Développement d'un module Lagrangien de collisions de particules afin de valider les algorithmes de résolution Eulérienne de gaz sans pression

Olivier Thomine

UMR 6614 CORIA
COMPLEXE DE RECHERCHE INTERPROFESSIONNEL EN AEROTHERMOCHIMIE

cnrs

UFR UNIVERSITÉ DE ROUEN

Rapport de stage de Master 2

présentée à la

Faculté des Sciences & Techniques de l'Université de Rouen

Discipline : Sciences physiques

Spécialité : Mécanique des Fluides, Numérique et énergétique

par

Olivier THOMINE

Développement d'un module Lagrangien de collisions de particules afin de valider les algorithmes de résolution Eulérienne de gaz sans pression.

Encadrants :

J. Reveillon	Directeur de stage
B. Delhom	Co-encadrant
F. Laurent (EM^2C)	Co-encadrante

Table des matières

Résumé

Le but de ce stage a été de développer un algorithme Lagrangien de détection et de prise en charge des collisions de gouttes. Tout d'abord, son intérêt sera expliqué, ainsi que les raisons pour lesquelles un tel algorithme est nécessaire dans bon nombre d'applications.

Ensuite son fonctionnement sera décrit en détail, les étapes utilisées et les méthodes qui ont été choisies. Puis, nous décrirons les optimisations effectuées ainsi que leur gain en complexité. Pour finir, une analyse des résultats obtenus sera effectuée, une fois ce module intégré dans une DNS[1] développé au sein du CORIA [2]. Trois configurations de simulation ont été mises en place. Les collisions jouent un rôle crucial dans celles-ci. Ces configurations consistent en deux jets croisés diphasiques (en domaine Cartésien). La première simulation est extrêmement dense en goutte, et les collisions ne sont pas prises en compte. La deuxième est peu dense en gouttes, mais les collisions sont effectives. Enfin, la troisième est extrêmement dense et les collisions sont prises en compte. Cela nous permettra d'étudier l'influence des collisions sur le résultat de la simulation.

1 Introduction

L'actualité climatique de notre planète met en valeur la nécessité d'améliorer les actuels vecteurs de dégradation de notre environnement. Parmi ceux-ci figurent principalement les moteurs à combustion interne, les réacteurs et autres machines utilisant la combustion pour fonctionner.

Leur nécessité est avérée, aussi la diminution des rejets de gaz à effet de serre, ainsi que la diminution de leur consommation et l'augmentation de leur rendement est aujourd'hui nécessaire. Tous ces paramètres dépendent d'un paramètre crucial dans la mise en oeuvre de tels moteurs : les injecteurs. Ce sont des paramètres importants dans la manière dont va s'atomiser et se vaporiser le jet du carburant. Cet atomisation va définir entre autre la géométrie de la flamme, donc son aptitude à brûler du carburant.

Afin d'améliorer cette injection, on a recours à la simulation. En effet, celle-ci est peu chère à mettre en oeuvre et permet la mise en place de configurations inexistantes actuellement. De plus, les conditions particulières des injections se font souvent dans des conditions critiques (jusqu'à une centaine de bar à des températures avoisinnant les 1000 K [2] dans les moteurs de voiture par exemple, sans parler des chambres de combustion type Vulcain).

L'inconvénient de la modélisation de tels système réside en leur complexité, ainsi qu'au très grand nombre de gouttes qui existent dans les chambres de combustions. Pour remédier à cela, une étude statistique Eulérienne a été développée. Afin de perfectionner la modélisation des injections, un module de prise en charge des collisions de gouttes Eulérien est actuellement en cours de développement au Laboratoire de

[1] Direct Numerical Simulation

[2] COmplexe de Recherche Interprofessionnel en Aérothermochimie, UMR-CNRS 6614

l'École Centrale Paris [3]. Le but de ce stage va être de développer un algorithme Lagrangien et déterministe de détection et de prise en charge de collisions de gouttes en vue de la validation de ce modèle.

Dans une première partie sera exposé les différentes méthodes actuellement utilisées ainsi que leurs domaines d'application. Ensuite la méthode utilisée ici, ainsi que les modèles appliqués seront décrits en détails. Puis, une étude sera faite sur la complexité de cet algorithme ainsi que des modifications apportés à celui-ci. Dans une quatrième partie seront exposées les différentes optimisations qui ont été développées ainsi que leurs intérêts et leur efficacité. Enfin le code sera commenté et plusieurs résultats seront étudiés.

2 Contexte

Le but d'un algorithme de collision est de repérer, parmi N gouttes – souvent très grand – , l'ensemble des couples de gouttes qui vont collisioner (cas déterministe) ou qui sont les plus susceptibles de collisioner (probabiliste). Il s'agira ensuite de prendre en charge ces collisions en appliquant des modèles simples (collision élastique) ou plus élaborés (coalescence, break-up, etc.).

Une solution serait de déterminer parmi chaque couple de particules ceux qui pourraient donner lieu à une éventuelle collision. En effet, il faudrait, parmi N particules, sélectionner tous les couples possibles, qui s'élèvent au nombre de $\frac{N(N-1)}{2} = \frac{N^2-N}{2}$. Pour 10 000 particules par exemple, cela reviendrait à effectuer environ 100 000 000 de tests, ce qui est parfaitement inenvisageable. Cet algorithme aurait une complexité de N^2, ce qui, au vu du nombre de gouttes présentes, est absolument impraticable.

Il s'agit donc de trouver des algorithmes permettant le traitement rapide d'un très grand nombre de particules. Ici, dans le cas déterministe que nous essayerons de développer, il faut réussir à sélectionner les particules susceptibles de collisioner avec chaque goutte, et d'ignorer tout simplement toutes celles qui ne le peuvent pas. Chaque particule ignorée sera un couple de moins à tester.

Ces algorithmes se décomposent en trois parties : détection des particules susceptibles de collisioner, détection des collisions, et enfin traitement de la collision.

En fonction du problème et de ses caractéristiques, plusieurs méthodes ont été développées pour chaque partie constituant cet algorithme. Quelques unes de ces méthodes vont être énumérées ici.

2.1 État de l'art

2.1.1 Détection des particules susceptibles de collisioner

La détection des particules susceptibles de "collisionner" relève souvent de la détection des plus proches voisins. Pour cela, on décompose l'espace en sous-domaines,

[3] Equipe : Marc Massot

dans lesquels on répartit les particules. Il suffit ensuite, pour chaque particule dont on veut trouver les partenaires potentiels de collision, chercher l'ensemble des particules se trouvant dans le même sous-domaine, ainsi que dans les sous-domaines voisins. Cela constitue la méthode Lagrangienne.

Une autre méthode, Eulérienne celle-ci, consiste à ne plus considérer les particules individuellement, mais statistiquement. Le nombre et les propriétés de ces particules sont caractérisées par une PDF[4]. La connaissance de cette PDF permet de déterminer le nombre de collision moyen devant s'y produire. Pour ce faire, la modélisation d'un opérateur de collision est nécessaire. En effet, une équation d'évolution de cette PDF a été créé, contenant un opérateur de collision [3, 4, 5, 6, 7], dont la modélisation la plus réaliste possible est souhaitée.

Cet methodologie est de plus en plus utilisé car il permet la gestion d'un nombre élevé de particules, contrairement à la méthode Lagrangienne.

Néanmoins, pour les simulations DNS de référence, nécessitant une grande précision dans la prédiction des collisions, on préférera un algorithme déterministe (Lagrangien) à modèle probabiliste (Eulérien).

2.1.2 Détection des collisions

Une fois que les particules susceptibles de "collisionner" sont détectées, trois méthodes sont envisageables.

La première consiste à détecter exactement les collisions dans les équations de trajectoires. C'est une méthode déterministe car elle rend compte de toutes les collisions, et n'en ignore aucune. Elle est utilisée lorsque de bonnes prédictions sont nécessaires. Par exemple, en dynamique moléculaire où le nombre de collisions de particules est important pour rendre compte de la réalité [8]. De même en astrophysique, par exemple pour simuler les disques d'accrétion autour d'astres, de galaxies, etc [9].

D'autres simulations déterministes l'utilisent [8, 10, 11]. L'inconvénient d'une telle méthode est qu'il faut détecter l'ensemble des particules susceptibles de collisionner, et de tester chaque couple de particules ainsi produit pour savoir si elles vont collisioner. La complexité d'une telle méthode est en N^2 (N : nombre de particules), elle est donc inenvisageable telle quelle si le nombre de particule devient très grand.

La deuxième méthode consiste à relever les particules dont la probabilité de collision est la plus élevée, d'en tirer cette proportion aléatoirement, puis de les faire arbitrairement collisioner [12, 13, 14, 15]. C'est une méthode probabiliste, qui globalement rendra compte de la réalité, mais qui pour certaines applications reste trop approximative.

La dernière est une application arbitraire de collisions. On détermine le nombre de collisions moyennes qu'il devrait se produire en fonction des PDF, puis il s'agit

[4]Probability Density Function

de rajouter un terme dans l'équation d'évolution de cette PDF afin de prendre en compte ces collisions. Tout l'art réside dans la modélisation de ce terme [6, 7]. Actuellement, une nouvelle modélisation est en cours à l'école Centrale Paris, réalisée par l'équipe de M. Marc Massot. L'algorithme de collision développé ici a pour but de valider ce nouveau modèle.

Nous utiliserons dans ce stage une détection déterministe, dans la mesure où notre but est de valider un modèle actuellement en développement [7], un résultat aussi précis que possible est nécessaire.

2.1.3 Prise en charge des collisions

Une fois que les couples (ou triplet [8]) de particules qui vont collisioner ont été déterminés, alors une fois de plus, plusieurs techniques existent.

- On considère une collision purement élastique, au quel cas les lois classiques de la mécanique suffisent à rendre compte du phénomène, en considérant que la totalité de l'énergie mécanique des particules est cinétique [10, 11, 8].

- Contrairement au traitement décrit ci-dessus, on considère que l'énergie mécanique des particules n'est pas que cinétique, mais peut aussi être de rotation. Pour cela, on utilise le moment cinétique d'une sphère dure, puis nous déterminons deux équations supplémentaires afin de rendre compte de la part d'énergie qui se transformera en énergie cinétique de celle qui partira en énergie de rotation [16]. Ce modèle a été validé par M. Sommerfeld [17] dans le cas d'écoulements THI [5].

- Encore plus complexe que précédemment, on considère aussi l'énergie de surface des particules. Cela revient à dire que l'on considère les particules comme pouvant coallescer, se rompre, donner naissance à des gouttes satellites etc. La résolution des équations de Navier-Stokes seraient bien trop coûteuse dans ce cas [18, 19]. Une généralisation a été faite : cette prise en charge dépend du paramètre d'impact de ces particules, ainsi que de leur nombre de Weber [13, 20]. Des études approfondies ont été menées afin d'avoir un schéma le plus précis possible concernant la conséquence de ces 2 paramètres d'impact sur la conséquence de la collision [21].
  C'est de loin la prise en charge des collisions la plus réaliste, mais cependant relativement difficile à mettre en oeuvre. Ce genre de phénomène n'a pas été pris en compte dans l'algorithme que nous avons développé.

3 Évolution temporelle des particules

Le code utilisé pour la simulation de dispersion de gouttelettes dans une phase porteuse gazeuse est un code DNS, nommé "Aphodele", développé au sein du CORIA. Afin de valider un opérateur collision développé à l'École Centrale de Paris,

[5] Turbulents Homogènes Isotropes

un module de détection et de gestion de collisions devra être ajouté dans ce code.

Pour déterminer la manière dont la goutte va évoluer dans le liquide, nous utilisons la force qu'exerce un fluide de vitesse $\overrightarrow{U}(\overrightarrow{x},t)$ sur une goutte se situant à $\overrightarrow{x}$, de vitesse $\overrightarrow{U_g}(t)$ et de section S. On a :

$$\overrightarrow{F}(t,\overrightarrow{x},\overrightarrow{U_g},S) = \frac{\overrightarrow{U}(\overrightarrow{x},t) - \overrightarrow{U_g}}{\tau_p(S)}$$

avec

$$\tau_p(S) = \frac{\rho_l S}{18\pi\mu_g}$$

le temps caractéristique de la particule. De cette force on déduit l'accélération $\overrightarrow{\gamma}(t,\overrightarrow{x},\overrightarrow{U_g},S) = \frac{\overrightarrow{F}(t,\overrightarrow{x},\overrightarrow{U_g},S)}{M_g}$. Afin de déterminer l'évolution temporelle de la position et la vitesse de la particule, il suffit d'intégrer par rapport au temps :

$$\overrightarrow{V}(t+\Delta t) = \overrightarrow{V}(t) + \int_t^{t+\Delta t} \overrightarrow{\gamma}(T,\overrightarrow{x},\overrightarrow{U_g},S)dT$$

Plusieurs hypothèses sont faites afin de résoudre cette équation numériquement. Ces hypothèses sont les suivantes :

- la goutte est sphérique (τ_p est ainsi définie pour un corps sphérique),
- $\overrightarrow{U_g}$ reste constant entre t et $t+\Delta t$ (cela revient à dire que $\tau_{turbulence} >> \Delta t$),
- S reste constant ($\tau_{evap} >> \Delta t$) pendant cette période,
- la force reste constante ($\tau_p >> \Delta t$) pendant cette période.

Ces hypothèses étant faites, on peut dire que $\overrightarrow{F}$ et M_g restent constant. On obtient par conséquent :

$$\overrightarrow{V}(t+\Delta t) = \overrightarrow{V}(t) + \frac{\overrightarrow{F}(T,\overrightarrow{x},\overrightarrow{U_g},S)}{M_g}\int_t^{t+\Delta t} dT = \overrightarrow{V}(t) + \Delta t\frac{\overrightarrow{F}(T,\overrightarrow{x},\overrightarrow{U_g},S)}{M_g}$$

Une fois $\overrightarrow{V}(t+\Delta t)$ déterminé, la déduction de $\overrightarrow{X}(t+\Delta t)$ est immédiate : $\overrightarrow{X}(t+\Delta t) = \overrightarrow{X}(t) + \Delta t\overrightarrow{V}(t)$.

Maintenant que la vitesse et la position de la particule au temps $t+\Delta t$ sont connus, ce résultat peut être utilisé dans chaque pas d'avancement en temps du Runge Kutta. Le Runge Kutta utilisé ici est d'ordre 3 (RK3).

Il va donc s'agir de détecter les collisions et d'en tenir compte. Pour ce faire trois étapes successives seront effectuées par le code :

- recherche des collisions à partir de la position et de la vitesse de toutes les particules,
- avancement en temps par le code (RK3),
- modification de la position et vitesse des particules qui ont collisionnées ensuite.

La détection et la prise en charge de ces collisions est détaillée dans le paragraphe suivant.

4 Description de l'algorithme de collision

4.1 Détection des plus proches voisins

Comme il a été expliqué au paragraphe précédent, la détection des collisions déterministe nécessite à un moment ou à un autre un test de chaque couple de particules sur un ensemble pré-sélectionné de particules. Cette phase est la plus longue (elle entraîne une complexité de l'ordre N^2 par rapport au nombre de particules pré-selectionné). Il s'agit donc de restreindre au maximum les particules pouvant entrer en collision. Pour ce faire, une méthode simple a été développée : il s'agit de restreindre la recherche aux plus proches voisins potentiellement collisionnables.

Pour un pas de temps Δt, il est évident que dans le pire des cas, une particule se trouvant à une distance $D_{max} = 2R_{max} + 2V_{max}\Delta t = 2(R_{max} + V_{max}\Delta t)$ pourra collisioner. On prend ici $V_{max} = \max_i ||\vec{V_i}||$.

