REPUBLIQUE ALGERIENNE
DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE
LA RECHERCHE SCIENTIFIQUE

**Ecole Nationale Polytechnique**
**Laboratoire de Recherches Sciences de l'Eau**

**Département Hydraulique**
Mémoire d'obtention du diplôme de

**MASTER EN HYDRAULIQUE**

Réalisé par
Mr **ZARAT Adel**

Thème :

# Application du parallélisme du code Palabos sur les écoulements turbulents dans les conduits à motifs périodiques

**Proposé et dirigé par :** Mr F. MEZALI
Dr S. BENMAMAR

**Promotion 2014**

*ENP, 10 Avenue Hassan Badi, BP.186 EL HARRACH, ALGER*

# *Dédicaces*

*Avant de commencer mes dédicaces, je suis certain que ce projet de fin d'étude est la meilleure chose qui puisse m'arrive durant cette année, j'en suis sûr, même en mi- année.*

*Grâce à Dieu nous avons réalisé ce travail que je dédie.*

*À mes parents, pour leurs aides appréciables qui ont tout fait pour que j'atteigne ce niveau.*

*Je le dédie également à mes frères qui ont été avec moi pour leur soutien moral et matériel.*

*À mes adorables sœurs et surtout mes anges sousou et marwa.*

*Je le dédie aussi à mes ami(e)s de polytechnique et d'autres pour leurs soutien et les moments inoubliables que nous avons passé tous ensemble, désole de ne pas pouvoir citer vos noms car vous êtes assez nombreux et que je crains d'oublier quelqu'un, et tous ceux avec qui je partage de gratitude, l'amitié, l'amour et le respect.*

# Remerciement

*En premier et en dernier, Avant tout et après tout, Dans le bonheur et dans le malheur, Merci à Allah le tout puissant pour toutes ses grâces et ses faveurs, pour nous avoir accordé la chance d'étudier à Ecole Nationale Polytechnique et de nous avoir donné le courage et la force d'accomplir ce modeste travail.*

*Nous remercions,* ***Mlle Benmamar Saâdia,*** *notre encadreur, pour le sujet qu'elle nous a proposé qui nous a passionné, mais également pour sa précieuse aide et ses conseils au cours de cette année. elle a su nous laissé la liberté nécessaire à l'accomplissement de notre tâche, tout en y gardant un œil critique et avisé malgré son emploi du temps chargé pendant toute l'année.*

*Nous tenons également à remercier* ***Mr. Mezali Farouk,*** *qui nous a éclairé et nous a mis dans la bonne voie tout au long de notre travail, ainsi que pour ses remarques et ses encouragements qui nous étaient de grande valeur.*

*A celui qui n'a jamais hésité le moindre instant à se tenir à nos côtés afin de nous soutenir, nous aider et nous encourager.*

*Nous exprimons nos vifs remerciements aux membres de Jury qui nous feront l'honneur d'apprécier ce mémoire de fin d'étude.*

*Et enfin, un grand remerciement destiné à nos enseignants et enseignantes, eux qui ont contribué à notre formation, depuis le cycle primaire jusqu'au cursus universitaire.*

## ملخص

*هذه الأطروحة هو جزء من جهد أوسع لتقييم محاكاة التدفق المضطرب في الأنابيب من نوع نمط الدوري متقاربة متباعدة أهداف هذه الأطروحة هو جعل الموازاة للحد من الوقت اللازم للحساب في الأنابيب عندما تكون الهندسة معقدة بحيث نقوم بوضع برنامج حساب يقوم على أحد أساليب بولتزمان*


***الكلمات المفتاحية*** *: قناة بأبعاد متغيرة؛ الموازاة ؛ بالابوس؛ الشروط النهائية؛ جريان مضطرب*

---


### *Résumé*

*Ce mémoire s'inscrit dans un effort plus large visant à évaluer et caractériser l'apport potentiel de la simulation des écoulements turbulents dans une conduite à motif périodique de type convergent-divergent.*

*L'objectif de ce mémoire est de faire une parallèlisation pour la réduction du temps de calcul lors de la simulation des écoulements turbulent dans des conduits à géométrie variable, en a utilisés un code open source. Ce dernier est basé sur la méthode de lattice Boltzmann.*


***Mots clés:*** *conduite à géométrie variable, Parallelisation, Palabos, conditions aux limites, écoulement turbulent.*

---


### *Abstract*

*This thesis is part of a broader effort to evaluate and characterize the potential contribution of the simulation of turbulent flows in a pipe with periodic pattern of convergent-divergent.*

*The objective of this paper is to make a parallelization to reduce computation time in the simulation of turbulent flows in ducts with variable geometry, has used an open source code. The latter is based on lattice Boltzmann method.*


***Key words:*** *driving variable geometry, parallelization, Palabos, boundary conditions, turbulent flow.*

# Table des matières

# Liste des figures



# Liste des tableaux

# Abréviations

| | |
|---|---|
| **CPU** | Utilisateur du calcul parallèle |
| **DNS** | Simulation numérique direct (Domain Name System) |
| **MIMD** | Multiple Instructions, multiples données |
| **MPI** | Message Passing Interface |
| **MPMD** | Multiple Instruction, Multiples Data |
| **RAM** | Mémoire à accès sélectif |
| **SMP** | Symmetric MultiProcessors |
| **SPMD** | Programme simple à multiples données |

# Notations

| **Symbole** | | **Définition** |
|---|---|---|
| A | ($m/s^2$) | L'accélération |
| *E* | (%) | L'efficacité |
| Nx | | Nombre de discrétisations suivant l'horizontale |
| Ny | | Nombre de discrétisations suivant la verticale |
| Np | | Nombre des processeurs |
| P | | Le processeur |
| Re | | Nombre de Reynolds |
| T | (s) | Temps de calcul |

