale des particules est uniformément espacée de 2 unités dans les directions x et y, et elles sont confinées dans une boîte de 20 unités de côté.

Dans cette configuration, l'unique contribution à l'énergie totale provient de l'interaction de Lennard-Jones entre les particules, car elles sont supposées ne pas avoir d'énergie cinétique (les vitesses initiales étant nulles). Par conséquent, chaque particule subit des forces d'interaction de ses voisines. Cependant, du fait de la symétrie de la disposition cristalline, les forces résultantes s'annulent pour chaque particule. Cela signifie qu'il n'y a aucun mouvement net des particules, et le système reste statique dans cette configuration initiale.

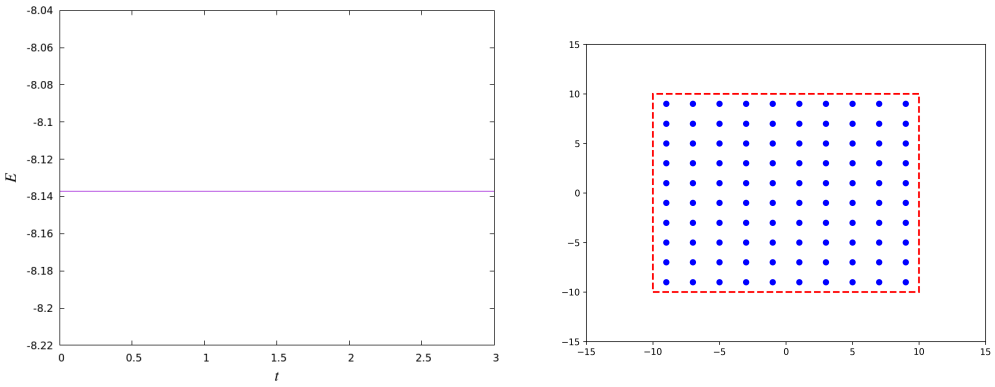

FIGURE 8 – Figure montrant la conservation de l'énergie totale du système pour une configuration cristalline.

Exemple 2

Dans un système composé de 100 particules initialement disposées dans une configuration aléatoire et sans vitesses initiales, on s'attend également à ce que l'énergie totale du système reste constante au cours du temps. Les particules sont réparties de façon à garantir une distance minimale de 1 unité entre chaque paire de particules et sont confinées dans une boîte de 80 unités de côté.

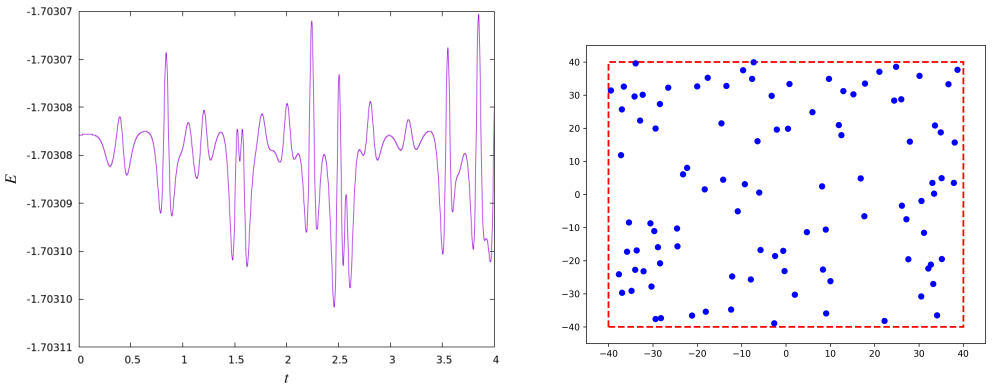

FIGURE 9 – Figure montrant la conservation de l'énergie totale du système pour une configuration aléatoire.

Il est important de noter que cette méthode de vérification est complexe, car de nombreuses

erreurs numériques peuvent affecter le calcul de l'énergie totale du système. Cela rend difficile de déterminer si la non-conservation exacte de l'énergie est due aux erreurs numériques ou aux limitations de l'algorithme. Nous pensons également qu'il serait préférable de réduire le nombre de particules pour cette vérification, car cela diminue le nombre de calculs nécessaires et, par conséquent, réduit les sources potentielles d'erreur.

5.4 L'Exposant ν : Caractérisation de l'Algorithme

Nous avons appris en cours que pour les méthodes d'intégration, telles qu'Euler et Velocity-Verlet, l'exposant ν prend respectivement les valeurs 1 et 2. Pour déterminer cet exposant dans notre simulation, nous nous appuyons sur le fait que l'énergie totale n'est pas strictement conservée en raison des erreurs numériques. Ainsi, en modifiant le pas de temps, l'amplitude varie en fonction de l'exposant ν. Pour définir la valeur de l'exposant ν, nous utilisons la relation suivante :

$$\Delta E_{\delta t}(t^*) \sim \delta t^{\nu}$$

où

$$\Delta E_{\delta t} = E_{\delta t}(t^*) - E(t_0)$$

Dans cette expression, $E(t_0)$ représente l'énergie initiale du système (c'est-à-dire à l'instant t_0). $E_{\delta t}(t^*)$ désigne l'énergie du système à un instant de référence t^*, que nous choisissons de manière appropriée pour observer les variations d'amplitude en fonction du pas de temps δt. Pour mettre en œuvre cette idée, nous avons suivi les étapes suivantes :

- Identifier une configuration initiale permettant d'observer la non-conservation de l'énergie, tout en choisissant une référence adéquate.
- Effectuer des simulations pour différents pas de temps δt, calculer l'énergie totale et en déduire $\Delta E_{\delta t}(t^*)$.
- Tracer le graphe de $\ln(|\Delta E_{\delta t}|)$ en fonction de $\ln(\delta t)$.

Nous avons donc choisi une configuration initiale composée de 10 particules, que nous avons initialisées aléatoirement et sauvegardées. Nous avons ainsi travaillé avec cette configuration pour le reste de cette partie.

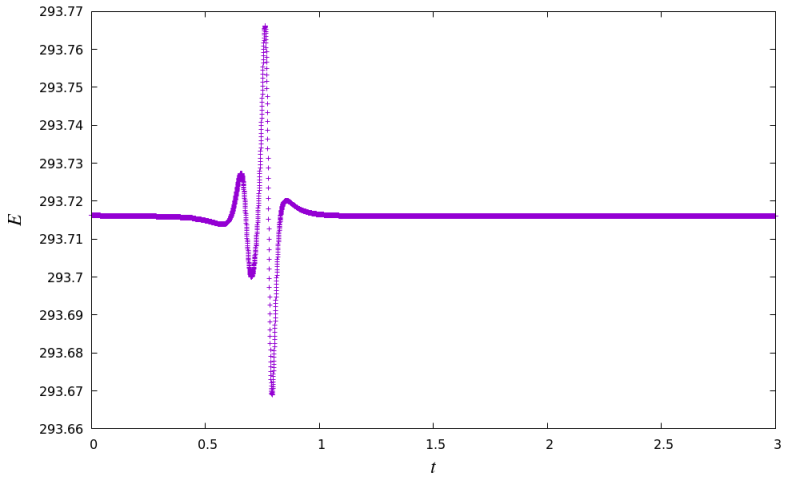

FIGURE 10 – Évolution de l'énergie totale de la configuration de 10 particules

Nous avons choisi comme référence le pic du maximum d'énergie.

