UNIVERSITE HASSAN II

FACULTE DES SCIENCES AIN CHOCK
UNIVERSITE HASSAN II DE CASABLANCA

Jouer avec les particules :

Simulation python de fission,fusion, collision et Modèle Cucker-Smale.

Rapport préparé par O.KHADRAOUI, L.FANNANE , H.EL MAHBOUBY,A.KHFIF

à l'égard de

Pr. K.TLEMÇANI

janvier 2024

Table des matières

1 Résumé

Dans le vaste domaine de la physique nucléaire, nous plongeons dans les phénomènes captivants de la fusion, de la fission et des collisions, des terrains où la théorie fondamentale et l'ingénierie appliquée se croisent. Notre exploration approfondie se concentre sur la programmation sophistiquée des codes dédiés à la modélisation nucléaire, mettant en lumière le mariage complexe entre les principes théoriques, les avancées algorithmiques et les compétences en programmation.

Au cœur de cette étude se trouve la prééminence du langage de programmation Python dans le domaine de la modélisation nucléaire. Ce choix éclairé repose sur la combinaison unique de la flexibilité intrinsèque de Python avec la puissance requise pour résoudre les équations complexes régissant la fusion, la fission et les collisions nucléaires. La clarté syntaxique de Python, sa riche bibliothèque et sa communauté active en font un choix judicieux pour le développement de codes de simulation nucléaire avancés. La mise en œuvre pratique de simulations nucléaires en Python s'appuie fréquemment sur des bibliothèques spécialisées telles que NumPy, SciPy et SymPy. Ces outils permettent le traitement numérique, la résolution symbolique d'équations et l'analyse statistique des résultats, fournissant ainsi un terrain fertile pour la modélisation précise des phénomènes nucléaires à différentes échelles.

L'approche objet en Python émerge comme un facilitateur essentiel, permettant la conception de codes modulaires pour une réutilisabilité efficace et une gestion simplifiée des composants complexes des simulations. En incorporant des techniques de programmation orientée objet, cette approche réaliste représente de manière fidèle les éléments nucléaires, leurs interactions et les multiples paramètres influant sur les processus étudiés. Le recours à des bibliothèques spécialisées dans la visualisation des données, telles que Matplotlib, enrichit davantage l'arsenal du programmeur, offrant la possibilité de créer des représentations graphiques explicites des résultats des simulations nucléaires. Cette visualisation cruciale facilite l'interprétation des modèles générés et la validation des hypothèses théoriques sous-jacentes.

Passant de la programmation à la physique des particules, nous explorons les collisions nucléaires, définissant ces interactions de courte durée et de courte portée entre particules. Les applications dans la physique des particules s'étendent de la découverte de nouvelles particules à l'étude de la structure fondamentale de la matière. Les collisions, qu'elles soient élastiques ou inélastiques, ouvrent des fenêtres sur la validation des théories, l'étude des interactions fortes et la recherche de matière noire.

En se plongeant dans le domaine de la fission nucléaire, ce rapport offre un aperçu approfondi de ce processus où un noyau atomique lourd se divise en deux noyaux plus petits,

libérant ainsi une quantité considérable d'énergie. L'importance cruciale de contrôler cette réaction en chaîne dans les réacteurs nucléaires est soulignée, mettant en avant les défis et les avantages de cette source d'énergie à faible émission de carbone.

Ainsi, cette exploration transcende les frontières entre la théorie et la pratique, offrant une plateforme souple et puissante pour explorer les mécanismes subtils de la fusion, de la fission et des collisions nucléaires.

Au cœur de cette étude se trouve la prééminence du langage de programmation Python dans le domaine de la modélisation nucléaire. Ce choix éclairé repose sur la combinaison unique de la flexibilité intrinsèque de Python avec la puissance requise pour résoudre les équations complexes régis...

2 Introduction générale

Dans le panorama complexe de la physique nucléaire, la fusion, la fission et les collisions constituent des phénomènes intrinsèquement captivants, marquant l'intersection entre la théorie fondamentale et l'ingénierie appliquée. Cette étude approfondie s'attache à dévoiler les intrications fascinantes de ces processus nucléaires, offrant un aperçu éclairant des mécanismes intimes régissant la libération et la manipulation de l'énergie à l'échelle atomique.

Au cœur de cette exploration réside la programmation sophistiquée des codes dédiés à la modélisation et à la simulation de la fusion, de la fission et des collisions nucléaires. Ces codes informatiques, véritables architectes virtuels, transcendent les limites de la compréhension humaine, sondant les arcanes de l'infiniment petit avec une précision inégalée. Leur élaboration méticuleuse requiert une symbiose exceptionnelle entre les principes théoriques de la physique nucléaire, les avancées algorithmiques et les compétences pointues en programmation.

Dans le domaine de la modélisation nucléaire, l'utilisation du langage de programmation Python émerge comme une approche privilégiée, combinant la flexibilité inhérente à ce langage avec la puissance nécessaire à la résolution des équations complexes régissant la fusion, la fission et les collisions nucléaires. La syntaxe claire de Python, sa bibliothèque riche et sa communauté active en font un choix judicieux pour le développement de codes de simulation nucléaire avancés.

La mise en œuvre de simulations nucléaires en Python repose souvent sur l'utilisation de bibliothèques spécialisées telles que NumPy, SciPy et SymPy, permettant le traitement numérique, la résolution symbolique d'équations et l'analyse statistique des résultats. Ces bibliothèques, combinées à des structures de données adaptées, offrent un terrain propice à

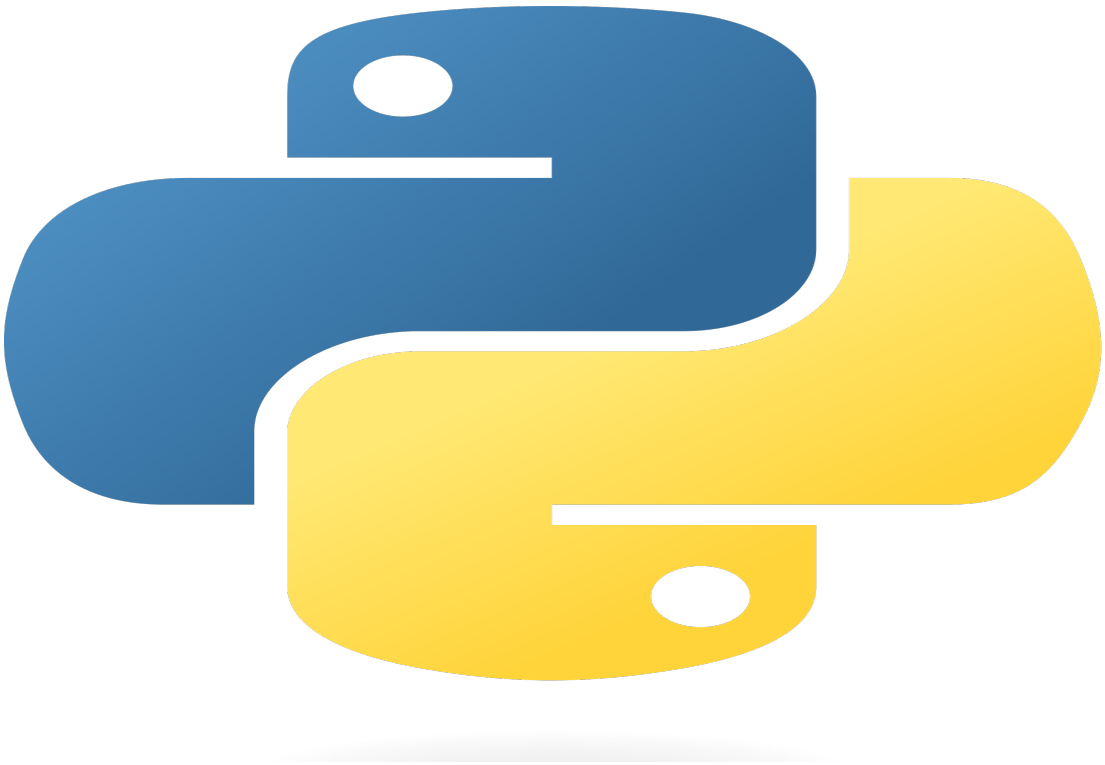
la modélisation précise des phénomènes nucléaires à différentes échelles.

Le recours à l'approche objet en Python facilite la conception de codes modulaires, favorisant la réutilisabilité du code et la gestion efficace des composants complexes des simulations. L'intégration de techniques de programmation orientée objet permet ainsi de représenter de manière réaliste les éléments nucléaires, leurs interactions et les multiples paramètres qui influent sur les processus étudiés.

L'utilisation de bibliothèques dédiées à la visualisation des données, telles que Matplotlib, enrichit davantage l'arsenal du programmeur, offrant la possibilité de créer des représentations graphiques explicites des résultats obtenus au cours des simulations nucléaires. Cette visualisation est cruciale pour l'interprétation des modèles générés et la validation des hypothèses théoriques sous-jacentes.

l'approche de programmation en Python pour la simulation nucléaire transcende les frontières entre la théorie et la pratique, offrant une plateforme souple et puissante pour explorer les mécanismes subtils de la fusion, de la fission et des collisions nucléaires. Ce rapport se penchera sur les nuances de cette implémentation, mettant en lumière les aspects cruciaux de la programmation Python dans le contexte captivant de la modélisation nucléaire avancée.

3 Introduction à l'utilisation de Python dans les Simulations Nucléaires



3.1 Rôle de Python dans la Recherche Nucléaire

Python joue un rôle crucial dans le domaine nucléaire en tant que langage de programmation polyvalent. Sa syntaxe claire et sa facilité d'utilisation en font un choix privilégié pour la modélisation mathématique complexe requise en physique nucléaire, tandis que des bibliothèques spécialisées telles que NumPy et SciPy offrent des outils puissants pour la manipulation de données et le calcul numérique. Python excelle également dans la création de simulations nucléaires, grâce à sa flexibilité et à des bibliothèques dédiées comme PyNE. Il est utilisé pour analyser des données expérimentales avec des outils tels que Pandas et Matplotlib, facilitant ainsi l'interprétation des résultats. En outre, sa capacité à interagir avec d'autres langages simplifie l'intégration avec des outils spécialisés, et sa rapidité de développement le rend idéal pour le prototypage. Python est largement adopté en éducation pour sa simplicité, et sa communauté active et sa nature open source encouragent la collaboration, permettant aux chercheurs de contribuer à des projets existants tout en développant de nouvelles solutions pour la physique nucléaire. La montée en popularité de Python dans ce domaine témoigne de son efficacité pour relever les défis complexes de la recherche nucléaire.

3.2 Python pour la Modélisation Nucléaire

Comme on avait déjà mentionné Python est largement utilisé dans la modélisation nucléaire en raison de sa polyvalence, de sa facilité d'utilisation et de son riche écosystème de bibliothèques scientifiques. En physique nucléaire, la modélisation implique souvent la résolution d'équations complexes, la simulation de phénomènes nucléaires tels que la fission, la fusion ou la diffusion de particules, ainsi que l'analyse de données expérimentales. Python, avec des bibliothèques comme NumPy, SciPy et Matplotlib, offre un environnement de développement puissant permettant de formuler, résoudre et visualiser ces modèles mathématiques et simulations. La flexibilité de Python en fait un choix idéal pour les chercheurs et ingénieurs qui peuvent rapidement prototyper, tester et itérer leurs modèles, contribuant ainsi à l'avancement de la recherche en physique nucléaire.

3.3 Avantages de Python dans les Simulations

Facilité d'Apprentissage : Python est réputé pour sa syntaxe simple et lisible, ce qui facilite l'apprentissage, même pour les débutants. Cela permet aux chercheurs en physique nucléaire d'entrer rapidement dans le développement de simulations sans passer beaucoup de temps sur la maîtrise du langage.

Polyvalence : Python est un langage polyvalent qui peut être utilisé pour une variété de tâches, de la modélisation mathématique à la manipulation de données et à la visualisation. Cette polyvalence est cruciale pour les simulations nucléaires qui peuvent impliquer divers aspects, tels que la fission, la fusion, la diffusion de particules, etc.

Écosystème de Bibliothèques Scientifiques : Python dispose d'un écosystème riche en bibliothèques scientifiques telles que NumPy, SciPy et Matplotlib. Ces bibliothèques facilitent la manipulation de données, le calcul numérique et la visualisation, offrant ainsi des outils puissants pour les chercheurs en physique nucléaire.