# Introduction générale

L'étude de l'écoulement des fluides par simulation numérique est une discipline en plein essor, elle repose sur la recherche de solution des équations qui décrivent la dynamique des fluides par les algorithmes appropries. Les simulations numériques ont deux types de finalité :

La première finalité est la réalisation d'étude à caractère fondamental, destinées à permettre une meilleur discipline de mécanisme physique de base qui régissent, la dynamique des fluides, en vu de leur compréhension, de leur modélisation et antérieurement de leur contrôle. Ces études requièrent une très grande précision de donnée fournies par la simulation numérique. Ceci implique que le modèle physique choisi pour représenter le comportement du fluide soit pertinent et que les algorithmes de résolution employés, ainsi que leur mise en œuvre informatique, n'introduisent qu'un faible niveau d'erreur. La qualité des informations fournies par la simulation numérique est également subordonnée ou niveau de résolution choisi: pour obtenir la meilleur précision possible, la simulation dois tenir compte de toutes les échelles spatio-temporelle qui contribuent à la dynamique de l'écoulement. Lorsque la gamme d'échelle est très large, ce qui est par exemple le cas pour les écoulements turbulents, le problème devient raide, en ce sens que le rapport entre les échelles caractéristiques associer ou plus grand et aux plus petites échelles devient très grand.

La seconde finalité concerne les études d'ingénierie, qui, pour la conception des matérielles, nécessitent la prévision de leur caractéristiques. Il s'agit ici non plus de produire des données en vue de l'analyse de la dynamique de l'écoulement, mais aussi de prédire certaine de ces caractéristique, ou plus précisément la valeur des paramètres physique qui en dépendent. Ces prédictions peuvent porter soit par les valeurs moyennes de ces paramètres, soit sur leurs valeurs extrêmes.

Devant la complexité des écoulements industriels dans les milieux à géométrie complexe il est nécessaire alors de faire appel à ces techniques de simulation et de modélisation de la turbulence.

Notre travail consiste à l'amélioration d'un outil numérique basé sur la méthode de Lattice

Boltzmann et faire une étude de performance pour réduire le temps de calcul afin de simuler l'écoulement turbulent. Notre mémoire structuré en trois chapitres.

En premier, nous avons abordé brièvement les généralités sur les écoulements turbulents dans des conduits à géométrie variable de type convergent-divergent ainsi les travaux faits sur les écoulements dans les conduites à motif périodique.

Le deuxième chapitre est consacré à la définition de la géométrie et aux équations de base régissant l'écoulement, une présentation de quelques détails sur le code que nous avons élaboré en C++ ainsi les différentes commandes d'excusions et les ressources informatiques utilisés y sont également exposés.

Le troisième et le dernier chapitre présente dans une première section quelques détails sur l'architecture des ordinateurs parallèles ainsi que les modèles utilisés lors de la simulation, une brève présentation des caractéristiques de la machine, ensuite les différentes conditions de la simulation numérique directe et le cout informatique, afin de terminer par la présentation de test de performance et l'analyse des résultats obtenu par la simulation.

**Chapitre I**

# Les écoulements turbulents dans les conduits à géométrie complexe

Lorsqu'un fluide s'écoule dans une conduite il exerce une résistance visqueuse qui crée une perte d'énergie. La chute de pression le long de la conduite s'appelle la perte de charge. Pour des conditions aux limites constantes imposées à l'écoulement, O. Reynolds a montré en 1883 qu'il existe deux sortes d'écoulements suivant la valeur d'un nombre sans dimension appelé nombre de Reynolds est noté $R_e$:

$$R_e = \frac{DV}{\upsilon}$$

Où, *V* est une vitesse typique de l'écoulement, *D* le diamètre de la conduite et υ la viscosité cinématique du fluide. Lorsque le nombre de Reynolds est faible, les lignes de courant sont stationnaires et l'écoulement est dit **Laminaire.** Au contraire lorsque le nombre de Reynolds est grand, les lignes de courant deviennent instationnaires et l'écoulement est dit **Turbulent**. A ces deux types d'écoulement fondamentalement différents, correspondent des pertes de charge différentes.

Le calcul de ces écoulements dans les conduites de forme circulaire ou non circulaire, est fréquemment rencontré dans la pratique de l'ingénieur hydraulicien. Les applications sont nombreuses et nous pouvons citer, à titre d'exemple, le cas de la conduite de refoulement depuis une station de pompage vers un réservoir d'alimentation d'une agglomération ou celui de la conduite gravitaire alimentant, pour des besoins énergétiques, et une usine hydroélectrique.

L'écoulement dans une conduite de forme circulaire ou non circulaire en charge est gouverné par trois principales relations qui sont les relations de *Darcy-Weisbach* (1845, 1854), de *Colebrook-White* (1939) et du nombre de Reynolds. Ces trois relations forment le système d'équations de base destiné au calcul de l'écoulement turbulent en conduites.

La première relation exprime le gradient de la perte de charge linéaire appelé aussi perte de frottement. Celui-ci dépend du coefficient de frottement, du débit, volume écoulé et des caractéristiques géométriques de l'ouvrage.

La seconde relation exprime le coefficient de frottement en fonction du nombre de *Reynolds* caractérisant l'écoulement et de la rugosité relative de la conduite considérée. Elle a été

proposée pour être appliquée au cas des conduites de commerce où la répartition locale des éléments de rugosité est accidentelle, par opposition à une répartition artificielle.

La troisième relation, où le nombre de Reynolds, traduit les effets des forces d'inertie et de viscosité. Lorsque les forces d'inertie sont prépondérantes, l'écoulement est caractérisé par une vitesse élevée et son régime est en règle générale dans le domaine de pleine turbulence. Le coefficient de frottement prend alors une valeur quasi constante et ne dépend que de celle de la rugosité relative. Par contre, lorsque les forces de viscosités sont prépondérantes, l'écoulement est caractérisé par une vitesse moins élevée et son régime appartient souvent au domaine de transition. Le coefficient de frottement dépend alors à la fois de la rugosité relative et du nombre de *Reynolds*. Pour les conduites à parois lisses ou pratiquement lisses, le coefficient de frottement ne dépend que de la valeur de la rugosité relative caractérisant l'état des parois internes de l'ouvrage.