Méthode d'Euler

La méthode d'Euler est une méthode numérique d'ordre 1, ce qui signifie que l'erreur d'approximation qu'elle introduit est proportionnelle au pas de temps, δt. En d'autres termes, l'erreur globale accumulée sur une période de temps donnée est linéaire par rapport à δt. Donc avec la configuration de 10 particules choisie. Le temps de référence est $t^* = 0.762500$ et $E(t_0) = 293.71623$.

Pour plus de précision, nous avons enregistré, pour chaque pas de temps, les valeurs de $\ln(|\Delta E_{\delta t}|)$ et de $\ln(\delta t)$. Ensuite, en appliquant la méthode des moindres carrés, nous déterminons la pente de la relation. Cependant, nous ne prenons pas en compte tous les points, mais sélectionnons plutôt un intervalle spécifique. En effet, inclure tous les points fausserait inévitablement les résultats, car pas de temps très grand ou très petits ne sont pas compatibles et produisent des valeurs erronées.

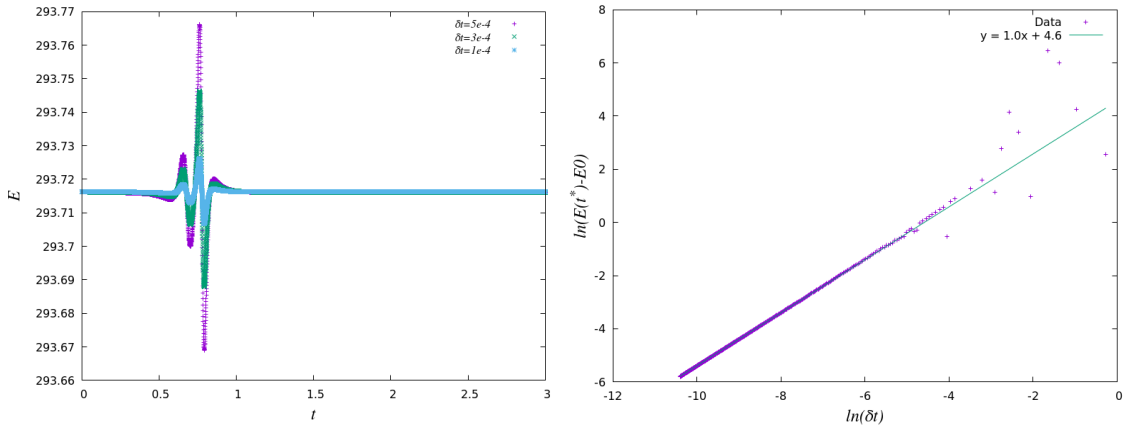

FIGURE 11 – À gauche l'évolution de l'énergie en fonction du pas de temps et à droite l'évaluation de l'exposant ν pour la méthode d'Euler

Nous remarquons donc que, comme attendu, $\nu = 1$ pour la méthode d'Euler, ce qui signifie que l'algorithme était bien implémenté et fonctionne correctement.

Méthode de Velocity-Verlet

En revanche, la méthode de Velocity-Verlet, qui est une méthode d'intégration symplectique, atteint un ordre d'approximation de 2. Cela lui permet de conserver l'énergie du système avec une plus grande précision sur de longues durées. En effet, les erreurs liées à l'approximation sont proportionnelles à δt^2.
Nous suivons la même démarche qu'avant, avec la même configuration initiale, mais cette fois-ci , le temps de référence est $t^* = 0.777500$.

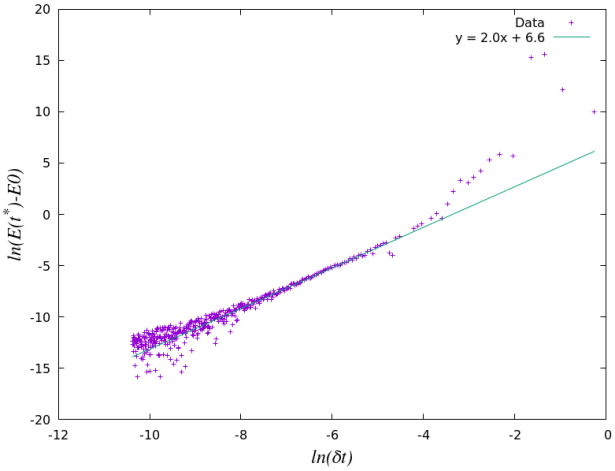

FIGURE 12 – Évaluation de l'exposant ν pour la méthode de Velocity-Verlet

Nous remarquons donc que, comme attendu, $\nu = 2$ pour la méthode de Velocity-Verlet, ce qui signifie que l'algorithme était aussi bien implémenté et fonctionne correctement.

6 Méthodes de calcul utilisées dans le code

6.1 Energie potentielle

Pour calculer l'énergie potentielle du système, on utilise le potentiel associé à chaque particule via le potentiel de Lennard-Jones, qui est l'élément central de notre programme. On effectue donc le calcul du potentiel de manière optimisée pour chaque couple de particules, et on somme ces contributions pour obtenir l'énergie potentielle totale du système. Le tout sans oublier le terme correctif U_{tail} vu précédemment :

$$E_p = \sum_{i,j}^{N} U_{ij} + U_{tail} \tag{9}$$

6.2 Température

Pour calculer la température du système, on utilise le théorème d'équipartition :

$$\frac{\langle E_{cin} \rangle}{N} = \frac{d}{2} k_b T$$

Dans notre cas, la constante de Boltzmann est normalisée pour faciliter la simulation, et nous sommes dans un cas en 2D, donc avec 2 degrés de liberté d.

