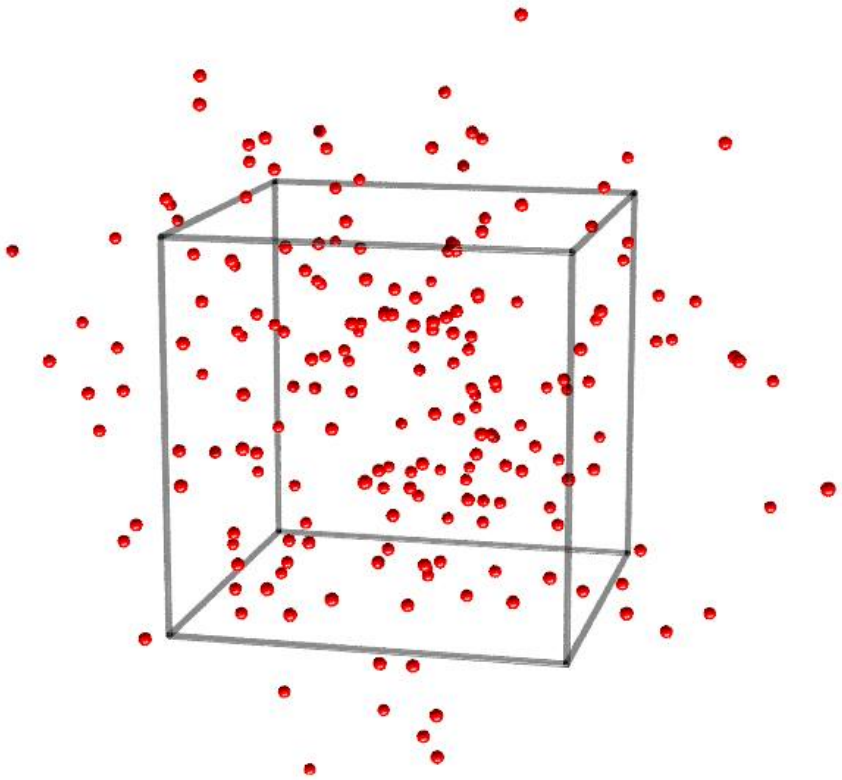
Rapport projet P6 : Collision moléculaire - Potentiel de Lennard Jones

Etudiants : Evenat Théo - Hervault Antoine - Leroux Simon - Wang Xiaomin - Xie Feilian

Enseignant responsable du projet : M. Jérôme Yon



Année 2017/2018

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES ROUEN

Date de remise du rapport : **18/05/2018**

Référence du projet : **STPI/P6/2018-38**

Intitulé du projet : **Collision moléculaire - Potentiel de Lennard-Jones**

Type de projet : **modélisation, simulation 3D**

Objectifs du projet :

Le but de ce projet est d'étudier puis de modéliser les interactions moléculaires au sein d'un gaz parfait ou réel, à partir du potentiel de Lennard-Jones.

Mots-clef du projet : **Dynamique, interactions, Lennard-Jones**

Table des matières

Introduction

Dans le cadre de notre projet physique, nous avons décidé de mettre en place diverses expériences faisant appel aux collisions moléculaires, en ayant pour base principale le potentiel de Lennard-Jones ainsi que notre cours de théorie cinétique des gaz. Il nous a donc fallu choisir des applications physiques intéressantes à étudier et des modélisations informatiques qui pouvaient en découler.

Les interactions, qu'elles soient moléculaires ou gravitationnelles, sont relativement simples à étudier lorsqu'il n'y a que deux corps mis en jeu. Cependant, dès lors que le nombre d'éléments devient supérieur, il est quasiment impossible de trouver une solution analytique à la main. Or, notre projet repose sur les interactions entre centaines voir milliers de particules : c'est donc pour cela qu'il nous a fallu développer différents programmes informatiques, tout d'abord en Scilab puis en Python, capables de déterminer ces solutions et de modéliser les interactions souhaitées, en les paramétrant avec les lois voulues.

Nous avons donc sélectionné trois cas pour étudier et modéliser les collisions moléculaires :
Dans un premier temps, nous avons voulu vérifier la validité du modèle du gaz parfait. Après avoir établi les conditions différenciant le gaz parfait du gaz réel, nous avons étudié le comportement de deux gaz dans des conditions de densité particulaire différentes, puis nous avons réalisé des vidéos illustrant le phénomène grâce à un logiciel de synthèse d'images : Pov-Ray.

Nous avons par la suite décidé d'étudier le phénomène de diffusion moléculaire vu en cours, avec deux angles d'approche, tout d'abord en nous intéressant uniquement à l'établissement général de l'équilibre des densités au sein d'un domaine, puis par la suite en essayant de retrouver numériquement le coefficient de diffusion moléculaire D d'un gaz, un coefficient approché en cours de P8-1 par la loi de Fick. Le tout a été réalisé en utilisant les deux modèles établis précédemment : le gaz parfait et le gaz réel. Afin d'illustrer les résultats obtenus lors des différentes expériences numériques et de les présenter de manière plus ludique, nous avons utilisé le logiciel de synthèse d'images précédemment évoqué. Cela nous a permis de réaliser des vidéos présentant le comportement des molécules au sein de nos différentes expériences.

Dans ce rapport, nous débuterons par un bref historique concernant le potentiel de Lennard-Jones. Par la suite, nous présenterons l'organisation du travail au sein du groupe ainsi que la méthodologie adoptée afin de mener à bien notre projet. Ensuite, nous développerons le travail réalisé, en expliquant les différents programmes que nous avons codé, en exploitant les résultats obtenus grâce à ces programmes, et en comparant ces derniers aux lois théoriques étudiées en cours de mécanique des fluides, comme la loi des gaz parfaits ou encore les lois théoriques d'équilibre des densités. Pour finir, nous tirerons des conclusions sur l'ensemble de notre travail en ayant un regard critique sur la justesse des résultats informatiques.

Chapitre 1

Histoire - John Lennard-Jones

1.1 Biographie de John Lennard-Jones

John Edward Lennard-Jones (27 octobre 1894 - 1er novembre 1954) était un mathématicien britannique, mais également un professeur de physique théorique à l'Université de Bristol, avant de devenir professeur de science théorique à l'Université de Cambridge. Il peut être considéré comme l'initiateur de la chimie computationnelle moderne.

En 1924, John Lennard-Jones propose une loi de force interatomique semi-empirique, c'est la première forme du potentiel de Lennard-Jones. C'est en 1931 qu'il propose la forme finale de son potentiel.

Il est aujourd'hui reconnu parmi les scientifiques pour son travail sur la structure moléculaire, la valence et les forces intermoléculaires. Beaucoup de recherches ont été effectuées sur ces sujets, et ce pendant plusieurs décennies, notamment suite à la publication de son article datant de 1929 "The electronic structure of some diatomic molecules" dans lequel il présente l'approximation de la combinaison linéaire des orbitales atomiques pour les orbitales moléculaires. [1]

1.2 Le potentiel de Lennard-Jones

1.2.1 Expression

Le potentiel de Lennard-Jones (également appelé potentiel L-J, potentiel 6-12 ou potentiel 12-6) est un modèle mathématiquement simple qui se rapproche de l'interaction existante entre une paire d'atomes ou de molécules neutres. Une forme de ce potentiel interatomique a été proposée pour la première fois en 1924 par John Lennard-Jones.

1. https://en.wikipedia.org/wiki/John_Lennard-Jones

L'expression la plus courante du potentiel de Lennard-Jones est la suivante [2] :

$$V_{L-J}(r) = 4\varepsilon\left(\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right) = \varepsilon\left(\left(\frac{r_m}{r}\right)^{12} - 2\left(\frac{r_m}{r}\right)^{6}\right)$$

où ε représente la profondeur du potentiel d'interaction, σ le diamètre atomique (c'est également la distance finie à laquelle le potentiel inter-particulaire est nul : forces attractive et répulsive égales), r représente la distance entre les particules, et r_m la distance à laquelle le potentiel atteint son minimum. À r_m, le potentiel a la valeur $-\varepsilon$.
Les distances précédemment présentées sont liées comme suit :

$$r_m = 2^{\frac{1}{6}}\sigma \simeq 1.122\sigma$$

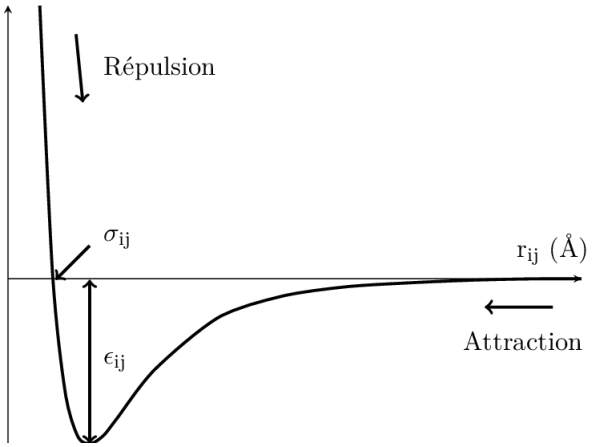


FIGURE 1.1 – Potentiel de Lennard-Jones

1.2.2 Explication physique de l'expression du potentiel de Lennard-Jones

Le terme r^{-6}, qui est le terme attractif à longue portée, décrit l'attraction à longue distance entre les particules appelée forces de Van der Waals. Un modèle simple a été développé par le physicien théoricien germano-américain Fritz Wolfgang London afin de formuler la théorie complète de ces forces. Pour établir sa théorie, London a étudié les forces existantes entre deux oscillateurs linéaires constitués d'une charge négative et vibrant autour d'une charge positive très lourde au repos. Celui-ci a remarqué que pour des distances entre les centres des oscillateurs, notées r, très grandes par rapport aux dimensions des oscillateurs, l'énergie du système était entre autre composée d'un terme négatif et inversement proportionnel à r^6. [3]
Alors que le terme attractif en r^{-6} a une justification physique claire, le terme répulsif en r^{-12} est empirique et n'a aucune justification théorique. [4]

2. http ://www.cax.free.fr/lennard/lennard.html
3. `https://www.universalis.fr/encyclopedie/fritz-london/`
4. `https://en.wikipedia.org/wiki/Lennard-Jones_potential`

Les forces intermoléculaires étant conservatives, on a la relation suivante :

$$\overrightarrow{F(r)} = -\overrightarrow{grad(E_p(r))}$$
$$\text{avec } E_p(r) = V_{L-J}(r) = 4\varepsilon\left(\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right).$$

La dérivation du potentiel L-J par rapport à r donne donc une expression de la force intermoléculaire entre 2 molécules a et b :

$$\overrightarrow{F_{b/a}} = \frac{24\varepsilon}{r}\left(2(\frac{\sigma}{r})^{12} - (\frac{\sigma}{r})^{6}\right)\overrightarrow{u}$$

avec $\overrightarrow{u}$ un vecteur unitaire orienté de b vers a. Cette force intermoléculaire peut être attractive ou répulsive, en fonction de la valeur de r. Lorsque r est très petit, les molécules se repoussent. Lorsque r devient grand, elles s'attirent.

Le potentiel L-J est une approximation relativement bonne. En raison de sa simplicité, il est souvent utilisé pour décrire les propriétés des gaz. Il est particulièrement précis pour les atomes de gaz rares et constitue une bonne approximation à des distances longues et courtes pour les atomes et les molécules neutres.

1.2.3 Des expressions alternatives

Forme AB

La forme AB est une formulation simplifiée utilisée par les logiciels de simulation [5] :

$$V_{L-J} = \frac{A}{r^{12}} - \frac{B}{r^{6}}$$

où $A = 4\varepsilon\sigma^{12}$ et $B = 4\varepsilon\sigma^{6}$. Inversement, $\sigma = \sqrt[6]{\frac{A}{B}}$ et $\varepsilon = \frac{B^2}{4A}$.
C'est dans cette forme que John Lennard-Jones a écrit le potentiel 12-6.