Notre travail consiste à vérifier la loi dite des pertes de charges dans le cas d'un écoulement turbulent dans des conduits à motif périodique, et de trouver par une étude de performance parallèle comment peut-on faire réduire le temps de calcul pour une simulation d'un écoulement turbulent dans une conduite composée de successions de convergents et de divergents.

## I.1. Ecoulement turbulent

Les écoulements turbulents se caractérisent donc par une apparence très désordonnée, un Comportement non prévisible et l'existence de nombreuses échelles spatiales et temporelles. Ils apparaissent lorsque la source d'énergie cinétique qui met le fluide en mouvement est relativement intense devant les forces de viscosité que le fluide oppose pour se déplacer. L'inverse est le régime laminaire pour lequel l'écoulement est prévisible et régulier. C'est à la fin du *XIXe* siècle que **Boussinesq** et **Reynolds** ont effectué les premières études sur le régime turbulent dans les fluides. La source d'énergie dans un écoulement turbulent peut Prendre plusieurs formes allant de perturbations non locales comme un gradient de pression dans un écoulement en canal, une différence de température ou de densité à des perturbations plus localisées comme des rugosités de paroi, des sources acoustiques extérieures, des sources Impulsionnelles liées à des forçages mécaniques ou des changements géométriques, bien d'autres sources d'énergie encore (électromagnétique, chimique, etc...) **[Robinet, 2010].**

En Hydraulique l'écoulement turbulent est celui que l'on rencontre dans une conduite cylindrique ou bien non cylindrique et qui correspond à des valeurs du nombre de Reynolds très élevées et à des valeurs de rugosité relative comprises en 0 et 0,05.

Trois régimes de l'écoulement turbulent peuvent donc être observés dans une conduite cylindrique. Il s'agit des domaines lisses, de transition et de pleine turbulence appelé aussi domaine du régime d'écoulement turbulent rugueux correspondant souvent aux fortes valeurs de la rugosité relative.

Ces trois régimes d'écoulement sont traduits graphiquement sur le diagramme universellement connu de *Moody* (1944), (figure I.1).

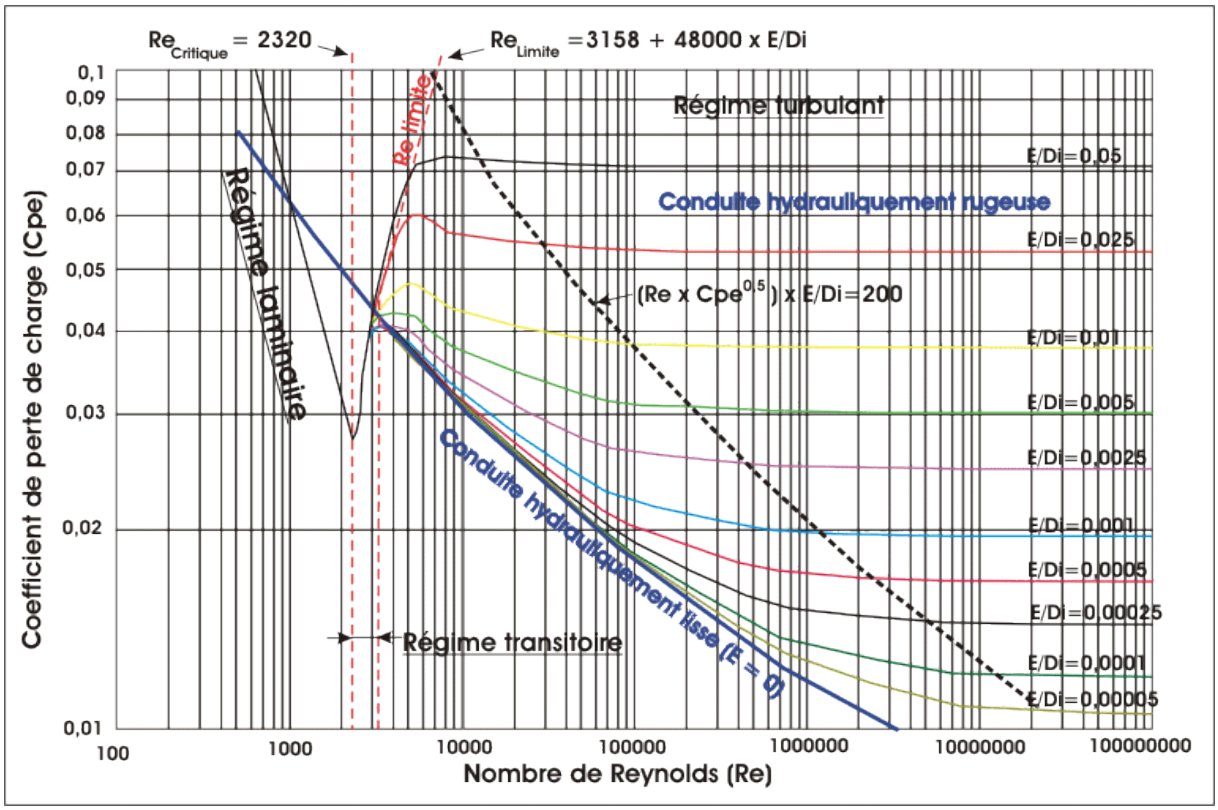


**Figure I. 1:** diagramme de Moody

Les résultats expérimentaux, décrits par l'abaque de Moody, montrent qu'il existe une plage de variation des paramètres dans laquelle le coefficient de perte de charge linéaire ne dépend pas du nombre de Reynolds, c'est la partie droite de l'abaque, dans lesquelles les courbes iso $\varepsilon / D$ . Sont presque horizontales. Dans cette zone, comme souvent en turbulence établie, les variations de nombre de Reynolds n'ont plus d'influence sur la nature des phénomènes.