Rapidité de Développement : La syntaxe concise de Python et sa facilité de prototypage permettent un développement rapide de simulations. Les chercheurs peuvent itérer plus rapidement sur leurs modèles, tester différentes approches et adapter rapidement leurs simulations en fonction des résultats.

Grande Communauté : Python bénéficie d'une vaste communauté d'utilisateurs et de développeurs. Cela signifie un accès facile à des ressources en ligne, des forums de discussion et des bibliothèques tierces, ce qui est précieux pour les chercheurs en physique nucléaire qui peuvent partager des connaissances et résoudre des problèmes communs.

Interactivité : L'interactivité de Python, notamment à travers des environnements comme

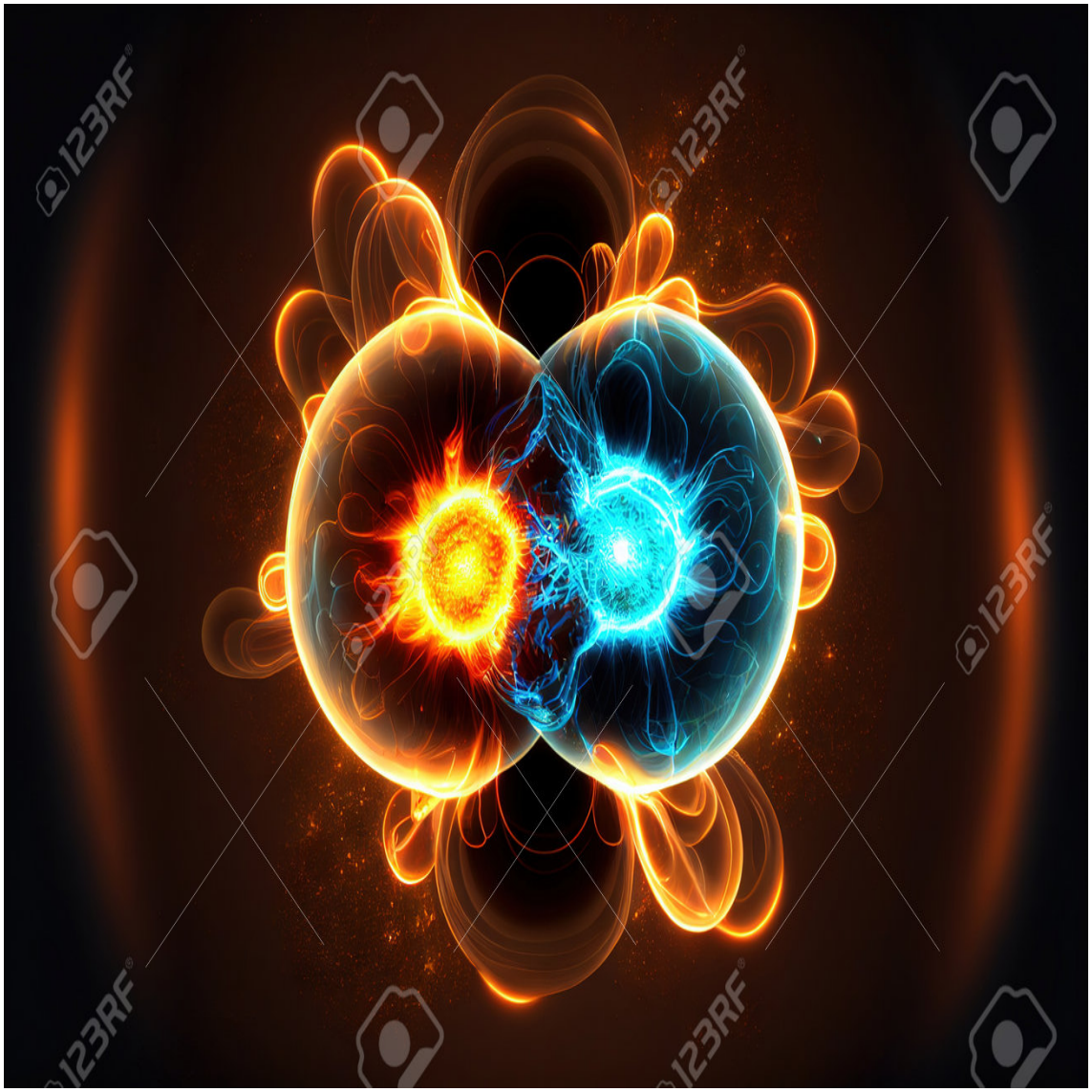
Jupyter Notebooks, permet aux chercheurs de tester et de visualiser leurs simulations pas à pas, ce qui facilite le processus de développement et de débogage.

Open Source : Python est un langage open source, ce qui signifie que les chercheurs ont un accès complet au code source. Cela favorise la transparence et la reproductibilité des simulations, des aspects cruciaux dans la recherche scientifique, y compris en physique nucléaire.

Intégration avec d'autres Langages : Python peut être facilement intégré avec d'autres langages de programmation, ce qui est utile lorsque des codes existants ou des outils spécialisés doivent être utilisés en conjonction avec les simulations.

3.4 Collision nucléaire

3.4.1 Définition



On dit qu'il y a collision ou choc entre deux ou plusieurs particules quand ces objets subissent une interaction mutuelle de courte durée et de courte portée. Le choc est localisé dans le temps et l'espace. En règle générale, les forces d'interaction sont négligeables quand les particules sont suffisamment éloignées. On peut donc distinguer un (avant) et un (après) la collision.

Ainsi, avant et après la collision, les particules se déplacent en ligne droite avec des vitesses uniformes. On notera Vi la vitesse d'une particule avant le choc et Vi celle après. On a deux sortes de collision :

Collision élastique :On dit qu'il y a collision élastique lorsque le nombre de particules

reste constant et que l'énergie interne de chaque particule reste inchangée avant et après le choc. En d'autres termes, les particules ne se déforment pas ni ne changent de nature.

collision inélastique :On dit qu'une collision est inélastique lorsqu'une partie de l'énergie cinétique initiale du système s'est transformée en d'autres formes d'énergie. La collision s'accompagne alors d'une variation d'énergie interne et/ou d'une modification du nombre de particules, certaines pouvant être créées par fragmentation ou par équivalence masse-énergie

3.4.2 Application dans la physique des particules

Les collisions dans la physique des particules jouent un rôle crucial pour comprendre la structure fondamentale de la matière. Voici quelques applications importantes de collisions dans ce domaine

Découverte de Nouvelles Particules : Les collisions à haute énergie, telles que celles réalisées au Grand Collisionneur de Hadrons (LHC) du CERN, permettent de créer des conditions similaires à celles qui existaient peu de temps après le Big Bang. Ces collisions permettent de produire de nouvelles particules qui peuvent être observées et étudiées.

Étude de la Structure des Particules : En analysant les produits de collisions, les physiciens des particules peuvent comprendre la structure interne des particules élémentaires. Par exemple, les collisions peuvent révéler la distribution des quarks à l'intérieur des protons et des neutrons.

Validation des Théories : Les collisions à haute énergie fournissent un moyen de tester et de valider les théories physiques, en particulier le Modèle Standard de la physique des particules. Les résultats des expériences peuvent confirmer ou réfuter les prédictions théoriques.

Étude des Interactions Fortes : Les collisions entre hadrons (particules composées de quarks) permettent d'étudier les interactions fortes qui régissent le comportement des quarks. Cela aide à comprendre la chromodynamique quantique (QCD), la théorie décrivant ces

interactions.

Recherche de Matière Noire et d'Antimatière : Les collisions à haute énergie peuvent fournir des indices sur l'existence de particules de matière noire ou d'antimatière, qui constituent une grande partie de l'univers mais restent largement méconnues.

Étude des Bosons W et Z : La découverte des bosons W et Z, médiateurs des forces faibles, a été réalisée grâce à des collisions à haute énergie. Cela a contribué à valider la théorie électrofaible, unifiant les forces électromagnétiques et faibles.

Étude des Quarks Top et Higgs : La découverte du quark top au Tevatron et du boson de Higgs au LHC a également été réalisée à travers des collisions à haute énergie, confirmant ainsi la prédiction théorique de l'existence de ces particules.

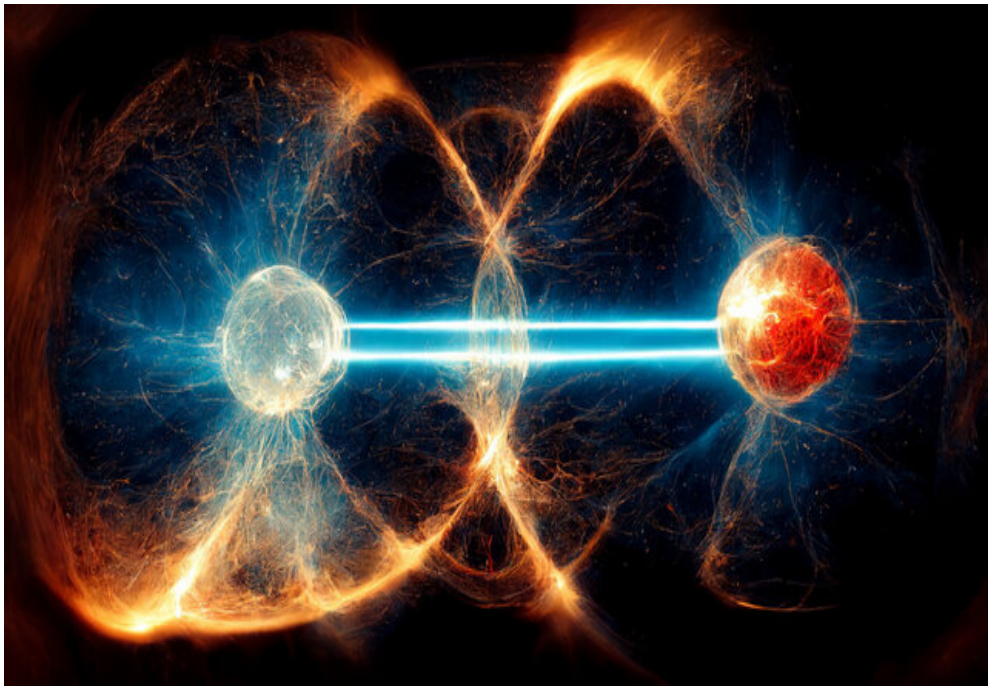
Prenons un exemple concrés pour mieux comprendre la collision : Dans l'expérience ATLAS au CERN, différents types de particules traversent les détecteurs MPX. Dans ce chapitre, nous allons voir comment chaque type de particule interagit avec le détecteur afin de distinguer chacune d'elles et mesurer leurs flux respectifs dans la caverne ATLAS. Les particules lourdes chargées Une particule lourge chargée (proton, alpha, deutérium, etc.) entrant dans la matière perd son énergie cinétique principalement par collision. Lors d'une collision avec un atome de silicium, la particule lourde chargée transfère de l'énergie au milieu entraînant une excitation ou une ionisation de l'atome. L'excitation de l'atome consiste en un passage d'un électron atomique d'une couche d'énergie à une autre d'énergie supérieure. Cette excitation n'entraîne pas la création de porteurs de charge. L'ionisation se produit quand un ou plusieurs électrons atomiques reçoivent une énergie suffisante pour être éjectés. Il y a donc une création de paires électron-trou dans un solide (ou de paires électron-ion dans un gaz). Lorsque l'électron éjecté a une énergie suffisamment grande,

il peut donner lieu à une ionisation à son tour, ce sont les rayons . Dans le silicium, il 20 faut une énergie de 3.62 eV pour créer une paire électron-trou. On peut décrire le taux moyen de la perte d'énergie d'une particule lourde chargée dans la matière par l'équation de Bethe-Bloch Une particule lourde chargée traversant la matière dépose son énergie surtout en ionisant le milieu. Le long de son parcours, la particule perd donc de l'énergie et son pouvoir d'arrêt dE/dx augmente avec la distance parcourue dans la matière. Lorsque l'énergie restante de la particule incidente n'est pas assez grande pour ioniser, elle s'arrête. Les particules lourdes chargées déposent la majeure partie de leur énergie à la fin de leur parcours. La courbe représentant la variation du pouvoir d'arrêt avec la distance parcourue dans la matière est la courbe de Bragg avec un pic appelé le pic de Bragg marquant le point d'arrêt de la particule Électrons et positrons Les électrons interagissent avec le milieu envi-

ronnant par collisions et radiation. Du fait de leur petite masse, ils subissent de multiples déviations avant de perdre de leur énergie et de s'arrêter. Un électron traversant la partie active du détecteur MPX perdra de l'énergie soit par collision, (dE/dx)coll, avec des électrons du silicium ou par Bremsstrahlung,(dE/dx)ra La perte d'énergie par radiation se fait par Bremsstrahlung, c'est-à-dire par émission de photons. C'est le processus majeur pour la perte d'énergie des électrons dans la matière si l'énergie de l'électron est supérieure à une énergie critique 2 . L'équation qui représente la perte d'énergie par Bremsstrahlung Les photons Les photons peuvent déposer leur énergie dans la matière via trois processus : l'effet photoélectrique, l'effet Compton et la création de paires. Pour le silicium, à petite énergie , l'effet photoélectrique domine, à grande énergie la création de paires et à moyenne énergie , la diffusion Compton . La détection de photons dans le détecteur MPX se fait donc par la détection des électrons ou positrons émis dans ces trois processus. Les neutrons Les neutrons ne sont pas électriquement chargés, donc ils n'ionisent pas directement le silicium. Pour les détecter, on place des convertisseurs au-dessus du silicium de façon à ce que les neutrons réagissent avec les matériaux des convertisseurs et produisent des particules chargées qui, elles, créeront des paires électron-trou dans le silicium. Pour cette raison, des couches de polyéthylène (PE) et de fluorure de lithium (6LiF) sont placées devant le détecteur. Il est toutefois possible qu'un neutron soit directement détecté dans le milieu actif du MPX en produisant une réaction nucléaire directement dans le silicium Collisions élastiques pour la détection des neutrons rapides Une façon efficace de détecter les neutrons rapides est par collision élastique avec des matériaux légers. Lors d'une collision élastique d'un neutron avec un noyau, une partie de l'énergie cinétique du neutron est transférée au noyau de recul. Le noyau de recul perd à son tour son énergie par collision ou par radiation comme nous avons vu plus haut (section des 4.1). Un neutron entrant dans la matière transfère son énergie aux particules chargées par collision élastique. Le transfert d'énergie dépend du nombre de masse du matériau. Par conservation de l'énergie et de l'impulsion