Forme tronquée et décalée

Le potentiel de Lennard-Jones est également utilisé numériquement sous une expression différente. Pour économiser le temps de calcul, le potentiel de Lennard-Jones est souvent tronqué à une distance de coupure $r_c = 2,5\sigma$, où :

$$V_{L-J}(r_c) = V_{L-J}(2,5\sigma) = 4\varepsilon\left((\frac{\sigma}{2,5\sigma})^{12} - (\frac{\sigma}{2,5\sigma})^{6}\right) \simeq -0,0163\varepsilon$$

5. https://en.wikipedia.org/wiki/Lennard-Jones_potential

c'est-à-dire qu'à $r_c = 2,5\sigma$, le potentiel de Lennard-Jones V_{L-J} est environ égal à $\frac{1}{60}$ de sa valeur minimale (la profondeur du puits de potentiel). Au-delà de r_c, les interactions ne sont plus prises en compte, le potentiel tronqué est mis à zéro.
Pour éviter une discontinuité à r_c, le potentiel L-J doit être légèrement décalé vers le haut, de sorte à ce que le potentiel tronqué soit nul exactement à la distance de coupure r_c.
Le potentiel de Lennard-Jones tronqué $V_{L-J_{trunc}}$ est donc défini comme suit :

$$V_{L-J_{trunc}}(r) = \begin{cases} V_{L-J}(r) - V_{L-J}(r_c) & \text{si } r \leq r_c \\ 0 & \text{si } r > r_c \end{cases}$$

On peut facilement vérifier que $V_{L-J_{trunc}}(r_c) = 0$, éliminant ainsi la discontinuité à $r = r_c$.

1.2.4 Les limites du potentiel de Lennard-Jones

Le potentiel L-J est une bonne approximation pour les atomes de gaz rares ainsi que pour les molécules neutres. Cependant, il ne peut être adapté à l'étude de l'interaction entre deux molécules polaires : celles-ci sont en effet plus complexes à décrire car elles dépendent de l'orientation des moments des molécules.

Avec le potentiel L-J, le nombre d'atomes liés à un atome n'affecte pas la force de liaison. L'énergie de liaison par atome augmente ainsi linéairement avec le nombre de liaisons par atome. Cependant, les expériences montrent plutôt que l'énergie de liaison par atome augmente de façon quadratique avec le nombre de liaisons.

Le terme de puissance 6 modélise les interactions dipôle-dipôle dues à la dispersion d'électrons dans les gaz rares (forces de dispersion de London), mais il ne représente pas bien d'autres types de liaisons. Le terme de puissance 12 apparaissant dans le potentiel est choisi pour sa facilité de calcul pour les simulations numériques (en mettant au carré le terme de puissance 6) et n'a pas de réelles significations physiques.

Le potentiel diverge lorsque deux atomes se rapprochent l'un de l'autre. Cela peut créer des instabilités qui nécessitent un traitement spécial dans les simulations de dynamique moléculaire. [6]

6. `https://en.wikipedia.org/wiki/Lennard-Jones_potential`

Chapitre 2

Méthodologie - Organisation du travail

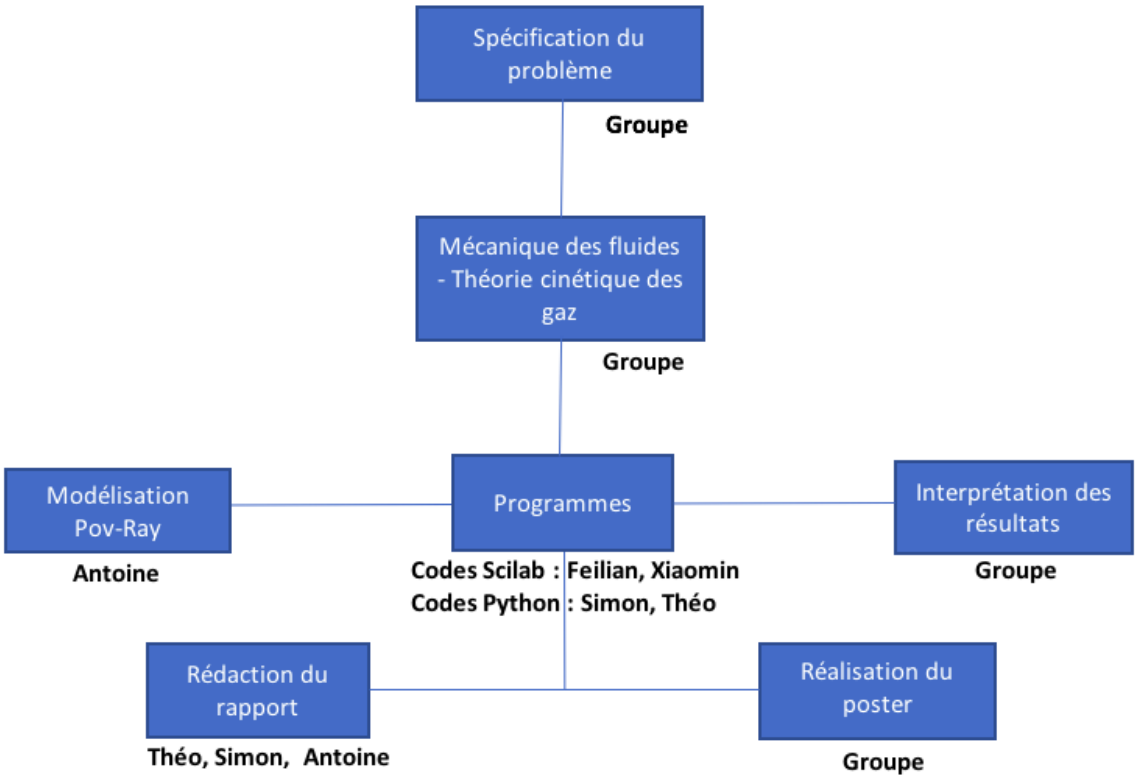


FIGURE 2.1 – Répartition du travail au sein du groupe

Au commencement de ce projet, nous avons pris comme base le code informatique réalisé par le groupe ayant travaillé sur un sujet similaire avec M.Yon l'année dernière. Ceci nous a permis de ne pas partir de zéro et de pouvoir obtenir plus rapidement des résultats tout en allant plus loin dans le travail qui avait été réalisé. Ce programme informatique était codé uniquement en Scilab, mais nous avons décidé de réaliser la suite du projet en Python afin d'avoir une puissance de calcul plus conséquente, mais aussi pour avoir d'avantage de moyens graphiques nous permettant ainsi de réaliser et étudier des courbes de manière plus complète. Cependant, aucun des membres du groupe ne savait coder en Python, il nous a donc fallu passer les premières séances à apprendre le langage de manière accélérée afin de pouvoir retranscrire le programme Scilab en un premier programme Python fonctionnel.

Le fond de notre projet reposant sur la modélisation de collisions moléculaires grâce au potentiel de Lennard-Jones dans différentes configurations (qu'elles soient 2D ou 3D), il était nécessaire de reprendre les formules vues en cours de théorie cinétique des gaz. Nous avons ainsi commencé à nous répartir le travail à réaliser.

— Théo et Simon se sont occupés du codage des différents programmes en Python. Ceci consistait, dans un premier temps, en la réécriture de l'ancien programme Scilab (2D), puis en la création de nouveaux programmes 2D ou 3D qui nous ont permis de réaliser nos expériences : l'obtention des conditions de validité du modèle du gaz parfait, l'équilibre des densités ou encore la recherche du coefficient de diffusion moléculaire.
— En parallèle, Feilian et Xiaomin travaillaient sur Scilab pour améliorer les techniques utilisées dans les programmes Python mais aussi pour réfléchir à de nouvelles solutions aux problèmes que posaient les différentes expériences.
— Une fois que les programmes informatiques fonctionnaient et nous donnaient les résultats escomptés, il a fallu modéliser les expériences afin d'obtenir des vidéos permettant d'expliquer simplement et d'illustrer les phénomènes complexes à l'œuvre. C'est donc Antoine qui s'est occupé de la réalisation de l'ensemble des vidéos sur Pov-Ray.

Durant tout l'avancement du projet, la communication entre les différents membres du groupe était primordiale. En effet, il fallait que chacun se tienne au courant de l'avancée dans le codage des programmes, codés en Python ou Scilab, ou encore dans la réalisation des vidéos puisque ces différentes parties étaient toutes liées. Il était essentiel de vérifier la cohérence des différents programmes avec le sens physique qui en découlait, afin de se rapprocher un maximum de la réalité lors de la modélisation.

Chapitre 3

Travail réalisé et résultats

3.1 Explication des programmes Python et PovRay

Après avoir réussi à coder en Python le programme réalisé l'an dernier, programme qui modélisait les interactions entre un ensemble de particules au sein d'un domaine fermé du plan, nous l'avons réécrit pour l'adapter à un domaine en 3 dimensions mais aussi pour que celui-ci prenne en compte le potentiel d'interaction de Lennard-Jones et non plus la loi définie l'an dernier. Dans cette simulation, plusieurs paramètres pouvaient être modifiés, à savoir le nombre de particules, la température, la taille du domaine ou encore le type de gaz. (programme disponible ici : 3.4.2)

Pour commencer, il est nécessaire de déclarer l'ensemble des variables qui seront par la suite utiles dans le programme, comme par exemple des constantes (la constante de Boltzmann k_b, la constante d'Avodagro N_a, ...), la température, la largeur du domaine ou encore les différentes constantes en relation avec le gaz considéré comme sa masse molaire ou la distance intermoléculaire correspondante. Il est ensuite nécessaire de déclarer les différentes fonctions qui vont nous servir pour modéliser les interactions. Nous commençons donc par déclarer la fonction "proba" qui représente la densité de probabilité des vitesses et en quantifie donc la répartition statistique, selon la loi de Maxwell-Boltzmann : $dN = 4\pi v^2 \left(\frac{m}{2\pi k_b T}\right)^{\frac{3}{2}} \times e^{-\frac{mv^2}{2k_b T}} dV$. Vient ensuite la fonction "CalculEP" qui nous donne, à chaque pas de temps et à partir des positions des différentes particules, l'énergie potentielle d'interaction via le potentiel de Lennard-Jones ($E_p = 4E_0((\frac{d}{r})^{12} - (\frac{d}{r})^6)$) et par conséquent la force exercée entre chacune des particules prises deux à deux puisque celle-ci dérive du potentiel précédent ($\vec{F} = -\vec{grad}(Ep)$).

Ensuite, nous devons générer les vitesses et positions initiales de l'ensemble des particules. Dans cette première version du programme, celles-ci seront générées aléatoirement dans l'ensemble du domaine, mais cela changera dans les prochaines expériences. Pour les vitesses, il a été nécessaire d'introduire un repère sphérique. Nous générons tout d'abord des angles θ et ϕ sur une sphère. Ensuite, par une petite interpolation de la densité de probabilité de Maxwell-Boltzmann, nous générons un tableau de vitesses (réparties entre 0 et v_{max} grâce aux lois précédemment décrites) puis nous multiplions ces dernières avec les angles créés via les formules du repère sphérique. Concernant

les positions des particules, nous créons un tableau de nombres aléatoires entre 0 et 1, puis nous multiplions ceux-ci par la taille du domaine pour une répartition dans tout le cube de coté L.

Ensuite, nous initialisons différents tableaux, tels que les tableaux d'énergies (cinétique, potentielle et mécanique), mais aussi ceux du temps et de la température.

Vient ensuite la boucle principale du programme qui va se charger à chaque pas de temps de recalculer les positions et vitesses de chaque particule pour que nous puissions, après exécution de l'ensemble des boucles, les voir évoluer dans le temps et dans l'espace. On notera qu'à chaque boucle, le temps est augmenté de deltat, une variable choisie arbitrairement.

Pour chaque pas de temps, nous commençons par calculer l'énergie potentielle d'interaction, la force résultante entre chaque couple de particules présentes dans le domaine et donc l'accélération de la particule concernée. Grâce à ces valeurs, nous pouvons recalculer les vitesses et positions de chaque particule :

$$Vitesse_2 = Vitesse_1 + \frac{F}{m*deltat} \; ; \; Position_2 = Position_1 + Vitesse_2 * deltat.$$

De plus, nous avons dû imposer des conditions aux limites pour les particules, qui étaient ici des conditions de rebond sur les parois de notre domaine. Nous verrons par la suite, dans les trois expériences suivantes, que ces conditions seront amenées à changer, et même à disparaître dans le cas du programme de diffusion. En effet, lorsqu'une particule arrive au bord du domaine, nous ne voulons pas qu'elle disparaisse en sortant de notre domaine d'affichage, mais bien qu'elle rebondisse sur les bords. Un rebond qui se doit d'ailleurs d'être élastique pour toutes les particules : la vitesse et la quantité de mouvement de celles-ci restent inchangées après ce contact avec un bord du domaine, afin que nos expériences ne se basent que sur les interactions entre particules et non celles avec d'autres corps.