Dans une autre plage, au voisinage de la courbe du régime lisse, $\lambda_t$ ne dépend que du nombre de Reynolds, toutes les iso $\varepsilon / D$ sont confondues, cette zone correspond aux cas où l'épaisseur de la couche visqueuse est supérieure à la taille moyenne des rugosités de paroi.

## I.2. Les écoulements turbulents dans les conduites à géométrie complexe

Différentes approche ont été utilisées simultanément pour étudies l'influence des ondulations de faible amplitude sur l'écoulement turbulent en conduite. Ces études, en faisant ressortir le caractère complexe de l'écoulement étudié, ont montré que les résultats classiques relatifs à

l'écoulement turbulent établi en tube droit étaient profondément modifiés particulièrement à cause de l'effet de variation de la section, de l'effet de courbure des parois et de la présence de courants de retour instationnaires derrière chaque crête.

### I.2.1. Travaux de Gschwind et Kotkke (2000)

Gschwind et Kotkke ont étudié expérimentalement et numériquement les effets de transfert de la chaleur et de masse ainsi que les pertes de pression dans des conduites avec axes symétriques (FigureI.1) et asymétriques (figure I.2)

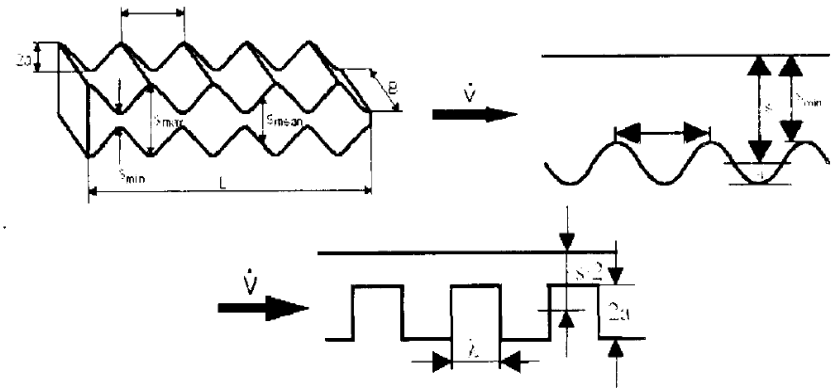

**Figure I. 2 :** conduites à motifs périodiques symétriques

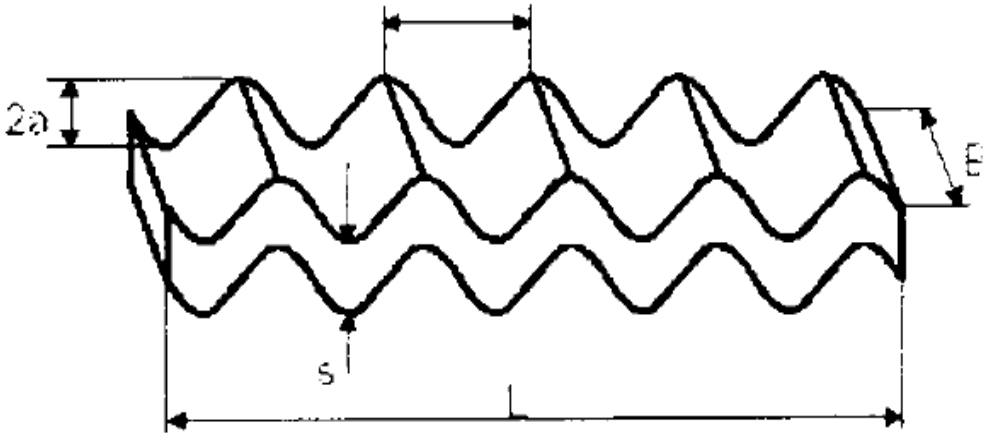


**Figure I. 3 :** conduite à motif périodique asymétrique

Ils ont remarqué des instabilités et formation de vortex dans les surfaces concaves, et un transfert de la chaleur et de la masse fortement non homogène à travers la largeur du conduit **[Sayoud, 2004].**

### I.2.2. Travaux de Luo (2003)

En 2003, Luo a étudié expérimentalement les écoulements autours des obstacles carrés, employant comme méthode de mesure un colorant fluorescent et la visualisation laser. Il a

déterminé le nombre Reynolds critique. Il a observé qu'au fur et à mesure que le nombre de Reynolds augmente, le caractère onduleux de sillage augmente, et à partir d'un nombre de Reynolds plus Elevé, le sillage derrière l'obstacle devient complètement déformé, et les longueurs d'ondes sont de plus en plus faibles **[Luo, 2003].**

### I.2.3. Travaux de Benmamar (2006)

Devant la complexité des écoulements dans les conduit à géométrie complexe, l'auteur a réalisé une expérimentation sur maquettes, afin d'observer les écoulements dans un conduit à motifs périodiques **[Benmamar, 2006].**

Pour visualiser les phénomènes hydrodynamiques existants dans les écoulements dans des conduits à géométrie complexe, Benmamar a conçu un modèle de conduite en acier à motif périodique de dimension 4,0cm x 4,0cm x 79,1cm. Les expériences ont été effectuées en écoulement en charge.

Elle a proposé de relever pour un débit donné, la pression sur la face supérieure du modèle, en installant sur chaque crête un piézomètre. Elle a constaté que pour un débit constant la pression diminue périodiquement le long de la conduite et elle a aussi évalué numériquement la distribution de la vitesse le long de la conduite à 4 motifs.

