Aix-Marseille Université
Faculté des sciences, UNIMECA
Master 1 Mécanique

Rapport de Stage TER

Professeur Encadrant

Prof. P. Vigliano

Auteur

Aamr ABDELHADI
Ahmed EL JAAFARI
Aiman BELHAJ

Année 2019/2020

1 Remerciements

En premier lieu, on tient à remercier notre professeur, Mr VIGLIANO, tuteur du stage, pour nous avoir permis d'effectuer notre stage et travailler avec lui. Il a eu la patience et la pédagogie pour nous accompagner durant toute la période de notre stage.

On remercie ensuite très vivement notre professeur, Mr CANDELIER, pour nous avoir encadré et proposé plusieurs sujets de stage.

Et on remercie finalement, tous les professeurs qui nous ont encadrés durant toute cette année scolaire.

Contents

2 Introduction

2.1 Introduction générale

La mécanique des fluides et la cinétique sont des sciences qui concernent l'étude de la conduite des fluides à l'état statique et dynamique. Avec ces sciences on peut trouver plusieurs applications dans les domaines tels que l'hydraulique, l'aérodynamique, la thermique... etc.

Dans la réalité, l'écoulement de fluide autour d'un obstacle (cylindres) est un phénomène très courant. Il est très important dans plusieurs domaines que cela soit de la conception mécanique ou thermique de plusieurs systèmes en industrie d'ingénierie comme par exemple : les bâtiments, les automobiles, les aubes de turbines et les formes géométriques à section circulaire.

L'assimilation et l'étude des cas hydrodynamiques et aérodynamiques qui accourent dans le sillage des obstacles sont des sujets d'actualité dans plusieurs domaines.

Le cylindre à section circulaire est le modèle de base des corps non profilés d'après des recherche effectués et il est devenu la configuration conforme pour étudier de tels écoulements externes. L'écoulement atour d'un cylindre est un problème de la dynamique des fluides et agit de milieu à la validation de nouvelles méthodes numériques.

On note que ce problème académique a provoqué un regain d'intérêt durant les dernières années expérimentalement ou numériquement, dû à l'apparition de nouvelles méthodes de résolutions des équations de la dynamique.

La dynamique des fluides computationnelle CFD qui repose sur les équations à l'échelle macroscopique discrétisées et résolues via : les différences finies, les éléments finis ou les volumes finis, n'est pas vraiment adaptée pour modéliser les écoulements multi-fluides, notamment quand l'interface est soumise à des changements de topologie.

Quant à la dynamique moléculaire, elle simule l'écoulement en suivant le mouvement de chacune des particules du fluide. Puisqu'il faut un grand nombre de molécules pour 5 retrouver le comportement de l'écoulement à l'échelle macroscopique, cette dernière s'avère trop coûteuse en terme de temps de calcul.

Dans notre cas on va utiliser un modèle numérique simplifié basé sur la méthode " Lattice Boltzman Methode " (LBM) en configuration D2Q9 pour qu'on puisse faire la modélisation de l'écoulement bi-dimensionnel (2D) :

Tout d'abord autour d'un cylindre de section circulaire puis à travers une

grille simulée par 5 cylindres en choisissant des Reynolds de compris entre 50 et 200.

Ces méthodes (LBM), sont basées en fait sur les équations à l'échelle mésoscopique. De nombreux avantages découlent de cette caractéristique : on obtient la pression à l'aide d'une équation d'état et non pas en résolvant l'équation de Poisson. Ces méthodes permettent également d'intégrer dans le modèle toute sorte d'intéractions, afin de simuler des fluides complexes.

NB : Cette méthode est fondée sur la théorie cinétique des gaz.

2.2 Objectif

Le sujet de notre stage est ainsi un écoulement à travers une grille.

Notre travail consiste en plusieurs étapes :

- programmer la simulation en Python.
- représenter notre écoulement.
- Interpréter finalement les résultats obtenus.

L'idée consiste à utiliser un modèle simplifié : du 2D (la méthode D2 Q9 est une méthode bidimensionnelle D2 avec 9 directions de propagations Q9).

Pour les réseaux LBM, la dénomination DnQm est généralement utilisée. Un réseau DnQm signifie que le réseau a n dimensions spatiales et m vitesses discrètes.

La méthode numérique en quelques sorte résout une forme de l'équation de Boltzmann en lieu et place de Navier-Stokes et le calcul des moments statistiques de la fonction de distribution des vitesses permet de revenir aux grandeurs macroscopiques. L'intérêt de cette résolution est d'être hautement parallélisable donc rapide, et très flexible (adaptée aux changements de conditions aux limites, parois mobiles etc..).

Le but de notre rapport est de bien comprendre cette autre vision de la mécanique des fluides ainsi que la méthode de calcul qui en découle.

L'avantage d'un modèle 2D est d'être vraiment rapide et de permettre de bien comprendre la méthode " LBM ". Les généralisations à 3D avec plus ou moins de directions de vitesses (19 ou 27 pour les modèles D3Q19 ou D3Q27) seront en effet plus facilement accessibles.

Le but au final est de bien comprendre le contexte et la méthode qu'on a utilisée.

3 Ecoulement autour d'un cylindre

L'écoulement autour d'un cylindre est un phénomène qui fait partie de quelques problèmes classiques de la mécanique des fluides, pour lesquelles les équations de Navier Stokes accordent une solution. Ce type d'écoulement affecte un caractère pratique, vu qu'il se rencontre dans plusieurs domaines soit en aérodynamique, en hydrodynamique etc.

Ce dernier à cause de son aspect pratique est rencontré dans plusieurs domaines de l'ingénierie mécanique, aussi bien, que dans plusieurs applications technologiques d'intérêt comme les pipelines, les structures maritimes et les échangeurs de chaleur.

L'étude des écoulements laminaires et turbulents autour d'un cylindre de section circulaire a fait l'objet de nombreuses études expérimentales et numériques. Le champs d'écoulement autour du cylindre circulaire est symétrique à de faibles valeurs du nombre de Reynolds. En effet, à bas Reynolds on a une solution analytique exacte en fluide visqueux et de nombreux travaux expérimentaux permettant de comparer les résultats des méthodes numériques et donc de pouvoir valider les calculs numériques.

Comme le nombre de Reynolds augmente, l'écoulement commence à se détacher derrière le cylindre en provoquant le détachement des tourbillons, alors que l 'écoulement devient instationnaire. Dans de telles situations, la décélération de l'écoulement, provoquée par le contact du liquide avec l'obstacle solide, donne lieu à un certain nombre de phénomènes complexes ; dans certaines circonstances, on remarque la formation d'un sillage turbulent.

Du point de vue phénoménologique, quand un corps se déplace dans un fluide au repos ou, de façon équivalente, lorsqu'un fluide s'écoule autour d'un corps solide, une force sur le corps apparaît.

Comme le montre la figure 1, la projection de cette force dans le sens perpendiculaire à l'écoulement est appelée le support (FL), et la composante parallèle à l'écoulement libre est appelée traînée (FD). Ces dernières peuvent être calculées par les expressions:

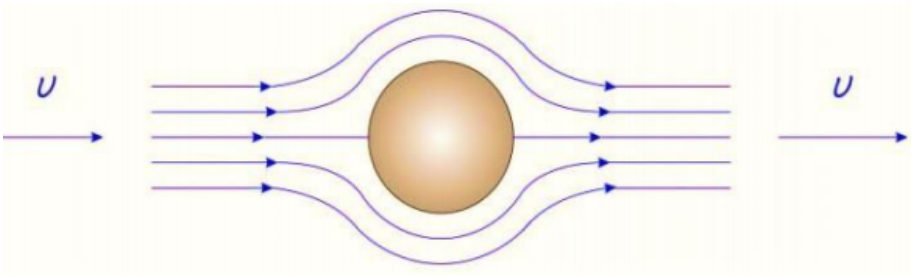


Figure 1: Écoulement autour un cylindre.

$$F_D = C_D \frac{1}{2} \rho U^2 A \tag{1}$$

$$F_L = C_L \frac{1}{2} \rho U^2 A \tag{2}$$

Où :
ρ : La masse du fluide,
A : Surface de la projection,
C_D : Le coefficient de la traînée
C_L : Le coefficient de portance.

NB : La force de traînée a toujours le même sens de l'écoulement, ce qui empêche la pénétration de liquide corporel.

3.1 Différents régimes autour d'un cylindre

L'écoulement d'un fluide est considéré incompressible et obéit aux équations de Navier- Stokes. L'adimensionnalisation de ces équations avec une échelle de vitesse U_0 et une échelle de Longueur D implique que l'écoulement dépend du nombre de Reynolds Re et des conditions aux limites.

L'écoulement autour d'un cylindre est présenté dans ce paragraphe pour des conditions aux limites de sorte que la surface du cylindre est la plus lisse. possible.

L'écoulement incident s'étend sur de grandes dimensions par rapport au cylindre et est le moins turbulent possible. Dans ces conditions, L'écoulement autour du cylindre dépend uniquement du nombre Reynolds défini comme :

$$Re = \frac{U_0.D}{v} \tag{3}$$

U_0 est la vitesse en amont, D le diamètre du cylindre et la viscosité cinématique du fluide considéré. Ce nombre adimensionnel caractérise le rapport entre les forces d'inertie et les forces visqueuses.

3.1.1 Ecoulement rampant

L'écoulement est dit rampant pour un Re < 5, quand le fluide (visqueux) s'écoule très lentement en une place étroite autour du cylindre. Les forces de viscosité étant prépondérantes, le fluide reste " attaché " au cylindre et il n'y a pas de décollement.

L'écoulement est symétrique par rapport à l'axe central du courant (axe longitudinal) et également entre l'amont et l'aval du cylindre.

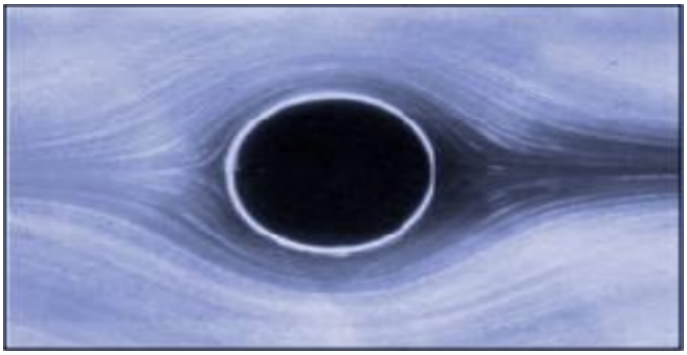


Figure 2: Écoulement rampant (visualisation réelle)

3.1.2 Régime stationnaire décollé

Pour $5 <$ Re < 48, les forces d'inertie augmentent et empêchent la couche limite de rester attachée au cylindre et commence à favoriser une dépression dans la zone de sillage.

Ainsi, on observe un décollement de chaque côté du cylindre. Le point de décollement se déplace vers l'amont du cylindre quand le nombre de Reynolds augmente. L'écoulement est stable et reste stationnaire et symétrique par rapport à l'axe longitudinal. En aval du décollement, se forment deux lobes presque symétriques de recirculation contrarotatifs attachés au cylindre.

Le point de rattachement, qui est définit comme le lieu où la vitesse longitudinale est nulle sur l'axe central du sillage, s'éloigne du cylindre quand le nombre de Reynolds augmente. L'abscisse de ce point définit la longueur de recirculation.

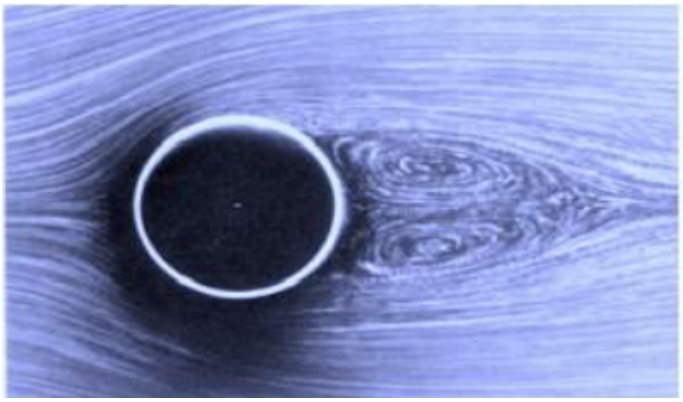


Figure 3: Écoulement dans un régime stationnaire décollé

3.1.3 Régime laminaire instationnaire

Pour $48 < \mathrm{Re} < 300$, l'écoulement devient instationnaire.

Les différentes perturbations possibles ne peuvent plus être amorties et une instabilité se déclenche ainsi. Les deux tourbillons perdent leur symétrie par rapport à l'axe longitudinal, se détachent du cylindre successivement et il est connecté dans le sillage pour former l'allée tourbillonnaire de Von-karman, les vecteurs et le champ de vitesse pour le cylindre.

Cette instabilité absolue est de nature bi-dimensionnelle et est caractérisée par une périodicité fortement prononcée. Ainsi le spectre temporel de la vitesse ou de la pression en un point de l'écoulement présente un pic important à la fréquence du lâcher tourbillonnaire. Cette fréquence adimensionnée par la vitesse de l'écoulement incident et le diamètre du cylindre définissent le nombre de Strouhal St. Ce dernier définit la fréquence adimensionnée du lâcher tourbillonnaire, avec :

$$S_t = \frac{f.D}{U} \tag{4}$$

Dans ce régime, cette fréquence adimensionnée augmente avec le nombre de Reynolds. On doit choisir un pas de temps qui soit une fraction assez fine de la période correspondante, soit pour un vingtième d'une période :

$$\Delta\tau = \frac{D}{20S_t.U} \tag{5}$$

Figure 4: Écoulement dans un régime instationnaire laminaire

3.1.4 Régime turbulent

Pour $300 < \mathrm{Re} < 2.10^5$, l'écoulement devient turbulent et le décollement augmente. Cependant, la couche limite est toujours laminaire.