FIGURE 3.1 – Rebond élastique d'une particule sur une paroi

Pour cela, nous devons prendre en compte deux cas différents : si les rebonds se font sur les faces où x,y ou z=L ou bien en x,y ou z=0. Pour le premier cas, on voit que si la position d'une particule est supérieure à L la largeur de notre domaine à l'instant t, c'est-à-dire si sa position recalculée au début de la boucle est hors des limites du domaine, on change la position de la particule par $2 * L - Posi$. Cette petite formule nous permet de remettre la particule à l'endroit où elle aurait dû être avec un réel rebond. Ceci étant, nous inversons la vitesse (V_x,V_y ou V_z suivant la face sur laquelle a lieu le rebond) de la particule afin qu'elle reparte dans le domaine à la suite de son rebond. Dans l'autre cas, il nous suffit simplement de prendre l'opposé de la position et de la vitesse sur l'axe correspondant pour simuler le rebond. On réalise ainsi ces conditions sur les six faces pour le domaine 3D pour obtenir un domaine fermé et des conditions de rebonds optimales.

Une dernière opération est alors nécessaire : faire une somme de la quantité de mouvement (Ppart) accumulée par les particules lors de chaque rebond sur une paroi, ce qui nous sera utile par la suite. On notera que dans chaque boucle, nous enregistrons dans les tableaux crées précédemment certaines valeurs pertinentes : l'énergie cinétique avec $Ec = somme(\frac{1}{2}mV^2)$, l'énergie potentielle (somme des E_p calculées lors des collisions) ou encore la température expérimentale avec $T_{exp} = \frac{2}{3}Ec_{moy}k_b$.

Pour finir, nous récupérons les valeurs finales de pression expérimentale avec $P_{exp} = \frac{Ppart}{6*t*L^2}$, de température du milieu T_{exp}, mais aussi de pression théorique avec $P_{th} = \frac{Nk_bT_{exp}}{L^3}$.

À l'issue de chaque programme, nous souhaitons réaliser une vidéo de modélisation afin d'obtenir un meilleur rendu visuel de notre travail. Pour cela, au cours de l'exécution de chaque programme, dans la boucle principale qui recalcule les positions et vitesses de chaque particule à chaque pas de temps, nous écrivons au sein d'un fichier texte, les coordonnées de chaque particule. On récupère ensuite ce fichier qui sera utilisé dans le programme Pov-Ray correspondant.

Dans le programme Pov-Ray, on commence par déclarer les variables définissant la taille du domaine, la densité particulaire et le gaz concerné. Pour obtenir une image fidèle, ces variables doivent correspondre à celles définies dans le programme Python correspondant. Il est également nécessaire de définir et de placer une caméra ainsi qu'une source de lumière afin de pouvoir obtenir les images escomptées. Ensuite, le programme fait appel au fichier texte précédemment évoqué et lit les coordonnées qu'il contient. Ces coordonnées sont ensuite attribuées aux variables de position qui définissent les coordonnées de chaque particule, représentée par une sphère, au sein de l'image à l'instant correspondant. Un programme secondaire répète l'exécution de ce programme principal, nous permettant ainsi d'obtenir une succession d'images représentant l'évolution du système au cours du temps. Le programme secondaire s'arrête lorsque le nombre d'images souhaitées est atteint. On obtient alors un ensemble d'images qu'il suffit de compiler avec n'importe quel logiciel de montage vidéo (nous avons ici utilisé iMovie) afin d'obtenir une vidéo de modélisation. (Des exemples de ces deux programmes sont disponibles page 42 : 3.4.2).

3.2 Conditions de validité du modèle du gaz parfait

3.2.1 Programmes, objectifs et méthodologie

On sait que la différence entre un gaz parfait et un gaz réel réside dans la différence de densité particulaire du domaine d'étude ($N = \frac{Nbr}{V}$) : nous avons en effet vu en cours que, dans le modèle du gaz parfait, les particules sont très éloignées les unes des autres, que les collisions entre ces particules sont quasi-inexistantes et qu'il n'y a aucune interaction à distance (modèle des "boules de billard"), la densité de particules y est donc faible, contrairement au gaz réel. Notre objectif va donc être ici de déterminer les conditions de densité particulaire représentatives des modèles du gaz parfait et du gaz réel en essayant de vérifier la loi des gaz parfaits. Le programme utilisé lors de la réalisation de cette expérience est celui décrit dans la partie 3.1 page 13. Comme expliqué précédemment, ce programme nous permet de calculer une pression expérimentale pour n'importe quelles conditions initiales de densité, de température ou d'espèce gazeuse.

Afin de déterminer les densités représentatives de chaque état, nous avons décidé de nous intéresser à deux gaz différents : l'Argon, ayant déjà un comportement type gaz parfait et le CO_2, afin de voir si des différences ou similitudes pouvaient être trouvées. Nous avons aussi décidé de fixer au début de l'expérience la température ainsi que le nombre de particules, noté Nbr, dans le domaine (un excès de particules entraîne en effet des programmes trop longs à exécuter pour nos ordinateurs) mais de changer uniquement la taille du domaine : $L = r0Nbr^{\frac{1}{3}}R$, avec r0 le rayon de répulsion électrostatique dépendant du gaz utilisé et R un coefficient que nous avons défini qui nous permet de rapidement changer la taille du domaine à notre guise.

Ceci étant fait, chaque simulation va nous donner une pression expérimentale et une pression théorique ainsi qu'une température. Avec environ dix couples de valeurs, nous pouvons tracer le graphique $P = Nk_bT$ et on s'attend alors à ce que la courbe expérimentale tracée soit identique à la droite théorique présente sur le même graphique si le gaz a un comportement parfait, ou à ce qu'elle diffère si le gaz a un comportement réel. Les résultats obtenus seront analysés dans la partie 3.2.2 page 17.

Cependant, après avoir réalisé quelques tests avec différentes densités, nous avons réalisé que ce programme n'était pas optimal lorsque la densité particulaire dépassait une certaine limite (limite décrite par la suite). En effet, lorsque les particules sont initialisées aléatoirement et que la densité est très élevée, il est probable qu'elles apparaissent très proches les unes des autres. Or, comme nous l'avons vu au début de ce rapport, à courte distance, les interactions entre particules sont répulsives et peuvent être extrêmement violentes : certaines de nos particules atteignaient des vitesses de plus de $10^{10}m/s$! (nous ne tenons en effet pas compte de toutes les autres lois physique, notamment celle de la relativité restreinte). Nous avons donc du revoir notre initialisation et certains de nos paramètres : comme présenté en annexe (ici : 3.4.2), nous avons commencé par placer de manière ordonnée les particules dans l'ensemble du domaine, pour que les interactions répulsives initiales soient le moins élevé possible : elles sont espacées régulièrement les unes des autres. De plus, le paramètre $r > d$, une condition mise en place pour qu'il calcule une éventuelle interaction entre deux particules distantes de r, présent dans la fonction "CalculEP" ne nous convenait plus : nous l'avons modifié par la condition $r > 0$, puisque les particules pouvaient être beaucoup plus proches les unes des autres.

Ces modifications étant prises en compte, passons aux résultats.

3.2.2 Résultats

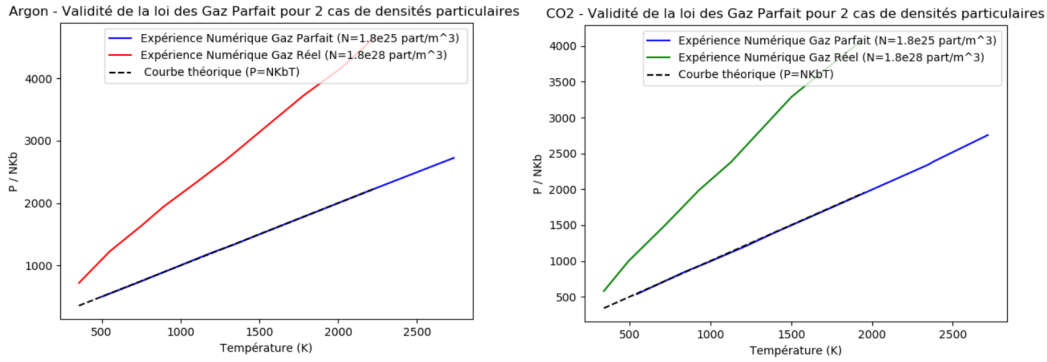


FIGURE 3.2 – Tracé de l'évolution de P/Nk_b en fonction de T pour deux gaz différents : l'Argon et le CO_2 et comparaison avec la courbe théorique (courbe noire). Les courbes bleues correspondent au modèle du gaz parfait, les courbes rouge et verte à celui du gaz réel (/1000 pour la lecture graphique)

Après avoir réalisé différents tests en modifiant à chaque fois la densité particulaire du milieu et en traçant l'évolution de la pression en fonction de la température, nous sommes arrivés à la conclusion que la densité maximale correspondant à l'existence d'un gaz parfait était d'environ $2.10^{25} particules.m^{-3}$,ce qui était équivalent à $R \simeq 7$ dans la formule de L. Comme nous le montrent les 2 graphiques ci-dessus représentant $P = f(T)$ pour deux gaz distincts, si la densité est supérieure à la densité limite déterminée préalablement alors le gaz (L plus petit avec R=1), quel qu'il soit, a un comportement type gaz réel : il ne suit pas la loi des gaz parfaits $P = Nk_bT$ (pente de 1) représentée par la droite noire sur les 2 graphiques (calculée grâce aux formules théoriques vues en P8.1 et appliquées à notre cas d'étude). On notera cependant une différence dans les coefficients directeurs de ces courbes (pente de 2,21 pour le CO_2 et pente de 2,1 pour l'Argon). Ces différences sont dues aux valeurs des constantes saisies initialement qui dépendent du gaz (d, E0, Mmol, ...), qui interviennent dans l'ensemble du programme et donc dans le calcul de la pression expérimentale. Si au contraire la densité est inférieure à la densité limite (L plus grand avec R=10), alors le gaz suit le modèle du gaz parfait, comme le montre la superposition des courbes expérimentale et théorique.

Par ailleurs, les modélisations obtenues confirment la différence entre les deux cas : celui du gaz parfait et celui du gaz réel. En effet, comme nous le montrent les images ci-dessous, dans le modèle gaz parfait, les collisions entre particules sont quasi-inexistantes et quand elles ont lieu, on retrouve des collisions type "boules de billard", ce qui confirme les propriétés du gaz parfait vues en P8.1 : au sein d'un gaz parfait, le libre parcours moyen (l.p.m.), qui représente la distance parcourue par une particule entre deux collisions, est bien supérieure à la distance inter-particulaire, ce qui explique la quasi-inexistence des collisions entre particules. Au contraire, au sein d'un gaz réel, les particules interagissent fréquemment et ce à distance : les trajectoires des particules ne sont plus des droites mais des courbes, ceci résultant de d'interactions à distance.

FIGURE 3.3 – À gauche : un gaz type Parfait et à droite, un gaz type Réel

On peut également confirmer la différence entre ces deux comportements via l'étude des courbes énergétiques : s'il n'y aucune interaction à distance, l'énergie doit se conserver, alors qu'elle fluctue lorsque ces interactions existent. C'est exactement ce que nous retrouvons dans notre expérience, comme le montrent les courbes ci-dessous.

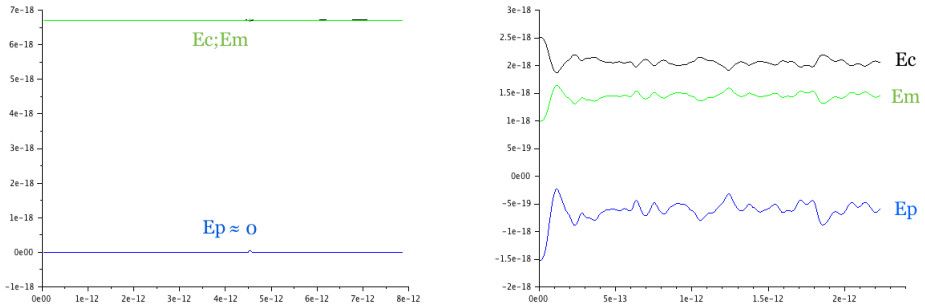


FIGURE 3.4 – Évolution de l'énergie au cours du temps au sein d'un gaz parfait à gauche et d'un gaz réel à droite

Ceci nous permet donc de définir deux zones de densité qui nous serviront pour nos prochaines expériences, dans lesquelles nous n'aurons pas à redéfinir le type de gaz mais juste à faire attention à sa densité.

Maintenant que nous savons à quel type de gaz nous avons affaire, nous allons pouvoir réaliser n'importe quelle expérience pour un gaz parfait et pour un gaz réel et ainsi comparer et analyser les résultats obtenus.