Le cas idéal serai de n'avoir à explorer qu'une surface correspondant à la surface parcourue par la particule pendant le pas de temps. C'est la plus petite surface qu'il est envisageable d'explorer. En effet, il suffit que 2 de ces surfaces se croisent pour savoir que potentiellement une particule peut la croiser. Cependant, détecter parmi N équations de surfaces comme celles-ci lesquelles ont une intersection non nulle est difficile. Cette surface est décrite par la figure 1.

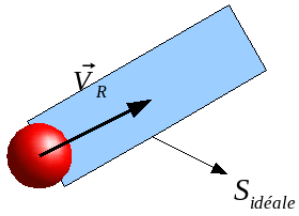


FIG. 1: Surface d'exploration idéale

Dans la bibliographie, la technique la plus majoritairement utilisée est de classer les particules dans une matrice, dont la largeur de chaque case correspondrai à D_{max}. Rechercher les partenaires potentiels de la collision revient donc à prendre l'ensemble des particules situées dans la même case et aux cases aux alentours de la particule considérée. Par la suite, nous appelerons la particule considérée 'R', et une particule candidate à la collision 'S'. La surface classiquement utilisée est montrée par la figure 2. La particule dont on cherche les plus proches voisins est en rouge, l'ensemble des candidats potentiels à la collision sont en jaune. On constate que cette surface est environ de $S_{classique} = 5D_{max}^2$. Nous verrons que nous pouvons faire beaucoup mieux par la suite.

Nous nous proposons ici de parcourir une surface de $2D_{max}^2$, mais cela nécessite un classement en abscisse des particules. Nous verrons par la suite pourquoi. Cette surface est symbolisée par la figure 3.

Dans tout le déroulement de l'algorithme, on prendra D_{max} constant. Afin de trouver toutes les particules se trouvant dans le domaine d'influence de la particule 'i' (en vert), nous allons classer les particules selon leur abscisse. Pour ce faire, un

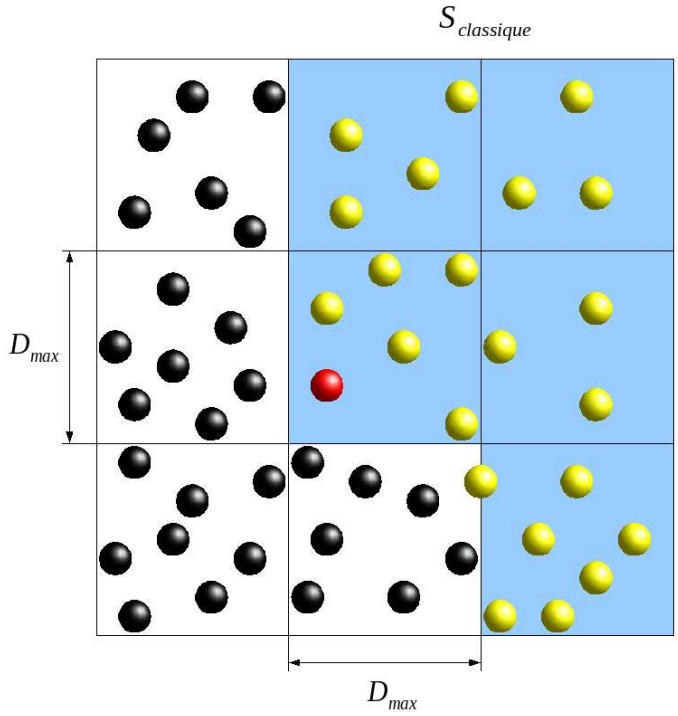


FIG. 2: Surface d'exploration classiquement explorée

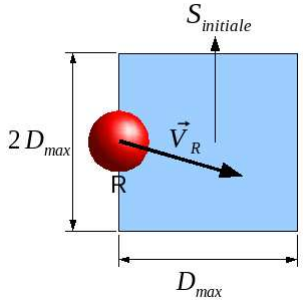


FIG. 3: Surface d'exploration initialement utilisée

système de tri rapide a été choisi. Celui-ci a une complexité allant de N à $N\log(N)$, ce qui pour notre cas est amplement suffisant. Une fois les particules triées dans une liste, il s'agira pour chaque particule 'i' de détecter, vers la droite, les particules se trouvant au plus à une distance D_{max} de 'i'. Pour chaque particule satisfaisant ce critère, on regarde sur (Oy) si ces particules sont à une distance inférieure à D_{max}.

La détermination des voisins les plus proches se décompose donc en deux parties :

- Premier tri
  On repère parmi l'ensemble des vitesses $\vec{V}$ des particules, la particule ayant la plus grande vitesse sur (Ox). On repère de même le plus grand rayon R_{max}. Dans le pire des cas, la particule INDEX(i) et la particule ayant cette vitesse maximale vont l'une contre l'autre à cette même vitesse. La distance maximale sur (Ox) d'une particule pouvant collisioner avec l'autre est D_{max}.

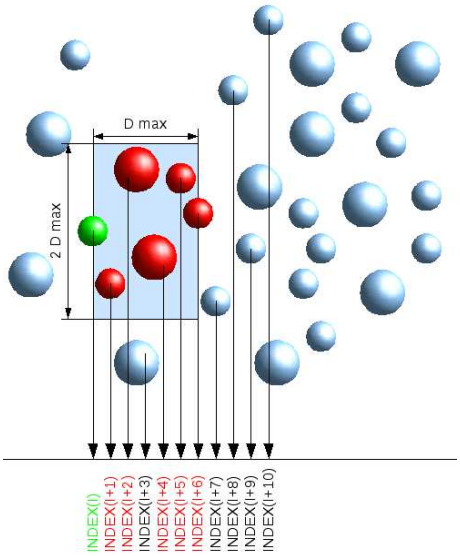


FIG. 4: Détection des plus proches voisins

– Second tri
Pour chaque particule INDEX(i), on regarde à sa droite l'ensemble des particules se trouvant à une abscisse au plus égale à D_{max} de la particule INDEX(i). Si une particule INDEX(i) venant de la gauche peut collisioner avec la particule INDEX(j), alors la particule INDEX(i) aura déjà été considérée, vue de la INDEX(j), comme candidate à la collision.
Pour détecter toutes les particules se trouvant au plus à D_{max} de la particule INDEX(j), un traitement préliminaire a été effectué afin d'obtenir une liste comportant l'index des particules, dans l'ordre de leur abscisse. Pour chaque particule INDEX(i), on parcourt donc la liste INDEX dans l'ordre croissant (donc vers les abscisses des particules supérieures à celle de la particule i). Cette recherche est matérialisé par la figure 4. On prend $INDEX(j) > INDEX(i)$, et on balaye INDEX(j) jusqu'à ce que $X_{INDEX(j)} - X_{INDEX(i)} > D_{max}$. Pour chaque particule se trouvant dans ce cas, on considère la particule candidate si $|Y_{INDEX(j)} - Y_{INDEX(i)}| < D_{max}$. Nous obtenons ainsi la liste des particules susceptibles de collisioner avec la particule INDEX(i). Cette liste est nommée CollP dans le code.

4.2 Détection des collisions par les trajectoires

Maintenant que nous avons l'ensemble des particules 'INDEX(j)' susceptibles de collisioner avec 'INDEX(i)', il s'agit de tester tous les couples INDEX(i),INDEX(j) afin de détecter une éventuelle collision.

On notera R=INDEX(i) et S=INDEX(j) pour simplifier les notations. Soit $L_{R,S}(t)$ la distance entre les centres des particules R et S. La distance entre deux particules R et S en fonction du temps est représenté par la figure 5. Il faut que $L(t) = R_R + R_S$.

On a :

$$L_{R,S}(t) = ||\overrightarrow{X_R(t)} - \overrightarrow{X_S(t)}|| = \sqrt{(X_R(t) - X_S(t))^2 + (Y_R(t) - Y_S(t))^2}$$

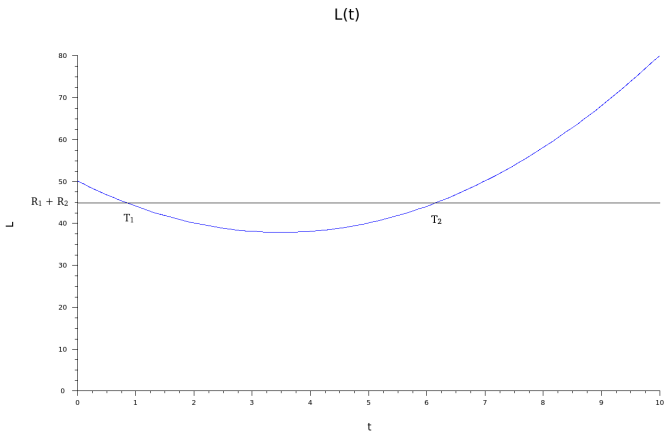


FIG. 5: Distance entre 2 particules en fonction du temps

avec

$$\overrightarrow{X}_k(t) = \overrightarrow{X}_{k,0} + \overrightarrow{V}_k t$$

Nous obtenons :

$$L(t)^2 = (X_R + V_{R,x}t - X_S - V_{S,x}t)^2 + (Y_R + V_{R,y}t - Y_S - V_{S,y}t)^2$$

Cela revient à changer de référentiel. On se met dans le référentiel de la particule S, se déplaçant à la vitesse constante $\overrightarrow{V}_S$.
On note ici :

$$\begin{cases} \Delta X = X_R - X_S \\ \Delta Y = Y_R - Y_S \\ \Delta V_X = V_{R,x} - V_{S,x} \\ \Delta V_Y = V_{R,y} - V_{S,y} \\ \Delta R = R_R + R_S \end{cases}$$

Nous avons :

$$L(t)^2 = (\Delta X^2 + \Delta Y^2) + 2t(\Delta V_X \Delta X + \Delta V_Y \Delta Y) + t^2(\Delta V_X^2 + \Delta V_y^2) \qquad (1)$$

Soient R_R et R_S le rayon respectivement de R et S.
Il y a collision s'il existe τ, comprit entre 0 et Δt tel que $L(\tau) = R_R + R_S$

$$(\Delta X^2 + \Delta Y^2) + 2t(\Delta V_X \Delta X + \Delta V_Y \Delta Y) + t^2(\Delta V_X^2 + \Delta V_y^2) - \Delta R^2 = 0 \qquad (2)$$

On notera : $\alpha = \Delta V_x^2 + \Delta V_y^2$, toujours positif
$\beta = 2(\Delta V_x \Delta X + \Delta V_y \Delta Y)$, signe quelconque
$\gamma = \Delta X^2 + \Delta Y^2 - \Delta R^2$, toujours positif car la distance des 2 particules est toujours supérieure à la somme de leur rayons
Finalement, nous obtenons $\alpha t^2 + \beta t + \gamma = 0$. Le discriminant est $\Delta = \beta^2 - 4\alpha\gamma$.$\Delta$ est positif si $\beta^2 > 4\alpha\gamma$. Sinon il n'y a pas collision.
Si Δ est positif, alors il existe au moins un solution τ qui est le temps auquel les particules entreront en collision. Sachant que α est positif, le seul τ acceptable est :

$$\tau = \frac{-\beta - \sqrt{\Delta}}{2\alpha}$$

car l'autre solution correspondrait au temps ou les deux particules étaient à une distance inférieure à la somme de leur rayon, et viendraient de se séparer.

Il faut que τ soit positif, cela signifie que :

$\beta + \sqrt{\Delta} < 0 \rightarrow \beta^2 > \beta^2 - 4\alpha\gamma \rightarrow \alpha\gamma > 0$, ce qui est systématiquement vrai car α et γ sont positifs. Cette condition n'a donc pas lieu d'être vérifiée (dans la mesure où toutes les collisions sont détectées et que les gouttes ne sont pas injectées l'une dans l'autre). La seconde condition sur τ est $\tau < \Delta t$. Pour satisfaire à cette condition, seule une comparaison peut être effectuée.

En conlusion, si τ existe et $0 \leq \tau < \Delta t$, alors il y a collision. La prise en charge de cette collision est expliquée dans le chapitre suivant.

4.3 Traitement de la collision élastique en 2D

Maintenant que nous connaissons les paires de particules qui collisionnent, nous allons traiter cette collision. Nous avons leur position , leur vitesse, leur masse, leur rayon et le temps exact τ de leur collision. Considérons leur vitesse comme constante entre 0 et τ et entre τ et Δt en faisant une approximation du premier ordre.

On définit un repère local de la collision $\overrightarrow{i}$, $\overrightarrow{j}$, avec :

$$\vec{i} = \frac{(\overrightarrow{X}_R + \tau\overrightarrow{V}_R) - (\overrightarrow{X}_S + \tau\overrightarrow{V}_S)}{R_R + R_S} = \begin{bmatrix} x_i \\ y_i \end{bmatrix} \tag{3}$$

avec : $\vec{j} \perp \vec{i} : \vec{j} = \begin{bmatrix} y_i \\ -x_i \end{bmatrix}$. Ce repère est décrit par la figure 6.

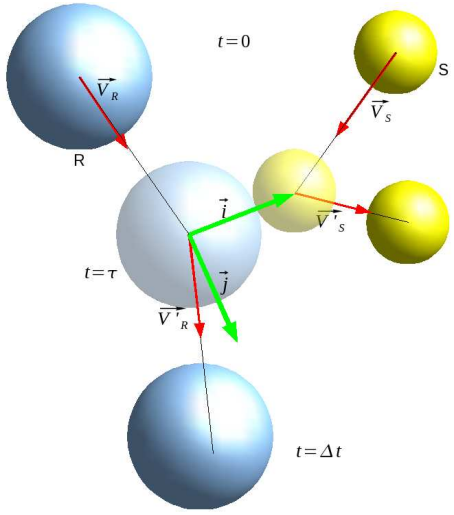


FIG. 6: Représentation du repère local de la collision

La collision se faisant sur l'axe $\vec{i}$, alors la vitesse de ces particules sur $\vec{j}$ reste constante. On connaît donc $v'_{R,j}$ et $v'_{S,j}$.