La température du système est donc simplement égale à l'énergie cinétique moyenne par particule :

$$T = \frac{\langle E_{\text{cin}} \rangle}{N} \tag{10}$$

6.3 Pression

La pression du système est calculée grâce au théorème du viriel. On utilise une variable directement tirée du calcul des forces afin d'optimiser les étapes de calcul. Nous appelons cette variable "viriel", et elle représente la somme pondérée des forces entre particules. Grâce à cette variable et au théorème du viriel, nous définissons dans notre code la pression de la manière suivante, avec P_{tail} comme correction sur la pression.

$$P = \frac{NT}{V} + \frac{viriel}{d * VN} + P_{tail} \tag{11}$$

7 Simulation de la fusion d'un cristal cubique simple

Afin de vérifier le bon fonctionnement et l'utilité du code il est utile de réaliser un essai sur une situation physique réelle. Il a été choisi pour ce code de dynamique moléculaire de prendre une configuration initiale de cristal cubique simple que l'on fait évoluer dans des conditions permettant de le faire fondre, simulant ainsi une fusion.

Au cours de cette simulation, nous observerons l'évolution de la température, de la pression et de l'énergie potentielle du système, et nous analyserons leur comportement pour juger de la fiabilité du code. Le processus de fusion, étant bien connu, sert de référence pour évaluer la validité de cette simulation et pour démontrer l'utilité des simulations de dynamique moléculaire dans l'étude des transitions de phase.

On devrait observer la structure ordonnée du cristal évoluer vers un arrangement désordonné, caractéristique d'un liquide.

7.1 Initialisation du cristal

La première étape pour réaliser cette simulation fut de faire en sorte d'avoir une configuration initiale de notre système régulière qui soit celle d'un cristal.

En 2D cela signifie un arrangement ordonné de particules dans un plan. Pour un cristal cubique simple, ce qui est notre cas ici, les particules vont ainsi être disposées sur une grille carrée où chaque particules aura 4 voisins. Cela forme un motif qui se répète dans le plan comme on peut le voir sur la FIGURE 13. Dans cette FIGURE le cristal se trouve dans la boîte de la simulation de taille L.

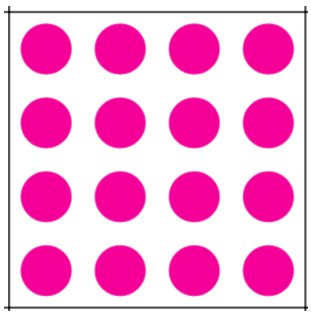

FIGURE 13 – Configuration pour un cristal cubique simple à 2D dans une boîte carré

Le nombre de particules N est ici bien évidemment choisi arbitrairement, il faut simplement un nombre carré pour pouvoir former une structure cubique.

Concrètement dans le code on utilise la fonction qui suit pour au début de la simulation avoir notre configuration initiale cristalline :

Listing 1 – Fonction en C pour avoir une configuration initiale cristalline (cubique simple)

```
1
2 //------------{Initialise la configuration}--------------------
3 /**
4  * @brief Initialise une configuration cristalline pour un ensemble de
      particules.
5  *
6  * Cette fonction dispose les particules dans une structure cristalline,
7  * typiquement en utilisant une grille reguliere dans l'espace.
```

```
 * Cela signifie que chaque particule est placee a une position fixe et ordonnee.
 *
 * @param tab_par Pointeur vers un tableau de structures 'Particule',
 *                representant les particules du systeme a initialiser.
 *                Le tableau doit etre prealablement alloue avec le nombre
 *                de particules souhaite.
 *
 * @note Cette fonction suppose que le tableau 'tab_par' est suffisamment
 *       grand pour contenir toutes les particules necessaires.
 *
 */
void initialiser_Configuration_Cristalline(Particule *tab_par) {
    // Calculer le nombre de particules par ligne et par colonne
    int particules_par_ligne = (int)sqrt(nbx_particules);
    int particules_par_colonne = (nbx_particules + particules_par_ligne - 1) / particules_par_ligne;

    // Espacement entre les particules
    double espacement_x = 2.3; // Ajustez pour obtenir l'espacement souhaite
    double espacement_y = 2.3;

    // Calcul de la largeur et hauteur du reseau
    double largeur_reseau = espacement_x * (particules_par_ligne - 1);
    double hauteur_reseau = espacement_y * (particules_par_colonne - 1);

    // Decalage pour centrer le reseau autour de (0, 0)
    double decalage_x = -largeur_reseau / 2.0;
    double decalage_y = -hauteur_reseau / 2.0;

    for (int i = 0; i <= particules_par_ligne; i++)
    {
        for (int j = 0; j < particules_par_colonne; j++)
        {
            int index = i * particules_par_colonne + j;
            if (index < nbx_particules)
            {
                // Position initiale centree autour de (0, 0)
                tab_par[index].x = i * espacement_x + decalage_x;
                tab_par[index].y = j * espacement_y + decalage_y;
                tab_par[index].vx = 0;
                tab_par[index].vy = 0;
                tab_par[index].actif = 1;
            }
        }
    }
}
```

L'attribut actif des particules sera explicité par la suite.

7.2 Cristal infini

Un paramètre important qu'il faut prendre en compte pour notre cristal ici est la présence de conditions aux bords périodiques. Étant donné que notre boîte principale est entourée de copies d'elle-même je dois m'assurer que la continuité de la structure cristalline d'origine soit maintenue.

En d'autres termes nous devons nous assurer que le réseau semble infini et homogène dans toues les directions. Si dans la boîte principale chaque particule est espacée de son voisin le plus proche d'une distance de 2a, alors chaque particule à un bord doit avoir son voisin périodique également à exactement 2a.

Un exemple du type de configuration qu'on doit avoir est représenté en FIGURE 14.

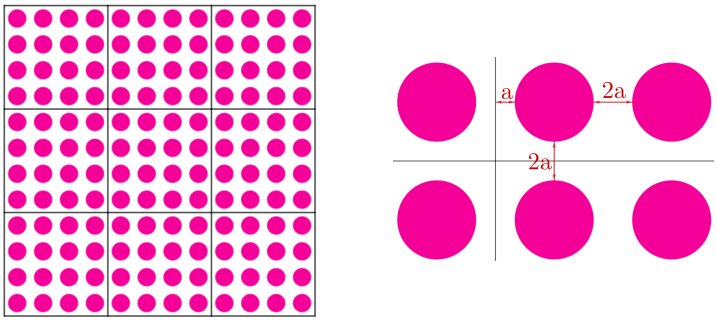

FIGURE 14 – Figure illustrant la mise en place d'un cristal infini.

Cette notion de cristal infini peut sembler anecdotique mais elle en réalité cruciale à cause de plusieurs aspects.

Tout d'abord si les distances entre les particules aux bords et leurs images périodiques ne sont pas conformes à la structure cristalline d'origine, les interactions entre ces particules ne seront pas correctes. Cela va causer l'apparition de forces *artificielles* qui vont fausser le mouvement des particules et ainsi de l'évolution du système au cours de notre fusion. Cela fausserait donc complètement notre simulation et on ne conserverait pas les propriétés physiques du système.

Faire en sorte que chaque particule "voit" un environnement identique, même si elle est au bord est donc fondamentale pour la validité de notre simulation.