3.3 Première application au phénomène de diffusion : Équilibre des densités

3.3.1 Programmes, objectifs et méthodologie

Nous nous sommes intéressés dans un second temps au processus de diffusion moléculaire, phénomène qui rééquilibre la répartition spatiale de la densité particulaire et amène, à terme, à un équilibre thermodynamique spatial. Nous avons donc voulu étudier l'évolution de quelques densités dans le temps au sein d'un domaine 3D, afin d'avoir une première approche expérimentale sur ce phénomène avant de l'aborder plus spécifiquement par la suite.

Pour cela, certaines conditions utilisées précédemment ont du être modifiées pour s'adapter à notre problème. La principale différence avec le programme précédent réside dans le positionnement initial des particules : à t=0, nous avons placé des particules uniquement dans la moitié gauche du domaine, la partie droite en étant dépourvue. La surface S (située en $\frac{L}{2}$) qui sépare les deux côtés n'est pas physique mais nous sert à différencier précisément les 2 parties d'étude du domaine. Le reste se passe exactement de la même manière que précédemment : les particules évoluent dans le domaine 3D, rebondissent sur les parois et nous mettons un terme à la simulation à l'aide d'une condition simple : quand le nombre de particules s'est équilibré des deux côtés de la boîte (à 99.8% près), on relève la valeur t_{eq} mais nous ne l'arrêtons que quand cet équilibre des densités a été atteint un certain nombre de fois afin d'avoir une courbe expérimentale optimale (nous le verrons dans la partie suivante). On peut alors relever l'évolution des densités dans les parties gauche et droite du domaine au cours du temps.

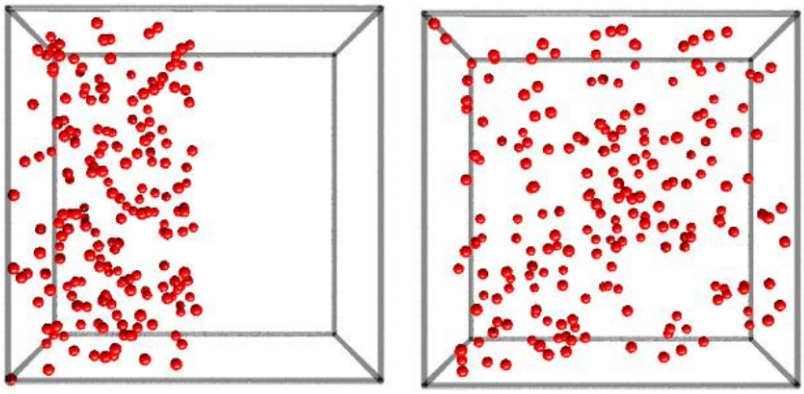


FIGURE 3.5 – Répartition des particules à la date t=0 sur l'image de gauche et équilibre des densités après diffusion sur celle de droite

Nous avions ainsi pour objectif de nous servir de l'expérience précédente et des conditions d'existence du gaz parfait et du gaz réel afin de comparer l'équilibre dans les 2 cas, en superposant les courbes pour le gaz parfait et le gaz réel. Nous avons encore une fois réalisé l'expérience avec deux gaz différents, afin de voir si des similitudes ou des différences pouvaient être trouvées, dans les mêmes conditions de densité et de température.

Là encore, dès que la densité dépassait les $10^{25} particules/m^3$, nous retrouvions les mêmes problèmes que dans l'expérience précédente (3.2.1 page 16) et nous avons alors repris le même type de solution, en positionnant notamment les particules de manière espacées dans la moitié gauche du domaine, afin de toujours limiter les interactions répulsives initiales.

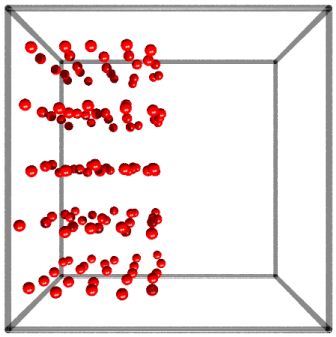


FIGURE 3.6 – Répartition initiale imposée des particules dans le cas d'une forte densité particulaire

3.3.2 Résultats

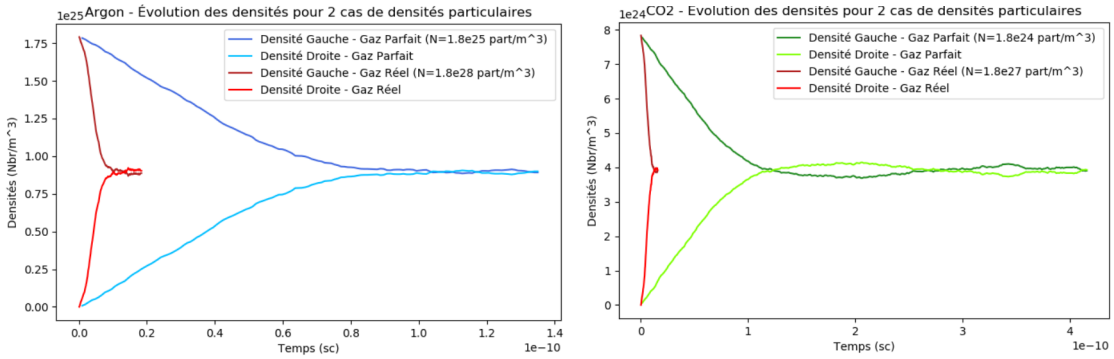


FIGURE 3.7 – Tracé de l'évolution des densités particulaires "gauche" et "droite" du domaine 3D en fonction de t pour deux gaz différents : l'Argon et le CO_2. Les courbes bleues correspondent au modèle du gaz parfait, les courbes rouge et verte à celui du gaz réel

En étudiant les deux courbes ci-dessus, on constate très bien que, pour tout type de gaz et dans cette configuration particulaire initiale, il y a retour à l'état d'équilibre : on remarque en effet que les deux densités "gauche" et "droite" s'égalisent au bout d'un certain temps (noté t_{eq}) et restent quasiment égales par la suite (les petites oscillations que l'on peut apercevoir après l'équilibre proviennent de légers mouvements d'ensemble des particules, ceci résultant de la manière dont elles ont été placées initialement et de la manière dont elles se déplacent dans le domaine) : il y a eu diffusion.

Quelques conclusions peuvent être tirées de cette expérience : elle nous a bien sûr permis de remarquer, comme évoqué précédemment, que pour tout gaz, qu'il soit parfait ou réel et peu importe

la manière dont sont placées les particules initialement (nous avons aussi réalisé d'autres tests en plaçant différemment les particules dans le domaine), sa diffusion dans l'ensemble de la boîte est inévitable et donc que l'équilibre est toujours atteint à partir d'un certain temps, plus ou moins long suivant les paramètres utilisés (température, taille du domaine, positions initiales, ...).

Nous pouvons aussi voir sur les graphiques que l'équilibre est atteint beaucoup plus rapidement lorsque la densité est élevée (supérieure à notre limite de 10^{25} $particules/m^3$), c'est-à-dire quand nous nous trouvons dans un modèle gaz réel, que quand on se place dans un gaz parfait avec une densité moindre. Les résultats obtenus ici (pour des pressions de l'ordre de grandeur de 10^5 Pa) sont sans appel : on atteint le stade d'équilibre, dans nos conditions, environ 9 fois plus rapidement au sein du gaz réel (8 fois pour l'Argon et près de 10 fois pour le CO_2). Cela peut s'expliquer par le fait que la densité y étant plus élevée et puisque le gaz a un comportement réel, il y a énormément d'interactions entre particules, qui évoluent ainsi plus rapidement dans l'ensemble du domaine. Alors que dans le gaz parfait, les interactions sont beaucoup plus rares, les particules se répartissent donc moins rapidement dans le domaine, d'où le fait que l'équilibre des densités soit atteint beaucoup moins rapidement. Bien que nos résultats semblent cohérents avec la réalité, ils restaient dépendants des positions initiales de nos particules et de leur caractère aléatoire, de leurs vitesses respectives et du mouvement des particules les unes par rapport aux autres.

Par ailleurs, il ne nous a pas été possible de comparer nos t_{eq} obtenus avec celui que nous avions déterminé lors d'une expérience similaire réalisée dans un cas particulier avec un gaz parfait, lors d'un TD de P8-1. Il aurait en effet fallu que les parties "gauche" et "droite" de notre domaine ne soient séparées que par une surface élémentaire dS, et pas par l'ensemble de la section comme c'est le cas ici. Nous aurions dans ce cas une formule du type : $t_{eq} = \frac{-ln(0.01)(V1+V2)}{S*v_{moy}}$ (V1 et V2 les volumes "gauche" et "droite", S la surface séparant les 2 parties du domaine et V_{moy} la vitesse moyenne des particules) et les particules auraient alors mis plus de temps à s'équilibrer.

Nous en sommes donc arrivés à la conclusion qu'une autre expérience allait être nécessaire afin de mieux cerner ce phénomène de diffusion mais aussi pour rapprocher nos simulations de la réalité physique.

3.4 Deuxième application au phénomène de diffusion : calcul du coefficient de diffusion

3.4.1 Programmes, objectifs et méthodologie

Nous avons donc décidé pour cette troisième expérience de déterminer une méthode numérique qui nous permettrait, à l'aide de nos simulations, de calculer expérimentalement le coefficient de diffusion D (en m^2s^{-1}) pour un gaz donné et dans des conditions connues. Pour ce faire, nous avons commencé par placer toutes les particules au centre du domaine, dans un petit cube d'une taille dépendante de l'utilisation qu'on veut en avoir (on reviendra par la suite sur ce détail), puis, comme dans les deux autres expériences, nous les avons laissé évoluer dans l'ensemble du domaine. Contrairement aux deux autres expériences, celle-ci s'est déroulée en 2D pour des soucis de simplifications de calculs (programme disponible ici : 3.4.2). Le programme utilisé ici, reprend les mêmes bases que le précédent. Cependant, quelques différences sont notables, comme le fait que les lois de Maxwell-Boltzmann ont du être recalculées pour un domaine 2D, avec par exemple la densité de probabilité qui devient $dN = v\left(\frac{m}{k_bT}\right) \times e^{-\frac{mv^2}{2k_bT}} d_v$ (démonstration ici : B), tout comme celle calculant la pression expérimentale ($P_{exp} = \frac{Ppart}{4Lt}$) par exemple. L'initialisation des vitesses a été beaucoup plus simple car nous n'avons plus besoin de repère sphérique et celle des positions reprend le même concept que celle de l'expérience précédente (3.3.1 page 19) en y enlevant une coordonnée.

Afin de réaliser correctement cette expérience, nous avons aussi dû nous séparer des conditions limites : les parois ont été supprimées afin que la diffusion à partir du centre du domaine se fasse d'une manière optimale (on ne veut pas que les particules ayant rebondi sur une paroi viennent frapper celles s'éloignant du centre) : en partant d'un petit amas de particules central, nous nous attendons donc à retrouver à la fin de l'expérience une boîte vide. C'est cette condition qui impose le plus grand changement par rapport aux simulations précédentes : ici, la diffusion se fait sous vide et c'est ce qu'on va retrouver dans le programme avec des pressions expérimentales de l'ordre de 10^{-7} Pascal.

Intéressons nous maintenant à la méthode de calcul du coefficient D. Nous sommes pour cela partis d'une loi que nous avions pu aborder en cours de théorie cinétique des gaz, et qui fait intervenir ce coefficient D : la loi de Fick. Cette loi, qui s'écrit $\vec{\varphi_p} = -D\vec{\nabla}(N)$, nous indique que la densité de flux $\vec{\varphi_p}$ du nombre de particules est proportionnelle au gradient de la densité particulaire et à D, le coefficient de diffusion.

Afin de déterminer une relation simple, que l'on pourrait vérifier à l'aide de nos programmes, qui ferait intervenir le coefficient de diffusion recherché D, nous sommes partis de l'équation de conservation du nombre de particules dans un domaine, une équation similaire à celles vues en mécanique des fluides (conservation de la masse) mais aussi en électromagnétisme (conservation de la charge électrique) :

$$div(\vec{j}) + \frac{\partial n}{\partial t} = 0$$

avec $\vec{j} = \vec{\varphi_p}$.