4 Fission

4.1 Définition



La fission nucléaire est un processus dans lequel le noyau d'un atome lourd, tel que l'uranium-235 ou le plutonium-239, est bombardé par un neutron, provoquant sa division en deux noyaux plus petits, appelés produits de fission. Lorsque la division du noyau se produit, une grande quantité d'énergie est libérée sous forme de chaleur, de rayonnement électromagnétique et de neutrons.

Ce processus est accompagné de la libération de plusieurs neutrons supplémentaires. Si ces neutrons supplémentaires sont capables de provoquer la fission d'autres noyaux, une réaction en chaîne peut se produire, libérant ainsi plus d'énergie et alimentant un réacteur nucléaire. Contrôler cette réaction en chaîne est crucial pour la production d'énergie nucléaire. Cependant, si cette réaction n'est pas contrôlée correctement, elle peut entraîner une libération incontrôlée d'énergie, potentiellement avec des conséquences destructrices, comme dans le cas d'une explosion nucléaire.

La fission nucléaire est utilisée dans les réacteurs nucléaires pour produire de l'électricité. Les réacteurs contrôlent soigneusement la réaction en chaîne pour générer de la chaleur, qui est ensuite utilisée pour produire de la vapeur et actionner des turbines afin de produire de l'électricité. Cette méthode de production d'énergie est considérée comme une source d'énergie à faible émission de carbone, mais elle présente également des défis en matière de gestion des déchets radioactifs et de sécurité.

4.1.1 Réactions en chaîne et contrôle de la fission

La réaction en chaîne dans la fission nucléaire est un processus où la division d'un noyau atomique lourd, comme l'uranium-235 ou le plutonium-239, libère des neutrons supplémentaires. Ces neutrons peuvent à leur tour provoquer la fission d'autres noyaux, libérant davantage de neutrons et d'énergie.

Pour contrôler cette réaction en chaîne dans un réacteur nucléaire, plusieurs méthodes sont utilisées :

Régulation neutronique : Les neutrons sont régulés pour maintenir la réaction à un niveau constant. Des matériaux appelés modérateurs sont utilisés pour ralentir les neutrons afin qu'ils interagissent plus efficacement avec les noyaux fissiles. Les barres de contrôle, souvent composées de matériaux absorbant les neutrons, sont insérées ou retirées du réacteur pour ajuster la réaction.

Gestion de la température : Le contrôle de la température est crucial pour éviter une surchauffe. Des systèmes de refroidissement, tels que des liquides de refroidissement, sont utilisés pour extraire la chaleur générée par les réactions de fission et éviter toute montée en température excessive.

Sûreté et dispositifs de sécurité : Les réacteurs nucléaires sont équipés de multiples dispositifs de sécurité pour prévenir les accidents. Ils incluent des systèmes de contrôle automatisés, des dispositifs d'arrêt d'urgence et des dispositifs de confinement conçus pour contenir tout rejet radioactif en cas de situation d'urgence.

La combinaison de ces éléments permet de maintenir la réaction en chaîne sous contrôle et d'assurer son fonctionnement sûr et stable. La régulation précise de la réaction et la gestion des conditions opérationnelles sont essentielles pour maximiser l'efficacité de la production d'énergie tout en garantissant la sécurité des installations et de l'environnement.

4.2 Applications pratiques (centrales nucléaires, armes nucléaires)

La fission nucléaire a des applications pratiques majeures dans deux domaines principaux : la production d'électricité via les centrales nucléaires et la création d'armes nucléaires.

Centrales nucléaires : Elles sont conçues pour générer de l'électricité de manière constante et contrôlée. Les réacteurs utilisent la fission nucléaire pour chauffer de l'eau dans un circuit fermé et produire de la vapeur. Cette vapeur fait tourner des turbines, qui à leur tour actionnent des générateurs pour produire de l'électricité. Les centrales nucléaires sont considérées comme une source d'énergie à faible émission de carbone et peuvent fournir une quantité significative d'électricité de base.

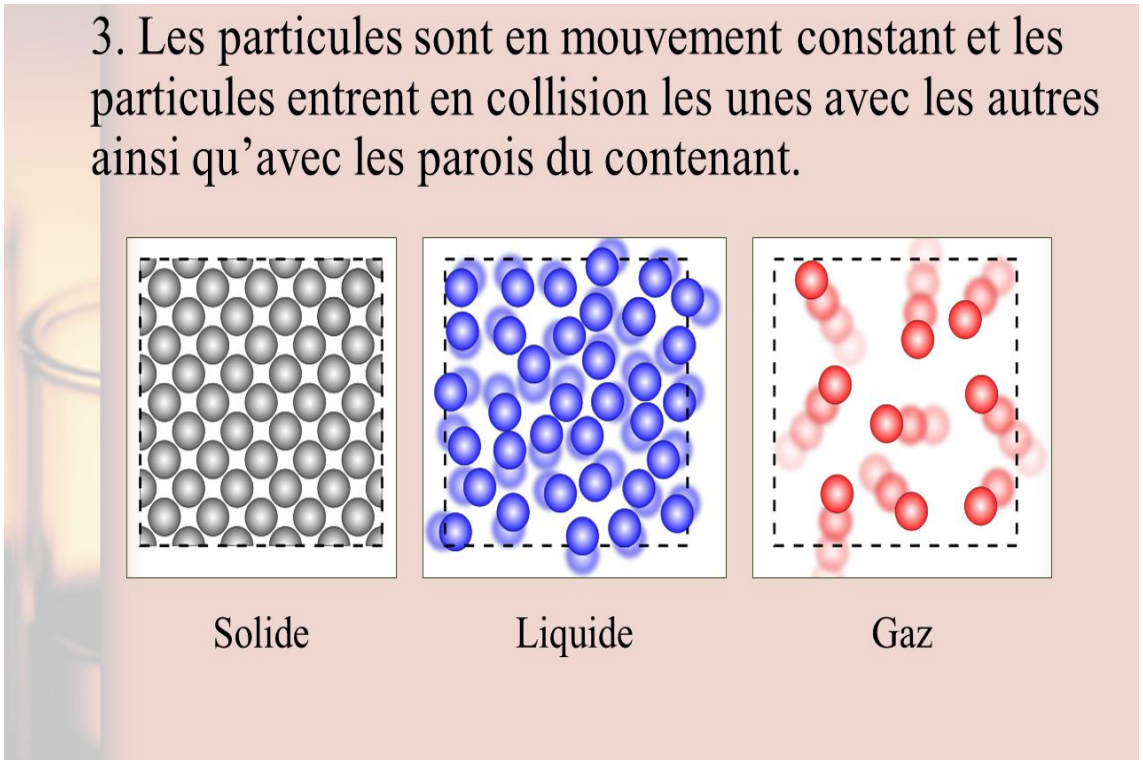
Armes nucléaires : Les armes nucléaires exploitent la fission pour créer des explosions extrêmement puissantes. Les bombes atomiques fonctionnent en concentrant une quantité suffisante de matériau fissile pour initier une réaction en chaîne non contrôlée, libérant une énergie colossale sous forme d'explosion. Les bombes thermonucléaires, ou bombes à hydrogène, utilisent également la fission pour déclencher une fusion nucléaire, produisant une explosion encore plus puissante.

Recherche scientifique : Les réactions nucléaires sont utilisées dans la recherche pour explorer les propriétés fondamentales de la matière. Des accélérateurs de particules et des réacteurs nucléaires sont employés pour étudier la physique nucléaire, la structure de la matière et la radioactivité.

Isotopes pour la médecine et l'industrie : Certains isotopes radioactifs résultant de la fission sont utilisés en médecine pour le diagnostic et la thérapie, par exemple dans l'imagerie médicale ou la radiothérapie pour traiter le cancer. Ils ont également des applications industrielles, comme dans la mesure de l'épaisseur des matériaux ou la détection de fuites. Propulsion nucléaire spatiale : La fission nucléaire a été étudiée pour propulser des engins spatiaux. Les concepts de réacteurs nucléaires utilisent la chaleur générée par la fission pour propulser des vaisseaux spatiaux, offrant potentiellement une source d'énergie très puissante et durable pour l'exploration spatiale.

Ces applications diverses montrent à quel point la fission nucléaire peut être utilisée dans des domaines variés, allant de la production d'énergie à des fins pacifiques à des applications de pointe en recherche et dans l'industrie, tout en soulignant les responsabilités et les défis associés à cette technologie.

5 Mouvement des particules



Le mouvement des particules peut être décrit de différentes manières selon le contexte,

qu'il s'agisse de particules subatomiques dans la physique quantique, de particules dans un fluide ou de particules dans un système en thermodynamique.

Physique classique : Les lois du mouvement de Newton, comme la deuxième loi (F = ma) ou la troisième loi (action-réaction), décrivent comment les objets se déplacent en réponse à des forces. Par exemple, un objet accélère dans la direction d'une force nette appliquée sur lui, tandis que des forces égales et opposées agissent entre deux objets en interaction.

Physique quantique : Au niveau subatomique, les particules peuvent présenter des comportements ondulatoires, décrits par des équations d'onde de Schrödinger. Ces équations décrivent la probabilité de trouver une particule à un endroit donné et à un moment donné. Les particules subatomiques peuvent également manifester des caractéristiques de dualité onde-particule, agissant à la fois comme des ondes et des particules selon le contexte expérimental.

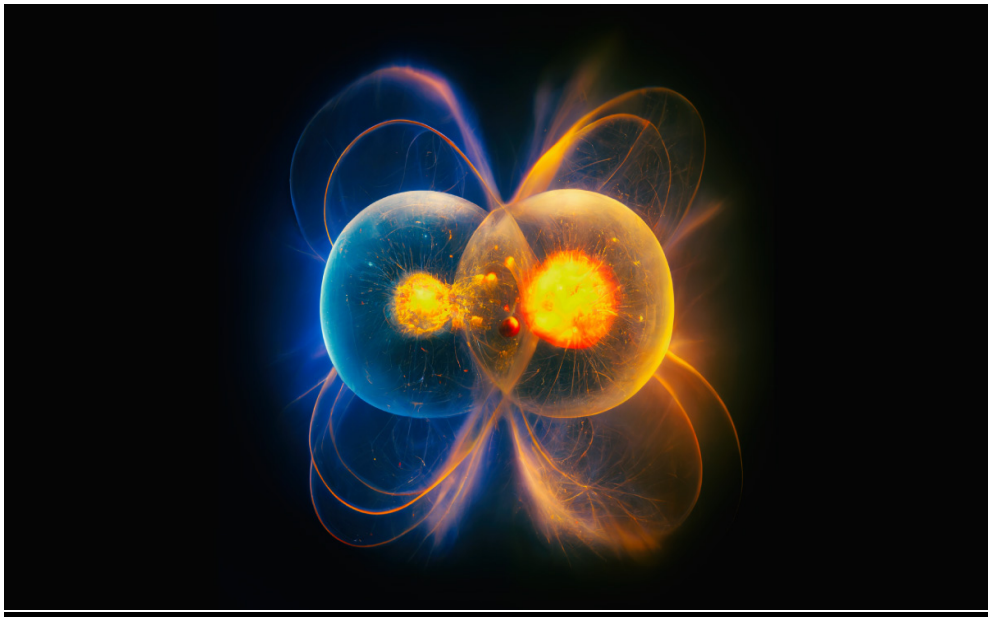
Thermodynamique : La théorie cinétique des gaz explique le mouvement des particules dans un gaz. Les particules de gaz se déplacent de manière aléatoire, se heurtent et échangent de l'énergie. La température d'un gaz est une mesure de l'énergie cinétique moyenne des particules, et la pression est le résultat des collisions des particules sur les parois du conteneur.