7.3 Fusion par injection d'une énergie initiale

Initialement bien qu'on a positionné les particules de notre simulation dans la configuration d'un cristal on ne leur a pas donné de vitesses. Ainsi lorsque l'on fait évoluer la simulation pour un pas raisonnable la configuration du cristal reste inchangé.

Une première manière d'inciter la fusion qui a été envisagée est une chauffe constante du système. En effet expérimentalement c'est ainsi qu'on procède, on chauffe l'échantillon jusqu'à ce qu'il fonde.

Dans le cadre de la simulation il est cependant difficile de procéder ainsi. En effet si l'on injecte de l'énergie au système constamment lors d'une chauffe on change d'ensemble statistique et la validité des formules que l'on utilise pour la température par exemple est remise en question. Un moyen de rester dans le même ensemble statistique serait de chauffer très lentement et d'attendre à chaque fois d'atteindre l'équilibre thermique. Cela serait cependant lent et laborieux, ce n'est donc pas la méthode que nous avons retenu.

Introduire un thermostat a également été envisagé même si cela a été écarté au profit d'une méthode plus simple.

En effet la méthode retenue a été simplement d'introduire une vitesse initiale aux particules du système, introduisant ainsi de l'énergie cinétique dans notre système.
Dans la simulation de dynamique moléculaire la température est directement reliée à l'énergie cinétique moyenne des particules selon la formule (10) . Le fait d'introduire une vitesse initiale aux particules permet de fixer une température initiale au système non nulle.

Or si la température devient suffisamment élevée alors les vibrations thermiques des particules deviennent également élevée et rompent la stabilité de la structure cristalline, pouvant provoquer la fusion du cristal. C'est donc cela qu'on a tenté de faire.

7.3.1 Modification du code

Modifier le code afin d'introduire une vitesse initiale aux particules fut assez aisé. Pour cela il a simplement fallu modifier la fonction d'initialisation **initialiser_Configuration_Cristalline** et changer les lignes fixant les vitesses des particules à 0 avec celles ci-dessous :

Listing 2 – Lignes permettant d'introduire des vitesses initiales

```
1 tab_par[index].vx = ((double)rand() / RAND_MAX - 0.5) * facteur;
2 tab_par[index].vy = ((double)rand() / RAND_MAX - 0.5) * facteur;
```

On aura ainsi des vitesses générées aléatoirement dans l'intervalle [-0.5,0.5] puis multipliées par un facteur choisit arbitrairement.

7.3.2 Configurations au cours de la simulation

En procédant comme expliquer au-dessus et en prenant les paramètres suivant :
→ 100 particules.
→ Avec une distance entre elles de 1.1 .
→ Avec une boîte de taille L=11.
→ Et un facteur pour générer les vitesses aléatoirement de 10.

On obtient des configurations au cours de la simulation telles que :

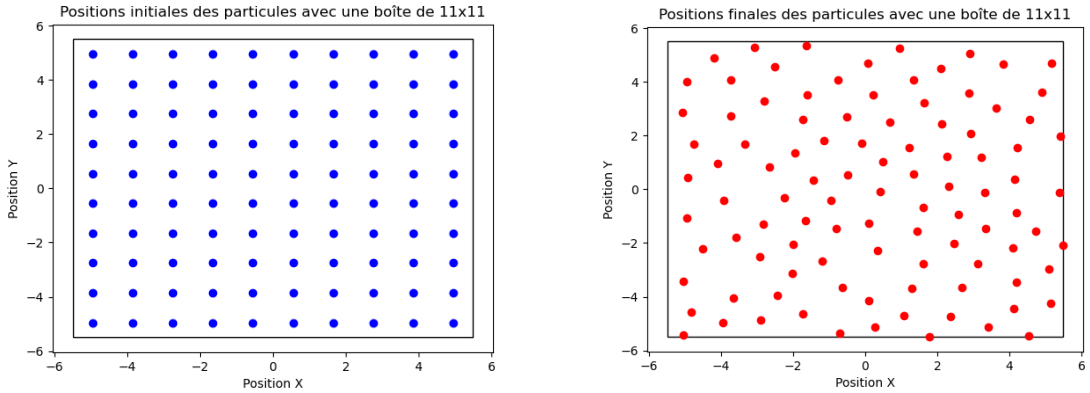

FIGURE 15 – Figures illustrant les configurations de la boîte au début et à la fin de la simulation.

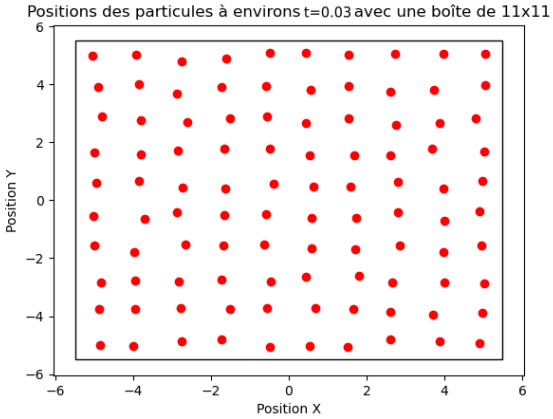

FIGURE 16 – Figure illustrant la configuration de la boîte à 1% de la simulation.

On a laissé la simulation évolué sur un temps suffisamment long pour que les courbes de températures, de pression et d'énergie potentielle se stabilisent, de 3τ.

Et on a pris un pas de temps de 5.0×10^{-4} qui correspond à la partie linéaire de la courbe de vérification que on peut voir en FIGURE 12.

On peut voir sur la FIGURE 15 que l'on passe d'un état initial ordonné à un état final complètement désordonné sans structures se répétant, un liquide. On a donc une transition de phase et plus particulièrement ici une fusion comme ce que l'on souhaitait.

7.3.3 Température, Pression et Énergie potentielle au cours de la fusion

Étudions à présent la température, la pression et l'énergie potentielle au cours de la fusion. Pour tracer ces dernières courbes on s'est servis des méthodes de calcul évoquées en section 6. On com-

mencera par étudier l'énergie potentielle.