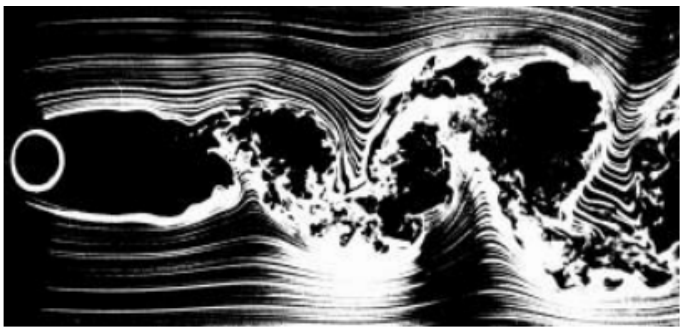


Figure 5: Écoulement dans un régime turbulent

Au-delà de $\mathrm{Re} = 2.10^5$ la couche limite est turbulente. Le point de décollement se déplace vers l'aval quand le nombre de Reynolds augmente,

car la couche limite est plus adhérente lorsqu'elle est turbulente. Ainsi, elle diminue la résistance au mouvement.

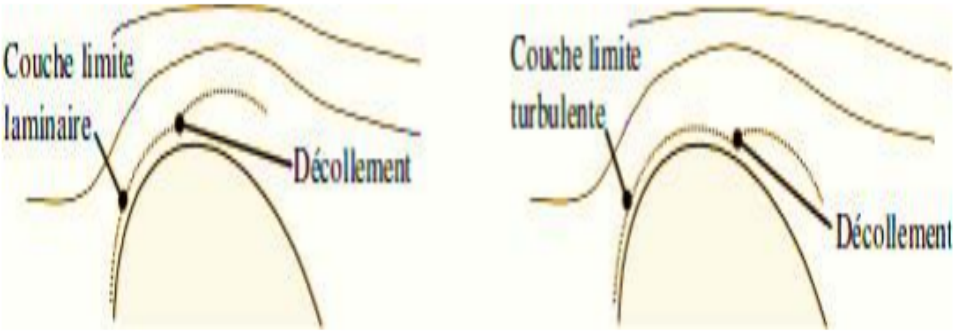


Figure 6: Décollement d'une couche limite laminaire ou turbulente

3.2 Gradient de pression

La transition de l'écoulement d'un régime laminaire vers un régime turbulent dépend du nombre de Reynolds. La zone de transition entre l'écoulement laminaire turbulent dépend aussi du nombre de Reynolds.

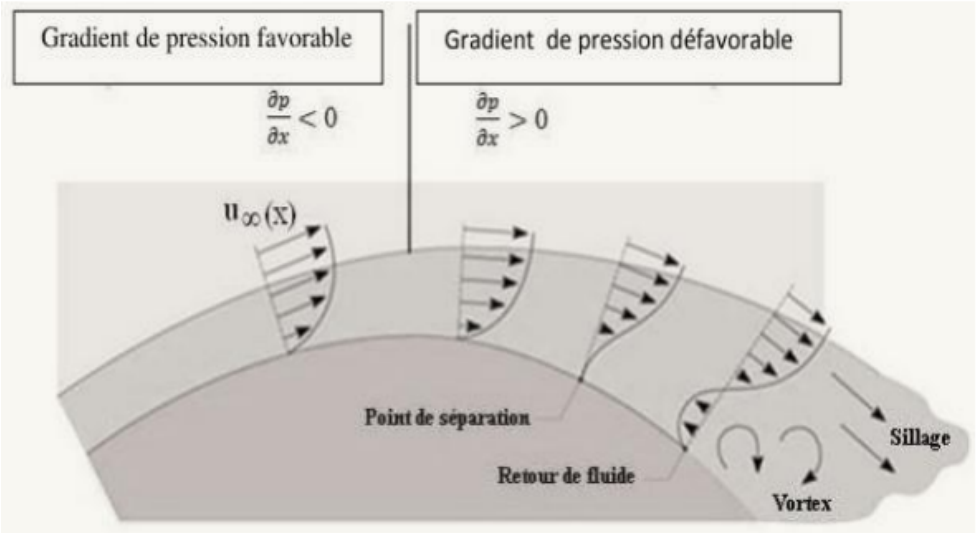


Figure 7: Profils de vitesse associée à la séparation sur un cylindre dans un écoulement transversal

3.3 Passage d'un régime à l'autre

On a trois types de transition : la première est appelée transition naturelle où la couche limite laminaire développe ce qu'on appelle les ondes de Tollmien-Schlichting suivie par une amplification d'instabilités et finalement d'un écoulement complètement turbulent.

La transition naturelle se produit habituellement avec des petites perturbations de l'écoulement libre.

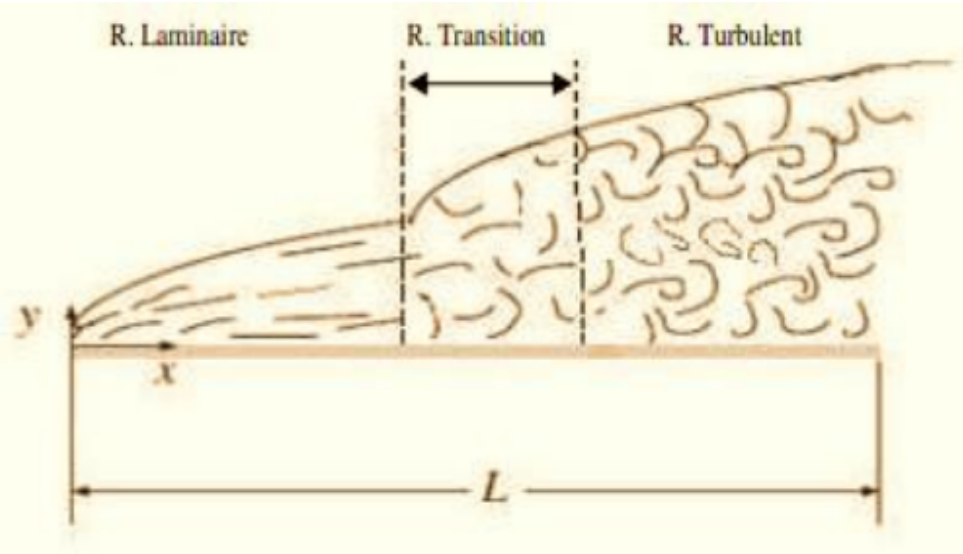


Figure 8: Phénomènes induits par la transition n

4 Méthode LBM

Le calcul local des fonctions de distribution des vitesses dans le temps et dans l'espace permet de calculer les grandeurs macroscopiques. En utilisant les bonnes modélisations, il est possible de calculer numériquement les grandeurs macroscopiques des écoulements de fluides. L'intérêt de ce type de modélisation réside en des modèles mathématiquement plus simples, dotés de propriétés de parallélisation des calculs plus simples à mettre en oeuvre.

C'est ce type de méthodes modernes pour le calcul des écoulements de fluides que nous nous proposons maintenant de présenter.

Dans notre cas on va utiliser la méthode de Lattice-Boltzmann (LBM).

La méthode Lattice Boltzmann est une méthode considérée comme étant puissante qui devient un concurrent sérieux avec les modèles traditionnels utilisés en CFD tels que les méthodes d'éléments finis ou de volumes finis etc.

C'est une méthode qui attire nombreux chercheurs, vu qu'elle permet l'incorporation de beaucoup de physique et est hautement parallélisable. De nombreux comportements fluides sont devenues accessibles, comme la turbulence ou la transition de phase par exemple, grâce à cette méthode. Elle est utilisée dans le domaine de la modélisation CFD pour le transport, la

météorologie etc.

À l'inverse des méthodes classiques qui considèrent une déscription macroscopique du volume représentatif (VRE), Notre méthode se base quant à elle sur une description mésoscopique pour résoudre une loi de distribution de vitesses (distribution de Maxwell- Boltzman) du VRE sur un réseau prédéfini.

La vitesse est calculée comme étant le moment de la fonction de distribution et par la suite pour retrouver les équations de Navier-Stokes, la LBM utilise le modèle BGK développé par Bhatnagar, Gross an Krook.

Plusieurs niveaux existent dans les méthodes actuelles pour décrire un système : microscopique, macroscopique et mésoscopique.

Le niveau microscopique est l'échelle à laquelle la matière est considérée discontinue alors que le niveau macroscopique est l'échelle à laquelle la matière est considérée continue.

Le niveau mésoscopique quant à lui est une échelle intermédiaire, grande devant le microscopique et petite devant le macroscopique. C'est cette dernière qui nous intéresse vraiment et où s'applique la méthode LBM.

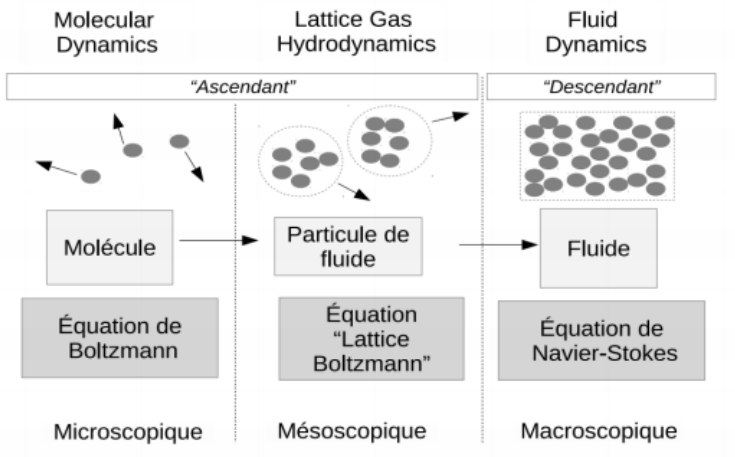


Figure 9: Équations permettant de décrire chaque phénomène

Le calcul local des fonctions de distribution des vitesses dans le temps et dans l'espace permet de calculer les grandeurs macroscopiques. En utilisant les modélisations appropriées il est ainsi possible de calculer numériquement les grandeurs macroscopiques des écoulements de fluides. L'intérêt de ce type de modélisation réside en des modèles mathématiquement plus simples, dotés de propriétés de parallélisation des calculs plus simples à mettre en oeuvre.

C'est ce type de méthodes modernes pour le calcul des écoulements de fluides que nous nous proposons maintenant de présenter.

L'équation de Boltzmann ci-dessus peut être discrétisée de cette façon :

$$f(x+v_i, v_i, t+1) - f(x, v_i, t) + F(v_i) = \frac{1}{\tau}[f^0(n, \vec{u}, t) - f(x, v_i, t)] \qquad (6)$$

Nous avons discrétisé l'espace des vitesses en un nombre infini de vecteurs vitesses vi, l'espace des positions est un réseau pour lequel x+vi est à nouveau une position dans le réseau et le temps peut prendre seulement des valeurs entières. Dès lors que les vecteurs vitesses sont maintenant définis, nous utiliserons la notation :

$$f(x, v_i, t) = f_i(x, t) \qquad (7)$$

et

$$F(v_i) = F_i \qquad (8)$$

Les termes de forces F_i sont définis comme une généralisation de la force dans l'équation de Boltzmann précédente, sous la forme :

$$F_i \iff \frac{Fa}{m}\frac{\partial f}{\partial va} \qquad (9)$$

En particulier, il faut que les moments vérifient :

$$\sum_i F_i = \int \frac{F_\alpha}{m}\frac{\partial f}{\partial v_\alpha}\underline{dv} = 0$$

$$\sum_i F_i v_{i\alpha} = \int \frac{F_\beta}{m}\frac{\partial f v_\alpha}{\partial v_\beta}\underline{dv} = -n\frac{F_\alpha}{m}$$

$$\sum_i F_i v_{i\alpha} v_{i\beta} = \int \frac{F_\gamma}{m}\frac{\partial f v_\alpha v_\beta}{\partial v_\gamma}\underline{dv} = -n(\frac{F_\alpha}{m}u_\beta + u_\alpha\frac{F_\beta}{m})$$

$$\sum_i F_i v_{i\alpha} v_{i\beta} v_{i\gamma} = \int \frac{F_\delta}{m}\frac{\partial f v_\alpha v_\beta v_\gamma}{\partial v_\delta}\underline{dv} =$$

$$= -n[\frac{F_\alpha}{m}(kT\delta_{\beta\gamma} + u_\beta u_\gamma) + \frac{F_\beta}{m}(kT\delta_{\alpha\gamma} + u_\alpha u_\gamma) + \frac{F_\gamma}{m}(kT\delta_{\alpha\beta} + u_\alpha u_\beta)]$$

On peut montrer que la limite hydrodynamique de cette version discrète de l'équation de Boltzman décrit bien les équations d'écoulements des fluides.

- On projette la fonction de distribution des particules (FDP) et la fonction de distribution à l'équilibre (FDE) sur un sous-espace de Hilbert grâce à leur développement en polynômes de Hermite orthonormés sur l'espace des vitesses
- On tronque les polynômes FDP et FDE à la résolution désirée qui est déterminée par le nombre N de moments des vitesses requis
- On applique une quadrature de Gauss-Hermite pour les distributions tronquées de particules et les moments des vitesses, conduisant à l'équation de Boltzmann sur réseau et aux expressions discrètes de la densité et des moments.

Projetons sur une base de Hermite et appliquons la troncature. Une base de Hermite est choisie car les moments de degré N sont complètement résolus par un développement des coefficients jusqu'au même ordre de grandeur. Ainsi, des troncatures des termes d'ordres plus élevés n'affecte en rien les termes d'ordres inférieurs.

Ainsi, les FDP peuvent être approximés par les projections sur les N premiers polynômes de Hermite tout en conservant une représentation exacte des moments des vitesses d'ordres inférieurs ou égaux à N.

La projection et la troncature s'écrivent mathématiquement :

$$f(\vec{r},\vec{v},t) = \omega\vec{v}\sum_{n=0}^{\infty}\frac{1}{n!}a^{(n)}(\vec{r},t)H^{(n)}(\vec{v})$$

$$f^{(N)}(\vec{r},\vec{v},t) = \omega\vec{v}\sum_{n=0}^{N}\frac{1}{n!}a^{(n)}(\vec{r},t)H^{(n)}(\vec{v})$$

Avec : $a^{(n)}$ et $H^{(n)}$ les tenseurs de rang n respectivement des coefficients de développement et des polynômes de Hermite.