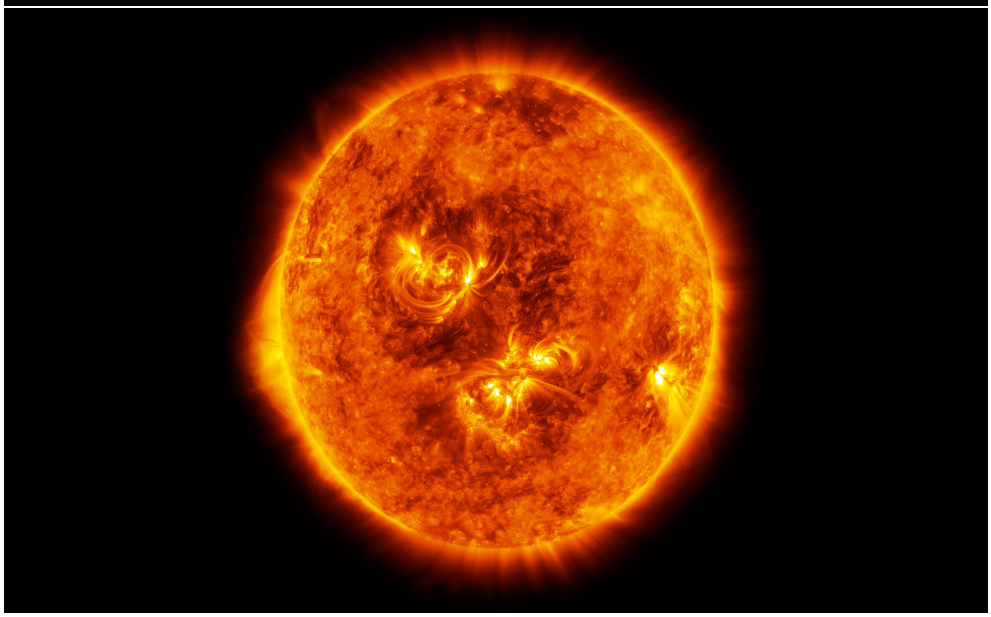
Le mouvement des particules peut également être influencé par des phénomènes comme la diffusion (la tendance des particules à se déplacer d'une région de concentration plus élevée vers une région de concentration plus faible), la convection (le déplacement de particules dans un fluide en raison de gradients de température ou de densité), ou encore l'effet de forces électromagnétiques et gravitationnelles.

La description précise du mouvement des particules dépend du niveau d'échelle étudié, des forces et des interactions spécifiques agissant sur les particules, ainsi que du cadre théorique utilisé pour modéliser le système. Chaque domaine de la physique fournit des outils et des modèles pour décrire et prédire le mouvement des particules dans des conditions spécifiques.

6 Fusion

6.1 Définition





La fusion nucléaire est un processus au cours duquel deux noyaux atomiques légers se combinent pour former un noyau plus lourd, libérant une quantité massive d'énergie. Contrairement à la fission nucléaire, qui implique la division d'un noyau lourd, la fusion fusionne des noyaux légers pour en former un plus grand.

Le processus de fusion nucléaire est celui qui alimente le soleil et les étoiles. Sur Terre, reproduire la fusion pour en tirer de l'énergie est un objectif majeur de la recherche en énergie.

Voici un aperçu du processus de fusion :

Conditions extrêmes : Pour réaliser la fusion sur Terre, il est nécessaire de recréer les conditions extrêmes présentes dans le cœur des étoiles. Cela implique des températures extrêmement élevées de plusieurs millions de degrés Celsius, ainsi que des pressions énormes.

Réactions de fusion : À ces températures extrêmes, les noyaux légers tels que ceux de l'hydrogène (ou ses isotopes, le deutérium et le tritium) sont forcés à se déplacer avec une énergie cinétique très élevée. Lorsqu'ils entrent en collision à grande vitesse, ils peuvent surmonter leur répulsion électrostatique et fusionner pour former un noyau plus lourd.

Libération d'énergie : Lorsque la fusion a lieu, une petite partie de la masse des noyaux légers est convertie en énergie, comme le prédit la célèbre équation d'Einstein, $E=mc^2$. Cette énergie est libérée sous forme de rayonnement électromagnétique et de particules à haute énergie.

La fusion nucléaire est considérée comme une source d'énergie potentiellement illimitée et relativement propre, car elle ne produit pas de déchets radioactifs à longue durée de vie comme ceux de la fission nucléaire. Cependant, reproduire ces conditions sur Terre pour réaliser une fusion contrôlée et soutenue demeure un défi technologique majeur. Des projets tels que les réacteurs à fusion expérimentaux, comme ITER, tentent de maîtriser la fusion pour une production d'énergie à grande échelle.

6.2 fusion nucléaire dans les étoiles

La « fusion nucléaire » aussi appelée « fusion thermonucléaire » est la réunion de deux noyaux atomiques légers pour former un noyau unique plus lourd et plus stable. Au cours de cette réaction de fusion, la masse du noyau produit est inférieure à la somme des masses des noyaux légers d'origine. Or, en vertu de la célèbre relation établie par Albert Einstein « E=mc2 », la différence de masse est convertie en énergie. On peut notamment observer ce phénomène de fusion au sein des étoiles dans lesquelles une énergie colossale est libérée. Le phénomène de fusion nucléaire se différencie donc de celui de la fission nucléaire dans lequel un atome lourd se scinde en deux atomes plus légers avec un dégagement d'énergie nettement inférieur.

Le processus de fusion nucléaire ne peut avoir lieu que dans des conditions de température et de pression particulières. A titre d'exemple, au cœur du Soleil, la pression est égale à 200 milliards de fois la pression atmosphérique terrestre et la température centrale atteint environ 15 millions de degrés. Dans ces conditions, les noyaux légers d'hydrogène (75 pourcent de la composition du Soleil) fusionnent en noyaux d'hélium (24 pourcent) approximativement deux fois plus lourds, créant ainsi la lumière et la chaleur que nous recevons. Selon les calculs,

620 millions de tonnes d'hydrogène y sont transformés en 615,7 millions de tonnes d'hélium chaque seconde.

6.3 fusion nucléaire dans la terre

De très grandes quantités d'énergie sont libérées par le processus de fusion nucléaire. Pouvoir reproduire ce phénomène sur Terre permettrait en théorie de satisfaire définitivement les besoins énergétiques de l'humanité. C'est précisément l'enjeu majeur de la recherche sur la fusion nucléaire « contrôlée ». Les combustibles nécessaires à la fusion sont deux isotopes de l'hydrogène : le deutérium, disponible en quantités pratiquement illimitées dans l'eau des mers, et le tritium que l'on produit à partir du lithium relativement abondant dans l'écorce terrestre.

La bombe thermonucléaire – couramment appelée bombe H – constitue aujourd'hui la seule application pratique de la fusion nucléaire. Celle-ci a été testée pour la première fois en 1952 aux Etats-Unis dans la foulée de la maitrise de la bombe A (à fission nucléaire). Les armes thermonucléaires ont joué un rôle clé dans l'équilibre dissuasif entre les deux blocs pendant la guerre froide.

Des efforts de recherche sont menés depuis plus de 50 ans pour recréer les conditions de la fusion nucléaire au sein d'un réacteur. Toutefois, la maîtrise d'un processus contrôlé de fusion n'est pas encore démontrée et les technologies et matériaux adaptés à ces températures et pressions extrêmes ne sont pas encore disponibles pour une utilisation industrielle. Recréer un processus de fusion nucléaire s'avère beaucoup plus complexe que d'exploiter la réaction de fission en chaîne.

6.4 Les enjeux de la domestication de la fusion nucléaire

6.4.1 Les avantages de ce procédé

Si le principe novateur des centrales à fusion nucléaire est validé scientifiquement et technologiquement il permettra de développer une nouvelle source abondante d'énergie complémentaire de la fission nucléaire.

Les avantages écologiques

La fusion génère peu de déchets radioactifs, en plus de courte durée de vie, et pas de gaz à effet de serre. De plus, elle écarte tout risque d'emballement de la réaction nucléaire et donc toute menace d'explosion. Contrairement au procédé de fission nucléaire, la moindre perturbation au sein d'un réacteur à fusion par confinement magnétique entrainerait un

refroidissement puis un arrêt spontané des réactions de fusion.

Les avantages économiques

La fusion nucléaire fait appel à des combustibles (deutérium, lithium) présents en grande quantités sur notre planète, de quoi alimenter les éventuels réacteurs à fusion pour de nombreux millénaires. Les risques de pénurie énergétique seraient donc écartés. Quelques grammes de combustible suffiraient pour déclencher et entretenir les réactions de fusion. Une centrale à fusion de 1 000 MWe aurait ainsi besoin de 125 kg de deutérium et de 3 tonnes de lithium (contre 2,7 millions de tonnes de charbon pour une centrale thermique de même puissance) pour fonctionner toute une année.

6.4.2 Les limites

Les limites technologiques

L'état actuel des connaissances scientifiques ne permet pas aujourd'hui d'extraire suffisamment d'énergie des réactions de fusion pour produire de l'électricité. De plus, on ne sait pas encore fabriquer de matériaux pouvant résister assez longtemps au rayonnement et au flux de neutrons libérés au cours de ces réactions. Les scientifiques estiment que les technologies nécessaires à la mise en œuvre de la fusion nucléaire contrôlée à des fins de production énergétique ne seront pas disponibles avant de nombreuses décennies.

Les limites financières

Le coût financier des installations de recherche se chiffre en milliards d'euros sur plusieurs décennies. Ce coût est donc très important pour des bénéfices potentiels éloignés dans le temps. L'investissement dans le programme ITER a par exemple été évalué initialement à 5 milliards d'euros(3). Selon les dernières estimations du programme en 2012, le coût prévisionnel de construction de la machine avoisinerait maintenant 13 milliards d'euros(4). Par ailleurs, les coûts de production de l'énergie de fusion restent une inconnue tant que le procédé n'aura pas atteint une maturité scientifique et technologique.

6.5 Fonctionnement technique ou scientifique

Fusionner des atomes sur Terre n'est pas simple. Il faut faire fondre deux atomes et provoquer la fusion de leurs noyaux alors que leurs charges électriques respectives ont tendance à les séparer. Les noyaux doivent pour cela se trouver dans un état d'agitation thermique intense. C'est le cas lorsqu'ils sont portés à des très hautes températures de l'ordre de la centaine de millions de degrés.

La fusion deutérium-tritium

Depuis une trentaine d'années, la quasi-totalité des recherches porte sur la fusion de deux isotopes de l'hydrogène : le deutérium et de tritium. Le premier existe à l'état naturel (présent dans l'eau de mer à hauteur de 33 g/m3) et le second peut être produit à l'intérieur d'un réacteur industriel à fusion, par interaction avec du lithium. Ce dernier est présent sur Terre à hauteur de 20 g/tonne dans la croûte terrestre et 0,18 g/m3 dans les océans(5). La température nécessaire à la fusion de ces deux isotopes est de l'ordre de 150 millions de degrés, soit dix fois la température du cœur du Soleil(6). Il se forme alors un plasma, quatrième état de la matière dans lequel les atomes sont totalement ionisés, c'est-à-dire que leurs noyaux et électrons ne sont plus liés. La réunion des noyaux atomiques légers pour former un noyau unique plus lourd et plus stable a alors lieu. Les neutrons dégagés lors de cette réaction irradient l'enceinte du réacteur qui emmagasine de l'énergie thermique.