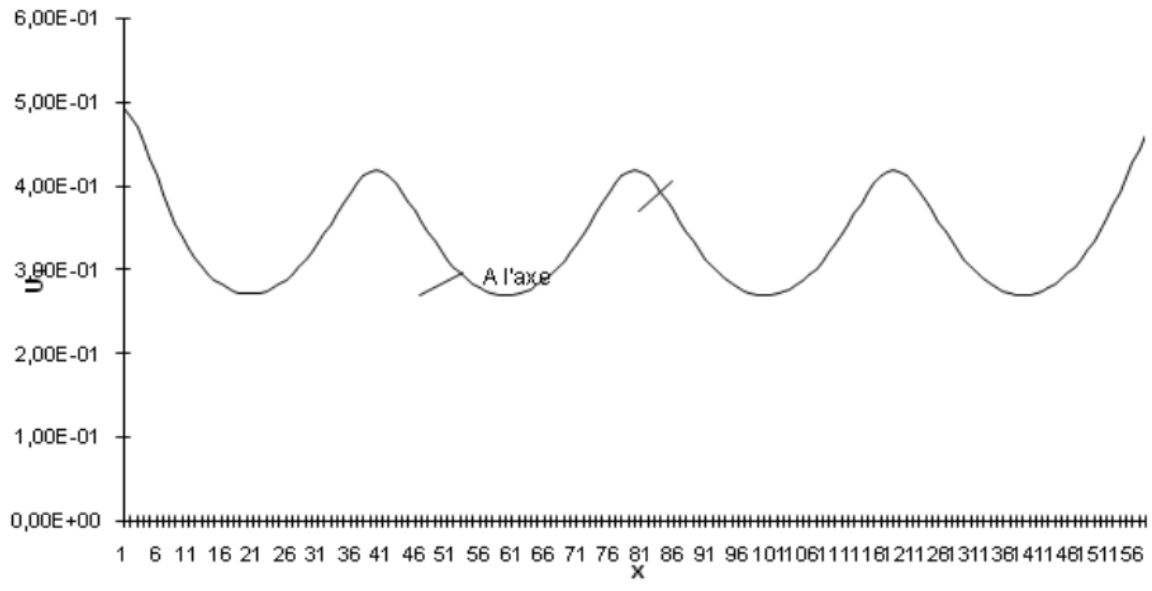


**Figure I. 4 :** Distribution de la vitesse dans une conduite à quatre motifs

**[Benmamar et al, 1995]**

### I.2.4. Travaux de Takafumi (2006)

Takafumi a effectué une étude numérique bidimensionnelle de l'écoulement autour d'un cylindre circulaire, en utilisant la modèle DES (Detached Eddy Simulation). Les résultats obtenus par la DES a prévu la cessation du décollement de tourbillon derrière le cylindre, et même résultat a été obtenu en utilisant la méthode simulation RANS (Reynolds Average Numerical Simulation) **[Takafumi, 2006].**

### I.2.5. Travaux de Cheng (2006)

Cheng a simulé un écoulement de cisaillement linéaire incompressible bidimensionnel au-dessus d'un tube carré. Il a montré l'effet du taux de cisaillement sur la fréquence du décollement de tourbillon du cylindre. Les résultats obtenus montrent que le vortex derrière le cylindre dépend fortement du taux de cisaillement et du nombre de Reynolds **[Cheng, 2006].**

### I.2.6. Travaux de Belakroum (2007)

Belakroum a étudié par la méthode des éléments finis, le modèle LES (Large Eddy Simulation) pour simuler l'écoulement instationnaire et turbulent d'un fluide incompressible autour d'un cylindre. Il a trouvé que le phénomène d'éclatement tourbillonnaire est nettement mis en évidence. Et c'est ce qu'on va vérifier pour une succession de convergent-divergent **[Belakroum, 2007].**

### I.2.7. Travaux de Khabbouchi et Guellouz (2008)

Khabbouchi et Guellouz ont effectué des mesures pa r PIV (vélocimétrie par images de particules) dans la zone du sillage proche derrière un cylindre placé près d'une paroi au niveau de son bord d'attaque. La configuration géométrique a permis d'isoler l'effet de l'écoulement type jet qui s'installe dans l'espacement entre la paroi et le cylindre. Ils ont montré l'existence de trois régions différentes d'écoulement lorsque le cylindre se rapproche de la paroi. L'effet de l'écoulement type jet se manifeste dans les faibles rapport- espace en détruisant la couche de cisaillement inférieure et empêchant, par la suite l'allée de Von Karman de s'installer dans le sillage **[Khabbouchi & Guellouz, 2008].**

### I.2.8. Travaux de Lam et Zou (2009)

Ils ont étudié numériquement et expérimentalement les écoulements turbulents autour de quatre cylindres dans une configuration carrée intégrée avec différents rapports d'espacement choisis. Les résultats obtenus de la fluctuation de vitesse sont avérés similaires aux résultats numériques **[Lam & Zou, 2009].**

### I.2.9. Travaux de Peixinho (2012)

L'écoulement dans un tube faiblement divergent (c'est-à-dire un tube cylindrique dont le diamètre augmente faiblement le long de l'axe de l'écoulement), tel que celui représenté dans

la Figure (I.6) n'est pas suffisamment documenté. Cet écoulement est observé dans de nombreuses applications, par exemple, dans le système circulatoire sanguin ou encore lorsque l'on utilise des pipettes. En effet, en mode prélèvement, le liquide, généralement peu visqueux, s´écoule le long de la pointe de la pipette dans un tube faiblement divergent. Le nombre de Reynolds, Re, basé sur le petit diamètre ou le diamètre de l'entrée, est de l'ordre de quelques centaines.