Les fonctions poids associées aux polynômes de Hermite sont définies par :

$$\omega(\vec{v}) = \frac{1}{(2\pi)^{D/2}}exp(\frac{v^2}{2}) \tag{10}$$

où $v^2 = \vec{v}.\vec{v}$ et les coefficients de développement donnés par :

$$a^{(n)}(\vec{r},t) = \int f(\vec{r},\vec{v},t)H^{(n)}(\vec{v})dv \tag{11}$$

Les quelques premiers développements en H(n) sont identifiés à des variables hydrodynamiques :

$$\rho = a^{(0)}$$
$$\rho\vec{u} = a^{(1)}$$
$$\Pi = a^{(2)} - \rho(\vec{u} - \delta)$$

4.1 Modèles 2D

4.1.1 Position du problème

Dans notre cas on va étudier et simuler l'écoulement autour d'un cylindre et aussi à travers une grilles de 5 cylindres.

L'écoulement est décrit par l'équation de Navier-Stokes qui est une équations aux dérivées partielles non linéaires dont l'existence de solutions et la résolution constituent dans le cas général l'un des problème complexes du prix du millénaire.

Vu cette complexité nous allons utiliser la méthode LBM qui est rarement utilisée dans les logiciels de simulation des fluides. On s'intéresse dans notre stage uniquement aux modèles 2D

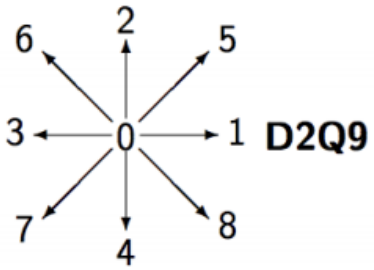


Figure 10: Schéma D2Q9

On considère classiquement les viscosités décrites à partir du temps de relaxation τ et de la célérité du son c_s sur le réseau :

Pour D2Q9, c'est à dire 2 dimensions et 9 vitesses nous avons dans ce cas :

$$w_0 = \frac{4}{9}$$

$$w_1 = w_2 = w_3 = w_4 = \frac{1}{9}$$

$$w_5 = w_6 = w_7 = w_8 = \frac{1}{36}$$

Avec : *w* nos différents vitesses.

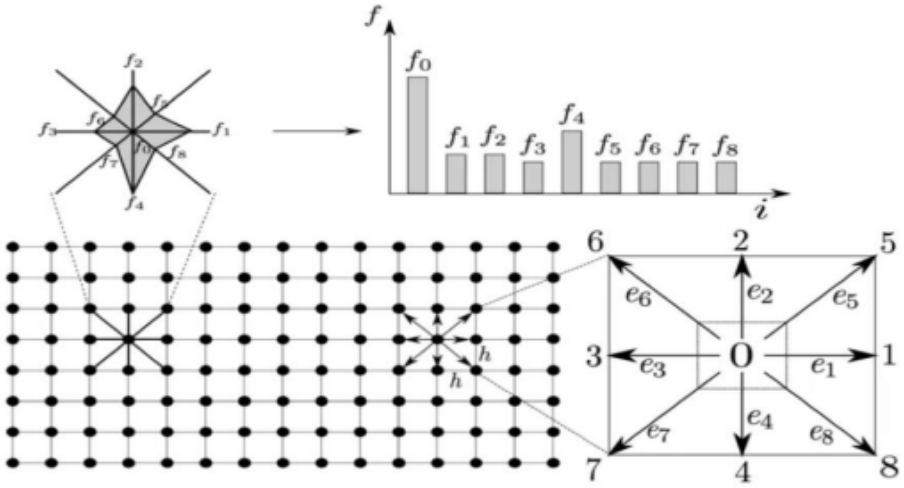


Figure 11: Problème 2D discrétisé dans le cadre Lattice Boltzmann

Les points noirs sur la figure 11 représentent des nœuds spatiaux occupés par des particules. Un site ne contient pas, en effet, qu'une seule particule mais un certain nombre de particules. Ce dernier dépend de la densité locale.

Dans un pas de temps, une particule ne peut se déplacer que vers ses sites voisins qui sont les plus proches, avec l'une des vitesses discrètes qui sont ici notées e_i, avec i allant de 0 à 8.

Les f_i, quant à eux, représentent le nombre de particules qui se déplacent avec une vitesse discrète e_i. Pour un site contenant N particules, par exemple : Les particules : f_i x N se déplacent avec une vitesse e_i.

NB : Une fois que les particules d'ensemble se sont déplacées vers celle du site voisin, elles entreront en collision avec d'autres ensembles de particules qui arrivent sur le même site en même temps.

5 Simulation numérique

L'écoulement autour d'un cylindre immobile est l'un des problèmes fondamentaux de la mécanique de fluide d'importance pratique. Ses applications sont très nombreuses en aérodynamique et en acoustique également.

Le but de ce projet consiste comme on l'a déjà mentionné à utiliser une méthode numérique (LBM) afin d'étudier le phénomène.

5.1 Géométrie

L'écoulement autour du cylindre est modélisé en 2D avec l'axe de rotation du cylindre perpendiculaire au sens de notre écoulement.

On modélise notre écoulement tout d'abord autour d'un cylindre de section circulaire puis à travers une grille simulée par 5 cylindres et dans tous les cas pour des Reynolds compris entre 50 et de 200.

5.1.1 Écoulement autour d'un cylindre

On représente notre domaine dans le programme (python) par un rectangle dont les dimensions choisies en pixels sont :

$$\begin{cases} Nx = 780 \\ Ny = 360 \end{cases}$$

Le rayon du cylindre quant à lui est : R $= \frac{N_y}{10} = 36$ (pixels). On travaillera avec des nombres de Reynolds compris entre 50 et 200.

Figure 12: Schéma pour un cylindre

5.1.2 Ecoulement autour de 5 cylindres

On représente le domaine dans le programme (python) par un rectangle dont les dimensions choisies en pixels sont :

$$\begin{cases} Nx = 780 \\ Ny = 360 \end{cases}$$

Le rayon de chaque cylindre est de : R $= \frac{N_y}{25} = 14$ (pixels). On travaillera de même avec des nombres de Reynolds compris entre 50 et 200.

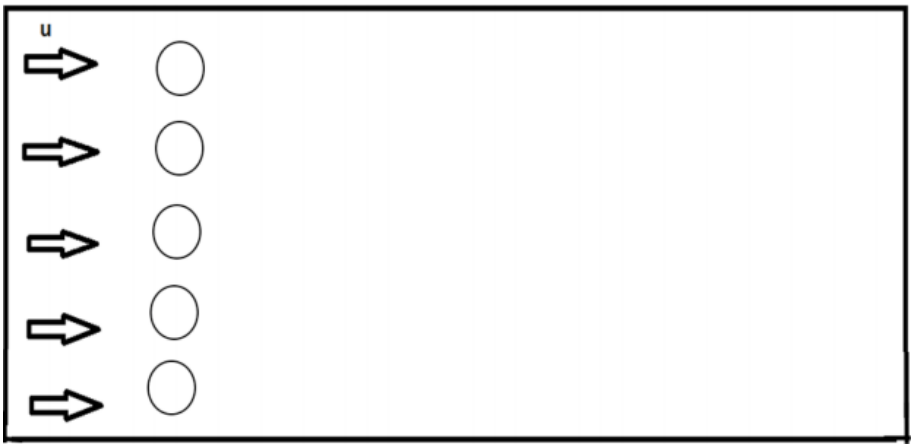


Figure 13: Schéma d'une grille composée de 5 cylindres

5.2 Programme

Le programme consiste à mettre en œuvre la méthode de Lattice Boltzmann sur un ou plusieurs obstacles qu'on définit. Cet obstacle peut être un disque (représentant un cylindre de base circulaire en 2D).

Dans un premier temps, le programme affiche une fenêtre de saisie pour choisir le type, la méthode utilisée et les paramètres dans la simulation. Cette fenêtre de sélection est une fenêtre de géométrie 800x250 avec les options : Cylindre ellipsoïdal, Helmholtz, 5 Cyl. Circ. Et 22 Cyl. circ.+Ellipse.

Dans notre cas, on utilise l'option 5 Cyl. Circ pour la simulation de 5 cylindres de section circulaire. Ensuite, on a à choisir les paramètres de la simulation, à savoir, le nombre de Reynolds, l'amplitude de la perturbation du HP, la demi-période de pulsation HP (multiple de 100), l'inclinaison du profil Theta ou la longueur et le rayon du tube.

Cependant, on a réussi à adapter le code original (11 cylindres et une ellipse) pour notre cas d'un et de 5 cylindres. On a donc modifié la partie des dimensions et des emplacements des obstacles.

On a développé l'algorithme dans le cas d'un problème résolu avec la méthode de Lattice-Boltzmann (LBM). On distingue plusieurs parties :

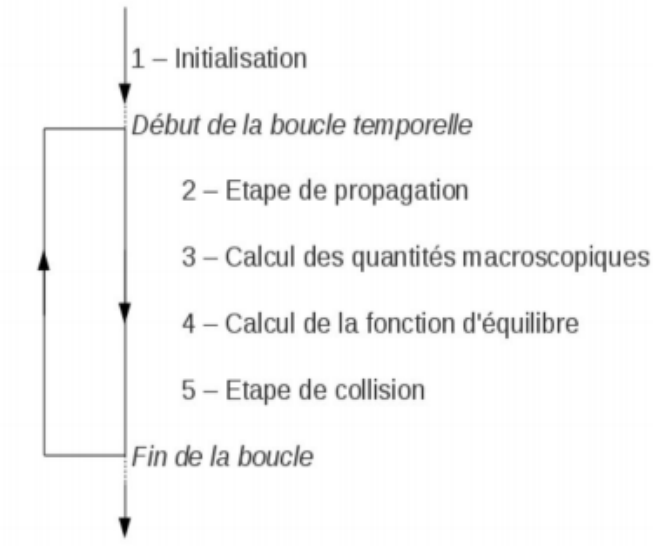


Figure 14: Algorithme de base de la LBM

On va montrer ci-de suite ce à quoi chaque partie sur la figure 14 correspond :

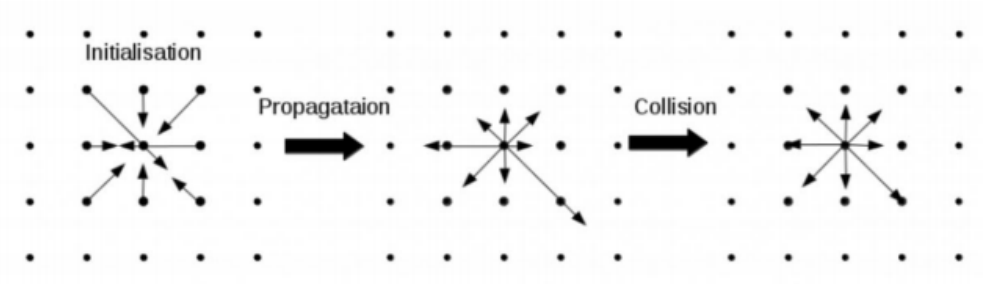


Figure 15: Schéma des étapes de propagation et de collision

5.2.1 Étape de propagation

Durant cette étape, les particules vont se propager d'un nœud à l'autre de par la direction de leur vitesse. On peut résumer cette situation par l'équation :

$$f_i^{temps}(\vec{x} + \vec{c_i}\delta t, t) = f_i(\vec{x}, t) \tag{12}$$

5.2.2 Calcul des quantités macroscopiques

On peut calculer les quantités macroscopiques à l'aide des fonctions de distribution précédentes telles que :

$$\rho(\vec{x}, t) = \sum_{i=0}^{q-1} f_i(\vec{x}, t)$$

$$\vec{v} = \frac{1}{\rho} \sum_{i=0}^{q-1} f_i(\vec{x}, t)\vec{c_i}$$

Avec : q le nombre de directions possibles. Par exemple : pour le schéma D2Q9, le nombre de direction possible est de 9.

5.2.3 Calcul de la fonction d'équilibre

L'expression de la fonction de distribution à l'équilibre peut être obtenue à partir de la fonction de distribution de Maxwell normalisée avec un développement

en séries de Taylor telle que :

$$f_i^{eq} = \Phi\omega_i[A + Bc_i.u + C(c_i.u)^2 + Du^2] \tag{13}$$

avec : u la vitesse macroscopique et c_i la vitesse microscopique dans la direction i et A, B, C et D des coefficients qui dépendent du problème, φ un paramètre scalaire et ω une constante qui dépend du type de noeud.

5.2.4 Etape de collision

Pendant cette étape, les fonctions de distribution et les vitesses des particules seront modifiées.

En remarque, on peut ajouter que deux modèles sont disponibles lors de cette étape de collision qui comprend soit un seul paramètre de relaxation soit plusieurs. En effet le modèle "Single Relaxation Time" (SRT) utilise un unique paramètre de relaxation w.

Ce paramètre est défini comme l'opérateur de collision.

NB : La principale limite de ce modèle est que les paramètres macroscopiques tels que la viscosité dépendent de ce paramètre de relaxation.

Le modèle "Multi-Relaxation Time" (MRT) permet d'utiliser plusieurs paramètres de relaxation ce qui apporte les avantages d'avoir un nombre de Prandtl différent de l'unité et une viscosité dynamique variable par exemple. Il est d'autant plus préféré pour sa stabilité.

5.2.5 Ecoulement autour d'un cylindre

Pour le cas d'un cylindre, notre programme a consisté à mettre la méthode (LBM) pour un seul obstacle sous forme de disque (représentation d'un cylindre de base circulaire en 2D).

La définition de l'obstacle se fait par la fonction " fromfunction " qui permet de construire un tableau dont le terme général obéit à une formule donnée, en utilisant une " lambda fonction ".