Sur $\vec{i}$, la conservation de la quantité de mouvement nous donne :

$$P_i = m_R v_{R,i} + m_S v_{S,i} = cste \tag{4}$$

avec $\overrightarrow{v}'_R$ et $\overrightarrow{v}'_S$ les vitesses des particules après la collision. m_k nous est donné simplement par $m_k = 4\pi R_k^3 \rho/3$. Nous avons deux inconnues ($v'_{R,i}$ et $v'_{S,i}$), il nous faut donc deux équations. La première est la conservation de la quantité de mouvement sur $\overrightarrow{i}$.
La seconde passera par la conservation de l'énergie (à une constante paramétrable près : η étant la perte énergétique suite à la collision).
On a : $\eta \sum_{i,j} E_c = \sum_{i,j} E'_c$. D'où,

$$\eta \left(\frac{1}{2} m_R v_{R,i}^2 + \frac{1}{2} m_S v_{S,i}^2 \right) = \eta E_i = \left(\frac{1}{2} m_R v_{R,i}'^2 + \frac{1}{2} m_S v_{S,i}'^2 \right) \tag{5}$$

$$\begin{cases} P_i = m_R v_{R,i} + m_S v_{S,i} = m_R v'_{R,i} + m_S v'_{S,i} \\ P_j = m_R v_{R,j} + m_S v_{S,j} = m_R v'_{R,j} + m_S v'_{S,j} \end{cases} \tag{6}$$

De l'équation 6 , on en déduit :

$$v'_{R,i} = \frac{P_i - m_S v'_{S,i}}{m_r} \tag{7}$$

qui est injecté dans l'équation 5, pour donner

$$v'_{S,i} = \frac{P_i \pm \sqrt{P_i^2 - \frac{m_R^2}{\mu} \left(\frac{P_i^2}{m_R} - 2\eta E_i \right)}}{m_R + m_S} \tag{8}$$

avec μ la masse réduite, définie par $\frac{1}{\mu} = \frac{1}{m_R} + \frac{1}{m_S}$.
Afin de déterminer le signe $\pm$ dans l'expression, considérons le cas simple où : $\overrightarrow{v_R} = \begin{bmatrix} v_0 \\ 0 \end{bmatrix}$, $\overrightarrow{v_S} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$, $m_R = m_S$ et $\eta = 1$. On obtient $\begin{cases} v'_{R,i} = 0 \\ v'_{S,i} = v_0 \end{cases}$
Appliquons dans le résultat donné par l'équation 8. Nous devons obtenir

$$v'_{S,i} = v_0 = \frac{P_i \pm \sqrt{P_i^2 - \frac{m_R^2}{\mu} \left(\frac{P_i^2}{m_R} - 2\eta E_i \right)}}{m_R + m_S}$$

.

$$v'_{S,i} = \frac{m_R v_0 \pm \sqrt{m_R^2 v_0^2 - m_R^2 \frac{2}{m_R} \left(\frac{m_R^2 v_0^2}{m_R} - m_R v_0^2 \right)}}{2m_R} = \frac{v_0 \pm v_0}{2} = v_0$$

C'est donc un +. Par conséquent, nous avons

$$v'_{S,i} = \frac{P_i + \sqrt{P_i^2 - \frac{m_R^2}{\mu} \left(\frac{P_i^2}{m_R} - 2\eta E_i \right)}}{m_R + m_S} \tag{9}$$

La déduction de $v'_{R,i}$ nous est donnée par l'équation 7.
Maintenant que les composantes de $\overrightarrow{V_R}$ et de $\overrightarrow{V_S}$ sont connues, nous les projetons

dans le repère d'origine :

$$\begin{cases} v'_{R,x} = v'_{R,i}\vec{i} \cdot \overrightarrow{Ox} + v'_{R,j}\vec{j} \cdot \overrightarrow{Ox} = (v'_{R,i}\vec{i} + v'_{R,j}\vec{j}) \cdot \overrightarrow{Ox} \\ v'_{R,y} = v'_{R,i}\vec{i} \cdot \overrightarrow{Oy} + v'_{R,j}\vec{j} \cdot \overrightarrow{Oy} = (v'_{R,i}\vec{i} + v'_{R,j}\vec{j}) \cdot \overrightarrow{Oy} \\ v'_{S,x} = v'_{S,i}\vec{i} \cdot \overrightarrow{Ox} + v'_{S,j}\vec{j} \cdot \overrightarrow{Ox} = (v'_{S,i}\vec{i} + v'_{S,j}\vec{j}) \cdot \overrightarrow{Ox} \\ v'_{S,y} = v'_{S,i}\vec{i} \cdot \overrightarrow{Oy} + v'_{S,j}\vec{j} \cdot \overrightarrow{Oy} = (v'_{S,i}\vec{i} + v'_{S,j}\vec{j}) \cdot \overrightarrow{Oy} \end{cases}$$

Ou, plus simplement, en passant par une matrice changement de repère $\overline{\overline{M}}$:

$$\overrightarrow{V'_{\mathbf{R}}} = \overline{\overline{M}}\overrightarrow{V'_{\mathbf{R}'}} \quad , \quad \overline{\overline{M}} = \begin{bmatrix} \overrightarrow{i}.\overrightarrow{Ox} & \overrightarrow{j}.\overrightarrow{Ox} \\ \overrightarrow{i}.\overrightarrow{Oy} & \overrightarrow{j}.\overrightarrow{Oy} \end{bmatrix} \tag{10}$$

Maintenant que $\overrightarrow{X}_R$, $\overrightarrow{X}_S$, $\overrightarrow{V}_R$, $\overrightarrow{V}_S$, $\overrightarrow{V'}_R$, $\overrightarrow{V'}_S$ et τ sont connus, alors la position de ces particules à la fin du pas de temps sera déduite de la manière suivante : $\overrightarrow{X'_R} = \overrightarrow{X_R} + \tau\overrightarrow{V_R} + (\Delta t - \tau)\overrightarrow{V'_R}$ $\overrightarrow{X'_S} = \overrightarrow{X_S} + \tau\overrightarrow{V_S} + (\Delta t - \tau)\overrightarrow{V'_S}$

4.4 Passage à la 3D

La méthode de détection des plus proches voisins reste la même. Nous trions les particules selon leur abscisse, puis pour chaque particule nous cherchons celles dont :

- l'abscisse se trouve au plus à D_{max} vers la droite de la particule que l'on considère,
- l'ordonnée se trouve dans $\pm D_{max}$ de la particule considérée,
- la profondeur se trouve dans $\pm D_{max}$ de la particule considérée.

Une fois les partenaires potentiels détectés, on calcule leur distance en fonction du temps de chacun des couples de particules en prenant en compte leur vitesse selon $\overrightarrow{z}$. Soient R,S le couple de particules dont on veut vérifier s'ils vont collisioner. On a

$$L(t)^2 = (\Delta X^2 + \Delta Y^2 + \Delta Z^2) + 2t(\Delta V_X \Delta X + \Delta V_Y \Delta Y + \Delta V_Z \Delta Z) + t^2(\Delta V_X^2 + \Delta V_Y^2 + \Delta V_Z^2)$$

Il faut qu'il existe $\tau \in [0, \Delta t]$ tel que $L(\tau) = R_R + R_S$ pour qu'il y ait collision. De manière similaire, cela revient à dire qu'il nous faut

$$0 < \tau = \frac{-\beta + \sqrt{\beta^2 - 4\alpha\gamma}}{2\alpha} < \Delta t$$

avec :

$$\begin{cases} \alpha = \Delta V_X^2 + \Delta V_Y^2 + \Delta V_Z^2 \\ \beta = 2(\Delta V_X \Delta X + \Delta V_Y \Delta Y + \Delta V_Z \Delta Z) \\ \gamma = \Delta X^2 + \Delta Y^2 + \Delta Z^2 - \Delta R^2 \end{cases}$$

ΔV_X, ΔV_Y, ΔV_Z, ΔX, ΔY, ΔZ et ΔR sont définis à la section précédente.

Si τ existe, alors nous passons à la collision qui se fait de la même manière que précédemment. Nous définissons un repère local de collision ($\overrightarrow{i}$, $\overrightarrow{j}$, $\overrightarrow{k}$), avec :

$$\vec{i} = \frac{(\overrightarrow{X}_R + \tau\overrightarrow{V}_R) - (\overrightarrow{X}_S + \tau\overrightarrow{V}_S)}{R_R + R_S} = \begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix}$$

Afin de former une base orthonormée directe, nous avons $\overrightarrow{k} = \overrightarrow{i} \wedge \overrightarrow{j}$, sachant que

$$\overrightarrow{i} \cdot \overrightarrow{j} = x_i x_j + y_i y_j + z_i z_j = 0 \tag{11}$$

Peu importe la direction de $\overrightarrow{j}$ tant qu'il satisfait l'équation 11. Une solution évidente de cette équation est $\overrightarrow{j} = \frac{1}{\sqrt{x_i^2+y_i^2}} \begin{bmatrix} -y_i \\ x_i \\ 0 \end{bmatrix}$, ce qui nous donne $\overrightarrow{k} = \frac{1}{\sqrt{x_i^2+y_i^2}} \begin{bmatrix} -x_i z_i \\ -y_i z_i \\ x_i^2 + y_i^2 \end{bmatrix}$.

La collision reste toujours selon $\overrightarrow{i}$, ce qui ne change en rien les bilans en quantité de mouvement et énergétiques que nous avions auparavant.

On obtient donc

$$\vec{V'_S} = \begin{bmatrix} \frac{P_i + \sqrt{P_i^2 - \frac{m_R^2}{\mu}\left(\frac{P_i^2}{m_R} - 2\eta E_i\right)}}{m_R + m_S} \\ V_{S,j} \\ V_{S,k} \end{bmatrix}$$

et

$$\vec{V'_R} = \begin{bmatrix} \frac{P_j - m_S v'_{S,i}}{m_R} \\ V_{R,j} \\ V_{R,k} \end{bmatrix}$$

Le passage dans le repère d'origine se fait exactement de la même façon que précédemment :

$$\overrightarrow{V'_{\mathbf{R}}} = \overline{\overline{M}}\overrightarrow{V'_{\mathbf{R'}}} \quad , \quad \overline{\overline{M}} = \begin{bmatrix} \overrightarrow{i} \cdot \overrightarrow{Ox} & \overrightarrow{j} \cdot \overrightarrow{Ox} & \overrightarrow{k} \cdot \overrightarrow{Ox} \\ \overrightarrow{i} \cdot \overrightarrow{Oy} & \overrightarrow{j} \cdot \overrightarrow{Oy} & \overrightarrow{k} \cdot \overrightarrow{Oy} \\ \overrightarrow{i} \cdot \overrightarrow{Oz} & \overrightarrow{j} \cdot \overrightarrow{Oz} & \overrightarrow{k} \cdot \overrightarrow{Oz} \end{bmatrix} \tag{12}$$

4.5 Complexité (coût calcul) de l'algorithme

Le but ici va être de déterminer le coût calcul de l'algorithme en 2D, et de sa généralisation en 3D. L'algorithme se décompose en trois parties, nous allons tenter de déterminer la complexité de chacune d'entre elles.

4.5.1 Complexité de la détection des partenaires potentiels

Cette première partie consiste principalement au tri des particules selon l'axe des abscisses. (Ce tri a été effectué selon un algorithme de tri rapide (Quicksort, [22])). La complexité d'un tel algorithme varie de N à $N\log(N)$ selon que la liste soit – dans les cas les plus favorables – complètement aléatoire ou – dans le pire des cas – strictement décroissante ou croissante. En effet, la technique du tri rapide (ou par pivot) se fait de la manière suivante :

On prend une liste constituée d'éléments aléatoires. On sélectionne un pivot et on cherche la liste des valeurs inférieures à ce pivot et on les mets à gauche. Les autres valeurs sont mises à droite (Fig. 7). En N opérations, on a classé une valeur. Puis

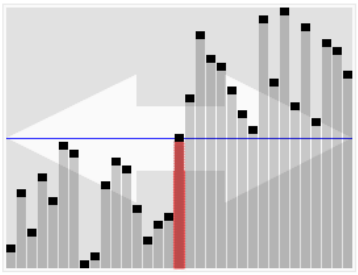


FIG. 7: Placement du premier pivot

on recommence avant et après le pivot qui a été classé (Fig. 8). Ici, nous avons en N

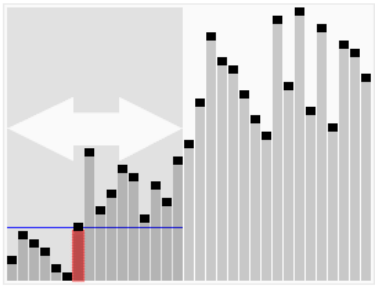


FIG. 8: Sélection et classement autour du deuxième pivot

comparaisons, classé deux valeurs. On recommence, en N comparaisons on en classe 4, puis 8 puis 16... Et finalement, la liste est classée (Fig. 9). En k itérations, dans le meilleur des cas, on aura classé $1+2+4+...+2^{k-1} = 2^k - 1$ valeurs. Il faut donc : $2^k + 1 = N \rightarrow k = \frac{\ln(N+1)}{\ln(2)}$ itérations (balayage de liste). Pour chaque itération, il faut effectuer N comparaisons, ce qui porte le nombre total de comparaisons à une complexité de $N\log(N)$.

Comme il a été dit, cette complexité se vérifie dans le meilleur des cas, c'est à dire que le nombre de valeurs triées en fonction de k est exactement de 2^{k-1}.

Imaginons que la liste soit strictement décroissante et que nous prenons pour pivot la première valeur non triée.

Première itération : le pivot va à la fin de la liste, le reste des valeurs ne bougent pas.

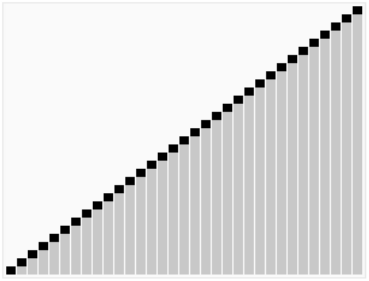


FIG. 9: Liste classée

Deuxième itération : le premier pivot est en fait l'avant dernier de la liste, il va donc à cette position et le reste ne bouge pas etc.
Finalement, à chaque itération nous ne trions qu'une valeur. Il faudra par conséquent faire N^2 comparaisons pour tout trier.
C'est pour cela que cet algorithme est de complexité $N\log(N)$ à N^2. Selon l'organisation de la liste, le tri variera entre ces deux complexités.

Sachant que dans notre cas que les particules sont toujours injectées par un des cotés les unes après les autres, la liste de leur abscisse sera globalement décroissante, ce qui correspond à un cas défavorable au tri. Afin de compenser ce problème, le système a été amélioré en prenant un pivot se trouvant exactement au milieu de chaque zone à trier, ce qui tend pour des cas croissants ou décroissants à s'approcher beaucoup plus du $Nlog(N)$ (statistiquement).

4.5.2 Coût de la détection par les équations de trajectoire

Cette partie est probablement la plus longue car elle consiste à tester parmi les plus proches voisins toutes les équations de trajectoires permettant de détecter les collisions. Le nombre de partenaires potentiels de collision est évidement fonction de la densité de particule. Dans le cas 2D, les particules étant disséminées dans le domaine, pour chaque "morceau" d'abscisse de longueur D_{max}, le nombre de particules présentes dans cette tranche sera proportionnelle à $\sqrt{N}$. Sachant que nous devons détecter pour chaque particule ses partenaires de collisions, alors la complexité sera en $N\sqrt{N} = N^{3/2}$.

En 3D, les particules sont dispersées selon $\sqrt[3]{N}$ dans le domaine, ce qui pour chaque particule, donne une quantité de plus proche voisins proportionnels à $\sqrt[3]{N^2} = N^{2/3}$. Cela entraîne une complexité de l'ordre de $N^{5/3}$.

Nous verrons dans la section 7 que c'est la partie la plus longue, mais néanmoins obligatoire pour cet algorithme tel qu'il a été conçu (ainsi que pour son optimisation, nous verrons ceci dans la section B.2).

4.5.3 Coût du traitement de la collision

Sachant que pour chaque particule, nous avons supposé par hypothèse qu'à chaque pas de temps elle ne pouvait collisioner qu'une seule fois, alors au final il y aura tout au plus N/2 couples de particules susceptibles de collisioner. Le traitement s'effectuant par des opérations arithmétiques simples, elle est de complexité

N. C'est largement négligeable face aux deux premières parties.

4.5.4 Complexité totale

La complexité de ce module est de l'ordre de $Nlog(N) + N^{3/2}$ en 2D, et de $Nlog(N) + N^{5/3}$ en 3D. Nous avons tracé, pour la modélisation du jet à contre courant, les temps de calcul en fonction du nombre de collisions (ce qui nous donnera indirectement le temps mis pour la prise en charge des collisions). Ce temps d'éxécution est représenté par la figure 14.

5 Optimisations

5.1 Décomposition de D_{max} selon les directions étudiées

Dans la littérature nous n'avons pas trouvé pour le moment d'algorithme qui fasse la distinction à proprement parler de la direction des vecteurs vitesses. Nous verrons ici que dans certains cas, cette séparation est très avantageuse. Cependant, dans un cas isotrope, nous verrons pourquoi elle ne l'est pas.

Nous avons vu précédemment que pour chaque particule, il nous faut tester les trajectoires de toutes les particules se trouvant dans la surface $D_{max} * 2D_{max}$. Sachant que pour N gouttes dans la zone d'interaction, le nombre de couples à tester est de $(N^2+N)/2$. Par conséquent, si nous réussissons à diminuer cette surface (tout en prenant en compte toutes les collisions possibles), nous diminuons le nombre de tests à effectuer de manière très significative.