Il existe deux voies de développement de réacteurs à fusion nucléaire :

Le réacteur à confinement inertiel : le mélange deutérium-tritium est enfermé dans des microbilles. Elles sont portées à très haute pression et température pendant un temps extrêmement court au moyen de lasers très puissants. La micro-explosion thermonucléaire obtenue produit une impulsion hyperpuissante de l'ordre du térawatt sur un laps de temps très court, environ 10 picosecondes(7).

Le Laser Mégajoule, dont l'installation doit être finalisée d'ici 2014 au centre CEA/-CESTA(8), près de Bordeaux, en est un exemple. Il a pour objectif de reproduire en laboratoire des conditions physiques analogues à celles créées lors du fonctionnement des armes nucléaires. Ce Laser est essentiellement destiné à un usage scientifique et à la simulation d'explosions d'armes atomiques, tout comme son semblable américain, le NIF (National Ignition Facility) situé en Californie. Le réacteur à confinement magnétique : les noyaux sont portés à plus de 100 millions de degrés Celsius dans des machines d'un volume important appelées tokamaks. Puisqu'aucun matériau ne peut résister à de telles températures, le plasma renfermant le mélange peu dense de deutérium et de tritium est confiné par un champ magnétique intense généré par des bobines situées autour de la chambre et par un fort courant électrique circulant dans le plasma. La fusion s'initie dès que la température, la densité et le temps d'isolation thermique du mélange atteignent les seuils critiques d'ignition.

Il existe plusieurs prototypes de tokamak dans le monde, dont l'installation Tore Supra à Cadarache. Le réacteur ITER, en construction sur ce site, appartient à la même famille.

Ces deux méthodes ont déjà permis d'obtenir de brèves réactions de fusion. Cependant, elles nécessitent pour le moment plus d'énergie qu'elles n'en créent. C'est pourquoi les axes principaux de la recherche dans les décennies à venir porteront sur l'allongement et

l'optimisation du processus de fusion.

7 Modèle Cucker-Smale

7.1 Définition

Le modèle Cucker-Smale est un modèle mathématique utilisé pour étudier le comportement collectif des systèmes dynamiques en mouvement. Il est nommé d'après les mathématiciens Felipe Cucker et Steve Smale, qui l'ont introduit dans les années 2000.

décrit l'interaction entre les particules en mouvement dans un espace. Les particules influencent les autres en ajustant leur vitesse en fonction de la position et de la vitesse relatives des particules voisines. Le modèle peut être utilisé pour étudier des phénomènes tels que la cohésion de groupe, la synchronisation ou l'auto-organisation.

Ce modèle est souvent utilisé pour modéliser le mouvement des bancs de poissons, des vols d'oiseaux, ou d'autres phénomènes de regroupement observés dans la nature. Il repose sur l'idée que les agents individuels, ou particules, ajustent leur vitesse et leur direction en réponse aux mouvements de leurs voisins les plus proches.

7.2 Les caractéristiques principales du modèle Cucker-Smale

Interactions à longue portée : Contrairement à certains modèles où les interactions sont limitées aux voisins les plus proches, dans le modèle Cucker-Smale, chaque particule ressent l'influence de toutes les autres particules dans le système.

Alignement : Les particules ont tendance à aligner leur vitesse et leur direction avec celles de leurs voisins.

Atteignabilité uniforme : Les particules visent à atteindre une configuration où la distance entre elles est uniforme, même si elle ne peut pas être atteinte parfaitement en pratique.

8 Programmes et Simulations

8.1 collision

particle_collision.py

C: > Users > lenovo > Downloads > particle_collision.py

```python
import pygame
import random
import math

pygame.init()

# Constants
WIDTH, HEIGHT = 800, 600
FPS = 60
PARTICLE_RADIUS = 10
NUM_PARTICLES = 40  # Change this to 40

# Colors
WHITE = (255, 255, 255)
RED = (255, 0, 0)
BLUE = (0, 0, 255)

# Particle class with wall collisions
class WallCollisionParticle:
    def __init__(self, x, y, mass, color):
        self.x = x
        self.y = y
        self.radius = PARTICLE_RADIUS
        self.mass = mass
        self.color = color
        self.velocity = [random.uniform(-2, 2), random.uniform(-2, 2)]

```

particle_collision.py

C: > Users > lenovo > Downloads > particle_collision.py

```
    def update(self):
        self.x += self.velocity[0]
        self.y += self.velocity[1]

        # Particle-wall collisions
        if self.x - self.radius < 0 or self.x + self.radius > WIDTH:
            self.velocity[0] *= -1

        if self.y - self.radius < 0 or self.y + self.radius > HEIGHT:
            self.velocity[1] *= -1

    def draw(self, screen):
        pygame.draw.circle(screen, self.color, (int(self.x), int(self.y)), self.radius)

# Create particles with wall collisions
particles = [
    WallCollisionParticle(random.uniform(PARTICLE_RADIUS, WIDTH - PARTICLE_RADIUS),
                          random.uniform(PARTICLE_RADIUS, HEIGHT - PARTICLE_RADIUS),
                          random.uniform(1, 5), random.choice([RED, BLUE])) for _ in range(NUM_PARTICLES)
]

# Create the screen
screen = pygame.display.set_mode((WIDTH, HEIGHT))
pygame.display.set_caption("Colorful Particle Collisions")

# Main loop
clock = pygame.time.Clock()
running = True
```

```
while running:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False

    # Update particles
    for particle in particles:
        particle.update()

    # Check for collisions and create new particles
    for i in range(len(particles) - 1):
        for j in range(i + 1, len(particles)):
            dx = particles[j].x - particles[i].x
            dy = particles[j].y - particles[i].y
            distance = math.sqrt(dx**2 + dy**2)

            if distance < particles[i].radius + particles[j].radius:
                # Collision occurred, create a new particle with a different color
                new_particle_color = (particles[i].color[0] + particles[j].color[0]) // 2, \
                                     (particles[i].color[1] + particles[j].color[1]) // 2, \
                                     (particles[i].color[2] + particles[j].color[2]) // 2
                new_particle = WallCollisionParticle(particles[i].x, particles[i].y,
                                                     particles[i].mass + particles[j].mass,
                                                     new_particle_color)
                particles.append(new_particle)

```

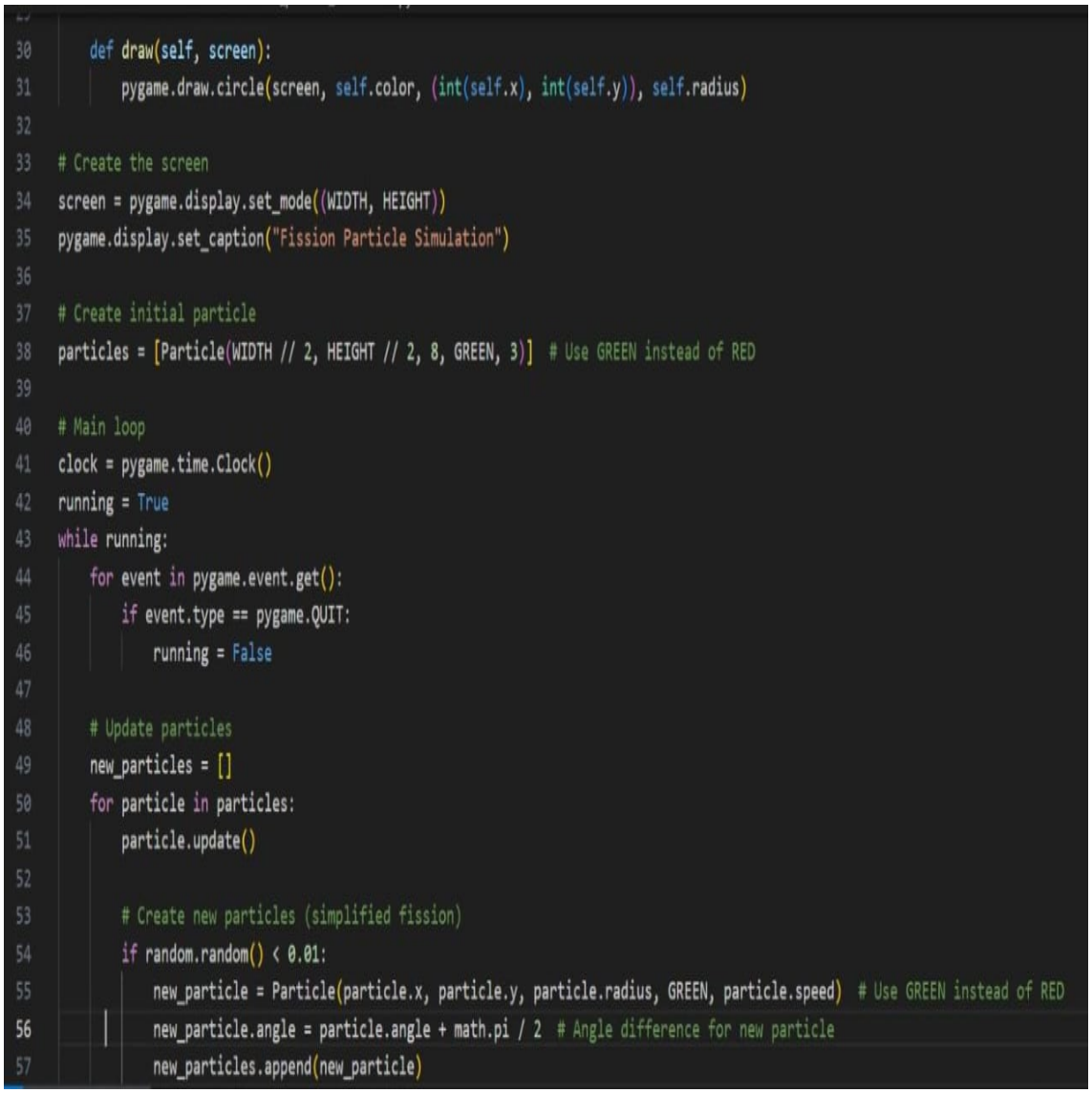
```
                    # Remove collided particles
                    particles.remove(particles[i])
                    particles.remove(particles[j - 1])
                    break

        # Draw background
        screen.fill(WHITE)

        # Draw particles
        for particle in particles:
            particle.draw(screen)

        # Update display
        pygame.display.flip()

        # Cap the frame rate
        clock.tick(FPS)

    # Quit Pygame
    pygame.quit()
```

8.2 fission

fission_particle_simulation.py

C: > Users > lenovo > Downloads > fission_particle_simulation.py

```
import pygame
import random
import math
import sys

pygame.init()

# Constants
WIDTH, HEIGHT = 800, 600
FPS = 60

# Colors
WHITE = (255, 255, 255)
GREEN = (0, 255, 0)  # Change color to green

# Particle class for fission simulation
class Particle:
    def __init__(self, x, y, radius, color, speed):
        self.x = x
        self.y = y
        self.radius = radius
        self.color = color
        self.angle = random.uniform(0, 2 * math.pi)
        self.speed = speed

    def update(self):
        self.x += self.speed * math.cos(self.angle)
        self.y += self.speed * math.sin(self.angle)
```



```
    def draw(self, screen):
        pygame.draw.circle(screen, self.color, (int(self.x), int(self.y)), self.radius)

# Create the screen
screen = pygame.display.set_mode((WIDTH, HEIGHT))
pygame.display.set_caption("Fission Particle Simulation")

# Create initial particle
particles = [Particle(WIDTH // 2, HEIGHT // 2, 8, GREEN, 3)]  # Use GREEN instead of RED

# Main loop
clock = pygame.time.Clock()
running = True
while running:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False

    # Update particles
    new_particles = []
    for particle in particles:
        particle.update()

        # Create new particles (simplified fission)
        if random.random() < 0.01:
            new_particle = Particle(particle.x, particle.y, particle.radius, GREEN, particle.speed)  # Use GREEN instead of RED
            new_particle.angle = particle.angle + math.pi / 2  # Angle difference for new particle
            new_particles.append(new_particle)
```

```
58
59          particles.extend(new_particles)
60
61          # Draw background
62          screen.fill(WHITE)
63
64          # Draw particles
65          for particle in particles:
66              particle.draw(screen)
67
68          # Update display
69          pygame.display.flip()
70
71          # Cap the frame rate
72          clock.tick(FPS)
73
74      pygame.quit()
75      sys.exit()
```