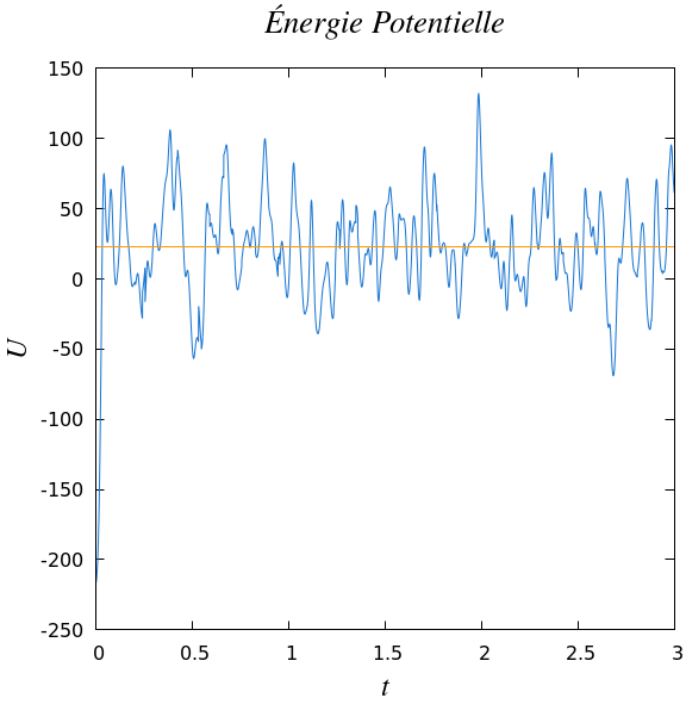

FIGURE 17 – Évolution de l'énergie potentielle au cours de la simulation

Dans la FIGURE 17 le temps est exprimé en secondes.

Si on étudie la figure on peut voir que l'énergie potentielle augmente brutalement au début de la simulation avant de se stabiliser et d'osciller autour d'une moyenne.

On a trouvé pertinent de faire afficher cette moyenne sous la forme d'une droite horizontale orange sur le graphe. Pour calculer cette valeur on a fait une moyenne pour toutes les valeurs de l'énergie potentielle après un temps de t=0.5, une fois que la simulation s'était bien stabilisée. En procédant de cette manière, dans les conditions de notre simulation, on a trouvé une valeur de moyenne de l'énergie potentielle à l'équilibre de $Ep_{eq} = 22.65\epsilon$.

D'un point de vue physique ce que l'on observe sur la FIGURE 17 semble bien correspondre à une fusion. L'énergie potentielle augmente brusquement en raison de la rupture des interactions ordonnées du cristal et de l'ajout d'énergie cinétique initiale. Puis une fois la fusion établie le système est à l'état liquide, désordonné, et l'énergie potentielle oscille autour d'une valeur relativement constante, ce qui est typique d'un équilibre dynamique atteint dans un liquide.

Si nous prenons un système plus grand, avec plus de particules, les oscillations de l'énergie potentielle autour de l'énergie d'équilibre auraient une amplitude moins élevées. En effet les fluctuations

individuelles auraient tendance à se compenser tout comme les fluctuations locales.

Nous avons ensuite étudié l'évolution de la température au cours de la simulation en utilisant à nouveau les formules et méthodes de la section 6.

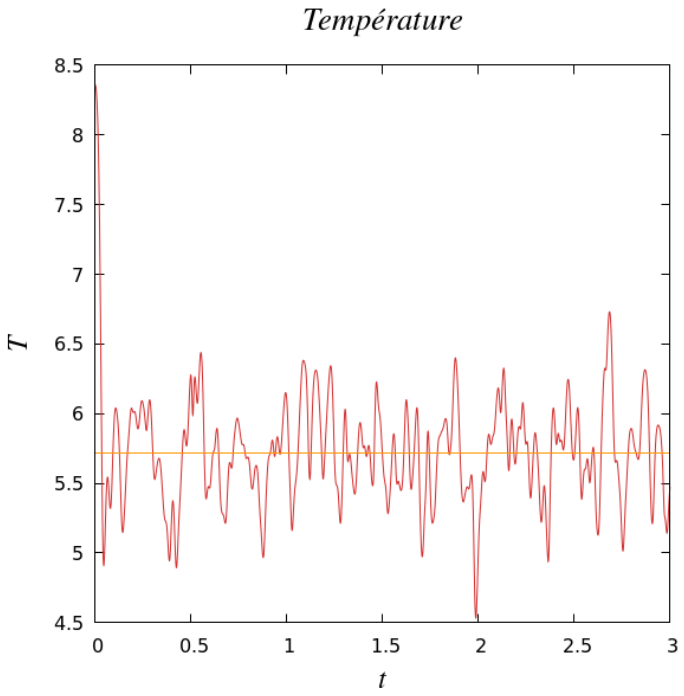

FIGURE 18 – Évolution de la température au cours de la simulation

Dans la FIGURE 18 la température est exprimée en ϵ/k_b. Le temps quant à lui est toujours en unité de τ

En analysant la figure comme pour l'énergie potentielle on peut en tirer des conclusions similaires. On commence avec une température initiale non nulle car les particules avaient une vitesse initiale. Cette température est de $8.36\epsilon/k_b$, elle chute ensuite brutalement avant d'osciller autour d'une valeur d'équilibre très rapidement, atteignant un équilibre dynamique.
Les mêmes conclusions physiques peuvent se tirer que pour l'énergie potentielle, on assiste bien à une fusion. Le système étant chauffé initialement (vitesses initiales non nulles) cela lui permet de surmonter les forces d'attractions qui maintenaient le système dans une configuration cristalline.

Physiquement, la température chute brutalement car lors de la fusion une partie de l'énergie cinétique des particules est convertie en énergie potentielle car ces dernières se réarrangent pour briser l'ordre cristallin et devenir désordonnées. De l'énergie cinétique étant consommée en toute logique la température baisse.

Puis une fois la fusion terminée la température se stabilise autour d'une valeur d'équilibres. Les

oscillations autour de cette valeur peuvent être interprétée comme étant due aux mouvements thermiques des particules même si elles se stabilisent autour d'une moyenne.
La valeur moyenne de la température a été calculée comme précédemment une fois la fusion finie après 0.5τ et on a obtenue $T_{eq} = 5.72\epsilon/k_b$, cela correspond à la température en équilibre.

Comme précédemment si le système était plus grand les oscillations autour de T_{eq} seraient de plus faibles amplitudes. A nouveau on aurait une compensation des fluctuations locales. Mais aussi, on ne l'a pas mentionné avant, mais plus un système est grand plus il se comportera suivant les prédictions thermodynamiques.

Finissons par étudier la pression en FIGURE 19. Le temps est toujours en unité de τ et la pression en unité de ϵ/σ^2 (car en 2D, V est en σ^2).