Nous avons donc mis en place un concept simple mais – selon les géométries – très efficace. Au lieu de définir les surfaces comme $D_{max} * 2D_{max}$, avec $D_{max} = 2(V_{max}\Delta t + R_{max})$ (paragraphe 4.1), on regarde la vitesse maximale sur (Ox) et sur (Oy) des particules, et nous ferrons la distinction des vitesses V_y positives et négatives.

Nous poserons par la suite :

$$\begin{cases} V_{max,x} = -\max_i(-V_{x,i}) = \min_i(V_{x,i}) \\ V_{max,y,+} = \max_i V_{y,i} \\ V_{max,y,-} = -\max_i(-V_{y,i}) = \min_i V_{y,i} \end{cases} \tag{13}$$

Supposons une goutte (R) ayant le vecteur vitesse $\overrightarrow{V_R} = \begin{bmatrix} V_{R,x} \\ V_{R,y} \end{bmatrix}$. Regardons à quelle distance maximale peut se trouver une goutte (i) en fonction de $\overrightarrow{V_R}$ et de $\overrightarrow{V_i}$ qui n'est pas connue d'avance.

On conçoit parfaitement que, dans le pire des cas possibles, la goutte R de vitesse sur x $V_{R,x}$ ira à l'encontre sur le même axe d'une goutte imaginaire allant vers la gauche avec une vitesse $V_{max,x}$ (la plus petite valeur des vitesses sur x, le signe est donc pris en compte).

La distance $D_{max,x}$ qu'il faut explorer pour la goutte R est donc définie par

$$D_{max,x} = (V_{R,x} - V_{x,max})\Delta t + R_R + R_{max} \tag{14}$$

Le sens positif est pris de la gauche vers la droite. Ceci est illustré par la figure 10.

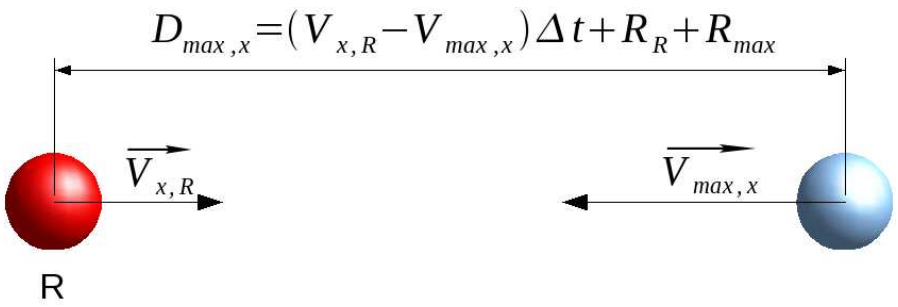


FIG. 10: Détermination de $D_{max,x}$

Faisons de même pour l'exploration sur y vers le haut et vers le bas. La distance maximale à parcourir vers le haut de la goutte R est notée $D_{max,y,1}$ et est donnée par l'équation 15 (toujours dans le pire des cas).

$$D_{max,y,1} = (V_{R,y} - V_{max,y,-})\Delta t + R_R + R_{max} \tag{15}$$

(le sens positif est pris du bas vers le haut)(cf la figure 11a). La distance maximale à parcourir vers le bas de la goutte R est notée $D_{max,y,2}$ et est symbolisée par la figure 11b. Cette distance nous est donnée par l'équation suivante :

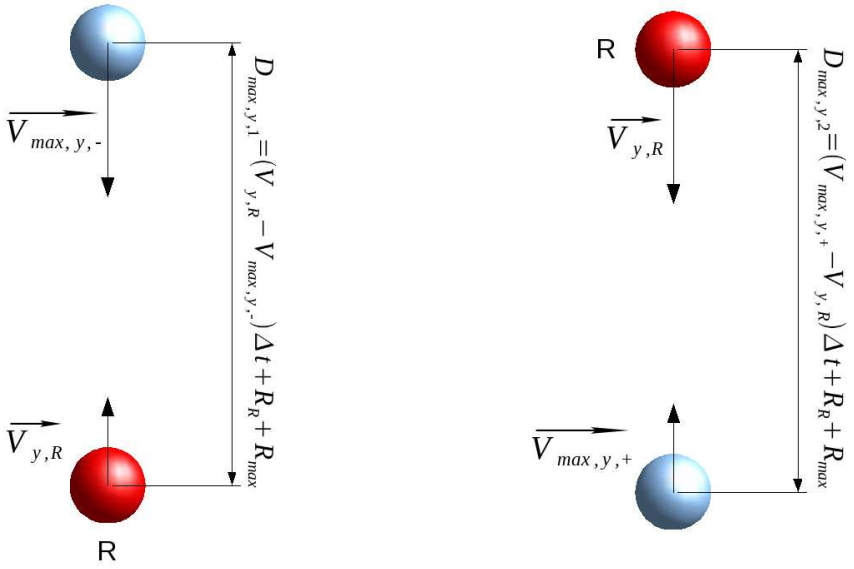
$$D_{max,y,2} = (V_{max,y,+} - V_{R,y})\Delta t + R_R + R_{max} \tag{16}$$

Au final la surface à explorer n'est plus de $S = 2 * D_{max}^2$ mais de $S_{optim} = D_{max,x} * (D_{max,y,1} + D_{max,y,2}) = ((V_{R,x} - V_{x,max})\Delta t + R_R + R_{max}) * ((V_{R,y} - V_{max,y,-})\Delta t + R_R + R_{max} + (V_{max,y,+} - V_{R,y})\Delta t + R_R + R_{max})$.
Finalement nous obtenons une surface donnée par l'équation 17.

$$S_{optim} = ((V_{R,x} - V_{x,max})\Delta t + (R_R + R_{max})) * ((V_{max,y,+} - V_{max,y,-})\Delta t + 2(R_R + R_{max})) \tag{17}$$

Cette surface possède bon nombre d'avantages. Tout d'abord, nous voyons que dans une géométrie préférentiellement axiale sur x (par exemple deux jets à contre courants), nous avons $V_{max,y,+} - V_{max,y,-} \to 0$. S_{optim} sera donc d'autant plus faible que la géométrie a cet axe privilégié. La surface à explorer tendra donc vers $S_{optim} = 2(R_R + R_{max})^2 << S_{initiale}$.

De plus, si $V_{R,x} \to V_{x,max}$, cette surface diminuera de même. Cela est parfait, par exemple dans le cas des jets croisés, ou la vitesse axiale initiale de toutes les



(a) Détermination de $D_{max,y,1}$ (b) Détermination de $D_{max,y,2}$

FIG. 11: Détermination des D_{max} sur y

particules est la même, la surface à explorer tendra donc vers $S_{optim} = (R_R + R_{max}) * ((V_{max,y,+} - V_{max,y,-})\Delta t + 2(R_R + R_{max})) << S_{initiale}$.

Une rapide comparaison de la définition de D_{max}, $D_{max,x}$, $D_{max,y,1}$ et $D_{max,y,2}$ nous montrera que la surface S_{optim} est très largement inférieure à la surface S que nous utilisions auparavant. Cette surface est symbolisée figure 12a.

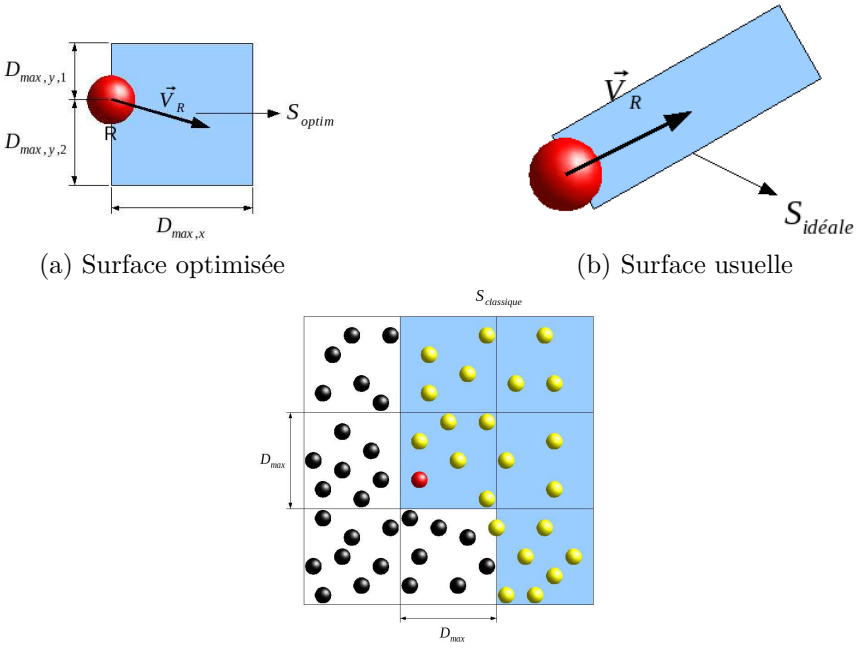
Nous pouvons comparer cette surface aux surfaces classiquement utilisées et aux surfaces que nous avions initialement (Fig. 12b et Fig. 12c). Ceci est extrêmement intéressant dans la mesure ou une division globale de ces surfaces d'exploration par $\frac{S_{optim}}{S}$ entraîne une division des couples à tester de $\left(\frac{S_{optim}}{S}\right)^2$. Plus le ratio $\frac{S_{optim}}{S}$ est petit, plus l'algorithme sera performant.

6 Description du module programmé

Dans cette partie, on décrira la mise en place ainsi que l'analyse de l'exécution du module une fois codé. Tous les temps d'exécution et l'étude temporelle ont été effectuées sur un Power4, en utilisant un seul processeur cadencé à 2,66 Ghz et avec 8Go de RAM cadencées à 667 Mhz. Les points largement en dehors des graphiques peuvent s'expliquer par l'encombrement de la pile processeur par diverses applications lors de l'exécution du code. Néanmoins, les résultats globaux restent d'une excellente qualité.

6.1 Codage de la détection des plus proches voisins

Une fois le code écrit, les premiers tests de détection de plus proches voisins ont été effectués. On a ensuite codé le système de séparation de $D_{max,x}$ et $D_{max,y}$, comme



(a) Surface optimisée

(b) Surface usuelle

(c) Surface initialement utilisée

FIG. 12: Comparaison des différentes surfaces d'exploration

elle a été décrite dans la section 5.1. Les résultats étaient parfaitement concluants. Ensuite, on a estimé la complexité réelle ainsi que la durée d'exécution de cette partie du module afin de voir si elles étaient conformes aux prédictions (c'est à dire une complexité située entre N et $N \log(N)$.

Voici le graphique représentant la durée d'exécution du classement en fonction du nombre de gouttes dans le cas de l'injection à contre-courant :

Nous voyons que l'évolution est quasi-linéaire, mais légèrement supérieure à une évolution purement linéaire, ce qui confirme notre prédiction.

De plus nous pouvons remarquer que la durée d'exécution de cette partie du module reste très faible face aux autres parties comme nous allons le voir par la suite.

6.2 Détection des collisions par les équations de trajectoire

Sachant que ces deux parties sont extrêmement liées de part le nombre de collisions exactes à détecter, ils nous a été impossible de discerner le temps mis par la détection par les équations de trajectoires du temps mis par la prise en charge de ces collisions.

Comme précédemment, une fois cette partie du module programmé, nous avons déterminé la durée d'exécution de cette partie complète du module.

Nous avons tracé ce temps mis par cette partie du module en fonction du nombre de collisions exactes détectées (Fig. 14). Nous voyons dans ce cas que la densité de gouttes est très importante. En effet, à chaque itération, entre 10% et 30% des

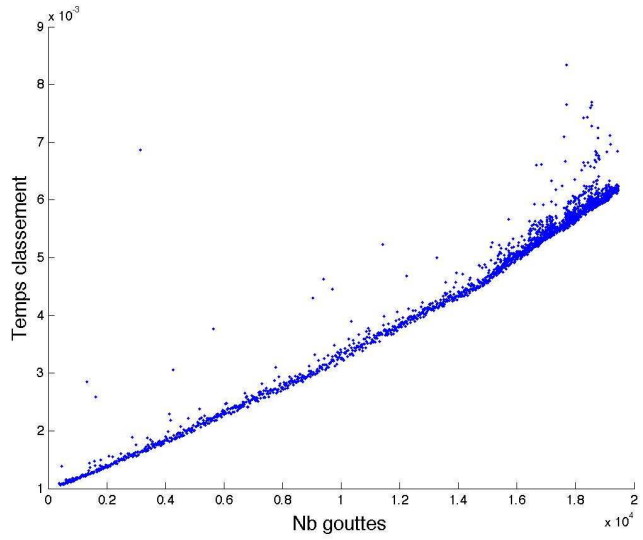


FIG. 13: Durée d'exécution du classement de l'abscisse des gouttes

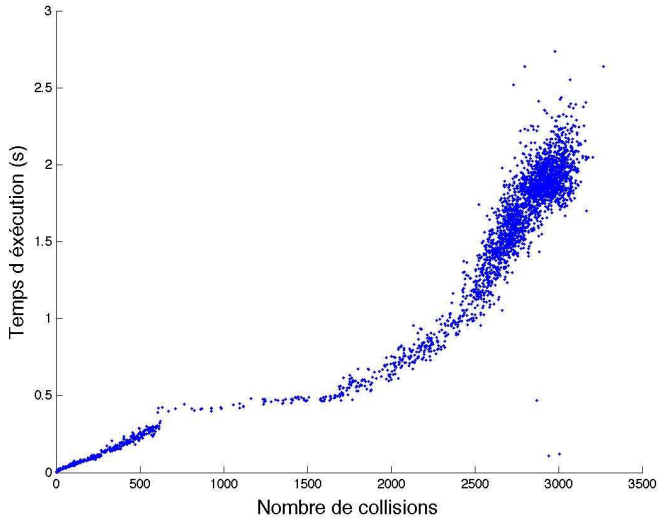


FIG. 14: Durée d'exécution de la détection et de la prise en charge des collisions

gouttes collisionnent. Dans la plupart des simulations effectuées par la suite, une telle densité n'a pas été atteinte car il sagit d'un test hors de tout critère physique. La discontinuité s'explique simplement. En effet, dans le cas du contre courant, les vitesses sont quasiment uniquement horizontale avant la rencontre des 2 jets, ce qui induit une surface à explorer très faible. A partir du moment où les jets sont en contact, les particules sont violemment éjectées vers l'extérieur, ce qui entraîne une isotropie des vecteurs vitesses. La surface à explorer s'en trouve grandement augmentée, ainsi que le nombre de couples à tester.

Néanmoins, ce fût l'occasion de tester la robustesse du code face à l'apparition de traînées importantes et de l'aptitude du module à prendre en charge autant de

gouttes dans un volume aussi restreint.
De plus, nous pouvons observer une évolution quasi-linéaire en début de simulation, ce qui correspond au cas où les 2 jets ne se sont pas encore croisés. Auquel cas le nombre de collision est faible.
C'est une fois les 2 jets en contact que le nombre de collision augmente significativement. Nous pouvons observer une évolution approximativement en $N^{\simeq 3/2}$ de cette partie du code, ce qui reste une fois encore conforme aux prédictions faites dans la partie 4.5.4.
De plus, comme il a été dit dans le paragraphe précédent, la durée de cette partie frôle les 2s pour la prise en charge de 3500 collisions, ce qui, face à la durée de détection des voisins, reste largement prépondérant.

7 Mise en place du module et résultats obtenus

Tout d'abord pour tester ce module, on a effectué un cas simple pour voir s'il prenait bien en compte les collisions, et que le résultat numérique correspondait avec le résultat théorique. On a donc envoyé 2 gouttes de même taille, de même vitesse avec un paramètre d'impact de 45° (Fig. 15).