8.3 fusuion

fusion.py

: > Users > lenovo > Downloads > fusion.py

```
1   import pygame
2   import random
3   import math
4   import sys
5
6   pygame.init()
7
8   # Constants
9   WIDTH, HEIGHT = 800, 600
10  FPS = 60
11
12  # Colors
13  WHITE = (255, 255, 255)
14  BLUE = (0, 0, 255)
15  RED = (255, 0, 0)
16
17  # Particle class for fusion simulation
18  class Particle:
19      def __init__(self, x, y, radius, color, speed):
20          self.x = x
21          self.y = y
22          self.radius = radius
23          self.color = color
24          self.angle = random.uniform(0, 2 * math.pi)
25          self.speed = speed
26
27      def update(self):
28          self.x += self.speed * math.cos(self.angle)
```

```
            self.y += self.speed * math.sin(self.angle)

            # Reflect particles off the boundaries
            if self.x - self.radius < 0 or self.x + self.radius > WIDTH:
                self.angle = math.pi - self.angle
            if self.y - self.radius < 0 or self.y + self.radius > HEIGHT:
                self.angle = -self.angle

        def draw(self, screen):
            pygame.draw.circle(screen, self.color, (int(self.x), int(self.y)), self.radius)

    # Create the screen
    screen = pygame.display.set_mode((WIDTH, HEIGHT))
    pygame.display.set_caption("Fusion Particle Simulation")

    # Create initial particles within the cadre
    particles = [Particle(random.randint(50, WIDTH - 50), random.randint(50, HEIGHT - 50), 8, BLUE, 3),
                 Particle(random.randint(50, WIDTH - 50), random.randint(50, HEIGHT - 50), 8, BLUE, 3)]

    # Main loop
    clock = pygame.time.Clock()
    running = True
    fusion_triggered = False

    while running:
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                running = False
```

C: > Users > lenovo > Downloads > fusion.py

```
    # Update particles
    for particle in particles:
        particle.update()

    # Draw background
    screen.fill(WHITE)

    # Draw particles
    for particle in particles:
        particle.draw(screen)

    # Check for fusion when particles collide
    if not fusion_triggered and pygame.Rect.colliderect(pygame.Rect(particles[0].x, particles[0].y, particles[0].radius, particles[0].radius),
                                                        pygame.Rect(particles[1].x, particles[1].y, particles[1].radius, particles[1].radius)):
        # Fusion event (combine particles)
        particles[0].color = RED  # Change color to indicate fusion
        particles[0].radius += particles[1].radius
        particles.pop(1)
        fusion_triggered = True  # Set the flag to avoid repeated fusion events

    # Update display
    pygame.display.flip()

    # Cap the frame rate
    clock.tick(FPS)

pygame.quit()
sys.exit()
```

8.4 mouvement des particules

C: > Users > lenovo > Downloads > particle_simulation.py

```python
import pygame
import random
import math

# Initialize Pygame
pygame.init()

# Constants
WIDTH, HEIGHT = 800, 600
PARTICLE_RADIUS = 8  # Change particle radius
NUM_PARTICLES = 100
FPS = 60

# Colors
WHITE = (255, 255, 255)
BLUE = (0, 0, 255)  # Change color to blue

# Particle class
class Particle:
    def __init__(self, x, y):
        self.x = x
        self.y = y
        self.radius = PARTICLE_RADIUS
        self.color = BLUE  # Change color to blue
        self.velocity = [random.uniform(-2, 2), random.uniform(-2, 2)]

    def update(self):
        self.x += self.velocity[0]
        self.y += self.velocity[1]

        # Collisions with walls
        if self.x - self.radius < 0 or self.x + self.radius > WIDTH:
            self.velocity[0] *= -1
        if self.y - self.radius < 0 or self.y + self.radius > HEIGHT:
            self.velocity[1] *= -1

    def draw(self, screen):
        pygame.draw.circle(screen, self.color, (int(self.x), int(self.y)), self.radius)

# Initialize screen
screen = pygame.display.set_mode((WIDTH, HEIGHT))
pygame.display.set_caption("Particle Simulation")

# Create particles
particles = [Particle(random.randint(PARTICLE_RADIUS, WIDTH - PARTICLE_RADIUS),
                      random.randint(PARTICLE_RADIUS, HEIGHT - PARTICLE_RADIUS))
             for _ in range(NUM_PARTICLES)]

# Main loop
clock = pygame.time.Clock()

running = True
while running:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False
```

```
57
58      # Update particles
59      for particle in particles:
60          particle.update()
61
62      # Draw background
63      screen.fill(WHITE)  # Change background color to white
64
65      # Draw particles
66      for particle in particles:
67          particle.draw(screen)
68
69      # Update display
70      pygame.display.flip()
71
72      # Cap the frame rate
73      clock.tick(FPS)
74
75  # Quit Pygame
76  pygame.quit()
77
```

8.5 Modèle Cucker-Smale

8.5.1 programme 2D

```
C: > Users > lenovo > Downloads > pdf2.py
1   import numpy as np
2   import matplotlib.pyplot as plt
3   from matplotlib.animation import FuncAnimation
4
5   # Potentiel d'interaction :
6   def g(r):
7       return 1 / (r**2 + 1)
8
9   # Modèle de Cucker Smale en dimension 2 :
10  def Cucker_Smale2D(x, y, vx, vy, dt, lamda):
11      N = len(x)
12
13      # Xi-Xj
14      xdiff = np.tile(x, (N, 1)) - np.tile(x, (N, 1)).T
15      ydiff = np.tile(y, (N, 1)) - np.tile(y, (N, 1)).T
16      normeXdiff = np.sqrt(xdiff**2 + ydiff**2)
17
18      # Vj-Vi
19      vxdiff = np.tile(vx, (N, 1)) - np.tile(vx, (N, 1)).T
20      vydiff = np.tile(vy, (N, 1)) - np.tile(vy, (N, 1)).T
21
22      # r évalué en Xi-Xj H = g(normeXdiff)
23      H = g(normeXdiff)
24
25      # x(t + dt)
26      x += dt * vx
27      y += dt * vy
28
```

```
        # Les particules restent dans la boîte x = x % lim
        y = y % lim

        # v(x + dt)
        vx += (np.sum(vxdiff * H, axis=1) / N) * dt * lamda
        vy += (np.sum(vydiff * H, axis=1) / N) * dt * lamda

        return x, y, vx, vy

    # Nombre de particules
    N = 100

    # Limite de la boîte
    lim = 100

    # Positions initiales aléatoires
    x = np.random.rand(N) * lim
    y = np.random.rand(N) * lim

    # Vitesses initiales aléatoires
    vx = np.random.rand(N)
    vy = np.random.rand(N)

    # Pas de temps
    dt = 0.1
```

```
# Facteur d'influence
lamda = 0.5

# Création de la figure
fig, ax = plt.subplots()
ax.set_xlim(0, lim)
ax.set_ylim(0, lim)

# Initialisation des éléments de la trame
quiver = ax.quiver(x, y, vx, vy, scale=20, color='blue', width=0.007)
scatter = ax.scatter(x, y, color='red')
annotations = [ax.annotate(f'{i+1}', (x[i], y[i]), textcoords="offset points", xytext=(0, 5), ha='center') for i in range(N)]
vitesse_annotation = ax.annotate('', xy=(0.5, 1.02), xycoords='axes fraction', ha='center')

# Fonction d'animation
def update(frame):
    global x, y, vx, vy, quiver, scatter, annotations, vitesse_annotation
    x, y, vx, vy = Cucker_Smale2D(x, y, vx, vy, dt, lamda)
    quiver.remove()
    scatter.remove()
    for annotation in annotations:
        annotation.remove()
    quiver = ax.quiver(x, y, vx, vy, scale=20, color='blue', width=0.007)
    scatter = ax.scatter(x, y, color='red')
    annotations = [ax.annotate(f'{i+1}', (x[i], y[i]), textcoords="offset points", xytext=(0, 5), ha='center') for i in range(N)]
    vitesse_moyenne = np.sqrt(np.mean(vx)**2 + np.mean(vy)**2)
    vitesse_annotation.set_text(f'Vitesse moyenne: {vitesse_moyenne:.2f}')
```

```
82
83  # Animation
84  ani = FuncAnimation(fig, update, frames=100, interval=50)
85
86  # Affichage de la vidéo
87  plt.show()
88
```

8.5.2 programme 3D

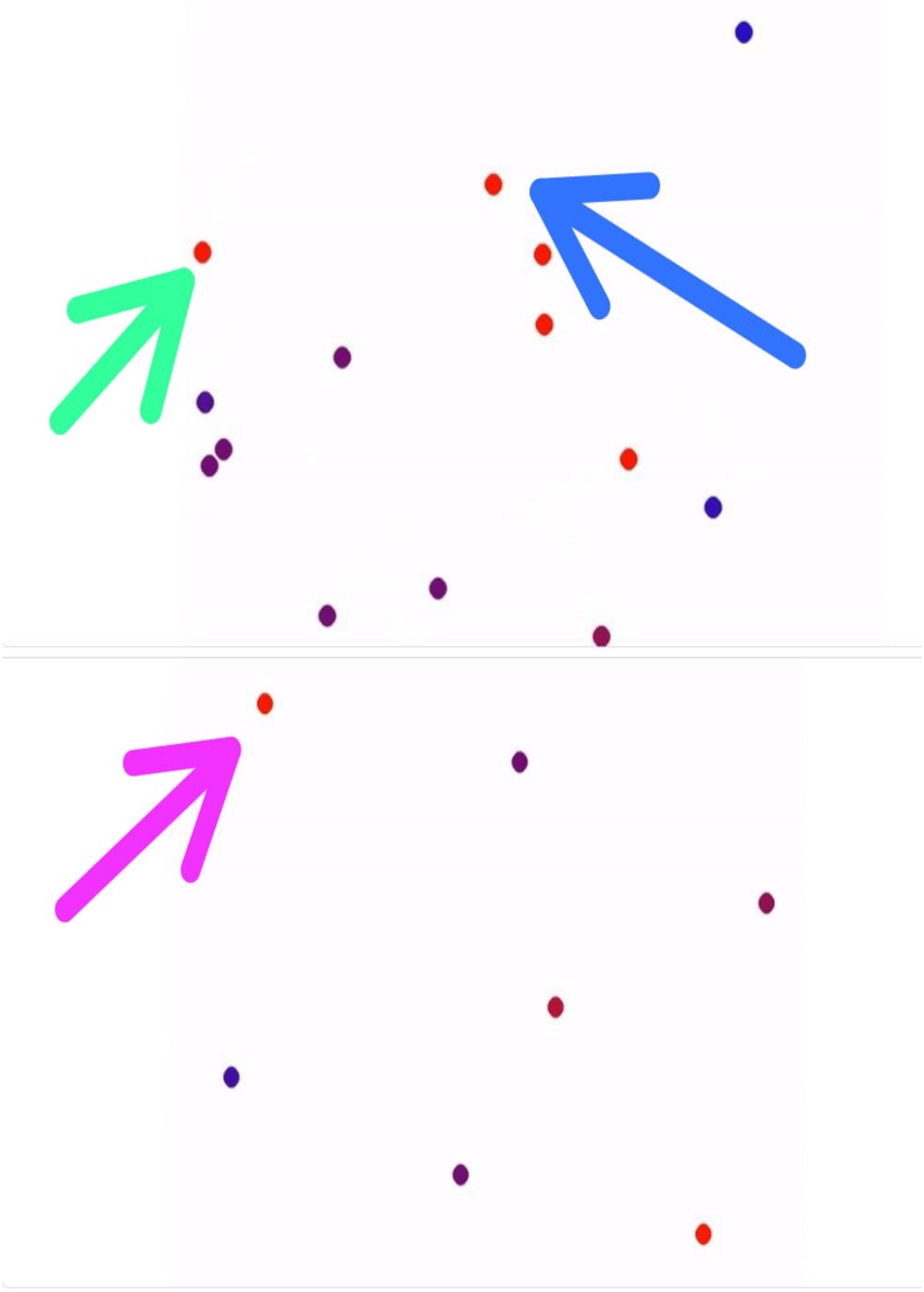
C: > Users > lenovo > Downloads > 3d (1).py