Dans ce cas, on choisit $R = \frac{Ny}{10} = 36$ pixels, et on emploie la formule : $(x - cx)^2 + (y - cy_1)^2 < r^2/$

$Cy_1 = \frac{Ny}{2}$ nous permet de positionner en position centrale verticalement

On ajoute la même lige en remplaçant cy_1 par cy_2 pour définir un deuxième obstacle sous forme d'un cylindre circulaire 2D (à près avoir défini cy_2 bien entendu). On a également choisi de laisser la position horizontale à $\frac{Nx}{10}$pour mieux analyser l'écoulement.

```
## Cylindres de bases circulaires##
cx = nx/10; cy1 = ny/2;
r = ny/10; cx2 = cx + 8*r;
deccy = ny/24; relips = ny/10
cxelips=nx/2; cyelips=ny/2
if (theta < -0.1):
    cyelips=ny/3
# Coordinates of the cylinders.
    obstacle = fromfunction(lambda x,y: (((x-cx)**2+(y-cy1)**2 <r**2)\
                                        | (y < 5) | (y > 355)), (nx,ny))
```

Figure 16: Lignes de code définissant l'obstacle

Après, on retrouve la boucle basée sur la méthode Lattice Boltzmann juste après la définition de l'obstacle. On retrouve notre programme dans l'annexe.

5.2.6 Ecoulement autour de 5 cylindres

Pour le cas de 5 cylindres, notre programme a consisté de même à mettre la méthode (LBM) pour 5 obstacles sous forme de disques (représentation de 5 cylindres de base circulaire en 2D).

On choisit R = $\frac{Ny}{25} = 14$

```
## Cylindres de bases circulaires##
cx = nx/10; cy1 = ny/6; cy2= ny/3; cy3 = ny/2; cy4 = 2*ny/3; cy5 = 5*ny/6; r = ny/25; cx2 = cx + 8*r;
deccy = ny/24; relips = ny/10
cxelips=nx/2; cyelips=ny/2
#if (theta < -0.1):
    #cyelips=ny/3
# Coordinates of the cylinders.
if(Cas == 2):
    obstacle = fromfunction(lambda x,y: (((x-cx)**2+(y-cy1)**2 <r**2)\
                                        | ((x-cx)**2+(y-cy2)**2 <r**2) | ((x-cx)**2+(y-cy3)**2 <r**2)\
                                        | ((x-cx)**2+(y-cy4)**2 <r**2) | ((x-cx)**2+(y-cy5)**2 <r**2)\
                                        | (y < 5) | (y > 355)), (nx,ny))
```

Figure 17: Lignes de code définissant les 5 obstacle

On a positionné nos 5 cylindres à égale-distance, de $\frac{ny}{6}$.

C_x reste constante vu que c'est une grille. On retrouve notre programme pour ce cas dans l'annexe.

5.3 Représentation de l'écoulement

On a effectué des simulations numériques d'un écoulement autour d'un cylindre et à travers une grille composée de cinq cylindres. Pour chaque simulation on a travaillé avec différentes valeurs de Reynolds (50, 100, 150, 200).

Nous avons, en effet, effectué une simulation qu'en instationnaire avec une vitesse de 0.04 m/s et 29000 itérations.

On va étudier et analyser le comportement du fluide après le contact avec l'obstacle ou les obstacles solides (derrière le cylindre) pour chaque nombre de Reynolds.

5.3.1 Écoulement autour d'un cylindre

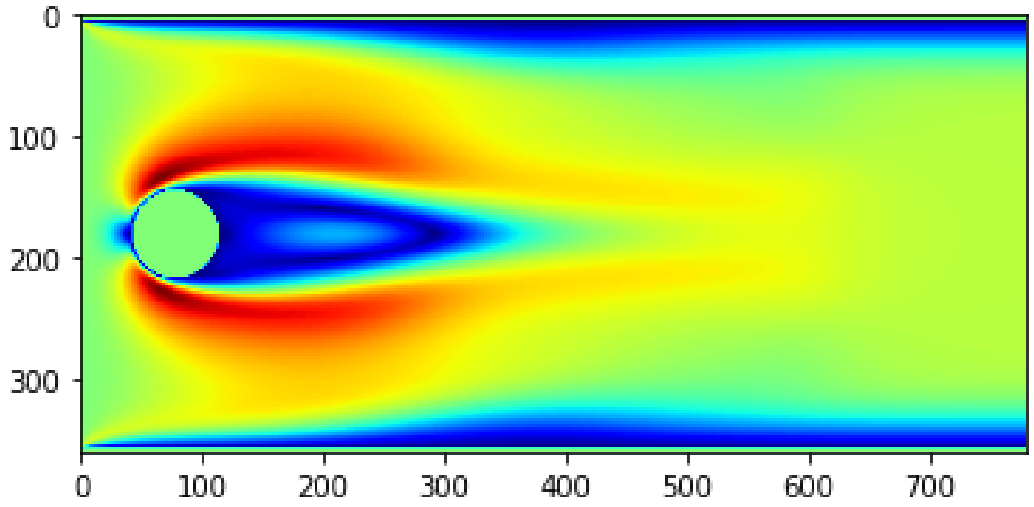


Figure 18: Re = 50

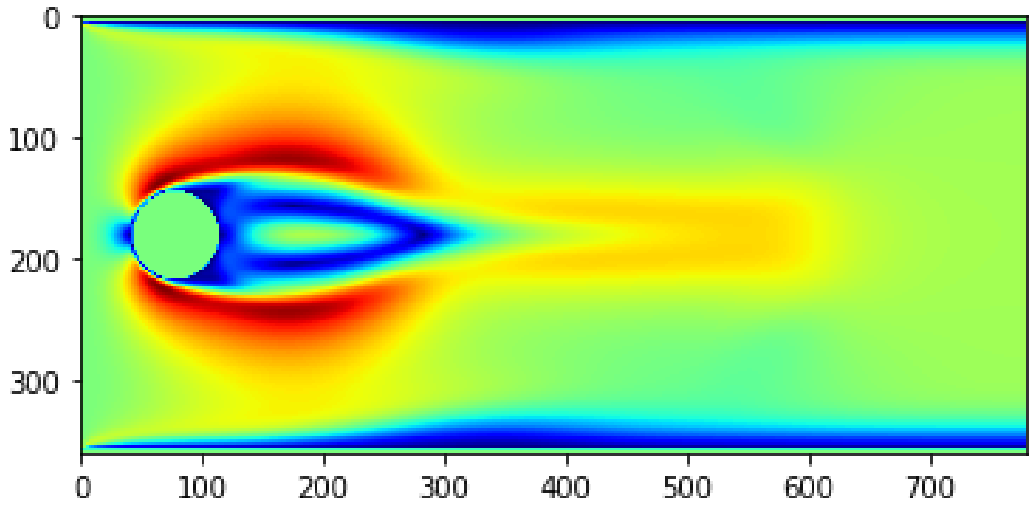


Figure 19: Re = 100

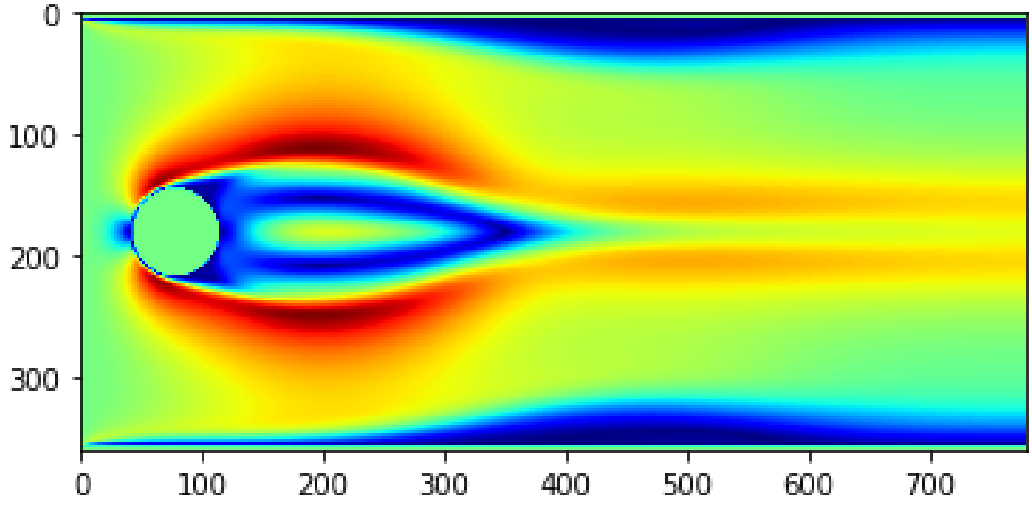


Figure 20: Re = 150

On constate que comme le nombre de Reynolds augmente, l'écoulement commence à se détacher derrière le cylindre, alors que l 'écoulement devient instationnaire.

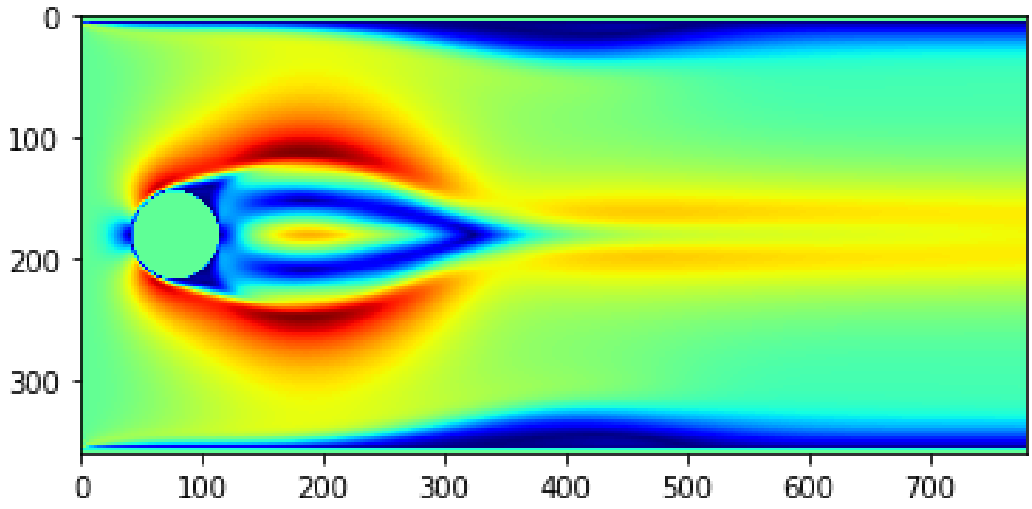


Figure 21: Re = 200

Dans de telles situations, la décélération de l'écoulement, provoquée par le contact du liquide avec l'obstacle solide, donne lieu à un certain nombre de phénomènes qu'on peut considérer comme étant complexes.

On voit bien que :

- une zone rigide enveloppe la zone cisaillée.
- Deux zones rigides statiques " collées " au cylindre se trouvant aux points de stagnation.

On voit aussi qu'une zone de recirculation stable se forme en aval du cylindre et sa taille augmente avec le nombre de Reynolds.

Le sillage de l'écoulement quant à lui reste symétrique et stable même en augmentant le nombre de Reynolds. En effet on s'attendait à la formation du phénomène de détachement tourbillonnaire qui consiste à la formation de tourbillons de chaque côté du cylindre, formant ainsi une allée de tourbillons laminaire, appelée allée de Von Karman.

Et en augmentant encore le nombre de Reynolds (Re $>$ 100), on voit que le vortex (vortex de Karman) se crée et le flux progresse de son côté.