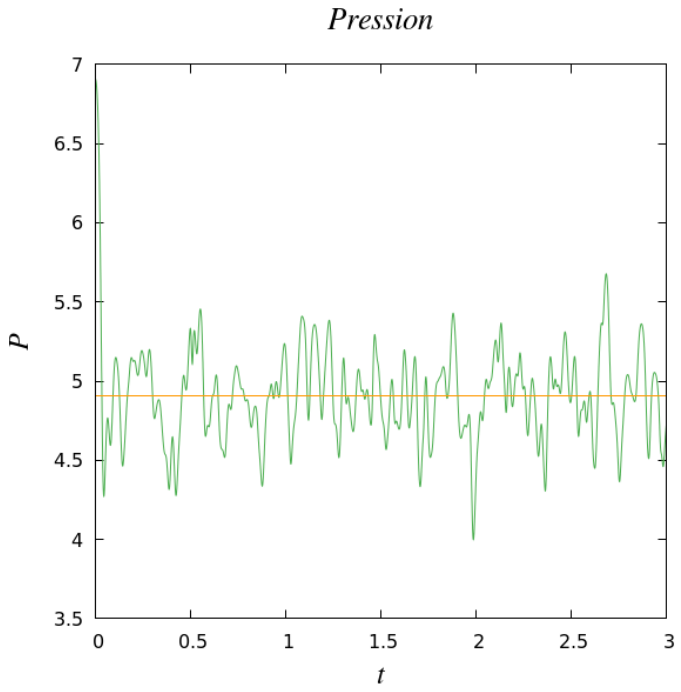

FIGURE 19 – Évolution de la pression au cours de la simulation

On observe à nouveau exactement le même comportement. La pression commence à une valeur initiale non nulle de $6.91\epsilon/\sigma^2$. Puis elle chute brutalement avant de se stabiliser et d'osciller autour de $4.91\epsilon/\sigma^2$.

Physiquement on en tire les mêmes conclusions.
Initialement, avant la fusion, le système est dans un état ordonné, les particules sont proches les unes des autres et interagissent fortement. Les particules "poussent" ainsi les unes contre les autres à cause de la structure compacte et cause une pression plus élevée.

Puis lors de la fusion le système se désorganise et les particules sont plus libres de leur mouvement, les interactions directes entre les particules sont moins intenses et la pression baisse rapidement. Après la fusion on atteint un nouvel équilibre, liquide cette fois, et la pression se stabilise donc autour d'une valeur plus basse. Les oscillations autour de cette valeur d'équilibre sont à nouveau dues aux mouvements thermiques des particules et seraient de plus faibles amplitudes dans un système plus grand, pour les mêmes raisons déjà évoquées.

On trouve avec la même technique que pour l'énergie potentielle et la température $P_{eq} = 4.91\epsilon/\sigma^2$.

Pour conclure la combinaison de ces comportements : baisse de température, de la pression et augmentation de l'énergie potentielle puis stabilisation autour d'un nouvel équilibre peuvent nous mener à confirmer la présence d'une transition de phase solide-liquide dans notre simulation. Notre système a absorbé suffisamment d'énergie pour surmonter les forces de liaison du cristal et passer à un état désordonné de liquide.

7.4 Insertion de défauts dans le cristal

Une autre piste pour la fusion qui nous a parut intéressant d'explorer est la présence de défauts dans notre configuration cristalline initiale, et plus précisément de lacunes.

Une lacune est un type défaut qui correspond à l'absence d'un atome à sa place normale dans la structure cristalline, on laisse derrière un trou. Dans la réalité, hors simulations numériques, il arrive assez fréquemment que des lacunes apparaissent naturellement dans des matériaux solides. Cela peut être dû à des fluctuations de température ou des imperfections lors de la formation du cristal. Il semble donc intéressant de voir l'effet de ces lacunes dans une fusion.

Pour procéder à la fusion avec ces défauts on a à nouveau chauffé initialement le système en donnant une vitesse initiale aléatoire aux particules. Pour pouvoir comparé les résultats avec le cas sans lacunes on a cependant fait en sorte que les résultats aléatoires soient reproductibles.
En effet on a utilisé la même graine pour générer les vitesses aléatoires que ce soit pour les cas avec lacunes et le cas sans lacunes à l'aide de la ligne de code dans le main :

Listing 3 – Ligne permettant la reproductibilité des résultats aléatoires

```
srand(42)
```

Cette ligne initialise le générateur de nombres aléatoires avec une valeur de départ (ou graine) qui est de 42. Ainsi nous nous assurons que chaque exécution du programme reproduit la même séquence de nombres aléatoires.

7.4.1 Activation des défauts

Concrètement on va à présent expliquer comment on a mis en place la présence de lacunes dans notre code.

Le plus simple pour enlever et mettre des lacunes à volonté fut d'ajouter un attribut à notre classe particule. Cette attribut se nomme **actif** et prend les valeurs binaires de 0 ou de 1.
Ainsi une particule qui aura comme valeur pour l'attribut actif de 0 sera considéré comme une

lacune, et pour 1 ce ne sera pas une lacune. Les schémas en FIGURE 20 permettent de mieux comprendre la manière de procéder sur un exemple d'un petit système.

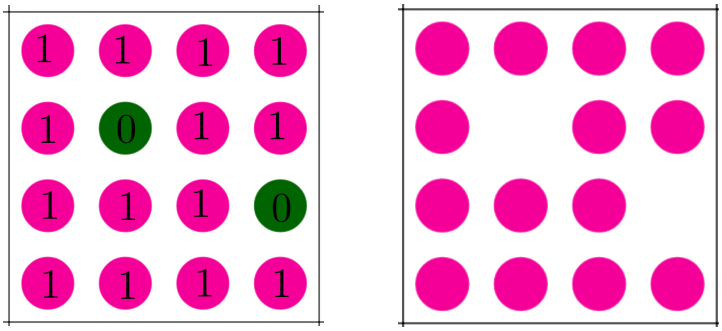

FIGURE 20 – Figure montrant le processus d'activation des particules.

Bien sûr le cristal suivant des conditions périodiques ces lacunes se "propagent" dans le cristal grâce aux images périodiques du cristal. Cela est démontré dans la FIGURE 24.

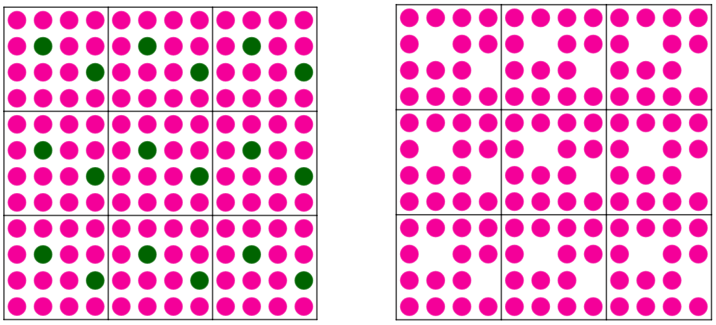

FIGURE 21 – Figure montrant le processus d'activation des particules à l'échelle du cristal infini.