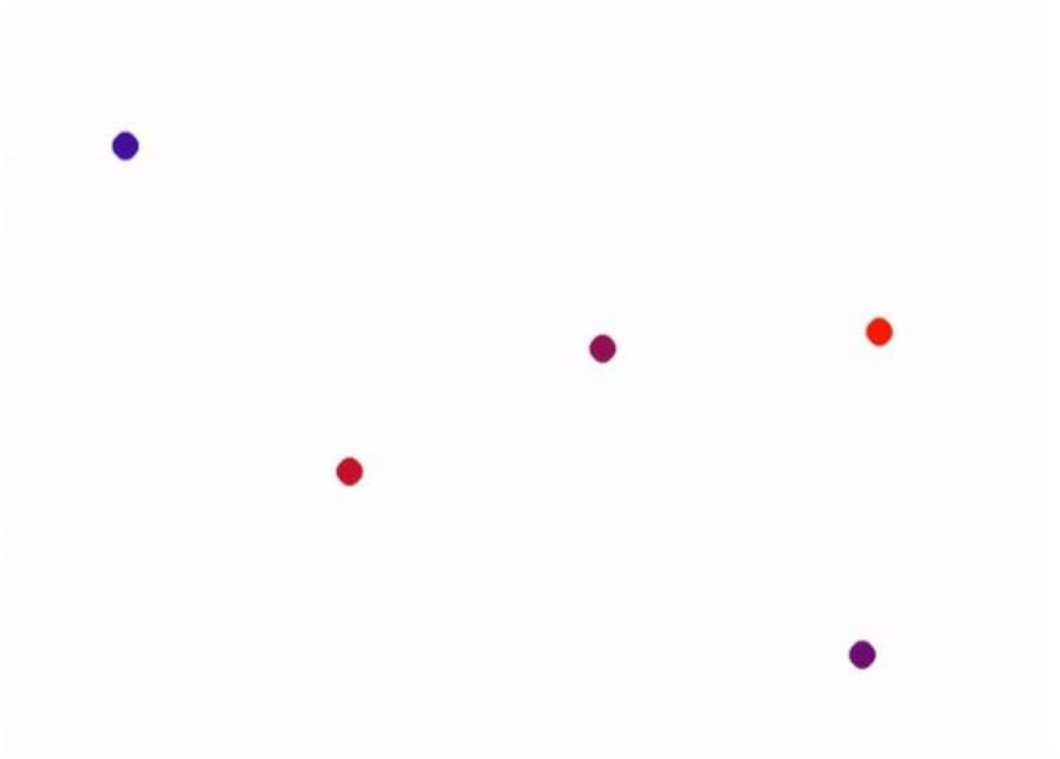
```python
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from matplotlib.animation import FuncAnimation

# Potentiel d'interaction :
def g(r):
    return 1 / (r ** 2 + 1)

# Modèle de Cucker Smale en dimension 3 :
def Cucker_Smale3D(x, y, z, vx, vy, vz, dt, lamda):
    N = len(x)

    # Xi-Xj
    xdiff = np.tile(x, (N, 1)) - np.tile(x, (N, 1)).T
    ydiff = np.tile(y, (N, 1)) - np.tile(y, (N, 1)).T
    zdiff = np.tile(z, (N, 1)) - np.tile(z, (N, 1)).T
    normeXdiff = np.sqrt(xdiff ** 2 + ydiff ** 2 + zdiff ** 2)

    # Vj-Vi
    vxdiff = np.tile(vx, (N, 1)) - np.tile(vx, (N, 1)).T
    vydiff = np.tile(vy, (N, 1)) - np.tile(vy, (N, 1)).T
    vzdiff = np.tile(vz, (N, 1)) - np.tile(vz, (N, 1)).T

    # r évalué en Xi-Xj H = g(normeXdiff)
    H = g(normeXdiff)

    # x(t + dt)
```

```
        x += dt * vx
        y += dt * vy
        z += dt * vz

        # Les particules restent dans la boîte x = x % lim
        y = y % lim
        z = z % lim

        # v(t + dt)
        vx += (np.sum(vxdiff * H, axis=1) / N) * dt * lamda
        vy += (np.sum(vydiff * H, axis=1) / N) * dt * lamda
        vz += (np.sum(vzdiff * H, axis=1) / N) * dt * lamda

        return x, y, z, vx, vy, vz

    # Nombre de particules
    N = 100
    lim = 100
    x = np.random.rand(N) * lim
    y = np.random.rand(N) * lim
    z = np.random.rand(N) * lim
    vx = np.random.rand(N)
    vy = np.random.rand(N)
    vz = np.random.rand(N)
    dt = 0.1
    lamda = 0.5

# Création de la figure 3D
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

# Ajouter des flèches dans les points qui indiquent la direction (avec une taille triplée)
quiver = ax.quiver(x, y, z, vx, vy, vz, color='blue', length=4.5, normalize=True, arrow_length_ratio=0.5)

# Ajout d'une annotation pour la vitesse moyenne
text_annotation = ax.text2D(0.5, 0.95, '', transform=ax.transAxes, ha='center')

# Fonction d'animation
def update(frame):
    global x, y, z, vx, vy, vz, quiver
    x, y, z, vx, vy, vz = Cucker_Smale3D(x, y, z, vx, vy, vz, dt, lamda)

    # Mettre à jour les données des flèches
    quiver.remove()
    quiver = ax.quiver(x, y, z, vx, vy, vz, color='blue', length=4.5, normalize=True, arrow_length_ratio=0.5)

    vitesse_moyenne = np.sqrt(np.mean(vx)**2 + np.mean(vy)**2 + np.mean(vz)**2)
    text_annotation.set_text(f'Vitesse moyenne: {vitesse_moyenne:.2f}')

# Animation
ani = FuncAnimation(fig, update, frames=100, interval=50)

# Affichage de l'animation
plt.show()
```