5.3.2 Ecoulement de 5 cylindres

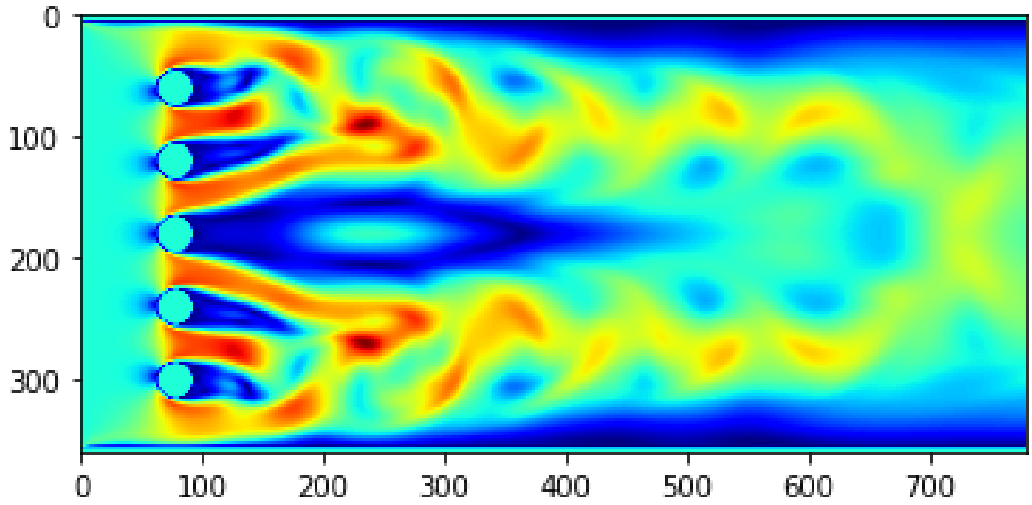


Figure 22: Re = 50

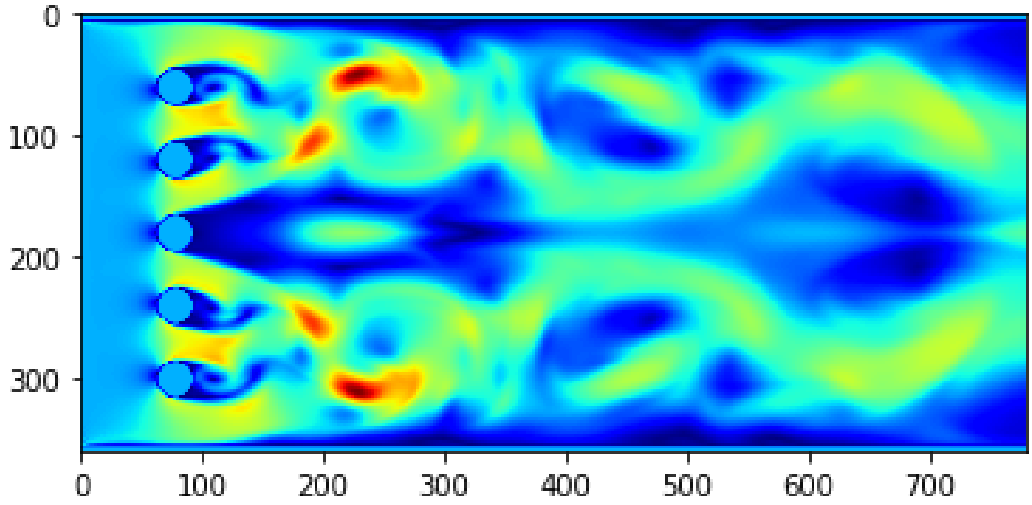


Figure 23: Re = 100

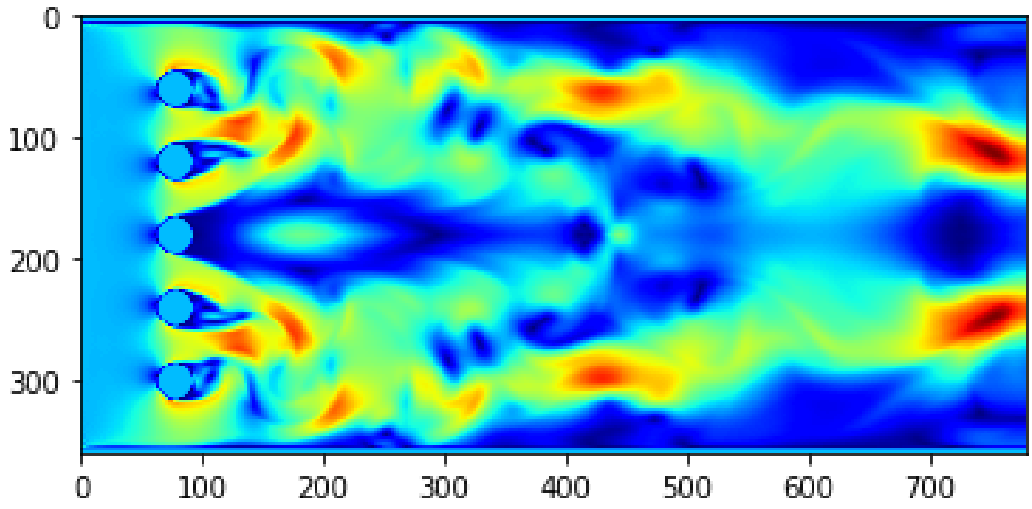


Figure 24: Re = 150

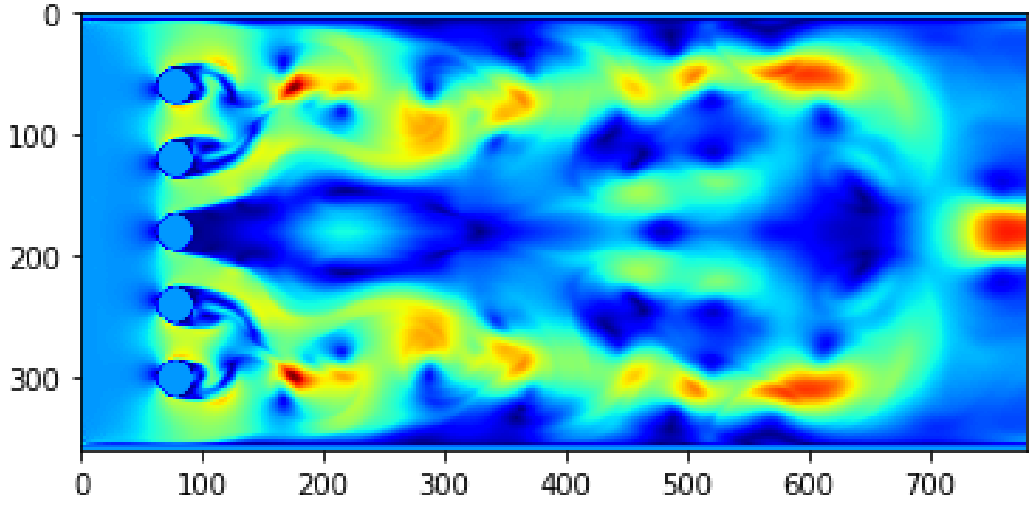


Figure 25: Re = 200

On remarque qu'en augmentant le nombre de Reynolds il y'a un développement des gros tourbillons. Ces dernières sont éjectées en alternance parfois vers la paroi supérieure et la paroi inférieure de chaque cylindre. On peut dire qu'il y a un roulement des tourbillons dans la zone de sillage et non un glissement.

À mesure que le nombre de Reynolds augmente, des tourbillons symétriques se détachent l'un après l'autre, et forment peu à peu l'allée caractéristique de Von Karman, comme le montre bien les simulations. Ceci est un phénomène instationnaire.

Nous avons en effet obtenu des allées tourbillonnaires complétement développées. 33 Les tourbillons restent toujours symétriques pour les différentes valeurs de Re jusqu'à l'arrivée à la paroi.

5.3.3 Allée de Von-Karman

Lorsque le nombre de Reynolds est compris entre 50 et 200, les tourbillons deviennent instables.

Les tourbillons qui sont disposés de part et d'autres de chaque cylindre et qui possèdent une vorticité inversée, s'enroulent et s'approchent des vorticités de signe opposé. Ils sont alors éjectés en aval.

Les tourbillons se détachent à tour de rôle alors que d'autres se créent à leur place.

Pour observer les allées de Von Karman, nous nous sommes placé dans le cas où le nombre de Reynolds est compris entre 50 et 200.

Pour le cas d'une grille à 5 cylindres :

1. Le comportement de l'écoulement a montré qu'un Re égale à 50 est la valeur critique pour l'apparition des allées de Von Karman à ß = 4 fixé

2. On a pu reproduire les allés de Von Karman pour toutes les valeurs de Re étudiées comprises entre 50 et 200 pour un ß fixé.

NB : Le nombre adimensionnel correspondant à l'étude de ce phénomène est le nombre de Strouhal, qui est étroitement lié au nombre de Reynolds. (cité à la partie 3).

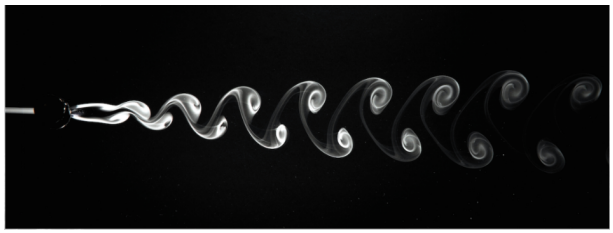


Figure 26: Allée de Von-Karman

5.4 Difficultés rencontrés

Dans un premier temps, on a eu un problème dans l'installation de la librairie ffmpeg qui est nécessaire à la réalisation d'animations pour l'assimilation de l'écoulement. Après plusieurs recherches sur internet, et après l'utilisation de deux éditeurs Python différents (Spyder et Pycharm) on a réussi à faire fonctionner le programme. En effet on a mis toutes les lignes responsables à la création et l'affichage des animation, en commentaire. (voir Annexe)

```
fig = plt.figure()
ims = []
#Writer = animation.writers['ffmpeg']
#writer = Writer(fps=15, metadata=dict(artist='Patrick VIGLIANO'), bitrate=1800)

###################################
nom_fichstr=''
for time in range(maxIter):
    fin[i1,-1,:] = fin[i1,-2,:] # Right wall: outflow condition.
    rho = sumpop(fin)           # Calculate macroscopic density and velocity.
    u = dot(c.transpose(), fin.transpose((1,0,2)))/rho
    u[:,0,:] =vel[:,0,:] # Left wall: compute density from known populations.
    #u[:,0,:] =vel[:,0,:]*(1. + AmpHP * sin(time * pi / HP)) # Left wall: vitesse oscillar
    #u[:,0,:] =vel[:,0,:]*(1. + AmpHP * sin(time * omega0helmholtz * 2 * pi)) # Left wall:
    #u[:,0,:] =vel[:,0,:]*(1. + AmpHP * sin(time * omega0helmholtz)) # Left wall: vitesse
    rho[0,:] = 1./(1.-u[0,0,:]) * (sumpop(fin[i2,0,:])+2.*sumpop(fin[i1,0,:]))
    feq = equilibrium(rho,u) # Left wall: Zou/He boundary condition.
```

```
    u[:,0,:] =vel[:,0,:] # Left wall: compute density from known populations.
    #u[:,0,:] =vel[:,0,:]*(1. + AmpHP * sin(time * pi / HP)) # Left wall: vitesse oscillar
    #u[:,0,:] =vel[:,0,:]*(1. + AmpHP * sin(time * omega0helmholtz * 2 * pi)) # Left wall:
    #u[:,0,:] =vel[:,0,:]*(1. + AmpHP * sin(time * omega0helmholtz)) # Left wall: vitesse
    rho[0,:] = 1./(1.-u[0,0,:]) * (sumpop(fin[i2,0,:])+2.*sumpop(fin[i1,0,:]))
    feq = equilibrium(rho,u) # Left wall: Zou/He boundary condition.
    fin[i3,0,:] = fin[i1,0,:] + feq[i3,0,:] - fin[i1,0,:]
    fout = fin - omega * (fin - feq)  # Collision step.
    for i in range(q): fout[i,obstacle] = fin[noslip[i],obstacle]
    for i in range(q): # Streaming step.
        fin[i,:,:] = roll(roll(fout[i,:,:],c[i,0],axis=0),c[i,1],axis=1)
    if ((time >= 1) & (time%100 == 1)): # Visualization
    #if (time%100 == 1): # Visualization
       # im=plt.imshow(((u[0]**2+u[1]**2)+rho*cs**2).transpose(),cmap='rainbow', animated=
        im = plt.imshow(sqrt(u[0]**2+u[1]**2).transpose(),cmap=cm.jet, animated=True) #vit
        # if (time/100 == 1): plt.savefig("test.png")
        ims.append([im])
        #ani = animation.ArtistAnimation(fig, ims, interval=50, blit=True, repeat_delay=10
        #ani.save(nom_fichstr, writer=writer)
        print(time,' sur ', maxIter)
plt.show()
```

Figure 27: lignes selectionnées sont mises en commentaire

Cependant, on a constaté que le programme prend la totalité de la mémoire de l'ordinateur et s'exécute très lentement.

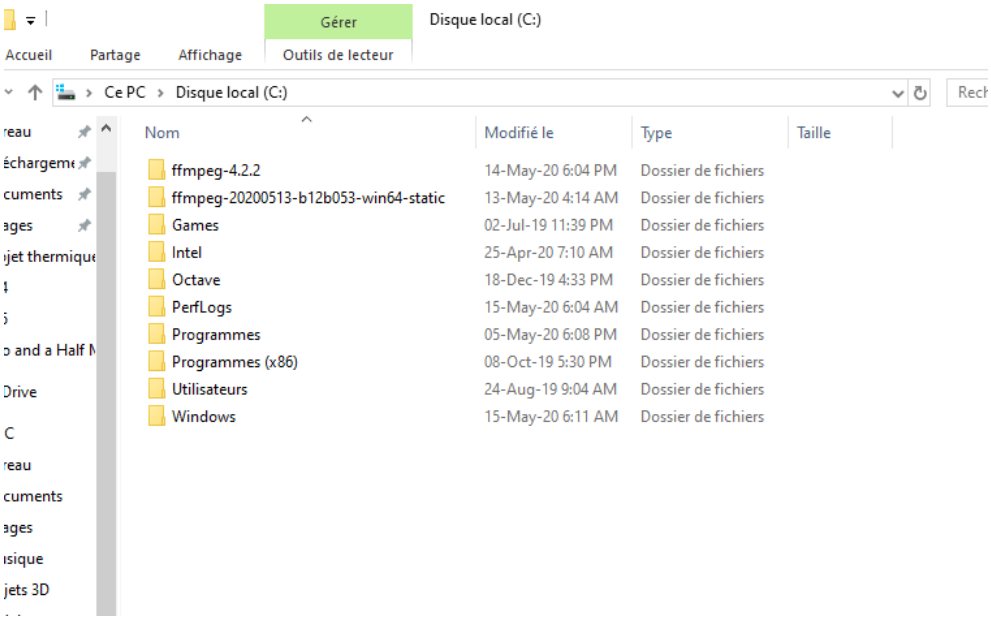


Figure 28: La librairie ffmpeg installée

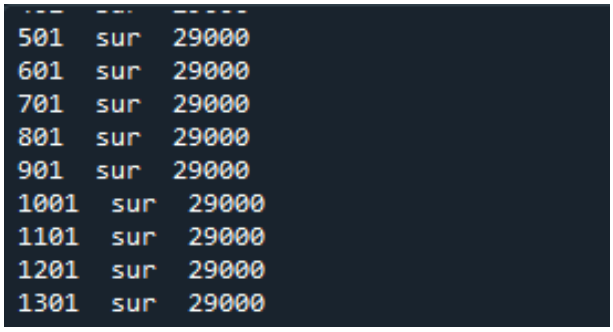


Figure 29: itérations du programme

Du coup, on a décidé d'utiliser l'outil " colaboratory " de Google qui joue le rôle d'un compilateur Python en ligne, avec une mémoire largement suffisante pour exécuter le programme de manière fluide.

Pour la modification du nombre de cylindres, on a reçu beaucoup de messages d'erreurs en utilisant " colaboratory ". Ceci est dû à la fenêtre de saisie du programme, puisque " colaboratory " utilise des commandes différentes et recourt à des librairies propres à lui, pour des tâches spéciales

telles que celles-ci. On a réutilisé Spyder pour contourner ceci. Ainsi, on a dû poursuivre nos simulations malgré les problèmes de mémoire saturée, en suivant un programme : un ordinateur compile le programme en permanence, en utilisant un second ordinateur pour la recherche et la rédaction.

6 Interprétation des résultats

En se basant sur l'équation de Boltzmann discrétisée avec le modèle BGK, on a modélisé notre écoulement dans une conduite en prenant en considération un obstacle cylindrique et une grille de 5 obstacles après.

Cette expérience nous a montré que l'on peut obtenir des allées tourbillonaires de Von Karman à partir de Re = 50 et jusqu'à Re = 200 pour le cas qui nous intéresse le plus (grille composée de 5 cylindres).