Il nous suffit donc pour introduire des lacunes de rajouter quelques petites lignes par-ci par-là dans notre code.
Ainsi dans la fonction d'initialisation de notre cristal, en même temps qu'on initialise les positions et vitesses de nos particules on initialise leur valeur de actif à 1. Et lorsque l'on veut créer des lacunes à des indexes spécifiques on va par exemple rajouter ce type de lignes dans la fonction d'initialisation du cristal :

Listing 4 – Lignes permettant l'introduction de défauts

```
if(index==12 || index==85)
    {
        tab_par[index].actif=0;
    }
```

Ensuite dans toutes les autres fonctions faisant une boucle sur les particules on rajoute ce type de ligne :

Listing 5 – Ligne permettant d'ignorer les particules lacunes

```
1 if (!tab_par[j].actif) continue;
```

Cela nous permet d'ignorer les particules étant à la position d'une lacune, ou autrement dit les particules non actives et de ne pas en tenir compte lors des calculs.
La dernière touche finale est d'introduire un nombre de particules actives et de l'utilisé dans les calculs au lieu du nombre de particules simples pour tenir compte du nombre vrai de particules dans le système. Ainsi si nous avons initialement une structure de 10 particules par 10 particules et donc au total de 100 particules mais qu'on introduit 3 lacunes mon nombre de particules actives est de 97 et c'est le vrai nombre de particules qu'il y a dans le système.

7.4.2 Configurations au cours de la simulation

Pour pouvoir comparer avec le cas sans lacunes on a pris exactement la même configuration initiale c'est à dire :

→ 100 particules.
→ Avec une distance entre elles de 1.1 .
→ Avec une boîte de taille L=11.
→ Et un facteur pour générer les vitesses aléatoirement de 10.
→ Avec une graine pour la génération des nombres aléatoires pour les vitesses initiales de 42.

Sauf que cette fois nous introduisons deux lacunes aux positions d'index 12 et 85 choisies arbitrairement.
On fait à nouveau évoluer la simulation durant 3 secondes avec le même pas de temps $\delta t = 5.0e-4$ que précédemment.

On commence par observer l'évolution de la configuration initiale au cours du temps. Dans la FIGURE 22 on peut voir la configuration finale et celle initiale de mon système.

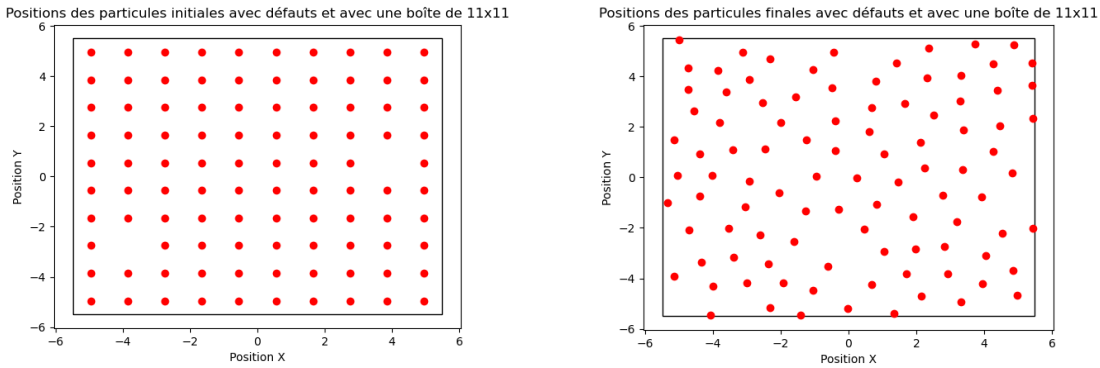

FIGURE 22 – Configurations finales et initiales des particules pour une simulation avec défauts

Comme pour le cas sans lacunes on peut voir que l'on passe d'un système cristallin ordonné où on peut cependant voir les lacunes à un système complètement désordonné qu'on peut déjà supposer être un liquide.

7.4.3 Température, Pression et Énergie potentielle au cours de la fusion

Comme pour le cas sans lacunes on a tracé l'évolution de l'énergie potentielle, de la température et de la pression au cours de la simulation.

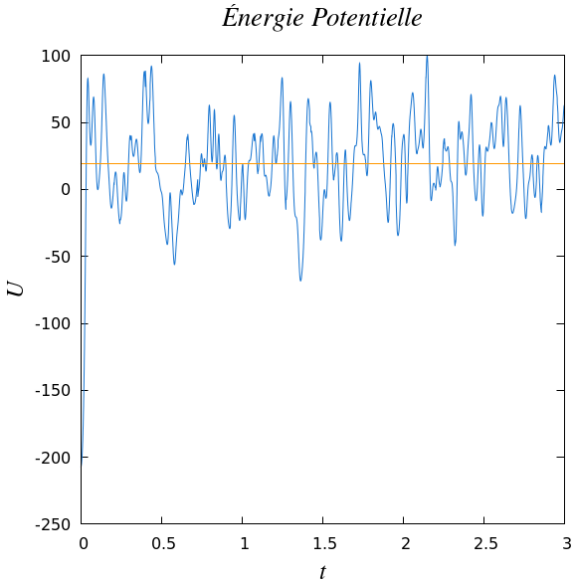

FIGURE 23 – Évolution de l'énergie potentielle au cours de la simulation avec des lacunes

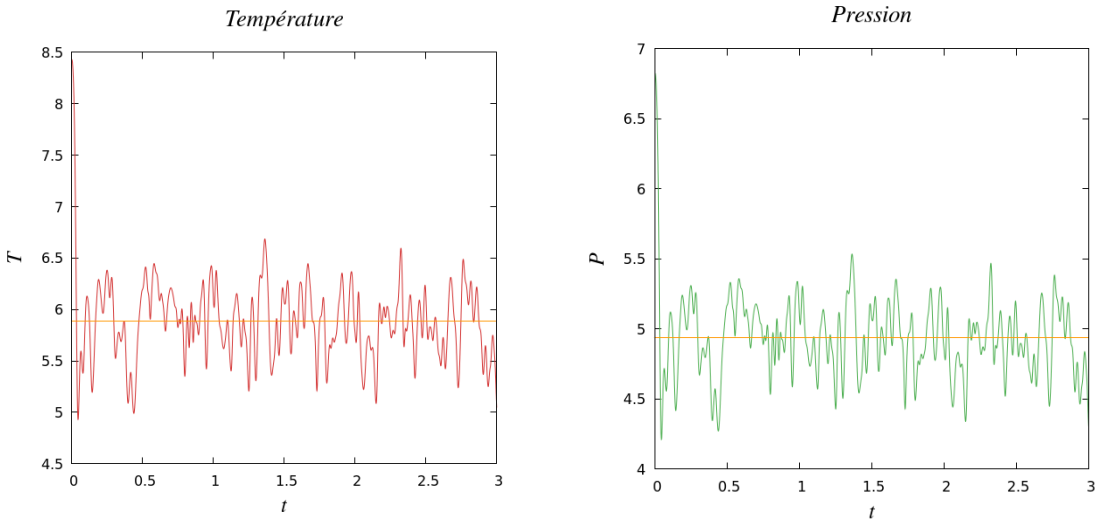

FIGURE 24 – Évolution de la température cinétique et de la pression au cours de la simulation avec des lacunes

On a exactement les mêmes comportements pour ces trois courbes que dans le cas sans lacunes. Les mêmes explications physiques sont donc de mises et on peut en conclure qu'une fusion a bel et bien eut lieu.