Le problème bidimensionnel de la stabilité de l´écoulement entre deux parois planes qui se coupent avec un angle, α, en un point source est appelé le problème de Jeffery-Hamel. Ce problème a été étudié théoriquement et numériquement. Des bifurcations, qui indiquent un ensemble de solutions avec des alignements de tourbillons le long de l'axe de l'écoulement, ont été trouvées. D'autres travaux ont traité le cas de l'écoulement dans un élargissement brusque (avec des coins à 90°) et il a été montré que l'écoulement devient dissymétrique à partir de Re = 80 Dans le cas de tubes de section circulaires et avec une expansion brusque, il apparait une bifurcation supercritique autour de Re = 1000.

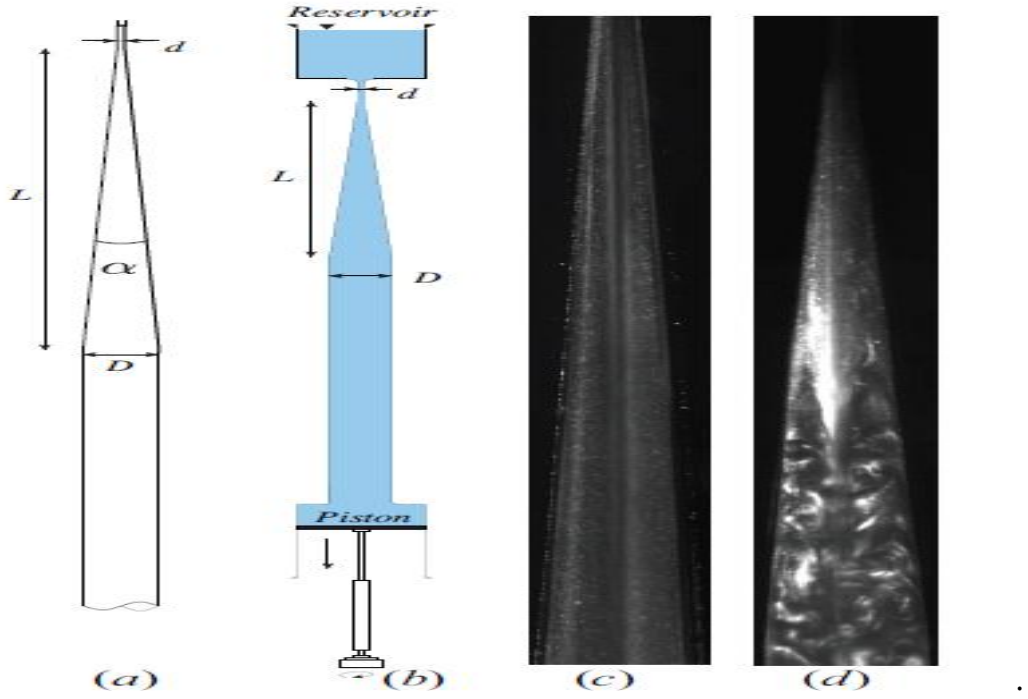


**Figure I. 5 :** schéma du dispositif expérimental et visualisation d'une bouffée turbulente

Les résultats consistent en l'analyse de photographies de visualisations d'écoulements. Pour de faibles débits, toutes les particules de fluides se déplacent dans le sens de l'écoulement. On pense alors à un écoulement laminaire type Poiseuille dont l'amplitude ou la vitesse maximale diminue fortement au fur et à mesure que l'on se déplace le long de la section divergente. Lorsque l'on augmente le débit, une recirculation, qui a une forme annulaire et étendue le long de l'axe de l'écoulement, apparait. En observant le point de d'écoulement de couche

limite, des estimations quantitatives de la taille de la recirculation peuvent être obtenues. En résumé, aucune recirculation n'est observée pour Re < 500 dans le divergent de 4° et pour Re < 700 dans le divergent de 6°.

Avec une nouvelle augmentation du débit, la recirculation peut, dans certain cas, donner naissance à des bouffées turbulentes comme celle représentée sur la **(Figure I.7)**. Ces taches localisées turbulentes ont certaines similitudes avec les puffs turbulents observes dans les conduites rectilignes. Par exemple, elles ont des longueurs précises pour un Re donne et une onde en déclin à l'avant. Ces taches turbulentes sont localisés et d´dépendent de perturbations ou imperfections du système.

Une deuxième série de résultats concerne des expériences de rélaminarisation. En pratique, on établit un écoulement à fort nombre de Reynolds, puis on diminue le débit afin de rélaminariser la bouffée turbulente. Les temps de rélaminarisation sont mesurés et utilisés pour construire les diagrammes décrivant la transition dans des tubes divergents. En résumé, le domaine sous-critique pour l'apparition de taches turbulents est de Re = 720 à 860 dans le

divergent de 4° et entre Re = 800 et 1000 dans le tube de 6°. Au cours de la rélaminarisation d'une tache turbulente, un jet de liquide passe au travers de la tache turbulente. Le jet de liquide ondule et des oscillations similaires de jet liquide dans une section divergente ont été observées dans des expériences en micro fluidiques **[Peixinho, 2012].**