On a vu qu'au-delà de Re = 50, la couche limite devient instable et ses oscillations donnent naissance à des tourbillons lâchés alternativement de chaque côté du cylindre.

À l'aval, on a observé une l'Allée de Von-Karman.

Quand Re augmente encore, le sillage est devenu turbulent et le point de décollement est déplacé vers l'aval.

On a remarqué qu'il y a de gros tourbillons qui se développent et sont éjectés en alternance parfois vers la paroi supérieure et la paroi inférieure. On peut dire qu'il y a un roulement des tourbillons dans la zone de sillage et non un glissement.

Le flux quant à lui progresse en augmentant le Reynolds.

On voit que cela est en accord avec la théorie, vu qu'à un certain nombre Re le régime devient turbulent et l'allée de tourbillons se forme.

7 Conclusion

L'intérêt de ce stage est d'acquérir de nouvelles connaissances numériques en utilisant une nouvelle méthode : La méthode LBM. Il nous a permis de bien comprendre la méthode et de l'utiliser en l'intégrant dans un code python, afin de simuler en 2D le comportement d'un fluide en charge à l'encontre d'un et plusieurs obstacles (5 cylindres).

En effet, la méthode de Lattice-Boltzmann reste encore peu mature même si de plus en plus de chercheurs se tournent vers ce mode de résolution.

Nous avons pu comparer nos résultats obtenues numériquement à ceux donnés dans la partie théorie. Ceci nous a permis de les mieux comprendre.

Nous avons appris encore une fois l'influence que le nombre de Reynolds a sur les écoulements. Nos résultats numériques ont montré qu'il existe une plage de Reynolds pour lesquels l'instabilité se développe. De plus, on a réussi à avoir une autre vision de la mécanique des fluides ainsi que la méthode de calcul qui en découle.

Nous avons aussi dû faire face à de nombreuses difficultés cependant on a pu obtenir des résultats relativement satisfaisants au final.

8 Bibliographie

Cinétique-LBC1.pdf (Mail Patrick Vigliano) /

https://feaforall.com/implementation-lattice-boltzmann-method-lbm/

http://feaforall.com/creating-cfd-solver-lattice-boltzmann-method/

http://hmf.enseeiht.fr/travaux/beiepe/book/export/html/45

http://hmf.enseeiht.fr/travaux/projnum/book/export/html/1916

https://tel.archives-ouvertes.fr/tel-01450340/document

https://tel.archives-ouvertes.fr/tel-01281360/document

http://www-sop.inria.fr/tropics/MAIDESC/COMMUNICATIONS/These-Emmanuelle-Itam.pdf

http://hmf.enseeiht.fr/travaux/projnum/book/export/html/900#:~:text=2.1%20%C3%89coulement%20autour%20d'un%20cylindre&text=Ainsi%2C%20la%20circulation%20de%20fluide,chemin%C3%A9e%20ou%20d'un%20ob%C3%A9lisque.&text=O%C3%B9%20la%20masse%20volumique%20du,coefficient%20de%20portance%20CL.

9 Annexe

9.1 Ecoulement autour d'un et de 5 cylindres

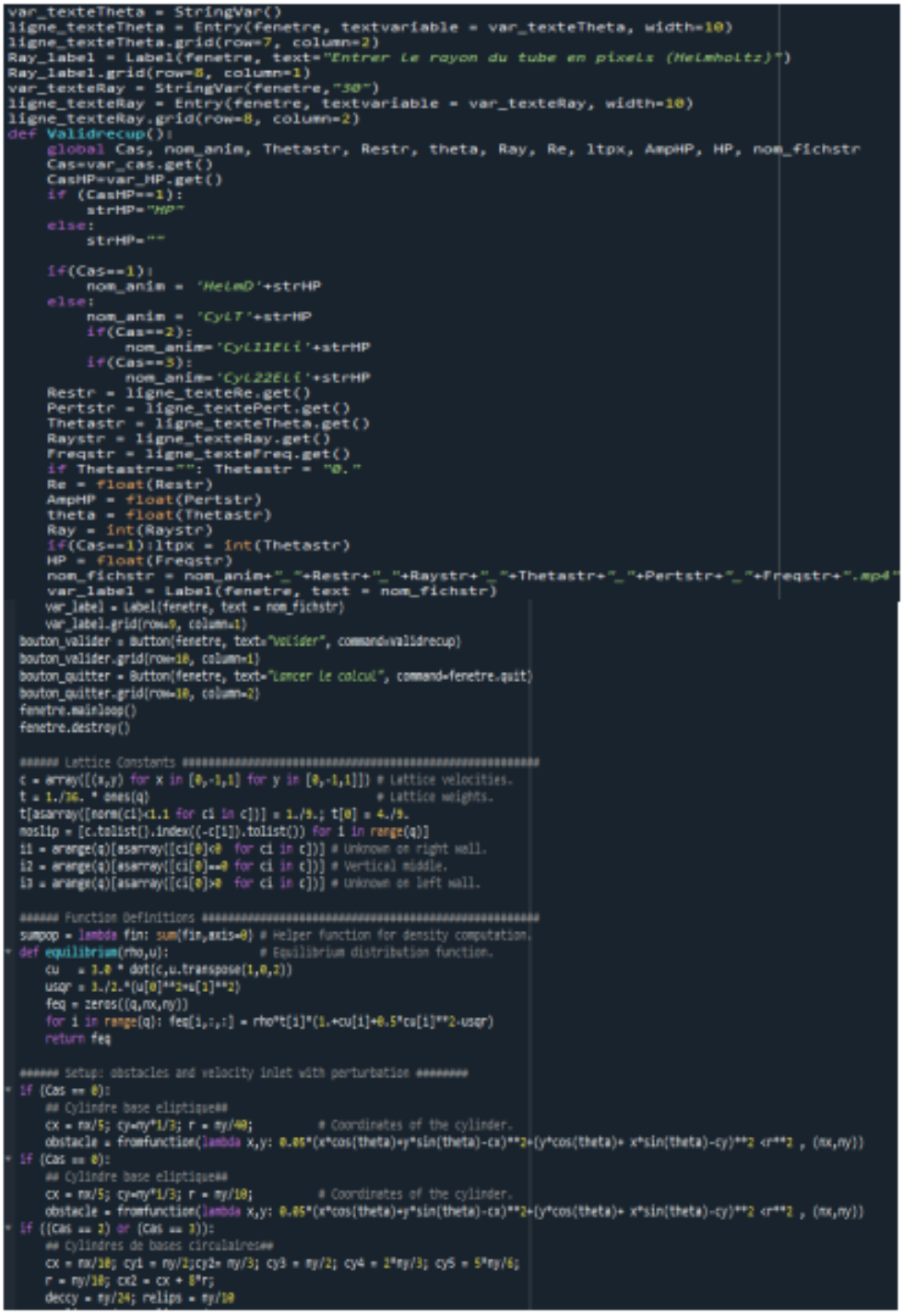
Dans les deux cas, le nombre de Reynolds est 100.

```
from numpy import *;
from numpy.linalg import *;
from tkinter import *
import matplotlib.pyplot as plt;
from matplotlib import cm
from matplotlib import animation

###### Flow definition #########################################################

maxIter = 29000 # Total number of time iterations initialement 200000.
Re      = 100.0  # Reynolds number.
nx = 780; ny = 360; ly=ny-1.0; q = 9 # Lattice dimensions and populations.
uLB     = 0.04                       # Velocity in lattice units.
theta=0.
nom_anim = "DefaultETU2020"
Cas=0
d=0.
#pert=1.e-4
#pert=0.
Ray = 30
beta = 4
cs = 1

##### Fenêtre de saisie ####

fenetre = Tk()
fenetre.geometry("800x250")
var_cas = IntVar()
var_HP = IntVar()
var_Ocyl=IntVar()
cas_cyl = Radiobutton(fenetre, text="Cylindre elipsoïdal", variable=var_cas, value=0)
cas_helm = Radiobutton(fenetre, text="Helmholtz", variable=var_cas, value=1)
cas_onzecyl=Radiobutton(fenetre, text="5 Cyl. circ.", variable=var_cas, value=2)
cas_vingtdeuxcyl=Radiobutton(fenetre, text="22 Cyl. circ.+Elipse", variable=var_cas, value=3)
cas_cyl.grid(row=1, column=1)
cas_helm.grid(row=1, column=2)
cas_onzecyl.grid(row=1,column=3)
cas_vingtdeuxcyl.grid(row=1,column=4)
cas_HP = Checkbutton(fenetre, text="HP", variable = var_HP)
cas_HP.grid(row=1, column=5)
# champ_label = Label(fenetre, text="Entrer le nom du fichier mp4")
# champ_label.grid(row=2, column=1)
# ligne_texteNF = Entry(fenetre, textvariable = var_texteNF, width=30)
# ligne_texteNF.grid(row=3, column=1)
Re_label = Label(fenetre, text="Entrer le Reynolds")
Re_label.grid(row=4, column=1)
var_texteRe = StringVar(fenetre,"220.")
ligne_texteRe = Entry(fenetre, textvariable = var_texteRe, width=10)
ligne_texteRe.grid(row=4, column=2)
Pert_label = Label(fenetre, text="Entrer l'amplitude de la perturbation du HP: ")
Pert_label.grid(row=5, column=1)
var_textePert = StringVar(fenetre,"0.04")
ligne_textePert = Entry(fenetre, textvariable = var_textePert, width=10)
ligne_textePert.grid(row=5, column=2)
Freq_label = Label(fenetre, text="Entrer la demi période de pulsation HP (multiple de 100): ")
Freq_label.grid(row=6, column=1)
var_texteFreq = StringVar(fenetre,"500")
ligne_texteFreq = Entry(fenetre, textvariable = var_texteFreq, width=10)
ligne_texteFreq.grid(row=6, column=2)
Theta_label = Label(fenetre, text="Entrer l'inclinaison du profil Theta ou la longueur du tube en pixels")
Theta_label.grid(row=7, column=1)
var_texteTheta = StringVar()
```

Figure 30: Paramètres principaux et fenêtre de saisie pour un cylindre

.

.