Ce qui nous intéressera ici plutôt ce sont les valeurs initiales et à l'équilibre de nos grandeurs qui sont rassemblées dans le tableau ci-dessous :

Valeurs	**Énergie potentielle (ϵ)**	**Température (ϵ/k_b)**	**Pression (ϵ/σ^2)**
Initiale avec lacunes	-207.53	8.43	6.83
Initiale sans lacunes	-217.36	8.36	6.91
A l'équilibre avec lacunes	18.95	5.89	4.94
A l'équilibre sans lacunes	22.65	5.72	4.91

Tableau 1 - Valeurs initiales et à l'équilibre de l'énergie potentielle, de la température cinétique et de la pression pour une simulation avec lacunes.

Étant donnée que la graine pour les nombres aléatoires est la même on peut donc en conclure que la présence ou l'absence de lacunes a une influence sur les valeurs de la fusion mais pas sur la présence ou non de la fusion directement.

Commençons pas étudier l'énergie potentielle. Initialement sans lacunes l'énergie potentielle est plus petite que avec lacunes. Cela s'explique assez simplement. En l'absence de lacunes les atomes sont bien alignés ce qui maximise leurs interactions attractives et minimise l'énergie potentielle. Quand il y a des sites vides (des lacunes) la stabilité globale de la structure est diminuée et l'énergie potentielle est donc plus élevée.

Initialement la température initiale augmente légèrement avec des lacunes, cela s'explique par le fait que les lacunes entraînent un désordre local dans la structure du cristal, laissant plus de liberté de mouvement aux atomes et donc une température qui peut être légèrement plus élevée même si cet effet reste faible.
A l'équilibre on a moins de particules dans la boîte si on avait initialement des lacunes. Donc les particules ont plus de liberté de mouvement ce qui explique que la température à l'équilibre puisse être plus élevée dans le cas avec lacunes.

Initialement la pression peut être plus élevée dans le cas sans lacunes. En effet quand on a des lacunes le réseau cristallin devient moins compact, et la densité du cristal est directement lié à sa

pression. Donc avec des lacunes on a un cristal moins dense ce qui explique que la pression diminue. Mais à nouveau cet effet est ici assez faible.
A l'équilibre on a toujours un système plus dense pour le cas sans lacunes initialement et donc une pression qui devrait en toute logique être plus élevée dans le cas sans lacunes. Ce n'est pas ce qu'on observe ici mais la différence étant très faible cela n'indique rien sur la validité de notre simulation. On peut cependant en conclure que l'impact des lacunes sur la pression à l'équilibre est très faible.

Maintenant que l'on comprend l'effet des lacunes sur les grandeurs initiales et à l'équilibre on peut se poser la question de l'utilité d'en mettre dans le cadre d'une fusion.

On peut comprendre assez rapidement que s'il y a des lacunes le système est initialement moins stable comme on l'a vu. Or un système moins stable est plus facile à déstabilisé et à désordonné. Donc il faut injecter moins d'énergie pour rompre les interactions dans le cas avec lacunes comparé au cas sans lacunes. Cela veut dire que pour observer une fusion, la température de fusion est moins élevée si mon cristal a des lacunes comparé au cas sans lacunes. Cela peut s'avérer utile.
Si on voulait donc plus étudier cet effet des lacunes il faudrait chauffer progressivement notre système pour évaluer la température de fusion au lieu de faire comme dans notre cas et d'injecter une énergie aléatoire initialement. Mais cela sort du cadre de ce TP et serait relativement long comme expliqué plus tôt dans ce rapport.

7.5 Remarques sur la taille du système choisi

Une critique qui pourrait nous être faites concernant la simulation est la taille du système choisi.

En effet si on choisissait un système avec plus de particules et plus grand les oscillations autour des valeurs d'équilibres auraient des amplitudes moins importantes ce qui serait sûrement plus agréable.

Seulement il y a deux raisons principales pour lesquelles on a pas choisit un système plus grand.
Tout d'abord un système plus grand veut dire un temps d'exécution plus lent. Même si le code que l'on a au finale est assez rapide car optimisé. On a donc jugé qu'un système de 100 particules était raisonnable en terme de temps.
Ensuite si on a un système plus grand mais toujours avec le même nombre de lacunes les effets des lacunes seront diluées et auront moins d'impact à cause de l'effet statistiques. Ainsi si on ne veut pas mettre trop de lacunes et en observer les effets il faut se contenter d'un système de taille relativement modeste.

Il serait cependant très intéressant avec plus de temps d'effectuer des simulations de fusion avec de plus grands systèmes.
Même si un système de 100 particules permet déjà amplement d'observer une fusion et son impact sur l'énergie potentielle, la température cinétique et la pression.

8 Conclusion

Pour brièvement faire le point sur ce rapport on a pu au cours de ce TP mettre en œuvre un code de dynamique moléculaire relativement élaboré.

En effet on a utilisé un potentiel d'interaction de Lennard-Jones tronqué et décalé, une méthode d'intégration de "Velocity Verlet" et intégré des conditions périodiques aux bords entre autres. Le tronquage du potentiel a également été pris en compte dans le calcul des différentes valeurs comme l'énergie et la pression.

Il a aussi été démontrer avec plusieurs cas, la conservation de l'énergie totale du système, l'exposant ν et des systèmes simples la fiabilité de notre code de dynamique moléculaire.

Une fois le code établi et sa fiabilité confirmée on a pu présenter un exemple qualitatif de son utilité avec une simulation de fusion pour le cas d'une configuration initiale cristalline. Cela nous a permis de confirmer encore un peu plus la fiabilité du code tout en démontrant l'utilité d'un tel code. Il permet dans certains cas de se passer d'expérimentations pratiques par exemple.

On a finit par étendre ce rapport en étudiant l'influence des lacunes dans notre cristal sur la fusion et sur les différentes grandeurs importantes au cours de la simulation. Cela sort un peut du cadre initiale de ce rapport mais nous semblait pertinent car cela peut être utile pour effectuer des fusions à température moindre et que c'est un cas souvent rencontré dans la nature.

Ce TP nous a donc donné de solides bases en dynamique moléculaire et nous a beaucoup enseigné.