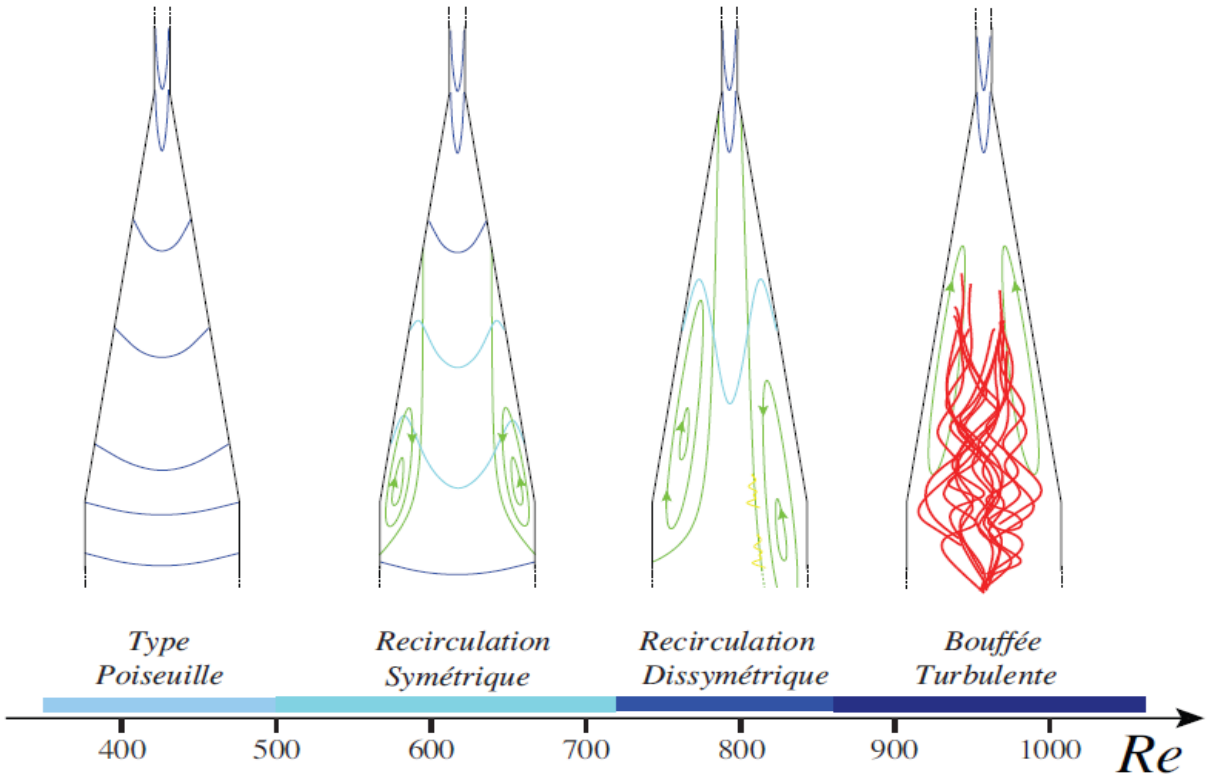


**Figure I. 6 :** diagramme de stabilité décrivant les différentes structures d'écoulements

Dans les schémas, les lignes bleues représentent les profils de vitesse et les lignes vertes représentent des lignes de courant. L'écoulement est du haut vers le bas.

### I.2.10. Travaux de Bouffenech et Djamai (2012)

Bouffenech et Djamai ont fait une simulation d'un Ecoulement turbulent dans une conduite à 4 motifs périodiques type convergent-divergent de longueur L (Figure I.7).

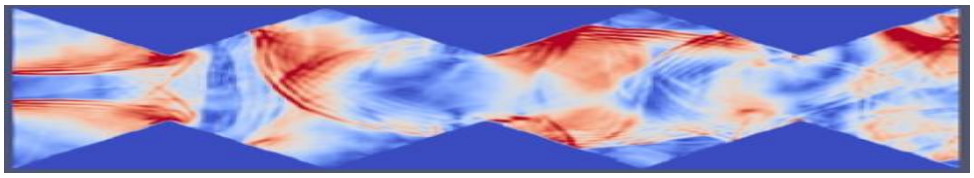

**Figure I. 7 :** Rouleaux de recirculations pour un écoulement turbulent dans une conduite à motif périodique **[Bouffenech et Djamai, 2012]**

Ils ont évalué la vitesse dans différents régions de la conduite à l'entré, près de la paroi et à la sortie. Pour deux nombres de Reynolds $R_e$ =10000 et $R_e$ = 30000. Ils ont constaté que pour un *Re* égale à 10000, le comportement de la vitesse à l'entrée est caractérisé par une perturbation suivie d'une oscillation autour d'une valeur constante. Ce n'est pas le cas sur la paroi ni à la sortie, qui sont caractérisées par l'apparition de vecteurs vitesses de sens opposé (ayant une composante négative) et qui représentent les rouleaux de recirculations.

Pour un $R_e$ égale à 30000 le comportement de la vitesse est le même à l'entrée, à la sortie et sur la paroi, il est caractérisé par deux facteurs importants, une oscillation autour d'une certaine valeur et l'apparition des vecteurs vitesses de sens opposés. Cet écoulement est assez chaotique, ce qui se manifeste par l'apparition des rouleaux de recirculations tout au long de la conduite [**Bouffenech** et **Djamai, 2012**].