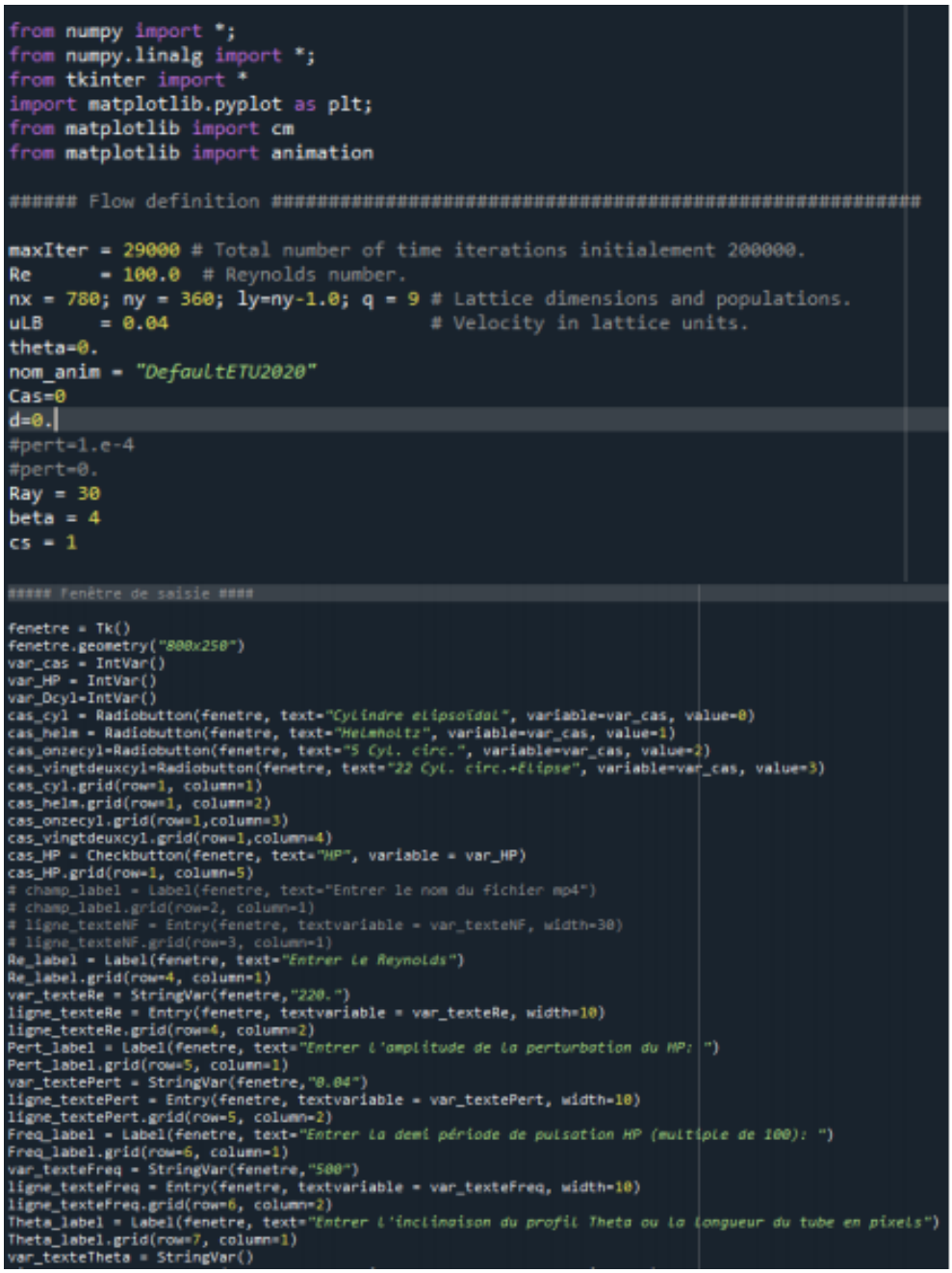


```
var_texteTheta = StringVar()
ligne_texteTheta = Entry(fenetre, textvariable = var_texteTheta, width=10)
ligne_texteTheta.grid(row=7, column=2)
Ray_label = Label(fenetre, text="Entrer le rayon du tube en pixels (Helmholtz)")
Ray_label.grid(row=8, column=1)
var_texteRay = StringVar(fenetre,"30")
ligne_texteRay = Entry(fenetre, textvariable = var_texteRay, width=10)
ligne_texteRay.grid(row=8, column=2)
def Validrecup():
    global Cas, nom_anim, Thetastr, Restr, theta, Ray, Re, ltpx, AmpHP, HP, nom_fichstr
    Cas=var_cas.get()
    CasHP=var_HP.get()
    if (CasHP==1):
        strHP="HP"
    else:
        strHP=""

    if(Cas==1):
        nom_anim = 'HelmD'+strHP
    else:
        nom_anim = 'CylT'+strHP
        if(Cas==2):
            nom_anim='Cyl11Eli'+strHP
        if(Cas==3):
            nom_anim='Cyl22Eli'+strHP
    Restr = ligne_texteRe.get()
    Pertstr = ligne_textePert.get()
    Thetastr = ligne_texteTheta.get()
    Raystr = ligne_texteRay.get()
    Freqstr = ligne_texteFreq.get()
    if Thetastr=="": Thetastr = "0."
    Re = float(Restr)
    AmpHP = float(Pertstr)
    theta = float(Thetastr)
    Ray = int(Raystr)
    if(Cas==1):ltpx = int(Thetastr)
    HP = float(Freqstr)
    nom_fichstr = nom_anim+"_"+Restr+"_"+Raystr+"_"+Thetastr+"_"+Pertstr+"_"+Freqstr+".mp4"
    var_label = Label(fenetre, text = nom_fichstr)
    var_label = Label(fenetre, text = nom_fichstr)
    var_label.grid(row=9, column=1)
bouton_valider = Button(fenetre, text="Valider", command=validrecup)
bouton_valider.grid(row=10, column=1)
bouton_quitter = Button(fenetre, text="Lancer le calcul", command=fenetre.quit)
bouton_quitter.grid(row=10, column=2)
fenetre.mainloop()
fenetre.destroy()

###### Lattice Constants ##################################################
c = array([(x,y) for x in [0,-1,1] for y in [0,-1,1]]) # Lattice velocities.
t = 1./36. * ones(q)                                   # Lattice weights.
t[asarray([norm(ci)<1.1 for ci in c])] = 1./9.; t[0] = 4./9.
noslip = [c.tolist().index((-c[i]).tolist()) for i in range(q)]
i1 = arange(q)[asarray([ci[0]<0  for ci in c])] # Unknown on right wall.
i2 = arange(q)[asarray([ci[0]==0 for ci in c])] # Vertical middle.
i3 = arange(q)[asarray([ci[0]>0  for ci in c])] # Unknown on left wall.

###### Function Definitions ###############################################
sumpop = lambda fin: sum(fin,axis=0) # Helper function for density computation.
def equilibrium(rho,u):              # Equilibrium distribution function.
    cu   = 3.0 * dot(c,u.transpose(1,0,2))
    usqr = 3./2.*(u[0]**2+u[1]**2)
    feq = zeros((q,nx,ny))
    for i in range(q): feq[i,:,:] = rho*t[i]*(1.+cu[i]+0.5*cu[i]**2-usqr)
    return feq

###### Setup: obstacles and velocity inlet with perturbation ########
if (Cas == 0):
    ## Cylindre base elliptique##
    cx = nx/5; cy=ny*1/3; r = ny/40;          # Coordinates of the cylinder.
    obstacle = fromfunction(lambda x,y: 0.05*(x*cos(theta)+y*sin(theta)-cx)**2+(y*cos(theta)+ x*sin(theta)-cy)**2 <r**2 , (nx,ny))
if (Cas == 0):
    ## Cylindre base elliptique##
    cx = nx/5; cy=ny*1/3; r = ny/10;          # Coordinates of the cylinder.
    obstacle = fromfunction(lambda x,y: 0.05*(x*cos(theta)+y*sin(theta)-cx)**2+(y*cos(theta)+ x*sin(theta)-cy)**2 <r**2 , (nx,ny))
if ((Cas == 2) or (Cas == 3)):
    ## cylindres de bases circulaires##
    cx = nx/10; cy1 = ny/2;cy2= ny/3; cy3 = ny/2; cy4 = 1*ny/3; cy5 = 5*ny/6;
    r = ny/10; cx2 = cx + 8*r;
    deccy = ny/24; relips = ny/10
```

Figure 31: LBM et définition des obstacles pour un cylindre

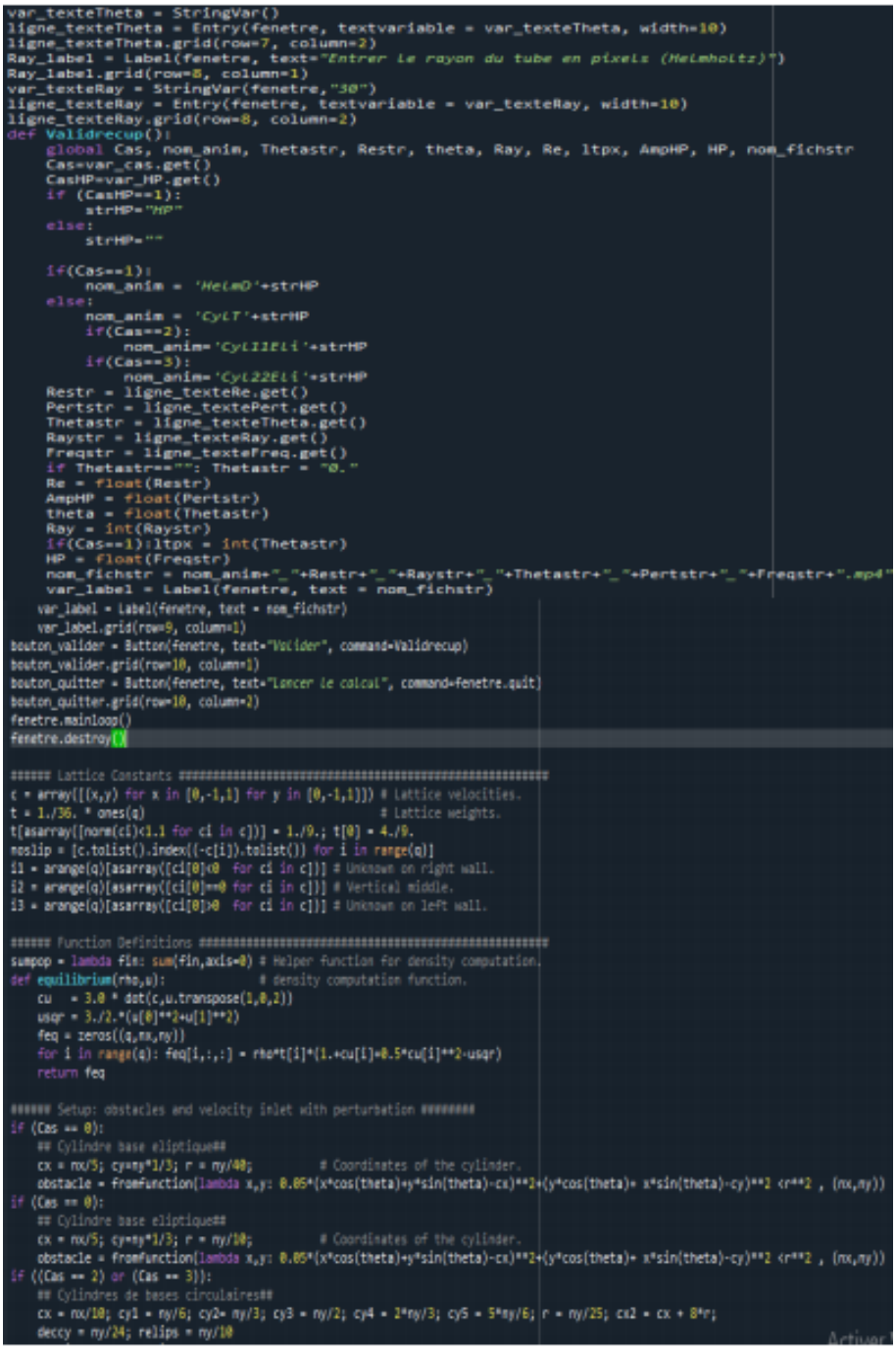
```
    deccy = ny/24; relips = ny/10
    cxelips=nx/2; cyelips=ny/2
    if (theta < -0.1):
        cyelips=ny/3
    # Coordinates of the cylinders.
    if(Cas == 2):
        obstacle = fromfunction(lambda x,y: (((x-cx)**2+(y-cy1)**2 <r**2)\
                                        | (y < 5) | (y > 355)), (nx,ny))

    if(Cas == 3):
        obstacle = fromfunction(lambda x,y: (((x-cx)**2+(y-cy1)**2 <r**2)\
                                        | ((x-cx)**2+(y-cy2)**2 <r**2) | ((x-cx)**2+(y-cy3)**2 <r**2)\
                                        | ((x-cx)**2+(y-cy4)**2 <r**2) | ((x-cx)**2+(y-cy5)**2 <r**2)\
                                        | ((x-cx2)**2+(y-cy1-deccy)**2 <r**2)\
                                        | ((x-cx2)**2+(y-cy2-deccy)**2 <r**2) | ((x-cx2)**2+(y-cy3-deccy)**2 <r**2)\
                                        | ((x-cx2)**2+(y-cy4-deccy)**2 <r**2) | ((x-cx2)**2+(y-cy5-deccy)**2 <r**2)\
                                        | (0.65*(x*cos(theta)+y*sin(theta)-cxelips)**2+(y*cos(theta)+ x*sin(theta)-cyelips)**2 <relips**2)\
                                        | (y < 5) | (y > 355)), (nx,ny))

if (Cas == 1):
    ## Helmholtz ##    Tx1 = 200; Tx2 = 220; Tx3 = Tx2 + ltpx; Ry1 = 160; Ry2 = 220; Ry3 = 156; Ry4 = 216; Ty2 = 4; Ty3 = 356 # Obstacle
    Tx1 = 200; Tx2 = 220; Tx3 = Tx2 + ltpx; Ry1 = 180 - Ray; Ry2 = 180 + Ray; Ry3 = Ry1 - 4; Ry4 = Ry2 - 4; Ty2 = 4; Ty3 = 356 # Obstacle
    #r = Tx3-Tx2 # Longueur du tube en pixels ####
    r = Ry2-Ry1 # Internal tube diameter in pixels
    #obstacle = fromfunction(lambda x,y: ((((x <= Tx2)&( x >= Tx1))&((y <= Ry1)|(y >= Ry2)))|((((y <= Ry1)&(y >= Ry3))|((y <= Ry2)&(y >= Ry4)))&((x >= Tx1)&
    obstacle = fromfunction(lambda x,y: ((((x < Tx2)&( x > Tx1))&((y < Ry1)|(y > Ry2)))|((((y < Ry1)&(y > Ry3))|((y < Ry2)&(y > Ry4)))&\
    ((x > Tx1)&(x < Tx3)))))|(((((xcTx2)&(y<Ty2))|((xcTx2)&(y>Ty3))))) , (nx,ny))
    omega0helmholtz = cs*sqrt(Ray*2/(ltpx*(Ty3-Ty2)*Tx1))
    print ('omega 0 résonnance de helmholtz unités LBM', omega0helmholtz)
nulb = uLB*r/Re; omega = 1.0 / (3.*nulb+0.5); # Relaxation parameter.
vel = fromfunction(lambda d,x,y: (1-d)*uLB,(2,nx,ny)) #*(1.0+pert*sin(beta*y/ly*2*pi))
feq = equilibrium(1.0,vel); fin = feq.copy()

###### Main time loop ##########################################################
fig = plt.figure()
ims = []
#Writer = animation.writers['ffmpeg']
#writer = Writer(fps=15, metadata=dict(artist='Patrick VIGLIANO'), bitrate=1800)

#################################
nom_fichstr=''
for time in range(maxIter):
    fin[i1,-1,:] = fin[i1,-2,:] # Right wall: outflow condition.
    rho = sumpop(fin)           # Calculate macroscopic density and velocity.
    u = dot(c.transpose(), fin.transpose((1,0,2)))/rho
    u[:,0,:] =vel[:,0,:] # Left wall: compute density from known populations.
    #u[:,0,:] =vel[:,0,:]*(1. + AmpHP * sin(time * pi / HP)) # Left wall: vitesse oscillante.
    #u[:,0,:] =vel[:,0,:]*(1. + AmpHP * sin(time * omega0helmholtz * 2 * pi)) # Left wall: vitesse oscillante pulsation résonnance?
    #u[:,0,:] =vel[:,0,:]*(1. + AmpHP * sin(time * omega0helmholtz))
     # Left wall: vitesse oscillante fréquence résonnance?
    rho[0,:] = 1./(1.-u[0,0,:]) * (sumpop(fin[i2,0,:])+2.*sumpop(fin[i1,0,:]))
    feq = equilibrium(rho,u) # Left wall: Zou/He boundary condition.
    fin[i3,0,:] = fin[i1,0,:] + feq[i3,0,:] - fin[i1,0,:]
    fout = fin - omega * (fin - feq)  # Collision step.
    for i in range(q): fout[i,obstacle] = fin[noslip[i],obstacle]
    for i in range(q): # Streaming step.
        fin[i,:,:] = roll(roll(fout[i,:,:],c[i,0],axis=0),c[i,1],axis=1)
    if ((time >= 1) & (time%100 == 1)): # Visualization
    #if (time%100 == 1): # Visualization
       # im=plt.imshow(((u[0]**2+u[1]**2)+rho*cs**2).transpose(),cmap='rainbow', animated=True) #pression
        im = plt.imshow(sqrt(u[0]**2+u[1]**2).transpose(),cmap=cm.jet, animated=True) #vitesse
        # if (time/100 == 1): plt.savefig("test.png")
        ims.append([im])
        #ani = animation.ArtistAnimation(fig, ims, interval=50, blit=True, repeat_delay=1000)
        #ani.save(nom_fichstr, writer=writer)
        print(time,' sur ', maxIter)
plt.show()
```