En appliquant cette équation à un domaine 1D (étude sur un seul axe) et en considérant que l'évolution de la densité particulaire sur cet axe devait avoir la forme d'une gaussienne, nous en sommes arrivés à l'équation suivante : (démonstration ici : A)

$$\sigma(t)^2 = 2Dt$$

Il nous faut maintenant calculer σ expérimentalement. En ne s'intéressant qu'à l'axe X, nous avons décidé de découper cet axe en un ensemble de bandes d"une certaine largeur, afin de pouvoir y mesurer la densité et pouvoir établir la courbe représentant cette diffusion en 1D. Nous sommes, comme annoncé juste au dessus, partis du principe que l'évolution de la densité dans notre boîte reportée sur l'axe X devait avoir la forme d'une gaussienne, les particules étant concentrées au centre et s'en éloignant progressivement.

Nous avons donc introduit une nouvelle méthode de calcul : la Gaussian-fit. Cette technique, que l'on peut apercevoir un peu plus bas (3.8 page 24) a pour but de créer une courbe de Gauss avec des paramètres que nous pouvons ajuster à chaque boucle : le maximum (en nombre de particules, présent dans une des zones de l'axe X), la moyenne (la position X moyenne, facilement calculable) mais aussi l'écart-type. Cette dernière mesure a été un peu plus délicate. En effet, cette méthode crée une gaussienne (représentative d'une loi normale en probabilité) et cherche, si les valeurs que nous avons saisies lui correspondent précisément, des valeurs utiles tels que son écart-type, mais aussi sa covariance par exemple. Pour que cela soit le cas et pour que nous puissions obtenir des valeurs, nous avons donc créé wid, une variable représentant l'écart-type que nous pensions avoir pour chaque boucle, le plus proche possible de la réalité : $wid = (\frac{tailleCotéX}{2})^2$. En regardant la formule de plus près, on remarque nous nous calculons ici la moitié de la distance sur X entre le X_moyen et le X de la particule la plus à droite du domaine, le tout au carré. Finalement, en connaissant cette valeur à $t = 0$, à t_final et en connaissant le nombre de boucles qu'allait réaliser le programme, nous en avons déduit de combien l'augmenter à chaque boucle pour que la gaussienne créée s'ajuste de manière optimale à nos valeurs. Il ne nous reste plus qu'à sortir cet écart-type expérimental σ à chaque boucle avec le temps correspondant, les insérer dans des tableaux, créer la courbe $\sigma(t)^2 = 2Dt$ pour en sortir le coefficient directeur : 2D.

On notera aussi que la valeur théorique de D est calculée grâce aux formules vues en cours : $D = \frac{1}{3}v_{moy}lpm$, et qu'il ne faut pas oublier de remettre nos valeurs à l'échelle : on divise donc le D obtenu par 10^8 puisque nous avions tout multiplié au début afin d'avoir un affichage plus lisible.

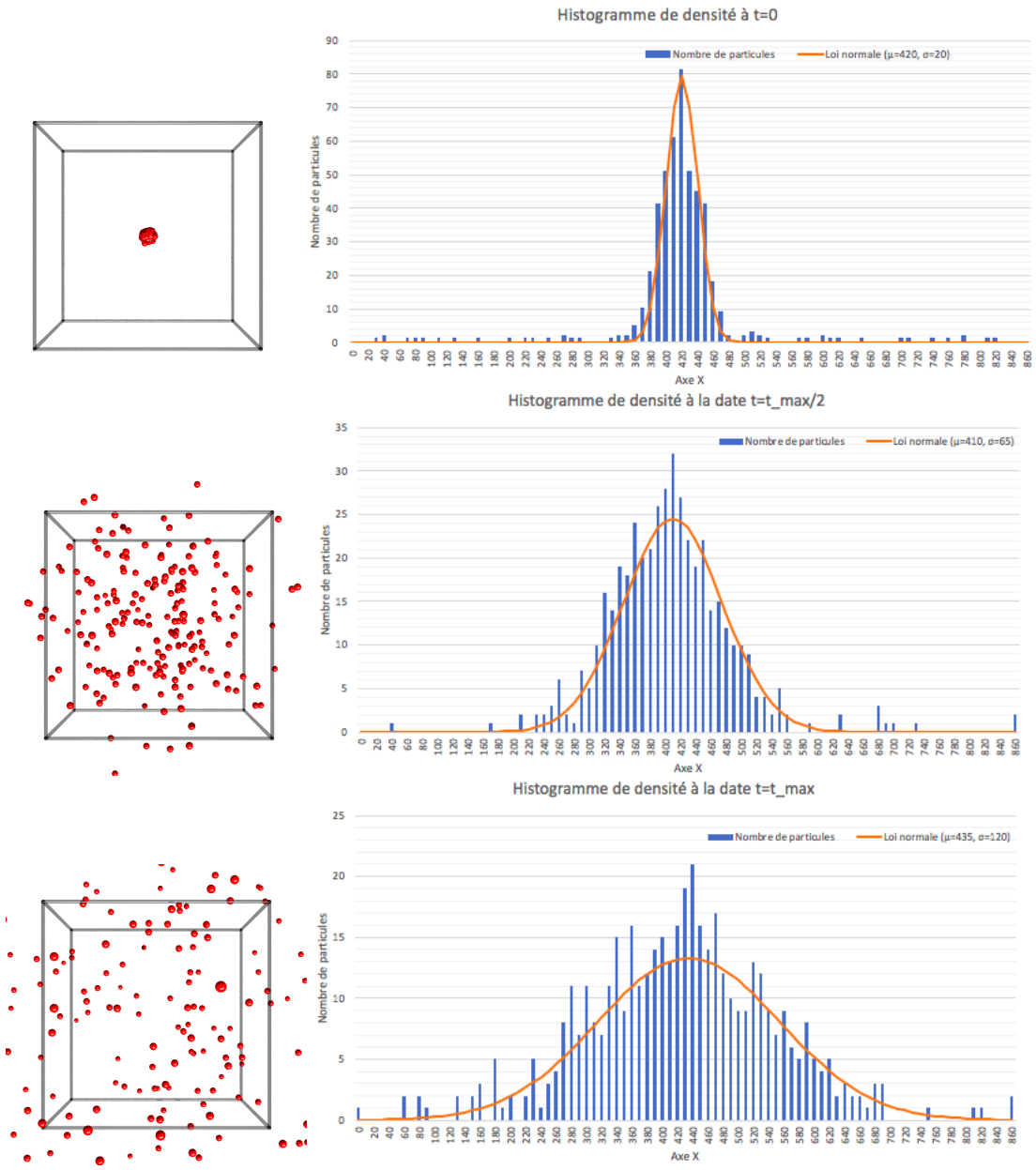


FIGURE 3.8 – Modélisation du phénomène de diffusion moléculaire pour de l'Argon. La seconde colonne représente les histogrammes montrant la répartition des particules à la date $t = 0$ en haut, $t = \frac{t_{max}}{4}$ au milieu et $t = \frac{t_{max}}{2}$ en bas et en trait continu, le résultat de l'ajustement des histogrammes par une distribution normale : le Gaussian-fit

Nous pouvons donc mesurer D grâce à une mesure de l'écart-type de notre courbe représentant l'évolution de la densité au sein du domaine.
Passons maintenant aux résultats que nous avons pu obtenir avec cette technique.

3.4.2 Résultats

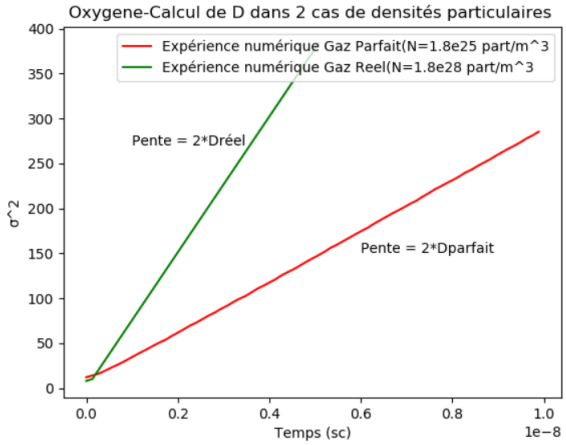


FIGURE 3.9 – Tracé de l'évolution de l'écart-type σ^2 de notre gaussienne en fonction du temps, pour l'oxygène et dans le vide. La courbe rouge correspond au modèle du gaz parfait, la verte à celui du gaz réel (/1000 pour notre graphique)

Comme expliqué dans la partie précédente, cette expérience n'était pas destinée à pouvoir faire une comparaison entre plusieurs gaz, mais uniquement de voir si il était possible de retrouver un coefficient de diffusion D dans des conditions particulières. Les droites présentes sur le graphique, de coefficient directeur 2D, nous permettent alors de comparer nos résultats aux valeurs théoriques. Il faut cependant noter que nos conditions extrêmes de pression et de température ne nous permettent pas de comparer directement nos valeurs à celles vues en cours (trouvées à P_{atmo} et à 298K, avec $D_{th} = 1.5 * 10^{-5} m^2/s^{-1}$), mais nous pouvons quand même regarder l'ordre de grandeur des résultats trouvés et leur cohérence.

Les coefficients trouvés, $D = 756 m^2/s^{-1}$ pour le gaz réel et $D = 280 m^2/s^{-1}$ pour le gaz parfait, ont des ordres de grandeurs qui semblent cohérents avec la réalité physique : nous pouvons donc dire qu'aux conditions initiales près, notre méthode de calcul de D semble pertinente puisqu'en effet, ce coefficient D ne peut qu'être bien plus élevé dans le vide que dans l'air à P_{atmo}. Ici, rien ne les retient, les particules s'éloignant du centre ne vont pas se heurter aux millions d'autres présentes aux alentours dans une atmosphère, et vont par conséquent se diffuser bien plus rapidement. Il serait alors très intéressant de réaliser cette expérience en créant un nouveau programme nous permettant d'avoir les conditions idéales pour une simulation encore plus fidèle de la réalité.

On remarque par ailleurs que le coefficient D trouvé pour le gaz réel (un gaz plus dense initialement) est, comme le montre le graphique, beaucoup plus important que celui trouvé dans le cas parfait. Cela s'explique par le fait que, même si nous sommes dans le vide, la forte densité initiale a permis, comme dans l'expérience précédente, aux particules de se repousser bien plus rapidement et de manière plus marquée que dans le cas parfait, dans lequel les interactions sont faibles et très peu nombreuses.

L'expérience que nous avons pu mener ici est donc concluante, elle nous a donné des résultats satisfaisants. La refaire dans de meilleures conditions pourrait donner des résultats encore plus intéressants.

Conclusion

Pour conclure, nous pouvons dire que ce projet a été très enrichissant pour nous tous.

Tout d'abord, il nous a permis d'améliorer notre capacité à travailler en groupe, avec notamment la nécessité d'organiser des temps de travail pour faire des recherches sur notre projet, avancer le plus possible sur la réalisation de nos différents programmes ainsi que mettre en place des moments pour faire le point sur l'avancement de chacun. Le travail effectué chaque semaine se devait d'avoir un rythme soutenu pour que, lors de chaque séance de projet, nous puissions profiter des conseils de M.Jérôme Yon, qui nous ont été d'une aide précieuse.

Ce travail nous a aussi permis de nous familiariser avec de nouveaux logiciels mais aussi avec des langages de programmation qu'aucun de nous n'avait eu l'occasion d'expérimenter par le passé (notamment Python et PovRay). C'est grâce à ces nouveaux outils que nous avons pu avoir une nouvelle approche, plus numérique et informatique, de notre cours de théorie cinétique des gaz. Il nous a en effet été possible de réaliser des images et des vidéos rendant compte d'une manière ludique des phénomènes abordés, qui rendaient bien compte des différents concepts théoriques vus en cours, qui peuvent paraître assez difficile de prime abord. Cependant, nous avons rapidement été limités par la puissance de calcul de nos ordinateurs : le nombre de particules ne pouvait être trop élevé sans quoi plusieurs heures étaient nécessaires à l'exécution de nos programmes.

Ainsi, ce projet P6 nous a permis d'enrichir notre culture scientifique mais aussi nous préparer à notre futur rôle d'ingénieur, nous permettant notamment d'apprendre à gérer et mener à bien un projet nécessitant de nouvelles connaissances, l'apprentissage de nouvelles méthodes de résolution et la découverte de nouveaux environnements de travail.