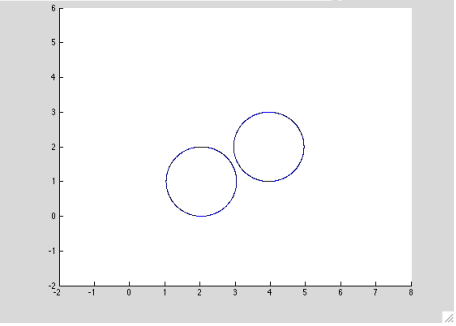


Fig. 15: Collision simple

La collision s'est effectuée correctement, les résultats ont été examinés et sont conformes aux résultats théoriques.

Puis, pour vérifier que les 2 principales parties du module fonctionnent correctement, nous avons simulé un mouvement Brownien (Fig. 16) d'une particule parmi 10000 en agitation.

Ainsi que la collision de 2 groupes de particules élancées l'un contre l'autre (avec collision au dessus, sans collision en dessous) comme nous le montre la figure 17.

Ensuite s'en est suivi la mise en place du module, ce qui fut chose relativement aisée. En effet, nous donnons les positions, vitesses des particules, ainsi que le Δt de la simulation. Celui-ci nous retourne une liste des nouvelles positions et vitesse de ces particules. Il suffit après l'avancement en temps des gouttes de remettre à jour ces même positions et vitesses pour que les collisions aient été prises en compte. Une petite manipulation d'index a été effectuée pour que la mise à jour ne prenne pas compte des gouttes qui auraient disparues (évaporation, sortie de domaine) pendant l'avancement en temps.

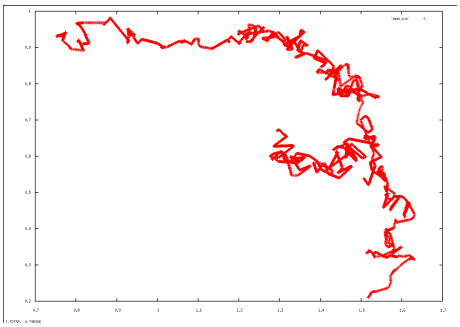


FIG. 16: Mouvement Brownien

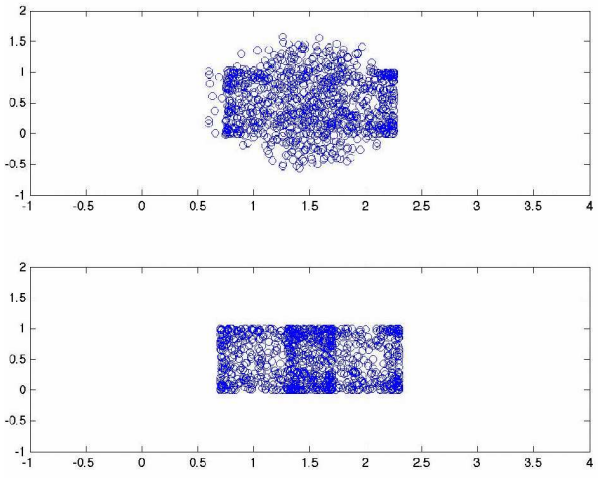


FIG. 17: Collision de 2 ensembles de particules

L'intégration dans le code a été faite ainsi :
- détection et prise en charge des collisions. Création d'une liste contenant l'index, les nouvelles vitesses et positions des particules concernées,
- avancement en temps du solveur,
- mise à jour des particules concernées par les collisions.

7.1 Résultats

Une fois l'intégration du module faite, plusieurs simulations ont été effectuées. Ces simulations sont décrites dans la section suivante.

8 Simulations effectuées

Quatre simulations ont été effectuées afin de comparer les statistiques du champ Eulérien avec et sans collision.

Ces quatre configurations ont été effectuées dans un domaine carré de 300 noeuds de coté, avec une taille de code de $6x_0$ par $6x_0$. Le pas de temps est choisi de telle sorte que le CFL n'excède pas 0.4. Nous avons par conséquent $\Delta x = \Delta y = 2.0\ 10^{-2}$ en dimension de code et $\Delta z = C_{vc}\Delta x$. Le coefficient C_{vc} a été introduit afin de prescrire

la richesse en entrée, tout en ayant suffisamment de gouttes pour que les collisions puissent être effectives. Le Reynolds de normalisation étant de 1000, nous pouvons estimer dans ce cas précis x_0 : $x_0 = \frac{\nu Re}{U_0}$ avec U_0 la vitesse du gaz à l'injection. Nous avons pris ici $U_0 = 1\ m.s^{-1}$. On obtient $x_0 = 1.5 \quad cm$ taille réelle.

Les conditions limites sont indiquées sur la figure 18.

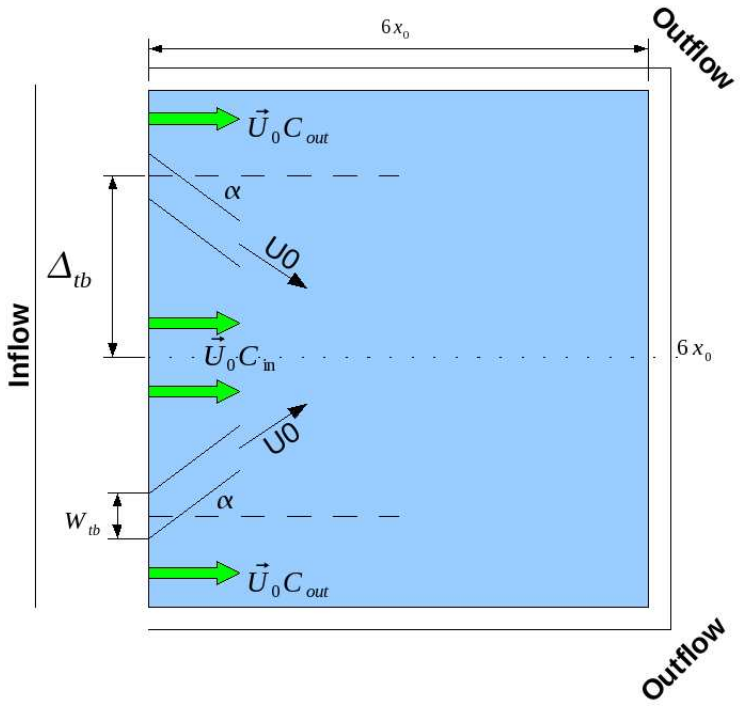


FIG. 18: Conditions limites et paramètres

Les profils de vitesse en entrée sont donnés par les équations 18 et 19. y_{mid} a été défini comme étant l'ordonnée du milieu du domaine : $y_{mid} = (y_{min} + y_{max})/2$.

$$\begin{cases} U_x(0,y) = U_0 \cos(\alpha)\left(P(x,\Delta h, W_{tb}, \Delta_{tb}) + C_{f-in}(1 - P(x,\Delta h, W_{tb}, \Delta_{tb}))\right) \\ \text{si} |y - y_{mid}| < \Delta_{tb} \\ U_x(0,y) = U_0 \cos(\alpha)\left(P(x,\Delta h, W_{tb}, \Delta_{tb}) + C_{f-out}(1 - P(x,\Delta h, W_{tb}, \Delta_{tb}))\right) \\ \text{sinon} \end{cases} \tag{18}$$

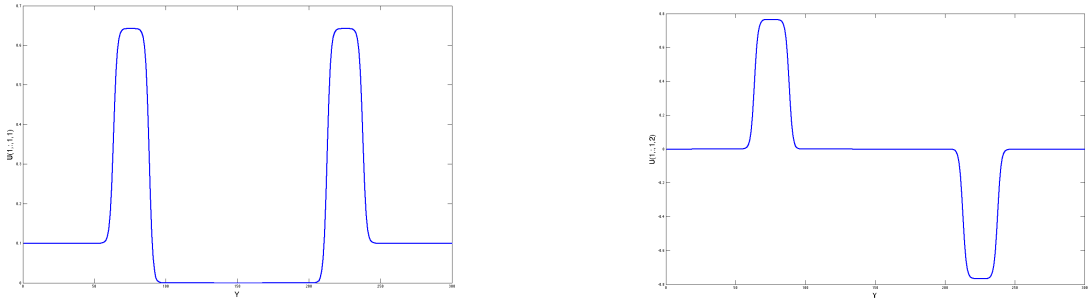
$$U_y(0,y) = -sign(y_{mid} - y) U_0 \sin(\alpha) P(x,\Delta h, W_{tb}, \Delta_{tb}) \tag{19}$$

La fonction $P(x,\Delta h, W_{tb}, \Delta_{tb})$ - le profil d'entrée - a été définit par l'équation 20.

$$P(x,\Delta h, W_{tb}, \Delta_{tb}) = \frac{1}{2}\left(1 + \tanh\left(\frac{W_{tb}}{\Delta h}\left(1 - 2\frac{|x - \Delta_{tb}|}{W_{tb}}\right)\right)\right) \tag{20}$$

Ces champs de vitesses sont paramétrés de telle sorte que l'angle à l'injection soit de $\alpha = 50^{\circ}$, avec un CoFlow intérieur de $C_{f-in} = 0\%$ et un Coflow extérieur de $C_{f-out} = 10\%$. La distance du centre des 2 injecteurs par rapport au centre du domaine est $\Delta_{tb} = 1.5x_0$, leur heuteur est de $w_{tb} = 0.5x_0$.

Afin d'éviter l'apparition de trop forts gradients, Δh à été mis égal à 0.1.
Les profils de U_x et U_y à l'entrée sont indiqués par les figures 19a et 19b.



(a) Vitesse horizontale (b) Vitesse verticale

FIG. 19: Profil des vitesses en entrée

Nous avons pris comme composition gazeuse à l'infini de l'air : $Y_{\infty,O_2} = 0.232$ et $Y_{\infty,N_2} = 0.768$ à une température de 300K.

Les paramètres des gouttes injectées ont été choisis pour avoir des statistiques significatives sans pour autant être sensible à un trop grand nombre de paramètres. Trois possibilités de diamètres sont possible. La première, bi-disperse, leur impose une taille à l'injection de $a_{0,1}$ et $a_{0,2}$. Le volume de Fuel injecté par chaque famille de goutte est identique. La deuxième possibilité, Mono $a_{0,1}$, n'injecte que des gouttes de diamètre $a_{0,1}$.

Les 3 configurations sont recensées dans le tableau 8

	Nom	Mono/BiDisperse	Collision (O/N)	Densité
Simu 1	NC-NEVAP-B-U0	Bi $a_{0,1}$ et $a_{0,2}$	N	Importante
Simu 2	2C-NEVAP-B-U0	Bi $a_{0,1}$ et $a_{0,2}$	O	Faible
Simu 3	C-NEVAP-B-U0	Bi $a_{0,1}$ et $a_{0,2}$	O	Importante

Quelques résultats et interprétations sont présentées dans la section 9.

9 Résultats des deux simulations

Ces simulations sont basées sur les configurations décrites dans la section 8. Les résultats produits permettront d'étudier l'importance de la prise en compe des collisions dans des configurations telles que celles-ci.

Deux études ont principalement été menées afin de donner tout son sens à ces résultats. Nous étudierons principalement la répartition en masse des particules, ainsi que la répartition en famille de goutte (Les gouttes de famille 1 sont les gouttes issues de l'injecteur du haut, celles de la famille 2 issues de l'injecteur du bas).

9.1 Simulation 1 : NC-NEVAP-B-U0

Les statistiques Eulériennes ont été effectuées en moyennant des "clichés" Lagrangiens, tel que celui représenté par la figure 9.1.

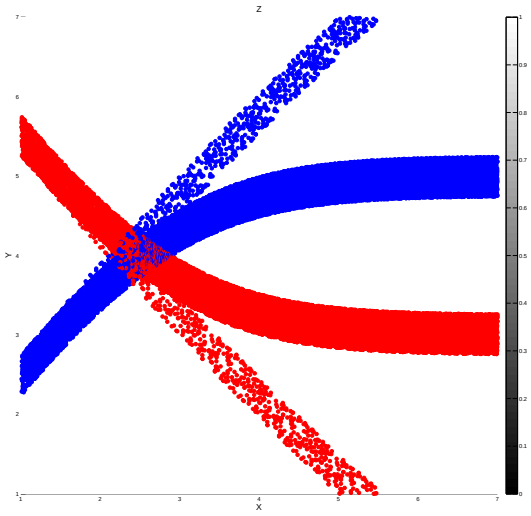


FIG. 20: Cliché Lagrangien de la simulation sans collision

Les particules lourdes (de diamètre $2a_0$) sont peu influencées par la présence du gaz porteur, leur trajectoire est donc quasi-rectiligne. Inversement, les gouttes légères (de diamètre a_0) sont elles très influencées par le gaz, leur trajectoire est beaucoup plus incurvée.

La masse moyenne des gouttes par maille est représentée par la figure 21a. On constate qu'au croisement des deux jets, la densité massique est presque doublée. Cela est tout à fait normal dans la mesure où les gouttes ne se voient pas.

Nous avons définit 2 familles de gouttes. Les gouttes de la famille 1 sont les gouttes issues de l'injecteur du haut, tandis que les gouttes issues de l'injecteur du bas font partie de la famille 2. La figure 22a nous donne la répartition Eulérienne des gouttes de la Famille 1.

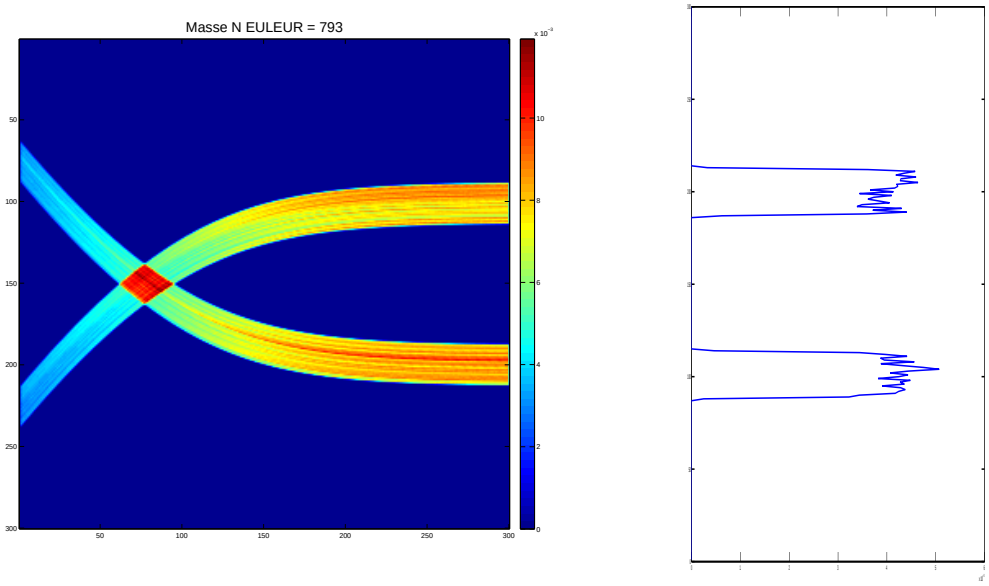
La figure 22b nous donne la répartition en famille de goutte à la sortie du domaine. La courbe en rouge est la densité en sortie des gouttes de la famille 1 en fonction de Y, la courbe en bleu est la densité en sortie des gouttes de la famille 2.

Nous pouvons constater qu'en l'abscence de collision, les gouttes restent regroupées par famille, et-ce tout au long de leur évolution dans le domaine. Il n'y a aucun mélange, ce qui est parfaitement logique.

Sachant que nous sommes dans une distribution de taille bi-disperse, nous avons des gouttes de rayon a_0 et $2a_0$. Ce qui fait que selon leur diamètre, une catégorie de goutte aura une masse 8 fois plus élevée que l'autre. Cela explique pourquoi quelques gouttes vont tout droit, car elles sont beaucoup plus inertielles que les autres.