Figure 32: Boucle principale pour un cylindre



```
from numpy import *;
from numpy.linalg import *;
from tkinter import *
import matplotlib.pyplot as plt;
from matplotlib import cm
from matplotlib import animation

###### Flow definition ##########################################################

maxIter = 29000 # Total number of time iterations initialement 200000.
Re      = 100.0  # Reynolds number.
nx = 780; ny = 360; ly=ny-1.0; q = 9 # Lattice dimensions and populations.
uLB     = 0.04                       # Velocity in lattice units.
theta=0.
nom_anim = "DefaultETU2020"
Cas=0
d=0.
#pert=1.e-4
#pert=0.
Ray = 30
beta = 4
cs = 1

##### Fenêtre de saisie ####

fenetre = Tk()
fenetre.geometry("800x250")
var_cas = IntVar()
var_HP = IntVar()
var_Dcyl=IntVar()
cas_cyl = Radiobutton(fenetre, text="Cylindre elipsoïdal", variable=var_cas, value=0)
cas_helm = Radiobutton(fenetre, text="Helmholtz", variable=var_cas, value=1)
cas_onzecyl=Radiobutton(fenetre, text="5 Cyl. circ.", variable=var_cas, value=2)
cas_vingtdeuxcyl=Radiobutton(fenetre, text="22 Cyl. circ.+Elipse", variable=var_cas, value=3)
cas_cyl.grid(row=1, column=1)
cas_helm.grid(row=1, column=2)
cas_onzecyl.grid(row=1,column=3)
cas_vingtdeuxcyl.grid(row=1,column=4)
cas_HP = Checkbutton(fenetre, text="HP", variable = var_HP)
cas_HP.grid(row=1, column=5)
# champ_label = Label(fenetre, text="Entrer le nom du fichier mp4")
# champ_label.grid(row=2, column=1)
# ligne_texteNF = Entry(fenetre, textvariable = var_texteNF, width=30)
# ligne_texteNF.grid(row=3, column=1)
Re_label = Label(fenetre, text="Entrer le Reynolds")
Re_label.grid(row=4, column=1)
var_texteRe = StringVar(fenetre,"220.")
ligne_texteRe = Entry(fenetre, textvariable = var_texteRe, width=10)
ligne_texteRe.grid(row=4, column=2)
Pert_label = Label(fenetre, text="Entrer l'amplitude de la perturbation du HP: ")
Pert_label.grid(row=5, column=1)
var_textePert = StringVar(fenetre,"0.04")
ligne_textePert = Entry(fenetre, textvariable = var_textePert, width=10)
ligne_textePert.grid(row=5, column=2)
Freq_label = Label(fenetre, text="Entrer la demi période de pulsation HP (multiple de 100): ")
Freq_label.grid(row=6, column=1)
var_texteFreq = StringVar(fenetre,"500")
ligne_texteFreq = Entry(fenetre, textvariable = var_texteFreq, width=10)
ligne_texteFreq.grid(row=6, column=2)
Theta_label = Label(fenetre, text="Entrer l'inclinaison du profil Theta ou la longueur du tube en pixels")
Theta_label.grid(row=7, column=1)
var_texteTheta = StringVar()
```

Figure 33: Paramètres principaux et fenêtre de saisie pour 5 cylindres



```
var_texteTheta = StringVar()
ligne_texteTheta = Entry(fenetre, textvariable = var_texteTheta, width=10)
ligne_texteTheta.grid(row=7, column=2)
Ray_label = Label(fenetre, text="Entrer le rayon du tube en pixels (Helmholtz)")
Ray_label.grid(row=8, column=1)
var_texteRay = StringVar(fenetre,"30")
ligne_texteRay = Entry(fenetre, textvariable = var_texteRay, width=10)
ligne_texteRay.grid(row=8, column=2)
def Validrecup():
    global Cas, nom_anim, Thetastr, Restr, theta, Ray, Re, ltpx, AmpHP, HP, nom_fichstr
    Cas=var_cas.get()
    CasHP=var_HP.get()
    if (CasHP==1):
        strHP="HP"
    else:
        strHP=""

    if(Cas==1):
        nom_anim = 'HelmD'+strHP
    else:
        nom_anim = 'CylT'+strHP
        if(Cas==2):
            nom_anim='CylIIEli'+strHP
        if(Cas==3):
            nom_anim='Cyl22Eli'+strHP
    Restr = ligne_texteRe.get()
    Pertstr = ligne_textePert.get()
    Thetastr = ligne_texteTheta.get()
    Raystr = ligne_texteRay.get()
    Freqstr = ligne_texteFreq.get()
    if Thetastr=="": Thetastr = "0."
    Re = float(Restr)
    AmpHP = float(Pertstr)
    theta = float(Thetastr)
    Ray = int(Raystr)
    if(Cas==1):ltpx = int(Thetastr)
    HP = float(Freqstr)
    nom_fichstr = nom_anim+"_"+Restr+"_"+Raystr+"_"+Thetastr+"_"+Pertstr+"_"+Freqstr+".mp4"
    var_label = Label(fenetre, text = nom_fichstr)
var_label = Label(fenetre, text = nom_fichstr)
var_label.grid(row=9, column=1)
bouton_valider = Button(fenetre, text="Valider", command=Validrecup)
bouton_valider.grid(row=10, column=1)
bouton_quitter = Button(fenetre, text="Lancer le calcul", command=fenetre.quit)
bouton_quitter.grid(row=10, column=2)
fenetre.mainloop()
fenetre.destroy()

###### Lattice Constants ##################################################
c = array([[x,y] for x in [0,-1,1] for y in [0,-1,1]]) # Lattice velocities.
t = 1./36. * ones(q)                                  # Lattice weights.
t[asarray([norm(ci)<1.1 for ci in c])] = 1./9.; t[0] = 4./9.
noslip = [c.tolist().index((-c[i]).tolist()) for i in range(q)]
i1 = arange(q)[asarray([ci[0]<0  for ci in c])] # Unknown on right wall.
i2 = arange(q)[asarray([ci[0]==0 for ci in c])] # Vertical middle.
i3 = arange(q)[asarray([ci[0]>0  for ci in c])] # Unknown on left wall.

###### Function Definitions ###############################################
sumpop = lambda fin: sum(fin,axis=0) # Helper function for density computation.
def equilibrium(rho,u):              # Density computation function.
    cu   = 3.0 * dot(c,u.transpose(1,0,2))
    usqr = 3./2.*(u[0]**2+u[1]**2)
    feq = zeros((q,nx,ny))
    for i in range(q): feq[i,:,:] = rho*t[i]*(1.+cu[i]+0.5*cu[i]**2-usqr)
    return feq

###### Setup: obstacles and velocity inlet with perturbation ########
if (Cas == 0):
    ## Cylindre base eliptique##
    cx = nx/5; cy=ny*1/3; r = ny/40;          # Coordinates of the cylinder.
    obstacle = fromfunction(lambda x,y: 0.05*(x*cos(theta)+y*sin(theta)-cx)**2+(y*cos(theta)- x*sin(theta)-cy)**1 <r**2 , (nx,ny))
if (Cas == 0):
    ## Cylindre base eliptique##
    cx = nx/5; cy=ny*1/3; r = ny/10;          # Coordinates of the cylinder.
    obstacle = fromfunction(lambda x,y: 0.05*(x*cos(theta)+y*sin(theta)-cx)**2+(y*cos(theta)- x*sin(theta)-cy)**1 <r**2 , (nx,ny))
if ((Cas == 2) or (Cas == 3)):
    ## Cylindres de bases circulaires##
    cx = nx/10; cy1 = ny/6; cy2= ny/3; cy3 = ny/2; cy4 = 2*ny/3; cy5 = 5*ny/6; r = ny/25; cx2 = cx + 8*r;
    deccy = ny/24; relips = ny/10
```

Figure 34: LBM et définition des obstacles pour 5 cylindres

```
    deccy = ny/24; relips = ry/10
    cxelips=nx/2; cyelips=ny/2
    #if (theta < -0.1):
        #cyelips=ny/3
    # Coordinates of the cylinders.
    if(Cas == 2):
        obstacle = fromfunction(lambda x,y: (((x-cx)**2+(y-cy1)**2 <r**2)\
                                          | ((x-cx)**2+(y-cy2)**2 <r**2) | ((x-cx)**2+(y-cy3)**2 <r**2)\
                                          | ((x-cx)**2+(y-cy4)**2 <r**2) | ((x-cx)**2+(y-cy5)**2 <r**2)\
                                          | (y < 5) | (y > 395)), (nx,ny))
    if(Cas == 3):
        obstacle = fromfunction(lambda x,y: (((x-cx)**2+(y-cy1)**2 <r**2)\
                                          | ((x-cx)**2+(y-cy2)**2 <r**2) | ((x-cx)**2+(y-cy3)**2 <r**2)\
                                          | ((x-cx)**2+(y-cy4)**2 <r**2) | ((x-cx)**2+(y-cy5)**2 <r**2)\
                                          | ((x-cx2)**2+(y-cy1-deccy)**2 <r**2)\
                                          | ((x-cx2)**2+(y-cy2-deccy)**2 <r**2) | ((x-cx2)**2+(y-cy3-deccy)**2 <r**2)\
                                          | ((x-cx2)**2+(y-cy4-deccy)**2 <r**2) | ((x-cx2)**2+(y-cy5-deccy)**2 <r**2)\
                                          | (0.05*(x*cos(theta)+y*sin(theta)-cxelips)**2
                                            +(y*cos(theta)- x*sin(theta)-cyelips)**2 <relips**2)\
                                          | (y < 5) | (y > 395)), (nx,ny))

if (Cas == 1):
    ## Helmholtz ##    Tx1 = 200; Tx2 = 220; Tx3 = Tx2 + ltpx; Ry1 = 160; Ry2 = 220; Ry3 = 156; Ry4 = 216; Ty2 = 4; Ty3 = 356
    # Obstacle
    Tx1 = 200; Tx2 = 220; Tx3 = Tx2 + ltpx; Ry1 = 180 - Rwy; Ry2 = 180 + Rwy; Ry3 = Ry1 - 4; Ry4 = Ry2 - 4; Ty2 = 4; Ty3 = 356
    # Obstacle
    #r = Tx3-Tx2 # Longueur du tube en pixels ####
    r = Ry2-Ry1 # Internal tube diameter in pixels
    #obstacle = fromfunction(lambda x,y: ((((x <= Tx2)&( x >= Tx1))&((y <= Ry1)|(y >= Ry2)))|(((y <= Ry1)&(y >= Ry3)))
    #|(((y <= Ry2)&(y >= Ry4)))&(((x >= Tx1)&(x < Tx3)))|((((x<=Tx1)&(y<=Ty2))|((x<=Tx1)&(y>=Ty3)))) , (nx,ny))
    obstacle = fromfunction(lambda x,y: ((((x < Tx2)&( x > Tx1))&((y < Ry1)|(y > Ry2)))
                                        |(((y < Ry1)&(y > Ry3)))|((y < Ry2)&(y > Ry4)))&\
    ((x > Tx1)&(x < Tx3)))|((((x<Tx2)&(y<Ty2))|((x<Tx2)&(y>Ty3)))) , (nx,ny))
    omega0helmholtz = cs*sqrt(Rwy*2/(ltpx*(Ty3-Ty2)*Tx1))
    print ('omega 0 résonnance de Helmholtz unités LBM', omega0helmholtz)
nulb = uLB*r/Re; omega = 1.0 / (3.*nulb+0.5); # Relaxation parameter.
vel = fromfunction(lambda d,x,y: (1-d)*uLB,(2,nx,ny)) #*(1.0+pert*sin(beta*y/ly*2*pi))
feq = equilibrium(1.0,vel); fin = feq.copy()

###### Main time loop ##########################################################
fig = plt.figure()
ims = []
#writer = animation.writers['ffmpeg']
#writer = Writer(fps=15, metadata=dict(artist='Patrick VIDILIANO'), bitrate=1800)

####################################
nom_fichstr=''
for time in range(maxIter):
    fin[i1,-1,:] = fin[i1,-2,:] # Right wall: outflow condition.
    rho = sumpop(fin)           # Calculate macroscopic density and velocity.
    u = dot(c.transpose(), fin.transpose((1,0,2)))/rho
    u[:,0,:] =vel[:,0,:] # Left wall: compute density from known populations.
    #u[:,0,:] =vel[:,0,:]*(1. + AmpHP * sin(time * pi / HP)) # Left wall: vitesse oscillante.
    #u[:,0,:] =vel[:,0,:]*(1. + AmpHP * sin(time * omega0helmholtz * 2 * pi)) # Left wall: vitesse oscillante pulsation résonnance?
    #u[:,0,:] =vel[:,0,:]*(1. + AmpHP * sin(time * omega0helmholtz))
     # Left wall: vitesse oscillante fréquence résonnance?
    rho[0,:] = 1./(1.-u[0,0,:]) * (sumpop(fin[i2,0,:])+2.*sumpop(fin[i1,0,:]))
    feq = equilibrium(rho,u) # Left wall: Zou/He boundary condition.
    fin[i3,0,:] = fin[i1,0,:] + feq[i3,0,:] - fin[i1,0,:]
    fout = fin - omega * (fin - feq)  # Collision step.
    for i in range(q): fout[i,obstacle] = fin[noslip[i],obstacle]
    for i in range(q): # Streaming step.
        fin[i,:,:] = roll(roll(fout[i,:,:],c[i,0],axis=0),c[i,1],axis=1)
    if ((time >= 1) & (time%100 == 1)): # Visualization
    #if (time%100 == 1): # Visualization
        # im=plt.imshow(((u[0]**2+u[1]**2)+rho*(s**2)).transpose(),cmap='rainbow', animated=True) #pression
        im = plt.imshow(sqrt(u[0]**2+u[1]**2).transpose(),cmap=cm.jet, animated=True) #vitesse
        # if (time//100 == 1): plt.savefig("test.png")
        ims.append([im])
        #ani = animation.ArtistAnimation(fig, ims, interval=50, blit=True, repeat_delay=1000)
        #ani.save(nom_fichstr, writer=writer)
        print(time,' sur ', maxIter)
plt.show()
```

Figure 35: Boucle principale pour 5 cylindres