Références

Histoire - John Lennard-Jones

https://en.wikipedia.org/wiki/Lennard-Jones_potential

https://en.wikipedia.org/wiki/John_Lennard-Jones

http://www.cax.free.fr/lennard/lennard.html

http://www.f-legrand.fr/scidoc/docimg/sciphys/dynmol/lennardjones2d/lennardjones2d.html

https://www.researchgate.net/figure/Potentiel-de-Lennard-Jones_fig7_316555206

https://users.lal.in2p3.fr/puzo/thermo/ch2_thermo.pdf

https://www.universalis.fr/encyclopedie/fritz-london/

Travail réalisé et résultats

https://femto-physique.fr/physique_statistique/phystat_C4.php

http://www.cpge.eu/documents/presentations/diffusion-particules.pdf

https://openclassrooms.com/courses/apprenez-a-programmer-en-python

Annexes

Programmes

Programme 3D - Recherche du comportement Gaz Parfait

```
# -*-coding:Latin-1 -*

import os
import numpy as np
from numpy import arange
from pylab import *
import random
from math import pi
from math import sqrt
from scipy.integrate import quad
from scipy import interpolate
import matplotlib.pyplot as plt
from matplotlib import pyplot
from spicy import *
from mpl_toolkits.mplot3d import Axes3D


""" Définition des constantes """



kb=1.3806E-23  #Boltzman
Na=6.022E23 #Avogadro
d=3.405E-10 #distance intermoléculaire (Argon)
Mmol=40  #Masse molaire (Argon)
vmax0=2500 #vitesse maxixmale des particules
NbrEtoile=400 # points utiles pour initialiser les vitesses
r0=((2*(d**6))**(1/6)) # rayon de répulsion électrostatique
Nbr=125 #nombre de particules
T=500# Température(Kelvin)
R=1 #Facteur Ecini/Ep

E0=1.65324E-21 #kb*119,2 (Argon)
L=(Nbr)**(1/3)*r0*R
m=(Mmol*1E-3)/Na # masse atome (Argon)
Densité=Nbr/(L**3)

print("La densité est de ", Densité, " particules/m^3")


""" Définition des fonctions utiles en 3D """



def proba(v):
    dP=4*pi*((m/(2*pi*kb*T))**(3/2))*(v**2)*exp((-m/(2*kb*T))*v**2)
    return dP
```

```
def CalculEP(Pi,Pj,d,E0):
    vectij=Posi[Pj,:]-Posi[Pi,:]
    r=np.linalg.norm(vectij)

    if r>d:
        Ep=4*E0*((d/r)**12-(d/r)**6)
        normeF=-(24*d**6*E0*(r**6-2*d**6))/r**13
        F=vectij/r*normeF
    else:
        Ep=0
        F=np.zeros(3)
    return Ep, F



""" Premières instructions """

# vecteur ligne des coordonnées de NbrEtoiles
vetoile=linspace(0, vmax0, NbrEtoile)
Pr=np.zeros(NbrEtoile)
# on intègre la fonction proba entre 0 et vétoile
for i in range(NbrEtoile):
    (a,b)=quad(proba, 0, vetoile[i])
    Pr[i]=a

(c,d)=quad(proba,0,vmax0)
probamax=c
Pt=probamax*(np.random.rand(Nbr,1))
# analyse splin de la courbe de proba
tck=interpolate.splrep(Pr, vetoile, s=0)
# interpolation pour avoir la vitesse
NormevitesseTab=interpolate.splev(Pt, tck, der=0)

Vmax1=max(NormevitesseTab)
deltaini=(L/Vmax1)/100 # pas de temps initialisé pour la boucle

""" Initialisation des vitesses dans un repère sphérique """

u=np.random.rand(Nbr,1)
phi=2*pi*u # tableau d'angles phi aléatoires
v=np.random.rand(Nbr,1)
thetaa=arccos(1-2*v) # tableau d'angles theta aléatoires
vmoy=0
Vitesse1=np.zeros([Nbr,1]) # création des tableaux de vitesse vides
Vitesse2=np.zeros([Nbr,1])
Vitesse3=np.zeros([Nbr,1])
```

```
for i in range(Nbr):# on crée nos vitesses random initiales
    Vitesse1[i]=NormevitesseTab[i]*(sin(thetaa[i])*cos(phi[i]))
for i in range(Nbr):
    Vitesse2[i]=NormevitesseTab[i]*(sin(thetaa[i])*sin(phi[i]))
for i in range(Nbr):
    Vitesse3[i]=NormevitesseTab[i]*(cos(thetaa[i]))

t=0

Posi=np.random.rand(Nbr,3) # création du tableau de position vide
# position des particules aléatoirement dans  la boîte de coté L
Posi[:,0]=Posi[:,0]*L
Posi[:,1]=Posi[:,1]*L
Posi[:,2]=Posi[:,2]*L
Vmax=max(NormevitesseTab)
deltat=deltaini
# initalisation des tableaux d'énergies utiles pur la suite
Ec=np.zeros([5000,1])
Ep=np.zeros([5000,1])
Em=np.zeros([5000,1])
Tver=np.zeros([5000,1]) # initalisation du tableau de température
Temps=np.zeros([5000,1]) # initalisation du tableau de temps
Ecmoy=0 # énergie cinétique initale
Epmoy=0 # énergie potentielle initale
cmpt=0 # compteur de boucle
cmpt2=0 # compteur seconde boucle
Ppart=0 # quantité de mouvement
P=0 #pression initiale
EcM=0
tmax=6000*deltaini # temps max pour arrêt du programme
vmax2=0

""" Boucle principale """

plt.clf()
while t < tmax:
    TabForce=np.zeros([Nbr,3]) # tableau des forces initialisé à 0
    TabEp=np.zeros([Nbr,1]) # tableau des EP initialisé à 0
    cmpt=cmpt+1
    # énergies résultantes de toutes les interactions entre particules
    for i in range(Nbr):
        for j in range(Nbr):
            EpF=CalculEP(i,j,d,E0)
            TabForce[j,:]=TabForce[j,:]+EpF[1] # force éxercée de particule i sur j
            TabEp[j]=TabEp[j]+EpF[0] # EP résultante
```

```
for a in range(Nbr):
    # vitesses changées après interactions
    Vitesse1[a]=Vitesse1[a]+(TabForce[a,0]/m*deltat)
    Vitesse2[a]=Vitesse2[a]+(TabForce[a,1]/m*deltat)
    Vitesse3[a]=Vitesse3[a]+(TabForce[a,2]/m*deltat)
    # positions changées après interactions
    Posi[a,0]=Posi[a,0]+Vitesse1[a]*deltat
    Posi[a,1]=Posi[a,1]+Vitesse2[a]*deltat
    Posi[a,2]=Posi[a,2]+Vitesse3[a]*deltat

for q in range(Nbr): # conditions limites (rebonds sur les bords)
    if Posi[q,0]>L:
        Posi[q,0]=2*L-Posi[q,0] # revient dans le domaine si elle touche un bord
        Ppart=Ppart+sum(2*m*Vitesse1[q])
        Vitesse1[q]=-Vitesse1[q] # on inverse sa vitesse lors du rebond

    elif Posi[q,0]<0:
        Posi[q,0]=-Posi[q,0]
        Vitesse1[q]=-Vitesse1[q]
        Ppart=Ppart+sum(2*m*Vitesse1[q])

    elif Posi[q,1]>L:
        Posi[q,1]=2*L-Posi[q,1]
        Ppart=Ppart+sum(2*m*Vitesse2[q])
        Vitesse2[q]=-Vitesse2[q]

    elif Posi[q,1]<0:
        Posi[q,1]=-Posi[q,1]
        Vitesse2[q]=-Vitesse2[q]
        Ppart=Ppart+sum(2*m*Vitesse2[q])

    elif Posi[q,2]>L:
        Posi[q,2]=2*L-Posi[q,2]
        Ppart=Ppart+sum(2*m*Vitesse3[q])
        Vitesse3[q]=-Vitesse3[q]

    elif Posi[q,2]<0:
        Posi[q,2]=-Posi[q,2]
        Vitesse3[q]=-Vitesse3[q]
        Ppart=Ppart+sum(2*m*Vitesse3[q])


NormevitesseTab=sqrt(Vitesse1**2+Vitesse2**2+Vitesse3**2)
vmax2=max(NormevitesseTab)
deltat=(L/vmax2)/25
t=t+deltat
```

```
    if cmpt%2==0: # deuxième boucle pour ne pas faire trop d'affichages
        #plot d'une figure montrant l'évolution des particules dans le domaine
        plt.close()
        fig=plt.figure()
        ax=fig.add_subplot(111, projection='3d')
        ax.scatter(Posi[:,0]/r0, Posi[:,1]/r0, Posi[:,2]/r0, s=4, c='r')
        ax.set_xlim3d(0, L/r0)
        ax.set_ylim3d(0, L/r0)
        ax.set_zlim3d(0, L/r0)
        plt.show(block=False)

        Ec[cmpt2]=sum(0.5*m*(NormevitesseTab**2)) # calcul de l'EC
        Ep[cmpt2]=sum(TabEp) # calcul de l'EP
        Em[cmpt2]=Ec[cmpt2]+Ep[cmpt2] # calcul de l'EM
        Temps[cmpt2]=t # temps dans un tableau si besoin par la suite
        EcM=mean(0.5*m*(NormevitesseTab**2))  # calcul de l'EC moyenne
        Tver[cmpt2]=2*EcM/kb/3 # température expérimentale ( voir cours P8-1 )
        cmpt2+=1

""" Affichage final des courbes d'énergies """
#On ne veut pas qu'il y ait des 0 inutiles à la fin
Ec2=np.zeros([cmpt2,1])
Ep2=np.zeros([cmpt2,1])
Tver2=np.zeros([cmpt2,1])
Temps2=np.zeros([cmpt2,1])
Em2=np.zeros([cmpt2,1])

for i in range(cmpt2):
    Ec2[i]=Ec[i]
    Ep2[i]=Ep[i]
    Em2[i]=Em[i]
    Tver2[i]=Tver[i]
    Temps2[i]=Temps[i]

P=Ppart/6/L/L/t # calcul de la pression expérimentale
Trecalcul=mean(Tver2) # température expérimentale moyenne
Pth=0
Pth=((Nbr*kb*Trecalcul)/(L**3)) # calcul de la pression théorique
EcMoy=mean(Ec) # Ec moyenne
EpMoy=mean(Ep) # Ep moyenne
"""plot(Temps2,Ep2,color='b') #courbes d'énergies
plot(Temps2,Em2,color='r')
plt.clf()
plot(Temps2,Ec2,color='y')
plot(Temps2,Tver2,color='g')
plt.show()"""
os.system("pause")
```

Programme 3D - Modifications apportées pour une forte densité particulaire

```
Posi=np.zeros([Nbr,3])

for i in range(1,6):
    for j in range(1,6):
        for k in range(1,6):
            Posi[a,0]=(i-0.5)*L/5
            Posi[a,1]=(j-0.5)*L/5
            Posi[a,2]=(k-0.5)*L/5
            a=a+1
```

FIGURE 3.10 – Initialisation organisée des particules dans le domaine

Pour cela, nous créons un tableau de coordonnées x,y et z et nous le remplissons avec des positions organisées et régulièrement espacées dans l'espace 3D. On remarquera qu'ici, la commande **for i in range(1,6)** prend des valeurs propres à cette modélisation. Il convient donc d'ajuster les bornes de l'intervalle en fonction de $\frac{L}{r0}$ (représentant la taille de l'affichage choisi sur Python, pour ne pas s'encombrer de valeurs trop petites), sachant que dans ce cas-là il vaut 6.

```
def CalculEP(Pi,Pj,d,E0):
    vectij=Posi[Pj,:]-Posi[Pi,:]
    r=np.linalg.norm(vectij)

    if r>0:
        Ep=4*E0*((d/r)**12-(d/r)**6)
        normeF=-(24*d**6*E0*(r**6-2*d**6))/r**13
        F=vectij/r*normeF
    else:
        Ep=0
        F=np.zeros(3)
    return Ep, F
```

FIGURE 3.11 – Révision de la condition r>d pour autoriser ou non les calculs résultants d'une interaction entre 2 particules