9.2 Simulation 2 : 2C-NEVAP-B-U0

Cette configuration est de densité moindre. Comme nous le verrons dans la section 9.3, une densité trop importante de particules induit un non croisement des jets. Cela se traduit par la répartition des gouttes selon leur famille en sortie (Fig.



(a) Densité massique (b) Densité massique en sortie

FIG. 21: Densité massique de gouttes dans le domaine

27b). Ici, les figures 24a et 24b nous montrent que la prise en compte des collisions lors de densités moyennes induit un mélange quasi-homogène des familles de gouttes.

La figure 23 nous montre un cliché de la simulation "2-collision". Nous voyons que suite aux collisions dans le croisement des 2 jets, les gouttes sont mélangées de manière quasi-uniforme comme nous le prouve la figure 24b, représentant le mélange des 2 familles en sortie de domaine.

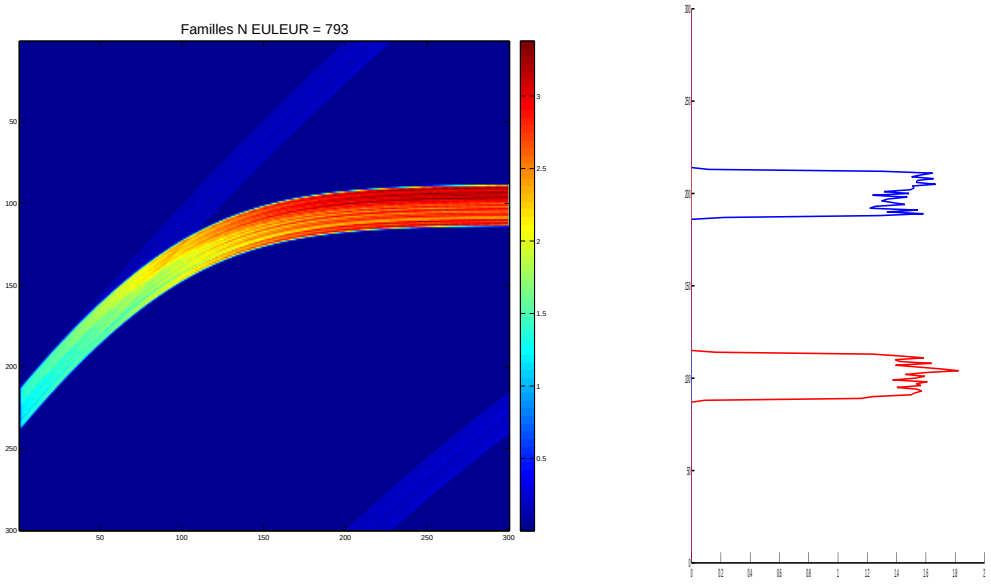
Cela est encore plus flagrant si on regarde la figure 25b, nous représentant la densité de masse moyenne en sortie de domaine. Certes, la répartition en masse est constituée globalement de 2 lobes symétriques, ce qui mets en apparence que certaines gouttes ne subissent aucune collision. Ce mélange est donc intermédiaire entre un mélange complet (Fig. 28b) et pas de mélange (Fig. 21b).

9.3 Simulation 3 : C-NEVAP-B-U0

Cette simulation est cele qui considère le plus de collisions. La densité de gouttes à l'injection est élevée. Un premier cliché de la simulation (Fig. 9.3) nous confirme qu'effectivement cette densité est très importante.

Cette configuration est tellement dense que les jets ne se croisent que très peu. Chaque goutte issue d'un injecteur voit, au croisement des jets, un mur de gouttes. Le mélange ne sera pas aussi efficace que celle que nous avions vu dans la section 9.2.

En effet, comme nous le montre la figure 24a, ainsi que la répartition de masse



(a) Famille 2 (b) Familles en sortie

FIG. 22: Distribution des familles en sortie

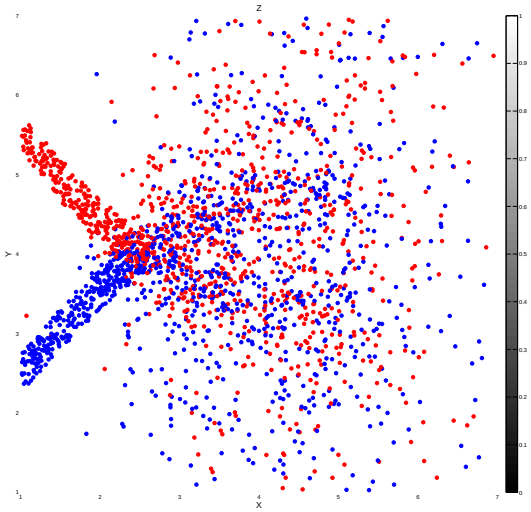
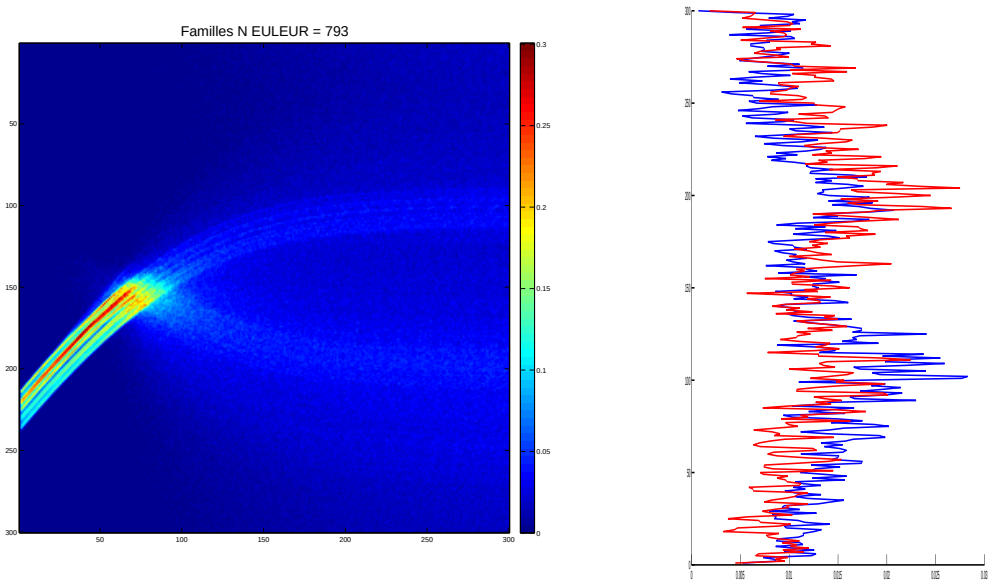


FIG. 23: Cliché Lagrangien de la simulation 2-collision

en sortie (Fig. 24b), le mélange des familles est extrêmement efficace dans le cas de collisions peu intenses. Ici, les figures 27a et 27b nous montrent qu'en cas de collision intenses, les familles sont très peu brassées.

Nous voyons néanmoins que la masse est extrêmement bien répartie après la zone



(a) Famille 2 (b) Familles en sortie

FIG. 24: Distribution des familles en sortie

de collision intense, comme nous le montre la figure 28a. La répartition de masse en sortie est similaire à la répartition que nous avions dans la deuxième simulation (Fig. 25b).

Nous pouvons en conclure que malgré le fait que les jets ne se croisent pas, l'intervention des collisions permet une répartition des quantités de mouvement entre les gouttes de manière plus ou moins aléatoire. Cela explique la très bonne répartition massique que nous avons en sortie de domaine (Fig. 28b).

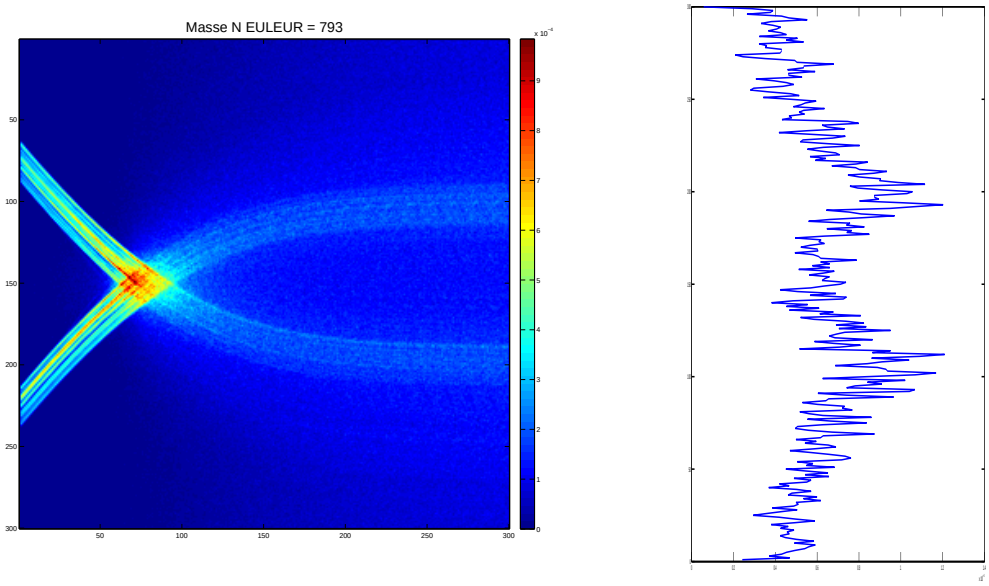
A Annexe A : Formule d'Olinde Rodrigues [1]

Soit le vecteur $\overrightarrow{N}$ ayant une orientation quelconque dans **R** qui sert à exprimer les composantes, alors nous effectuons le raisonnement de la manière suivante :

Définissons le plan Π, orthogonal à $\overrightarrow{N}$. Le vecteur $\overrightarrow{U}$ se décompose en la somme de $(\overrightarrow{U}.\overrightarrow{N})\overrightarrow{N}$, colinéaire à $\overrightarrow{N}$ et invariant par la rotation, et de $\overrightarrow{W} = \overrightarrow{U} - (\overrightarrow{U}.\overrightarrow{N})\overrightarrow{N}$, élément de Π et qui va subir une rotation dans ce plan. Le vecteur directement orthogonal à $\overrightarrow{W}$ dans le plan et de même norme est $\overrightarrow{N} \wedge \overrightarrow{W}$, de sorte que l'image de $\overrightarrow{W}$ dans la rotation d'angle ϕ est $(\cos\phi)\overrightarrow{W} + (\sin\phi)\overrightarrow{N} \wedge \overrightarrow{W}$.

Finalement, l'image de $\overrightarrow{U}$ par la rotation vaut :

$$\overrightarrow{V} = (\overrightarrow{U}.\overrightarrow{N})\overrightarrow{N} + (\cos\phi)\overrightarrow{W} + (\sin\phi)\overrightarrow{N} \wedge \overrightarrow{W}$$



(a) Densité massique (b) Densité massique en sortie

FIG. 25: Densité massique de gouttes dans le domaine

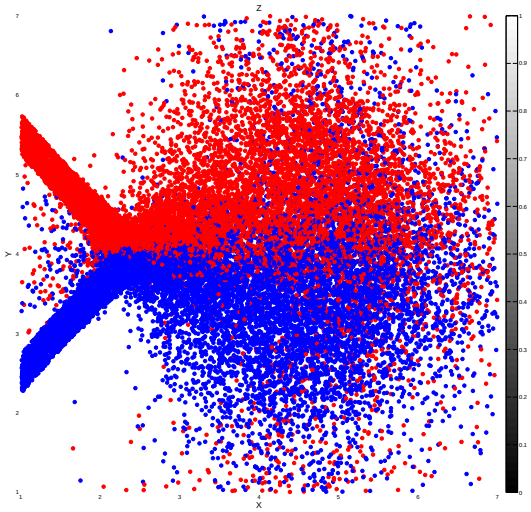
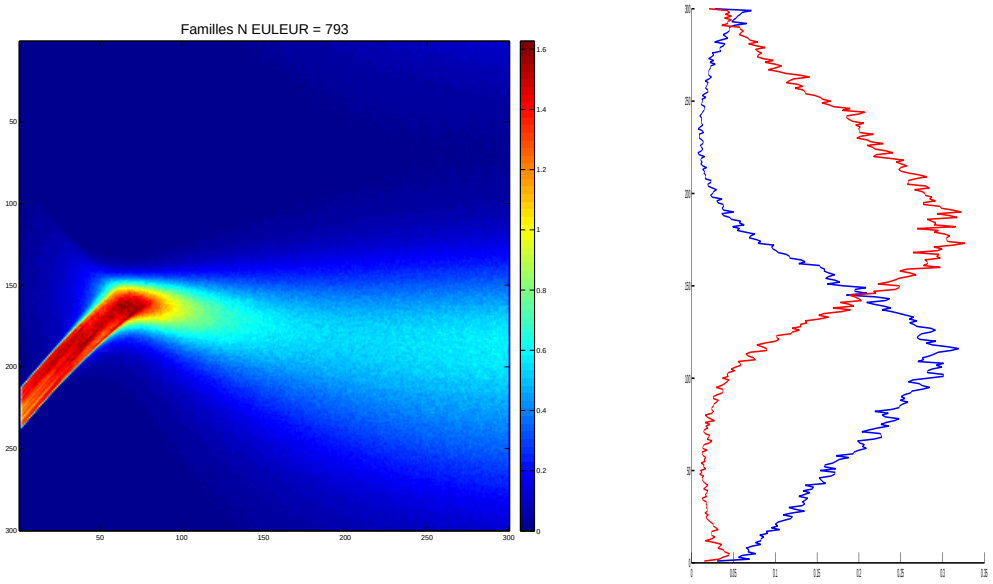


FIG. 26: Cliché Lagrangien de la simulation avec collision

En remplaçant $\overrightarrow{W} = \overrightarrow{U} - (\overrightarrow{U}.\overrightarrow{N})\overrightarrow{N}$, on obtient :

$$\overrightarrow{V} = (\cos\phi)\overrightarrow{U} + (1-\cos\phi)(\overrightarrow{U}.\overrightarrow{N})\overrightarrow{N} + \sin\phi[\overrightarrow{N} \wedge \overrightarrow{U}] \qquad (21)$$



(a) Famille 2 (b) Familles en sortie

FIG. 27: Distribution des familles en sortie

La formule 21 donne l'expression vectorielle du transformé $\overrightarrow{V}$ d'un vecteur $\overrightarrow{U}$ quelconque, dans la rotation $[\phi; \overrightarrow{N}]$ d'angle ϕ et d'axe $\overrightarrow{N}$ normé ($n_x^2 + n_y^2 + n_z^2 = 1$). On peut représenter le même résultat sous forme matricielle équivalente suivante :

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \overline{\overline{M}} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{22}$$

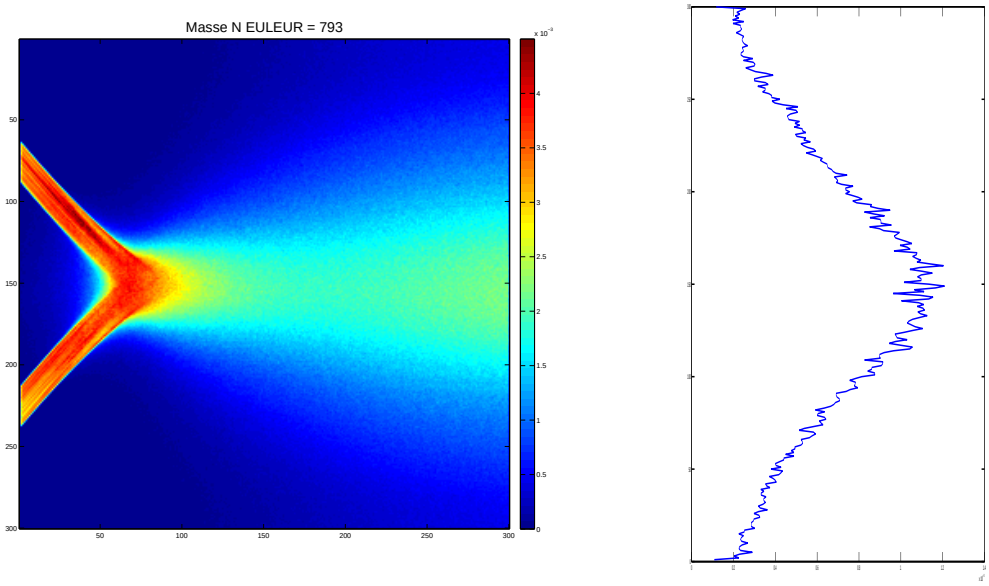
avec

$$\overline{\overline{M}} = \cos(\phi)\overline{I} + (1 - \cos\phi) \begin{bmatrix} n_x^2 & n_x n_y & n_x n_z \\ n_x n_y & n_y^2 & n_y n_z \\ n_x n_z & n_y n_z & n_z^2 \end{bmatrix} + \sin\phi \begin{bmatrix} 0 & -n_z & n_y \\ n_z & 0 & -n_x \\ -n_y & n_x & 0 \end{bmatrix} \tag{23}$$

B Annexe B : Perspectives

Deux optimisations ont été écrites de manière théorique mais n'ont pas été implémentées pour le moment. La première consiste en un système de changement de repère utilisant une matrice de transformation calculable très simplement.