9 Résultats, Discussions

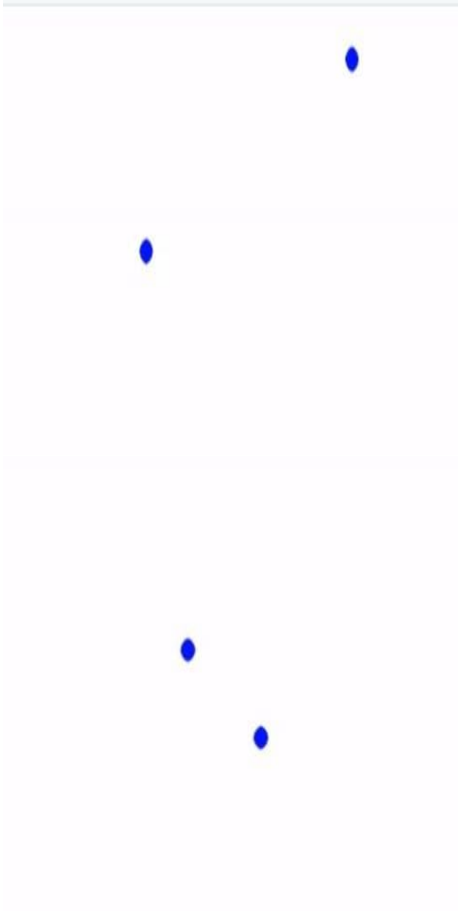
9.1 collision



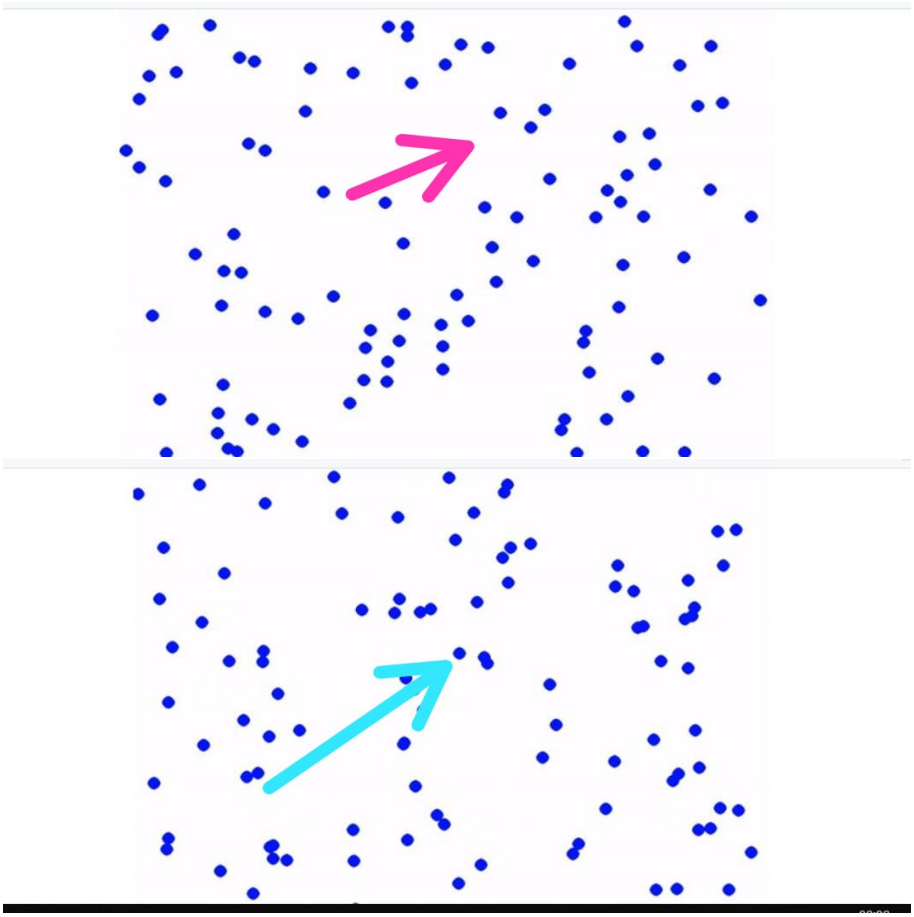


9.2 fission

9.3 fusion

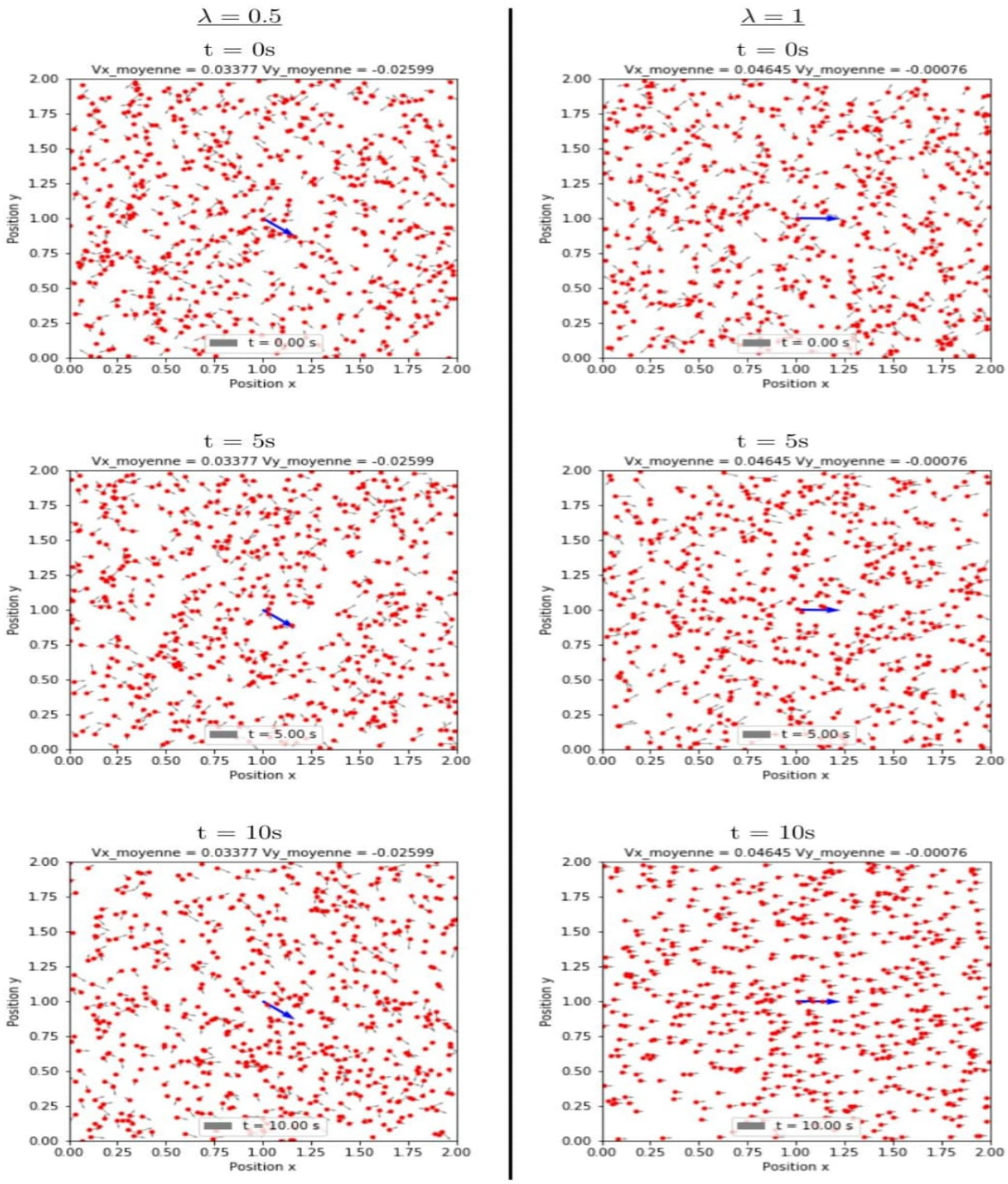


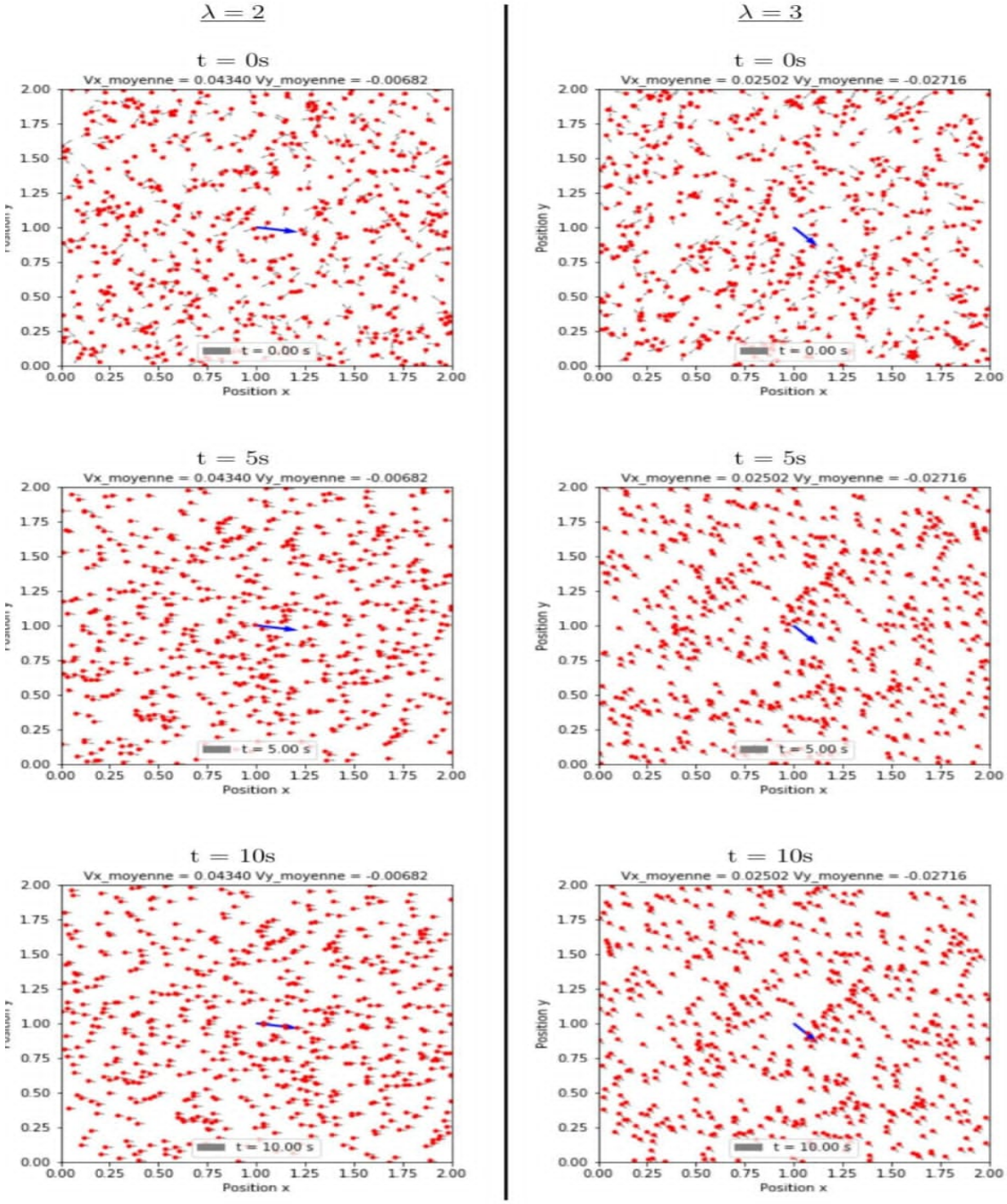
9.4 mouvement des particules



9.5 Modèle Cucker-Smale

En 2D :

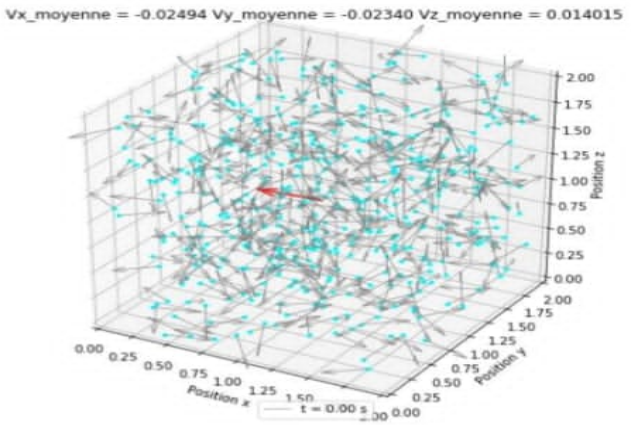



λ = 2
t = 0s
Vx_moyenne = 0.04340 Vy_moyenne = -0.00682
t = 0.00 s
t = 5s
Vx_moyenne = 0.04340 Vy_moyenne = -0.00682
t = 5.00 s
t = 10s
Vx_moyenne = 0.04340 Vy_moyenne = -0.00682
t = 10.00 s
Position x
λ = 3
t = 0s
Vx_moyenne = 0.02502 Vy_moyenne = -0.02716
t = 0.00 s
t = 5s
Vx_moyenne = 0.02502 Vy_moyenne = -0.02716
t = 5.00 s
t = 10s
Vx_moyenne = 0.02502 Vy_moyenne = -0.02716
t = 10.00 s
Position x
Position y

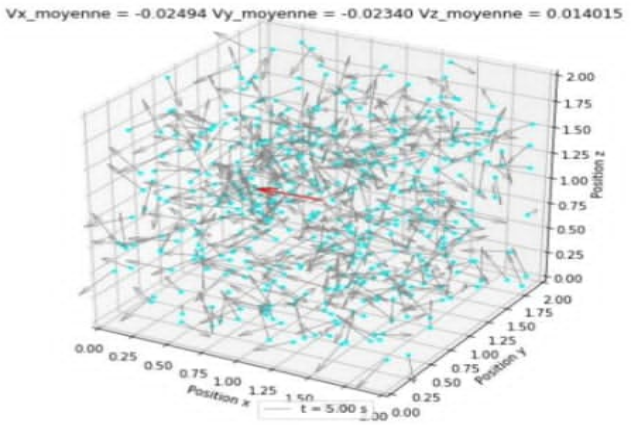
En 3D :

$\underline{\lambda = 1}$

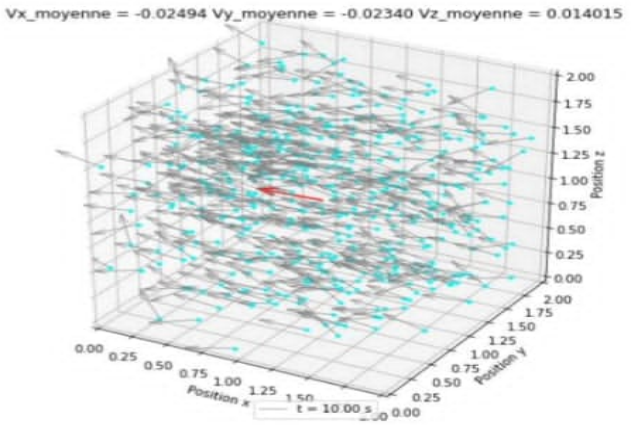
t = 0s



t = 5s

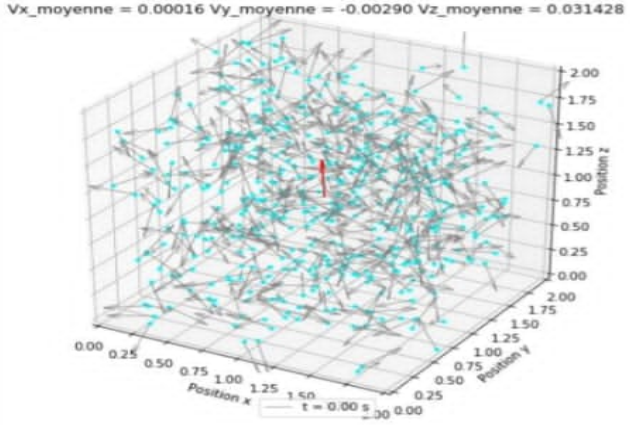


t = 10s

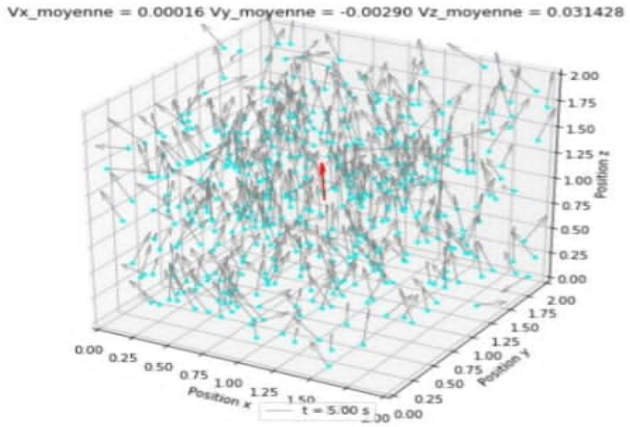


$\underline{\lambda = 2}$

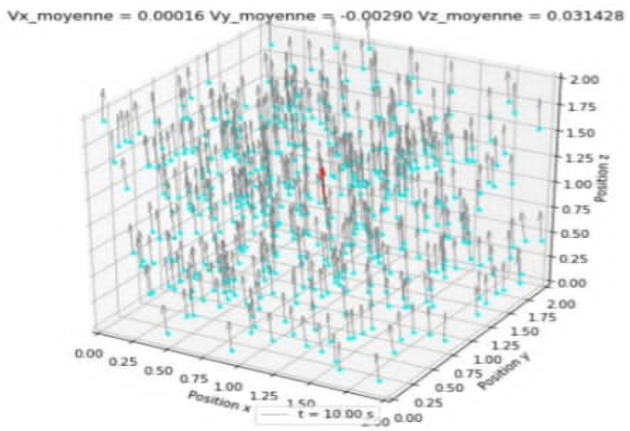
t = 0s



t = 5s



t = 10s



10 Conclusion

En conclusion, l'utilisation de Python est crucial dans le domaine des simulations nucléaires. À travers une exploration détaillée des différents aspects de la physique nucléaire, nous avons pu constater comment Python offre une plateforme flexible et puissante pour la modélisation et la simulation de phénomènes complexes.

L'intégration de Python dans la recherche nucléaire permet non seulement de comprendre les mécanismes fondamentaux à l'œuvre dans les collisions, les fissions, les fusions et le mouvement des particules, mais elle ouvre également la voie à de nouvelles découvertes et applications.

Malgré les défis rencontrés, tels que la précision des modèles et la validation des résultats expérimentaux, les avantages de l'utilisation de Python dans les simulations nucléaires sont indéniables. Ce langage de programmation offre une grande variété de bibliothèques et d'outils qui facilitent la mise en œuvre de modèles complexes et l'analyse des données.

Cependant, il est essentiel de reconnaître que les simulations ne remplacent pas les expériences réelles. Elles doivent être utilisées en complément des observations expérimentales pour valider et affiner nos modèles théoriques.

Enfin, ce rapport souligne l'importance de la collaboration entre les chercheurs, les ingénieurs et les scientifiques de divers domaines pour faire avancer la recherche en physique nucléaire. Ensemble, nous pouvons continuer à repousser les limites de notre compréhension de l'univers nucléaire et à développer des technologies innovantes qui auront un impact positif sur notre société et notre avenir.

En définitive, l'utilisation de Python dans les simulations nucléaires représente une avancée significative qui ouvre de nouvelles perspectives passionnantes pour la recherche et le développement dans ce domaine crucial.

11 bibliographie

https ://papyrus.bib.umontreal.ca/

https ://bib.umontreal.ca/guides/types-documents/revues-scientifiques-revues-populaires

https ://www.futura-sciences.com/

https ://images.cnrs.fr/

https ://github.com/UcenELma/PhysicsEngine