Programme 2D - Recherche du coefficient de diffusion D

```
# -*-coding:Latin-1 -*

import os
import numpy as np
from numpy import arange
from pylab import *
import random
from math import pi
from math import sqrt
from scipy.integrate import quad
from scipy import interpolate
import matplotlib.pyplot as plt
from matplotlib import pyplot
from spicy import *
from math import floor
from math import log
from math import exp
from numpy import exp, linspace, random
from scipy.optimize import curve_fit
from statistics import mean
from numpy import ravel
from scipy import stats
""" Définition des constantes """

kb=1.3806E-23  #Boltzman
Na=6.022E23 #Avogadro
tmax=1E-8 #Duree maximum
Mmol=32  #Masse molaire (O2)
vmax=1300 #vitesse maximale des particules
NbrEtoile=400 #nombre de points pour initialiser les vitesses
Nbr=2000 #nombre de particules
T=298 # température (Kelvin)
R=1000 #Facteur Ecini/Ep
d=3.46E-10 #distance intermoleculaire (O2)
E0=1.6291E-21 #kb*119 (O2)
r0=(2*(d**6))**(1/6) #rayon de épulsion electrostatique
m=(Mmol*1E-3)/Na #masse atome (O2)
L=sqrt(Nbr)*r0*R

""" quelques formule théoriques pour calculer D théorique"""
vmoyth=sqrt((8*kb*T)/(pi*m)) # Vmoy via Maxweel-Boltzman
lpm=90e-9 #libre parcours moyen (O2)
Dth=(1/3)*vmoyth*lpm # Coefficient D théorique ( voir cours P8-1 )
print("Coeff de difuss",Dth,"m²/s")

""" fonctions utiles """
```

```
def gaussian(x, amp, cen, wid):

    return (amp / (sqrt(2*pi) * wid)) * exp(-(x-cen)**2 / (2*wid**2))

def proba(v):
    dP=((m*v)/(kb*T))*exp((-m*v**2)/(2*kb*T))
    return dP

def CalculEP(Pi,Pj,r0,E0):
    vectij=Posi[Pj,:]-Posi[Pi,:]
    r=np.linalg.norm(vectij)

    if r>d:
        Ep=4*E0*((d/r)**12-(d/r)**6)
        normeF=-(24*d**6*E0*(r**6-2*d**6))/r**13
        F=vectij/r*normeF
    else:
        Ep=0
        F=np.zeros(2)
    return Ep, F

""" Premières instructions """

vetoile=linspace(0, vmax, NbrEtoile) #vecteur ligne des coordonnées de NbrEtoiles

Pr=np.zeros(NbrEtoile)
for i in range(NbrEtoile): # on intègre la fonction proba entre 0 et vétoile
    (a,b)=quad(proba, 0, vetoile[i])
    Pr[i]=a

Pt=np.random.rand(Nbr,1)
tck=interpolate.splrep(Pr, vetoile, s=0) # analyse splin de la courbe de proba
NormevitesseTab=interpolate.splev(Pt, tck, der=0) #interpolation pour avoir la vitesse


thetaTab=2*pi*np.random.rand(Nbr,1) #tableau de nombres aléatoires
Vitesse1=np.zeros([Nbr,1])
Vitesse2=np.zeros([Nbr,1])
for i in range(Nbr): #création des vitesses des particules dans le domaine + angles
    Vitesse1[i]=NormevitesseTab[i]*cos(thetaTab[i])
for i in range(Nbr):
    Vitesse2[i]=NormevitesseTab[i]*sin(thetaTab[i])


Posi=np.zeros([Nbr,2])
t=0
for i in range(Nbr): #positionnement au milieu du domaine
```

```
    Posi[i,0]=random.uniform(0.4999999,0.5000001)
    Posi[i,1]=random.uniform(0.4999999,0.5000001)

Posi[:,0]=Posi[:,0]*L
Posi[:,1]=Posi[:,1]*L

#calcul de la densité au sein du petit domaine pour"
#séparer les 2 cas initiaux : GP ou GR ( on aura 2 courbes différentes à la fin )
miniX=min(Posi[:,0])
maxiX=max(Posi[:,0])
miniY=min(Posi[:,1])
maxiY=max(Posi[:,1])

cote1=maxiX-miniX # taille petit carré coté x
cote2=maxiY-miniY # taille petit carré coté y
Surfini=cote1*cote2 # surface petit carré

D=Nbr/Surfini #densité initiale
Dmax=2E25  # trouvé dans la 1ère expérience

print("La densité initiale est de", D,"parts/m^3")
if D < Dmax:
    print("La densité limite étant fixée à", Dmax,"parts/m^3, on est dans le cas du GParfait à t=0")
elif D >= Dmax:
    print("La densité limite étant fixée à",Dmax,"parts/m^3, on est dans le cas du GRéel à t=0")


deltat=(L/vmax)/100 # pas de temps initialisé pour la boucle
print("deltat pour N=", Nbr," particules est de ",deltat)

cpmt=0
taille=L*(10**8)   # on crée des valeurs plus maniables ( de l'ordre de 10^1 )
taille=floor(taille)

#On simplifie les calculs suivants en arrangeant
#la taille ( ex : taille=378 -> taillepropre=380 )
x=0
u=100
taillepropre=0
while u>0:
    if taille%10 < 5:
        u=taille-x
        if u%10==0:
            taillepropre=u
            break
        else:
```

```
    end=end+largeur

#initialisation de quelques tableaux
Ec=np.zeros([5000,1]) #énergie cinétique
tabsigma=np.zeros([5000,1]) # écart type
tabh=np.zeros([nbre,1]) # hauteur h (voir la suite)
Temps=np.zeros([5000,1])
Tver=np.zeros([5000,1]) #température
wid=wid0
vmax2=0
EcM=0 # énergie cinétique moyenne
cpmt=0
Ppart=0

while t <= tmax:
    TabForce=np.zeros([Nbr,2])
    TabEp=np.zeros([Nbr,1])

    for i in range(Nbr):
        for j in range(Nbr):
            EpF=CalculEP(i,j,r0,E0)
            TabForce[j,:]=TabForce[j,:]+EpF[1]
            TabEp[j]=TabEp[j]+EpF[0]

# si une particule sort du domaine, on compte juste sa quantité de mouvement

    for a in range(Nbr):
        Vitesse1[a]=Vitesse1[a]+(TabForce[a,0]/m*deltat)
        Vitesse2[a]=Vitesse2[a]+(TabForce[a,1]/m*deltat)
        Posi[a,0]=Posi[a,0]+Vitesse1[a]*deltat
        Posi[a,1]=Posi[a,1]+Vitesse2[a]*deltat

    for q in range(Nbr):
        if Posi[q,0]>L:
            Ppart=Ppart+sum(2*m*abs(Vitesse1[q]))
            """
            Vitesse1[q]=0
            """
        elif Posi[q,0]<0:
            Ppart=Ppart+sum(2*m*abs(Vitesse1[q]))
            """
            Vitesse1[q]=0
            """
        elif Posi[q,1]>L:
            Ppart=Ppart+sum(2*m*abs(Vitesse2[q]))
            """
```

```
            Vitesse2[q]=0
            """
        elif Posi[q,1]<0:
            Ppart=Ppart+sum(2*m*abs(Vitesse2[q]))
            """
            Vitesse2[q]=0
            """
j=0
debut=0
nb=0
fin=largeur
# à chaque boucle, on rempli un tableau de la répartition des particules
# dans nos bandes sur l'axe x
while fin <= (taillepropre):
    nb=0
    for i in range(Nbr):
        if (Posi[i,0]*(10**8)) >= debut and (Posi[i,0]*(10**8)) <= fin:
            nb=nb+1
    tabsurx[j]=nb
    j=j+1
    debut=debut+largeur
    fin=fin+largeur
# eventuellement print les particules dans le domaine
"""
plt.clf()
plt.plot(Posi[:,0]*(10**8), Posi[:,1]*(10**8),"d", label="pas de ligne", color='k', markersize='3')
plt.axis([0,L*(10**8),0,L*(10**8)])
plt.show(block=False)
plt.pause(0.005)
 """
# calcul du maximum de particules présent sur notre graphique ci dessous
maximum=int(max(tabsurx))

# graphique du nombre de particules ( densité ) en fonction de sa
#répartition sur l'axe X au cours du temps : Gaussienne

plt.clf()
plt.plot(Tabtestx,tabsurx, color='r')
plt.axis([0,taillepropre,0,maximum+50])
plt.xlabel("Axe X")
plt.ylabel("Nombre de Particules / Gausienne")
plt.plot(Tabtestx,tabh,color='b')
plt.show(block=False)
plt.pause(0.005)

# h est une condition d'arrêt du programme
h=maximum*exp(-1/2)
```

```
    n=round(float(n))
    for i in range(nbre):
        tabh[i]=h
    # h est forcément supérieur à 1 : trop bas = résultats faux

    NormevitesseTab=sqrt(Vitesse1**2+Vitesse2**2)
    vmax2=max(NormevitesseTab)
    xmoy=int(mean((Posi[:,0])*(10**8))) #posisition x moyenne

    # métode Gaussian-fit
    print("wid",wid)
    print("xmoy",xmoy)

    axex=ravel(Tabtestx) # on passe les tableaux en degré
    axey=ravel(tabsurx)
    init_vals=[maximum,xmoy,wid] # création de notre gaussienne avec nos paramètres
    best_vals, covar =curve_fit(gaussian, axex, axey, p0=init_vals, maxfev=100000)

    wid=wid+widloop
    tabsigma[cpmt]=(best_vals[2]) # on récupère l'écart type au carré de la gausienne
    EcM=np.mean(0.5*m*(NormevitesseTab**2))
    Ec[cpmt]=sum(0.5*m*(NormevitesseTab**2))

    Tver[cpmt]=EcM/kb # formule 2D !
    Temps[cpmt]=t
    print("Temp à cett boucle =",Tver[cpmt])

    t=t+deltat
    cpmt=cpmt+1
    if h <=10: # on arrête là car après les valeurs ne sont plus fiables
        print("on est sorti de force au bout de", cpmt,"boucles")
        break

# on élimine les cases vides
      Temps2=np.zeros([cpmt,1])
tabsigma2=np.zeros([cpmt,1])
Ec2=np.zeros([cpmt,1])
Tver2=np.zeros([cpmt,1])

for i in range(cpmt):
    Temps2[i]=Temps[i]
    tabsigma2[i]=tabsigma[i]
    Tver2[i]=Tver[i]
    Ec2[i]=Ec[i]

Pth=((Nbr*kb*T)/(L**2)*Na) # calcul de la pression théorique
```

```
Pexp=Ppart/4/L/t  # calcul de la pression expérimentale

# calcul de D expérimental
Trecalcul=np.mean(Tver2)   # vrai température moyenne du domaine
# le coefficient directeur de sigma² en fonction de t vaut 2D donc :
Dcourbe=((tabsigma2[cpmt-1]-tabsigma2[1])-(Temps2[cpmt-1]-Temps2[1]))
D=Dcourbe/2 # sigma²=2D*t d'ou la droite théorique de coef 2D
D=D/(10**8)  ( on avait  *10^8 au début )

print("Le coefficient de diffusion de l'oxygène expérimental est de ", D,"m^2/s")
print("CONDITIONS : T ini =", T,"K, Tvrai =", Trecalcul,"K")
print("Pth=",Pth,"et Pexp test =",Pexp)
      print("Le D oxygène théorique est de ", Dth,"m^2/s")

plt.clf()
plt.plot(Temps2,tabsigma2, color='r')
plt.xlabel("Temps (sc)")
plt.ylabel("Ecart-type^2")
plt.show()

os.system("pause")
```