La seconde consiste en une optimisation très interessante qui consiste à décomposer le pas de temps Δt en k-sous pas de temps. Cela permettra – comme nous le verrons par la suite – de réduire significativement la surface à explorer. Cela est possible grace au système de tri qui a été mis en place. Dans le cas normal d'un tri en classement dans une matrice, la complexité est de N tandis que notre méthode



(a) Densité massique (b) Densité massique en sortie

FIG. 28: Densité massique de gouttes dans le domaine

de tri est en $N\log(N)$. Néanmoins nous avons vu que cette méthode de tri a permi des optimisations impossible à faire par la méthode précédente, et la méthode de décomposition en k-sous-pas de temps en fait partie.

B.1 Système de changement de coordonnées globalisé

Comme nous l'avons vu dans la gestion de collision en 2 et 3D, nous changeons deux fois de repère. Une fois pour se placer dans un repère spécifique à la collision, où un axe privilégié permet de dire que la vitesse des 2 particules sur cet axe restent constante, puis une seconde fois pour repasser le système de collision dans son repère d'origine. Ces changements de repère sont lourds, tant du point de vue syntaxique que du point de vue temps calcul, aussi nous nous proposons ici de définir une méthode complète et simple à mettre en oeuvre pour effectuer ces changements de repère rapidement, que ce soit en 2D ou en 3D.

Tout d'abord nous allons développer cette méthode pour le cas 3D, le passage en 2D se fera de manière naturelle.

B.1.1 Méthode de changement de référentiel généralisé en 3D

Nous avons vu que le choix d'un repère spécifique à la collision est restreint par plusieurs critères. Il faut un axe sur lequel les vitesses restent inchangées (un axe qui reste parallèle à la surface de chaque sphère au temps propre de la collision τ) ainsi qu'un axe passant par le centre de ces 2 sphères.

Nous nous proposons ici de trouver une méthode plus rapide pour déterminer la matrice de changement de repère utilisée dans les équations 10 et 12. Nous allons partir des même bases mais allons utiliser une méthode radicalement différente.

Le vecteur passant par le centre de ces 2 sphères est définit simplement dans la section 4.3 :

$$\overrightarrow{i}_{\mathbf{R}} = \frac{(\overrightarrow{X}_R + \tau\overrightarrow{V}_R) - (\overrightarrow{X}_S + \tau\overrightarrow{V}_S)}{R_R + R_S} = \begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix}$$

Il s'agit ici d'effectuer une rotation du repère local de collision $(\overrightarrow{i}, \overrightarrow{j}, \overrightarrow{k})$ afin de passer au repère d'origine $(\overrightarrow{x}, \overrightarrow{y}, \overrightarrow{z})$. Cette rotation se fera autour d'un vecteur $\overrightarrow{N}$ qui doit être perpendiculaire à $\overrightarrow{x}$ et à $\overrightarrow{i}$. Nous définissons donc $\overrightarrow{N_{\mathbf{R}}} = \frac{\overrightarrow{x_{\mathbf{R}}} \wedge \overrightarrow{i_{\mathbf{R}}}}{||\overrightarrow{x_{\mathbf{R}}} \wedge \overrightarrow{i_{\mathbf{R}}}||}$.

De plus, on connaît $\overrightarrow{x_{\mathbf{R}}} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$ ainsi que $\overrightarrow{i_{\mathbf{R}}} = \begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix}$.

On obtient :

$$\overrightarrow{N_{\mathbf{R}}} = \frac{\begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} \wedge \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}}{\|\overrightarrow{x_{\mathbf{R}}} \wedge \overrightarrow{i_{\mathbf{R}}}\|} = \frac{1}{\sqrt{y_i^2 + z_i^2}} \begin{bmatrix} 0 \\ -z_i \\ y_i \end{bmatrix} \tag{24}$$

Pour obtenir ϕ, on sait que :

$$\begin{cases} \cos(\phi) = \frac{\overrightarrow{x}.\overrightarrow{i}}{\|\overrightarrow{x}\|\|\overrightarrow{i}\|} \\ \sin(\phi) = \frac{\|\overrightarrow{x} \wedge \overrightarrow{i}\|}{\|\overrightarrow{x}\|\|\overrightarrow{i}\|} \end{cases} \tag{25}$$

D'où

$$\phi = \cos^{-1}\left(\frac{\overrightarrow{x}.\overrightarrow{i}}{\|\overrightarrow{x}\|\|\overrightarrow{i}\|}\right) = \cos^{-1}\left(\overrightarrow{x}.\overrightarrow{i}\right) = \cos^{-1}(x_i) \tag{26}$$

Maintenant que nous avons le vecteur normal à la rotation $\overrightarrow{N}$ ainsi que l'angle de rotation définit par ϕ, la technique va consister à, non pas définir les vecteurs dans le nouveau repère - mais faire tourner les vecteurs d'un angle $-\phi$ autour de $\overrightarrow{N}$ pour retrouver la collision dans le repère $\mathbf{R}$. Pour ce-faire, nous allons utiliser la formule d'Olinde Rodrigues [1] (voir Annexe A). Ensuite, nous effectuerons l'opération inverse (c'est à dire une rotation d'angle $+\phi$ autour du vecteur $\overrightarrow{N}$).

On notera $\overrightarrow{U_{\pm\phi}}$ les vecteurs ayant subit une rotation $[\pm\phi; \overrightarrow{N}]$.

Le repère $(\overrightarrow{i}, \overrightarrow{j}, \overrightarrow{k})$ est défini de la manière suivante : $\overrightarrow{i}$ est le vecteur porteur de la droite passant par le centre des 2 sphères. C'est l'axe sur lequel nous devront effectuer les équations de conservation de mouvement et d'énergie. $\overrightarrow{j}$ est un vecteur parallèle à la surface des 2 sphères au point de contact de celles-ci. Ici, on prendra $\overrightarrow{j} = \overrightarrow{N}$. La rotation se faisant sur $\overrightarrow{N}$, ce vecteur est un invariant de la transformation $[\pm\phi; \overrightarrow{N}]$. Enfin, $\overrightarrow{k} = \overrightarrow{i} \wedge \overrightarrow{j}$ afin que le repère $(\overrightarrow{i}, \overrightarrow{j}, \overrightarrow{k})$ constitue une base orthonormée directe.

En résumé nous avons $\overrightarrow{i} = \begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix}$, $\overrightarrow{j} = \overrightarrow{N} = \frac{1}{\sqrt{y_i^2+z_i^2}} \begin{bmatrix} 0 \\ -z_i \\ y_i \end{bmatrix}$ et $\overrightarrow{k} = \frac{1}{\sqrt{y_i^2+z_i^2}} \begin{bmatrix} y_i^2+z_i^2 \\ -x_i y_i \\ -z_i x_i \end{bmatrix}$.

Nous pouvons en déduire immédiatement la matrice transformation $[-\phi; \overrightarrow{N}]$ grâce à la formule de Rodrigues :

$$\overline{\overline{M_{-\phi}}} = \cos(-\phi)\overline{I} + \frac{1-(\cos-\phi)}{y_i^2+z_i^2} \begin{bmatrix} 0 & 0 & 0 \\ 0 & z_i^2 & -y_i z_i \\ 0 & -y_i z_i & y_i^2 \end{bmatrix} + \frac{(\sin-\phi)}{\sqrt{y_i^2+z_i^2}} \begin{bmatrix} 0 & -y_i & -z_i \\ y_i & 0 & 0 \\ z_i & 0 & 0 \end{bmatrix}$$

D'après la définition de ϕ que nous avions dans l'équation 25, nous obtenons

$$\overline{\overline{M_{-\phi}}} = x_i\overline{I} + \frac{1-x_i}{y_i^2+z_i^2} \begin{bmatrix} 0 & 0 & 0 \\ 0 & z_i^2 & -y_i z_i \\ 0 & -y_i z_i & y_i^2 \end{bmatrix} + \frac{-\sqrt{1-x_i^2}}{\sqrt{y_i^2+z_i^2}} \begin{bmatrix} 0 & -y_i & -z_i \\ y_i & 0 & 0 \\ z_i & 0 & 0 \end{bmatrix}$$

Et comme $\overrightarrow{i}$ est normalisé, nous avons $x_i^2+y_i^2+z_i^2 = 1 \rightarrow y_i^2+z_i^2 = 1-x_i^2 = (1+x_i)(1-x_i)$ Nous obtenons donc l'équation 27.

$$\overline{\overline{M_{-\phi}}} = \begin{bmatrix} x_i & y_i & z_i \\ -y_i & x_i + \frac{z_i^2}{1+x_i} & \frac{-y_i z_i}{1+x_i} \\ -z_i & \frac{-y_i z_i}{1+x_i} & x_i + \frac{y_i^2}{1+x_i} \end{bmatrix} \quad (27)$$

Par la transformation $[-\phi; \overrightarrow{N}]$ dans $\mathbf{R}$, nous pouvons facilement vérifier que $\overrightarrow{i} \rightarrow \overrightarrow{x}$, $\overrightarrow{j} \rightarrow \overrightarrow{j}$ – car $\overrightarrow{j} = \overrightarrow{N}$ et $\overrightarrow{N}$ est un invariant de la transformation $[-\phi, \overrightarrow{N}]$ – et $\overrightarrow{k} \rightarrow \overline{\overline{M}}\overrightarrow{k}$, ce qui correspond aux hypothèses que nous avons émises au début.

Pour rendre compte de la collision, la méthode est donc la suivante :

Lorsqu'une collision est détectée, on détermine $\overrightarrow{i}_{\mathbf{R}} = \frac{(\vec{X}_R+\tau\vec{V}_R)-(\vec{X}_S+\tau\vec{V}_S)}{R_R+R_S}$.

Une fois $\overrightarrow{i}$ déterminé, on calcule $\overline{\overline{M}}$. Tout l'intérêt de cette méthode est la disparition complète de tout calcul trigonométrique pour le changement de base, qui ont la désagréable propriété d'être très long à calculer et qui par conséquent entraîne un ralentissement significatif du code. De plus un produit matriciel est facilement parallélisable. Nous pouvons même anticiper sur les futures architectures processeur en cours de développement (basées sur les processeurs graphiques actuels), optimisés sur les multiplications de matrices et les intégrations. Puis, on détermine $\overrightarrow{V_{R,-\phi}} = \overline{\overline{M}}\overrightarrow{V_R}$ et $\overrightarrow{V_{S,-\phi}} = \overline{\overline{M}}\overrightarrow{V_S}$. Cela permet l'obtention d'une collision dans le repère $\mathbf{R}$ ayant exactement les mêmes caractéristiques que la collision originale dans le repère $\mathbf{R}'$. C'est à dire :

– une vitesse des gouttes invariantes sur l'axe $\overrightarrow{y}$ et l'axe $\overrightarrow{z}$

– régie par la conservation de la quantité de mouvement et d'énergie sur $\overrightarrow{x}$

On détermine $P_{i,-\phi}$ par l'équation 4, ce qui permet de déduire $v'_{R,i,-\phi}$ par l'équation 7, puis $v'_{S,i,-\phi}$ par l'équation 9.

Enfin, on retourne dans le repère local :

$$\overrightarrow{v'}_{R,-\phi+\phi} = \overrightarrow{v'}_{R,\mathbf{R}} = \overline{\overline{{}^{T}M}}\,\overrightarrow{v'}_{R,-\phi} \quad ; \quad \overrightarrow{v'}_{S,-\phi+\phi} = \overrightarrow{v'}_{S,\mathbf{R}} = \overline{\overline{{}^{T}M}}\,\overrightarrow{v'}_{S,-\phi} \tag{28}$$

L'unique restriction que l'on a est que si $\overrightarrow{i} = \pm\overrightarrow{x}$, alors on a $x_i = \pm 1$.

Or, comme $\overrightarrow{i}$ est normalisé, nous avons $y_i = 0$ et $z_i = 0$. Par conséquent le vecteur $\overrightarrow{N} = \frac{1}{\sqrt{y_i^2+z_i^2}} \begin{bmatrix} 0 \\ -z_i \\ y_i \end{bmatrix}$ ne serai pas définit. Cela n'est pas bien grave, car en cas de colinéarité la rotation à effectuer est d'angle nul, et par conséquent tout vecteur dans $\mathbf{R}$ aura les mêmes composantes dans $\mathbf{R}'$. La matrice changement de repère serai donc $\overline{\overline{M}} = \bar{I}$.

B.1.2 Passage en 2D

Maintenant que nous avons la méthode pour effectuer ce changement de repère immédiatement en 3D, la 2D en sera d'autant plus simple. En effet, nous savons que $\overrightarrow{i} \in (\Pi)$ et que la rotation se fera dans le plan, donc que $\overrightarrow{N} \perp (\Pi)$. Donc $z_i = 0$, $n_x = 0$, $n_y = 0$, $n_z = \pm 1$. En appliquant toutes ces simplifications, on déduit : $\overline{\overline{M}} = \begin{bmatrix} x_i & y_i & 0 \\ -y_i & x_i & 0 \\ 0 & 0 & 1 \end{bmatrix}$.

La composante sur z devient de ce fait une composante totalement indépendante, ce qui peut paraître logique étant donné que la simulation se fait dans le plan $\Pi(Ox, Oy)$.

La méthode de prise en charge de la collision est exactement la même que celle décrite précédemment.

B.2 Décomposition du pas de temps en k-sous pas de temps

B.2.1 Principe et théorie

Cette optimisation est probablement la plus efficace comme nous allons le voir.

Le principe que nous avons trouvé ici est simple. Nous savons que dans le cas idéal, la surface à explorer (en 2D donc) est la surface que remplissent les gouttes durant l'intervalle de temps Δt. Celui ci (pour une particule quelconque R) est donné par

$$S_{ideale,R} = 2\left(R_R + V_R\Delta t\right)\left(2R_R\right) = 4R_R\left(R_R + V_R\Delta t\right)$$

Ceci est malheureusement mathématiquement infaisable (ou tout du moins fort complexe), car détecter parmi N équations de surfaces (surface occupée par les gouttes pendant l'intervalle de temps Δt) lesquelles ont une intersection non nulle se révélerai plus complexe et plus long que la méthode utilisée ici. La surface que nous explorions initialement (et généralement explorée dans la littérature), avant les différentes

optimisations est indépendante des caractéristique de la goutte et est

$$S = 2 * (2V_{max}\Delta t + 2R_{max})^2$$

Le cas optimisé utilisé actuellement, pour une particule quelconque R, décrit dans le paragraphe 5.1, équation 17 a une surface $S_{optim,R} = ((V_{R,x} - V_{x,max})\Delta t + R_R + R_{max}) * ((V_{max,y,+} - V_{max,y,-})\Delta t + 2(R_R + R_{max})$, décrite par la figure 12a.

Si le pas de temps est décomposé en $\frac{\Delta t}{k}$, alors nous allons tenter d'exprimer la surface à explorer. Retenons bien ici que le nombre de couples à tester décroît comme le carré de la diminution de surface. Pour une surface divisée par 2 nous aurons 4 N opérations en moins à faire. C'est la raison pour laquelle les optimisations portent globalement sur la minimisation de cette surface.

Si nous divisons le pas de temps principal Δt en k-sous pas de temps $\frac{\Delta t}{k}$, alors nous aurons k fois la surface à explorer. La surface ainsi décomposée est la surface optimisée décrite dans le paragraphe 5.1. Les distances maximales d'interaction deviennent donc :

$$\begin{cases} D_{k-it,x,max} = (V_{R,x} - V_{x,max})\frac{\Delta t}{k} + R_R + R_{max} \\ D_{k-it,max,y,1} = (V_{R,y} - V_{max,y,-})\frac{\Delta t}{k} + R_R + R_{max} \\ D_{k-it,max,y,2} = (V_{max,y,+} - V_{R,y})\frac{\Delta t}{k} + R_R + R_{max} \end{cases} \tag{29}$$

La surface totale à explorer par pas de temps pour une particule quelconque R est donc :

$$S_{k-it}(k) = \sum_k \left((V_{R,x} - V_{x,max})\frac{\Delta t}{k} + R_R + R_{max} \right) * \left((V_{max,y,+} - V_{max,y,-})\frac{\Delta t}{k} + 2(R_R + R_{max}) \right)$$

Afin d'estimer la surface à explorer ainsi définie, on peut s'affranchir de la somme en multipliant par k car s'il n'y a pas de collision (grande majorité des cas), la vitesse de la goutte reste constante. Les surfaces restent donc égales pour chaque sous-pas de temps. L'estimation de cette surface permet uniquement de trouver une condition de détermination de k.

$$S_{k-it}(k) = k \left((V_{R,x} - V_{x,max})\frac{\Delta t}{k} + R_R + R_{max} \right) * \left((V_{max,y,+} - V_{max,y,-})\frac{\Delta t}{k} + 2(R_R + R_{max}) \right) \tag{30}$$

Cette surface est symbolisée par la figure 29. Nous voyons par cette figure que

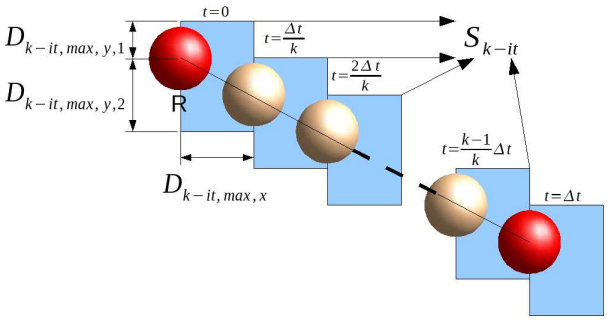


FIG. 29: Surface des k-sous pas de temps

nous nous rapprochons grandement de la surface idéale à explorer, symbolisée par la figure 1.

Le but va être de déterminer k afin que la surface à explorer soit minimale. Pour cela, il faut que k satisfasse ces 2 conditions :

$$\frac{\delta S_{k-it}(k)}{\delta k} = 0 \quad et \quad \frac{\delta^2 S_{k-it}(k)}{\delta k^2} > 0$$

Nous avons

$$\begin{aligned} S_{k-it}(k) &= k((V_{R,x} - V_{x,max})\frac{\Delta t}{k} + (R_R + R_{max})) * ((V_{max,y,+} - V_{max,y,-})\frac{\Delta t}{k} + 2(R_R + R_{max})) \\ &= (V_{R,x} - V_{x,max})(V_{max,y,+} - V_{max,y,-})\frac{\Delta t^2}{k} + \Delta t(R_R + R_{max}) * \\ &(2(V_{R,x} - V_{x,max}) + V_{max,y,+} - V_{max,y,-}) + 2k(R_R + R_{max})^2 \end{aligned} \tag{31}$$

Il nous faut donc trouver k tel que

$$\frac{\delta S_{k-it}(k)}{\delta k} = 2(R_R + R_{max})^2 - \frac{1}{k^2}((V_{R,x} - V_{x,max})(V_{max,y,+} - V_{max,y,-})\Delta t^2) = 0$$

Nous obtenons donc :

$$k = \left[\frac{\Delta t}{R_R + R_{max}}\sqrt{\frac{(V_{R,x} - V_{x,max})(V_{max,y,+} - V_{max,y,-})}{2}}\right] \tag{32}$$

k devant être un entier, on prendra la partie entière de ce résultat (notée [ ]).

Vérifions que c'est bien un minimum (et non pas un maximum) :

$$\begin{aligned} \frac{\delta^2 S_{k-it}(k)}{\delta k^2} &= -\frac{\delta\left(-\frac{1}{k^2}\right)}{\delta k}((V_{R,x} - V_{x,max})(V_{max,y,+} - V_{max,y,-})\Delta t^2) \\ &= \frac{2}{k^3}((V_{R,x} - V_{x,max})(V_{max,y,+} - V_{max,y,-})\Delta t^2) \end{aligned} \tag{33}$$

De part la définition de $V_{x,max}$, $V_{max,y,+}$ et $V_{max,y,-}$, données par la formule 13, on peut en conclure que :

$$\begin{aligned} &V_{R,x} \geq V_{x,max} \quad \forall R \Rightarrow V_{R,x} - V_{x,max} \geq 0 \\ &V_{max,y,+} \geq V_{max,y,-} \quad \forall R \Rightarrow V_{max,y,+} - V_{max,y,-} \geq 0 \end{aligned}$$

k est positif, ainsi que $V_{R,x} - V_{x,max}$ et $V_{max,y,+} - V_{max,y,-}$, donc $\frac{\delta^2 S_{k-it}(k)}{\delta k^2} > 0 \quad \forall k$. Il s'agit donc bien d'un minimum.

Cependant, chaque avancement en temps $\frac{\Delta t}{k}$ des particules doit être égal pour chaque particule. En effet, on fait avancer par une méthode du premier ordre la position des particules, il faut donc que cet avancement soit le même pour chaque particule afin que la recherche des collision ait un sens. Il est donc impossible de définir k selon une caractéristique spécifique à une seule particule, il faut donc prendre une valeur qui – globalement – nous donnera le k optimum. Or, d'après l'équation 32, nous voyons que k est définit selon une valeur spécifique de chaque particule :

$V_{R,x}$. Il va donc falloir trouver une valeur de substitution de $V_{R,x}$ et qui restera égal pour chaque particule, tout en essayant de garder la rapidité de cette méthode.

On sait que la surface à explorer pendant le k-sous pas de temps est inversement proportionnelle à k. Le nombre de couples à calculer (qui, rappelons-le, est le plus long) est donc proportionnel à $\left(\frac{1}{k}\right)^2$. Nous prendrons donc une valeur moyenne pour $V_{R,x}$, qui sera égale pour toutes les particules, et est donné par l'équation 34.

$$\widetilde{V_x} = \left(\widetilde{\sqrt{V_{R,x}}}\right)^2 = \left(\frac{1}{N}\sum_{i=1}^{N}\sqrt{V_{i,x}}\right)^2 \tag{34}$$

Nous obtenons donc la valeur de k pour l'intégralité du pas de temps, donné par l'équation 35.

$$k = \left[\frac{\Delta t}{R_R + R_{max}}\sqrt{\frac{(\widetilde{V_x} - V_{x,max})(V_{max,y,+} - V_{max,y,-})}{2}}\right] \tag{35}$$

Maintenant que nous connaissons la valeur de k pour que S_{k-it} soit minimale, nous pouvons alors exprimer cette surface en injectant la valeur de k donnée par l'équation 32 dans l'équation 31.

Nous obtenons donc la surface la plus petite atteignable par cette méthode :

$$\begin{aligned} S_{k-it} =& (\widetilde{V_x} - V_{x,max})(V_{max,y,+} - V_{max,y,-})\frac{\Delta t^2}{\frac{\Delta t}{R_R+R_{max}}\sqrt{\frac{(\widetilde{V_x}-V_{x,max})(V_{max,y,+}-V_{max,y,-})}{2}}} + \\ & \Delta t(R_R + R_{max}) * (2(\widetilde{V_x} - V_{x,max}) + V_{max,y,+} - V_{max,y,-}) + \\ & 2\frac{\Delta t}{R_R + R_{max}}\sqrt{\frac{(\widetilde{V_x} - V_{x,max})(V_{max,y,+} - V_{max,y,-})}{2}}(R_R + R_{max})^2 \end{aligned} \tag{36}$$

Qui donne après simplifications :

$$S_{k-it} = (R_R + R_{max})\Delta t\left(\sqrt{2(\widetilde{V_x} - V_{x,max})} + \sqrt{V_{max,y,+} - V_{max,y,-}}\right)^2 \tag{37}$$

Afin comparer la surface de cette méthode avec la surface que nous avions auparavant, une rapide justification s'impose. La surface que nous avions est donnée dans le paragraphe 5.1, équation 17 :

$$S_{optim}(\Delta t) = ((V_{R,x} - V_{x,max})\Delta t + R_R + R_{max}) * ((V_{max,y,+} - V_{max,y,-})\Delta t + 2(R_R + R_{max}) \tag{38}$$

Or, on constate que $\frac{\delta S_{optim}(\Delta t)}{\delta \Delta t} > 0$, ainsi que $\frac{\delta^2 S_{optim}(\Delta t)}{\delta \Delta t^2} > 0$. Par conséquent, sachant que $S_{optim}(\Delta t)$ est continue, on a

$$k \cdot S_{optim}\left(\frac{\Delta t}{k}\right) = S_{k-it}(\Delta t, k) < S_{optim}(\Delta t) \tag{39}$$

On sait que $\frac{\delta S(\Delta t)}{\delta \Delta t} > 0$ Cette surface est très inférieure à la surface à explorer trouvée dans la bibliographie ou la surface optimisée que nous utilisions précédemment donnée par l'équation 17.

B.2.2 Avantages et inconvénients de cette méthode

Un des principaux avantages de cette méthode est une très grande optimisation de la vitesse. Néanmoins, elle nécessite quelques précautions.

En effet, la décomposition en k-sous-pas de temps sous-entend un avancement de toutes les particules à chaque sous-pas de temps.

La complexité de cet avancement est de N, ce qui nous donne k*N calculs supplémentaires à effectuer. Certes pour de petites valeurs de k ceci est amplement négligeable au vu du temps gagné, mais pour de grandes valeurs de k, cette étape peut devenir rapidement imposante.

De plus, étant donné que nous effectuons une modification de la position des particules à chaque sous pas de temps, il faut reclasser leurs abscisses. Néanmoins ce classement sera très proche d'une complexité d'ordre N. En effet, comme nous l'avons vu dans le chapitre 4.5.1, si la liste est "pratiquement" classée, le classement est d'ordre N. Ici, nous allons classer les abscisses une fois, effectuer l'avancement en temps des positions, puis reclasser etc. L'abscisse des gouttes sera peu modifié, à fortiori leur classement aussi. La liste triée après un avancement en temps sera donc quasiment – si ce n'est complètement – triée. Le reclassement sera donc très rapide. Un autre avantage majeur est la possibilité de capturer plusieurs collisions par pas de temps Δt. En effet, puisque l'on fait un avancement tous les $\Delta t/k$, alors nous pouvons voir pour une seule particule la capture de k collisions. Cela permet donc une plus grande précision dans la gestion de ces collisions.

Références

[1] Wikipedia. Rotation vectorielle. *http ://fr.wikipedia.org/wiki/Rotation vectorielle*.

[2] C.D. Rakopoulos and G.C. Mavropoulos. Experimental evaluation of local instantaneous heat transfer characteristics in the combustion chamber of air-cooled direct injection diesel engine. *Energy*, 2007.

[3] PIALAT Xavier. *Développement d'une méthode hybride eulérienne-lagrangienne pour la modélisation numérique de la phase particulaire dans les écoulements turbulents gaz-particules*. PhD thesis, Ecole Nationnale Supérieure de l'Aéronautique et de l'Espace, 2007.

[4] Ho Y. Kim Sam S. Yoon and John C. Hewson. Effect of initial conditions of modeled pdfs on droplet characteristics for coalescing and evaporating turbulent water spray used in fire suppression applications. *Fire Safety Journal*, 2007.

[5] Stéphane Pascaud. *Vers la simulation aux grandes échelles des écoulements turbulents diphasiques réactifs : application aux foyers aéronautiques*. PhD thesis, Institut National Polytechnique de Toulouse.

[6] M. Massot J. Reveillon S. de Chaisemartin, F. Laurent. Evaluation of eulerian multi-fluid versus lagrangian methods for ejection of polydisperse evaporating sprays by vortices.

[7] Marc Massot Frédérique Laurent and Philippe Villedieu. Eulerian multi-fluid modeling for the numerical simulation of coalescence in polydisperse dense liquid sprays. *Journal of Computational Physics*, 2004.

[8] Andrew Stuart Hersir Sigurgeirsson and Wing-Lok Wan. Algorithms for particle-field simulations with collisions. *Journal of Computational Physics*, 2001.

[9] John M. Barbara and Larry W. Esposito. Moonlet collisions and the effects of tidally modified accretion in saturn's ring. *Icarus*, 2002.

[10] K.Kontomaris M. Chen and J.B. McLaughlin. Direct numerical simulation of droplet collision in a turbulent channel flow. part I : collision algorithm. *International Journal of Multiphase Flow*, 1998.

[11] Shivshankar Sundaram and Lance R. Collins. Numerical considerations in simulating a turbulent suspension of finite-volume particles. *journal of Computational Physics*, 1996.

[12] David P. Schmidt and Christopher J. Rutland. Numerical issues in droplet collision modeling.

[13] Jouke Jan Hylkema. *Modélisation cinétique et simulation numérique d'un brouillard dense de gouttelettes*. PhD thesis, Université Pierre et Marie Curie, 1999.

[14] T.A.G. Langrish and K. Kota. A comparison of collision kernels for sprays from one and two-nozzle atomisation systems. *Chemical Engineering Journal*, 2006.

[15] David P. Schmidt and C.J. Rutland. A new droplet collision algorithm. *Journal of Computational Physics*, 2000.

[16] Martin Sommerfeld Clayton Crowe and Yutaka Tsuji. *Multiphase Flows with droplets and particles*. 1998.

[17] Martin Sommerfeld. Validation of a stochastic lagrangian modelling approach for inter-particle collisions in homogeneous isotropic turbulence. *International Journal of Multiphase Flow*, 2001.

[18] A. Frohn M. Rieber. Three-dimensional navier-stokes simulation of binary collisions between droplets of equal size. *Journal of Aerosol Science*, 1995.

[19] M. Schelke and A. Frohn. Three-dimensional lattice boltzmann simulations of binary collisions between equal droplets. *Journal of Aerosol Science*, 1995.

[20] Gwon Hyun Ko and Hong Sun Ryou. Droplet collision processes in an interspary impingement system. *Aerosol Science*, 2005.

[21] Sebastien Tanguy and Alain Berlemont. Application of a level set method for simulation of droplets collisions. *International Journal of Multiphase Flow*, 2005.

[22] Numerical Recipies. Tri rapide.