Simulation sur Pov-Ray : exemple de programme 2D

```
#include "colors.inc"

#declare N=100;                                                          //Nombre de particules
#declare echelle=pow(10,7);                                              //Facteur d'échelle
#declare R=50;                                                           //Facteur Ecini/Ep
#declare ro=pow(10,-9);                                                  //Rayon de répulsion électrostatique
#declare D=2*ro*echelle*3;                                               //Diametre des particules
#declare Largeur=sqrt(N)*ro*R*echelle;                                   //Largeur du domaine

#declare NbImages=249;                                                   //Nombre d'images
#declare Camera_0 = camera {angle 25                                     //On déclare et on place une caméra
                            location    <Largeur/2 , Largeur/2, -3*Largeur> //Orientée de façon à ce que l'on puisse
                            //right x*image_width/image_height           //observer le domaine d'étude
                            look_at     <Largeur/2 , Largeur/2, 0>}
camera{Camera_0}
light_source { <2500,2500,-2500> color White }
sky_sphere { pigment { color White } }
light_source { <0,500,-500> color White }                                // On déclare une source de lumière

#fopen infos "2D_Ar_2540K_250Boucles_R=50_L=5E-7.txt" read  //On ouvre et on lit le fichier contenant les
                                                           //coordonnées des particules
#declare Count=1;                                          //Déclaration d'un compteur
#declare i=1;                                              //Initialisation des variables
#declare Sp_x=1;
#declare Sp_y=1;

       #while (Count <= NbImages)                    //Lecture associée à chaque image
            #for (i,1,N,1)                           //Lecture associée à chaque sphère
               #read (infos,Sp_x)                    //Coordonnée x de la sphère
               #read (infos,Sp_y)                    //Coordonnée y de la sphère
               //On ajoute les sphères aux instants antérieurs pour afficher la trajectoire de celles-ci
               #if (Count=frame_number-10)

                     sphere { <Sp_x*echelle,Sp_y*echelle,0>,D/2   //Ajout de la sphère à t=-10
                              texture {
                                 pigment {color Grey filter 0.9999999999} //Texture de la sphère
                                 finish {ambient rgbf <0.15, 0.15, 0.15, 0.15, 0.15>
                                         diffuse 0.85
                                         phong 1}
                                        }
                              }
               #end

               #if (Count=frame_number-9)

                     sphere { <Sp_x*echelle,Sp_y*echelle,0>,D/2  //Ajout de la sphère à t=-9
                              texture {
                                 pigment {color Grey filter 0.999999999} //Texture de la sphère
                                 finish {ambient rgbf <0.15, 0.15, 0.15, 0.15, 0.15>
                                         diffuse 0.85
                                         phong 1}
                                         }
                              }
               #end

                        #if (Count=frame_number-8)

                              sphere { <Sp_x*echelle,Sp_y*echelle,0>,D/2  //Ajout de la sphère à t=-8
                                       texture {
                                          pigment {color Grey filter 0.99999999} //Texture de la sphère
                                          finish {ambient rgbf <0.15, 0.15, 0.15, 0.15, 0.15>
                                                  diffuse 0.85
                                                  phong 1}
                                                 }
                                       }
                        #end

                        #if (Count=frame_number-7)

                              sphere { <Sp_x*echelle,Sp_y*echelle,0>,D/2  //Ajout de la sphère à t=-7
                                       texture {
                                          pigment {color Grey filter 0.9999999} //Texture de la sphère
                                          finish {ambient rgbf <0.15, 0.15, 0.15, 0.15, 0.15>
                                                  diffuse 0.85
                                                  phong 1}
                                                 }
                                       }
                        #end

                        #if (Count=frame_number-6)

                              sphere { <Sp_x*echelle,Sp_y*echelle,0>,D/2 //Ajout de la sphère à t=-6
                                       texture {
                                          pigment {color Grey filter 0.999999} //Texture de la sphère
                                          finish {ambient rgbf <0.15, 0.15, 0.15, 0.15, 0.15>
                                                  diffuse 0.85
                                                  phong 1}
                                                 }
                                       }
                        #end

                        #if (Count=frame_number-5)

                              sphere { <Sp_x*echelle,Sp_y*echelle,0>,D/2 //Ajout de la sphère à t=-5
                                       texture {
                                          pigment {color Grey filter 0.99999} //Texture de la sphère
                                          finish {ambient rgbf <0.15, 0.15, 0.15, 0.15, 0.15>
                                                  diffuse 0.85
                                                  phong 1
                                                  }

                                                 }
                                               }
                        #end
```

```
#if (Count=frame_number-4)

      sphere { <Sp_x*echelle,Sp_y*echelle,0>,D/2  //Ajout de la sphère à t=-4
               texture {
                  pigment {color Grey filter 0.9999} //Texture de la sphère
                  finish {ambient rgbf <0.15, 0.15, 0.15, 0.15, 0.15>
                          diffuse 0.85
                          phong 1}
                         }
               }
#end

#if (Count=frame_number-3)

      sphere { <Sp_x*echelle,Sp_y*echelle,0>,D/2 //Ajout de la sphère à t=-3
               texture {
                  pigment {color Grey filter 0.999} //Texture de la sphère
                  finish {ambient rgbf <0.15, 0.15, 0.15, 0.15, 0.15>
                          diffuse 0.85
                          phong 1}
                         }
               }
#end

#if (Count=frame_number-2)

      sphere { <Sp_x*echelle,Sp_y*echelle,0>,D/2  //Ajout de la sphère à t=-2
               texture {
                  pigment {color Grey filter 0.99}  //Texture de la sphère
                  finish {ambient rgbf <0.15, 0.15, 0.15, 0.15, 0.15>
                          diffuse 0.85
                          phong 1}
                        }
               }
#end

#if (Count=frame_number-1)

      sphere { <Sp_x*echelle,Sp_y*echelle,0>,D/2 //Ajout de la sphère à t=-1
               texture {
                  pigment {color Grey filter 0.9}  //Texture de la sphère
                  finish {ambient rgbf <0.15, 0.15, 0.15, 0.15, 0.15>
                          diffuse 0.85
                          phong 1}
                         }
               }
#end

#if (Count=frame_number) //On affiche les sphères de l'image courante "frame_number",
                         //pilotée par le fichier "Borwnian_anim2DGP.ini"
        sphere { <Sp_x*echelle,Sp_y*echelle,0>,D/2 //Ajout de la sphère à t=0
                 texture {
                  pigment {color Red}                //Texture de la sphère
                  finish {ambient rgbft <0.15, 0.15, 0.15, 0.15, 0.15>
                          diffuse 0.85
                          phong 1}
                         }
                }
 #end
                 #end
                 #declare Count=Count+1;
          #end

  #fclose infos                                //On ferme le fichier
```

FIGURE 3.12 – Programme principal Pov-Ray en 2D

Simulation sur Pov-Ray : exemple de programme 3D

```
#include "colors.inc"

#declare d=3.405*pow(10,-10);                                          //Distance intermoléculaire Argon
#declare echelle=pow(10,9);                                            //Facteur d'échelle
#declare R=10;                                                         //Facteur Ecini/Ep
#declare r0=pow((2*pow(d,6)),(1/6));                                   //Rayon de répulsion électrostatique
#declare N=125;                                                        //Nombre de particules
#declare Largeur=pow(N,(1/3))*r0*R*echelle;                            //Largeur du domaine
#declare D=2*r0*echelle;                                               //Diametre des particules

#declare NbImages=250;                                                 //Nombre d'images
#declare Camera_0 = camera {angle 25                                   //On déclare et on place une caméra
                            location    <2*Largeur, 2*Largeur, -4*Largeur> //Orientée de façon à ce que l'on
                            right x*image_width/image_height           //puisse observer le domaine d'étude
                            look_at     <Largeur/2, Largeur/2, Largeur/2>}
camera{Camera_0}
light_source { <2500,-2500,-2500> color White }
sky_sphere { pigment { color White } }
light_source { <0,-500,-500> color White }   //On déclare une source de lumière

#fopen infos "ExportPos.txt" read            //On lit le fichier contenant les coordonnées des particules
#declare Count=1;                            //Déclaration d'un compteur
#declare i=1;                                //Initialisation des variables
#declare Sp_x=1;
#declare Sp_y=1;
#declare Sp_z=1;

// Cube formé par des cylindres représentant le domaine d'étude
#declare Ra=D/4;                 //Rayon des cylindres
#declare L=Largeur;              //Longueur des cylindres
#declare Domaine = union{   sphere{<0,0,0>,Ra} //8 coins composés par des shpères
                            sphere{<L,0,0>,Ra}
                            sphere{<0,0,L>,Ra}
                            sphere{<L,0,L>,Ra}
                            sphere{<0,L,0>,Ra}
                            sphere{<L,L,0>,Ra}
                            sphere{<0,L,L>,Ra}
                            sphere{<L,L,L>,Ra}

                            cylinder {<0,0,0>,<L,0,0>,Ra} //4 cylindres dans la direction x
                            cylinder {<0,0,L>,<L,0,L>,Ra}
                            cylinder {<0,L,0>,<L,L,0>,Ra}
                            cylinder {<0,L,L>,<L,L,L>,Ra}
                                                          //4 cylindres dans la direction y
                            cylinder {<0,0,0>,<0,L,0>,Ra}
                            cylinder {<0,0,L>,<0,L,L>,Ra}
                            cylinder {<L,0,0>,<L,L,0>,Ra}
                            cylinder {<L,0,L>,<L,L,L>,Ra}
                                                          //4 cylindres dans la direction z
                            cylinder {<0,0,0>,<0,0,L>,Ra}
                            cylinder {<0,L,0>,<0,L,L>,Ra}
                            cylinder {<L,0,0>,<L,0,L>,Ra}
                            cylinder {<L,L,0>,<L,L,L>,Ra}

                            texture{pigment {color rgb<0.5,0.5,0.5> filter 0.99999}
                                    finish{ diffuse 0.9 phong 1}}
                           }

  object{Domaine scale 1              //Ajout du cube dans l'image
                 rotate<0,0,0>
                 translate<0,0,0>}

        #while (Count <= NbImages)                  //Lecture associée à chaque image
             #for (i,1,N,1)                         //Lecture associée à chaque sphère
                 #read (infos,Sp_x)                 //Coordonnée x de la sphère
                 #read (infos,Sp_y)                 //Coordonnée y de la sphère
                 #read (infos,Sp_z)                 //Coordonnée z de la sphère

                 #if (Count=frame_number)  //On n'affiche que les sphères de l'image courante "frame_number",
                                           //pilotée par le fichier "Borwnian_anim3D.ini"
                     sphere { <Sp_x*echelle,Sp_y*echelle,Sp_z*echelle>,D/2 //Ajout de la sphère
                              texture {
                                 pigment {color Red}                       //Texture de la sphère
                                 finish {ambient rgbft <0.15, 0.15, 0.15, 0.15, 0.15>
                                 diffuse 0.85
                                 phong 1}
                                      }
                            }
                 #end
             #end
             #declare Count=Count+1;
        #end
  #fclose infos                                     //On ferme le fichier
```

FIGURE 3.13 – Programme principal Pov-Ray en 3D

Simulation sur Pov-Ray : exemple de programme secondaire permettant la répétition du programme principal

```
+W800 +H600
Antialias=On
Antialias_Threshold=0.3
Antialias_Depth=2
Output_File_Type=S

Input_File_Name=2DGP.pov
Initial_Frame=1
Final_Frame=249
Initial_Clock=0
Final_Clock=1

Cyclic_Animation=off
Pause_when_Done=off
```

FIGURE 3.14 – Exemple de programme secondaire Pov-Ray

Démonstration des formules utilisées

A) **Coefficient de diffusion :**

$$div(\vec{j}) + \frac{\partial n}{\partial t} = 0$$

où $\vec{j}$ représente la densité de courant de particules et n le nombre de particules au sein du domaine considéré.
En 1D, cette équation se réécrit :

$$\frac{\partial n}{\partial t} = D\frac{\partial^2 n}{\partial x^2}$$

En effet, la loi de Fick vue en P8-1 nous dit que $\vec{j} = -D\vec{\nabla}n$. Ensuite, nous nous sommes basés sur le fait que la fonction représentant l'évolution de la densité particulaire de notre domaine dans le temps devait avoir une forme de Gaussienne. Nous avons donc posé la fonction :

$$n(x) = \frac{Nbr}{\sqrt{2\pi}\sigma} e^{\frac{-1}{2}(\frac{x}{\sigma})^2}$$
$$\frac{\partial^2 n}{\partial x^2} = \frac{Nbr}{\sqrt{2\pi}\sigma}(\frac{x^2}{\sigma^4} - \frac{1}{\sigma^2}) e^{\frac{-1}{2}(\frac{x}{\sigma})^2}$$

On remarque aussi que comme σ est fonction du temps, on peut aussi avoir :

$$n(t) = \frac{Nbr}{\sqrt{2\pi}} \frac{e^{\frac{-1}{2}(\frac{x}{\sigma(t)})^2}}{\sigma(t)} \quad \frac{\partial n}{\partial t} = \frac{\partial n}{\partial \sigma}\frac{\partial \sigma}{\partial t} = \frac{Nbr}{\sqrt{2\pi}}(\frac{x^2}{\sigma^4} - \frac{1}{\sigma^2}) e^{\frac{-1}{2}(\frac{x}{\sigma})^2} \frac{\partial \sigma}{\partial t}$$

Finalement, en réorganisant ces formules, on trouve que

$$\sigma(t)^2 = 2Dt$$

B) **Formules 2D :**
Calcul de la loi de Maxwell-Boltzmann :

$$F_0(v) = f(V_x)f(V_y)$$

$$F_0(v) = (\frac{\alpha}{\pi}) \times e^{-\alpha v^2}$$

$$d^2N = NF_0(v)d_v^2 = N(\frac{\alpha}{\pi}) \times e^{-\alpha v^2} v d_v d_\theta$$

On intègre θ entre 0 et 2π :

$$dN = 2\pi N(\frac{\alpha}{\pi}) \times e^{-\alpha v^2} v d_v$$

Ce qui est équivalent à :

$$dN = v\left(\frac{m}{k_b T}\right) \times e^{-\frac{mv^2}{2k_b T}} d_v$$