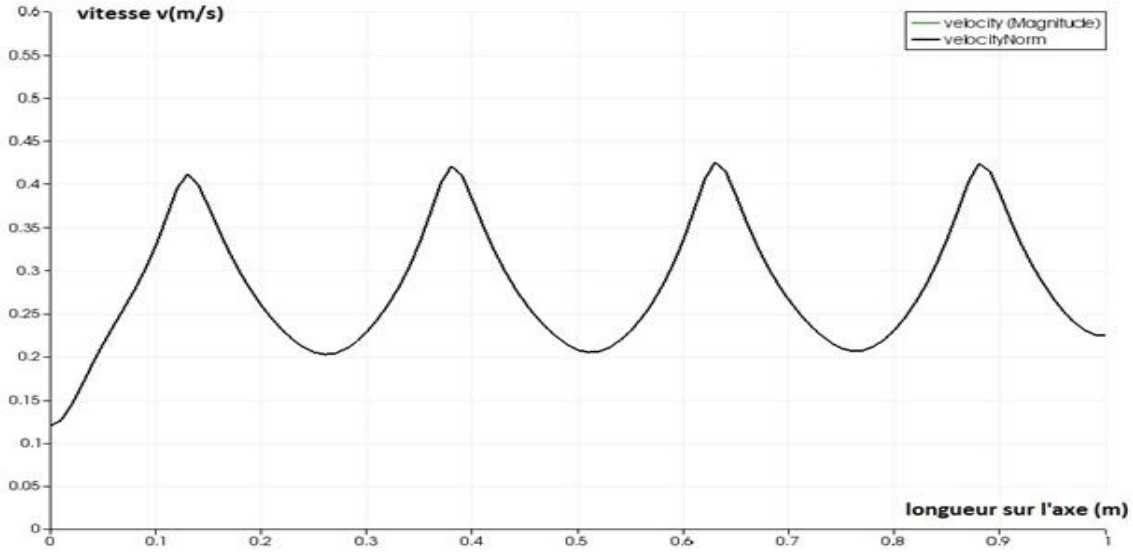


**Figure I. 8 :** Distribution de la vitesse dans conduite à quatre motifs

**[Bouffenech et Djamai, 2012]**

Ils ont aussi vérifié la relation liant le coefficient des pertes de charge et le nombre de Reynolds dans le cas d'un écoulement laminaire.

**Conclusion**

Dans les différents travaux présentés les écoulements turbulents dans des conduits à géométries complexes, les auteurs ont remarqué que la turbulence joue un rôle indispensable pour mélanger le plus rapidement possible les fluides.

La périodicité de la géométrie de la conduite rend l'écoulement assez chaotique ce qui se manifeste par l'apparition des rouleaux de recirculations tout au long de la conduite.

Ainsi, la complexité de la géométrie amène à l'apparition de trois régions différentes d'écoulement lorsque le fluide se rapproche de la paroi, il s'agit des domaines lisses, de transition et de pleine turbulence appelé aussi domaine du régime d'écoulement turbulent rugueux.

Dans notre travail, nous allons évaluer la variation de la pression dans différentes régions de la conduite et nous vérifierons l'équation des pertes de charge dans le cas d'un écoulement turbulent.

**Chapitre II**

# Présentation du programme en C++ introduit dans Palabos

La bibliothèque Palabos est un cadre mis en place de la dynamique des fluides computationnelle (CFD), avec un noyau basé sur la méthode Lattice Boltzmann (LBM). Elle est utilisée à la fois comme un outil de recherche et de conception. Son interface de programmation est directe et rend possible la simulation des fluides avec une facilité relative. Plusieurs codes de calcul sont disponibles sur Palabos pour faciliter son utilisation. C'est pourquoi le choix s'est porté sur cette bibliothèque pour l'accomplissement de notre travail. Le code C++ de Palabos fait appel à la généricité dans ses différentes facettes. La programmation générique est utilisée pour offrir un code singulier qui peut servir à plusieurs intentions. D'un côté, le code exécute la généricité dynamique par l'utilisation d'interfaces à orientés objets.

L'avantage de code calcul Palabos, c'est que il ne demande pas de très grand connaissances en C++ pour commencer à programmer. Une fois les lois de programmation connus, il deviendra plus facile de comprendre ce que le programme fait et éviter les différentes complications telle que l'insuffisance de la mémoire, exécution inconsistante, les coulisses inattendus de la programmation orientée objet (POO) . . . etc.

## II.1 Définition des Bibliothèques

Comme tout autre programme informatique, la première partie est dédiée à la déclaration des bibliothèques requises pour le bon fonctionnement du programme. Et celles-ci sont les suivantes :

#include "palabos2D.h" // L'accès aux différentes classes et structures du fichier Palabos traitant les géométries a deux dimensions telles que cylinder2D, Smagorinsky Model...etc. est donné par palabos2D.h.

#include "palabos2D.hh" // L'accès aux Templates du code est quant a lui assuré par palabos2D. hh, ce qui permettra la compilation des bibliothèques cis nommées.

#include <vector> // Bibliothèque qui contient le nouveau type vector pour la déclaration des vecteurs, en remplaçant pour le type array qui perd en efficacité lorsqu'il s'agit de simulation parallèle.

#include <cmath> // Bibliothèque qui contient les outils mathématiques nécessaires pour le bon déroulement du programme, tel que sqrt pour carré ou sin pour sinus.

#include <iostream> // Bibliothèque servant à utiliser le flux de données de sortie (output) pour écrire les résultats de la simulation dans des fichiers ou dans le terminal.

#include <fstream> // Bibliothèque servant à l'affichage des données de sortie sous forme de fichiers.

#include <iomanip> // Bibliothèque qui permet un contrôle total sur les données de sortie, tel que le nombre de chiffres après la virgule (stpeprecision).

#include <<complexDynamics/smagorinskyDynamics2D.h>>//définition du model de smagorinsky.

*Using namespace plb : :descriptors ;*

*Using namespace std ;*

Ces deux déclarations donnent accès aux fichiers qui sont contenus dans l'espace plb et aussi à la bibliothèque C++ qui est contenu dans plb.

### II.2 Définition du Modèle d'étude

La définition du modèle d'étude se fait par la commande suivante :

#define DESCRIPTOR D2Q9 Descriptor

Le modèle D2Q9 à neuf vecteurs de vitesse est très utilisé, particulièrement pour résoudre les problèmes d'écoulements des fluides. La vitesse centrale étant nulle, les vitesses sont :

c(0,0), c(1,0), c(-1,0), c(0,1), c(0,-1), c(1,1), c(-1,1), c(-l,-1) et c(1,-1) pour f0, f1, f2, f3, f4, f5, f6, f7 et f8 respectivement, les facteurs de pondération. Pour les fonctions de distribution correspondantes sont 4/9, 1/9, 1/9, 1/9, 1/9,1/36, 1/36, 1/36 et 1/36.

### II.3 Introduction des paramètres Hydrauliques

Les paramètres Hydraulique sont très importants lors de la simulation, ils sont en générale définis par la commande suivante :

**IncomprFlowParam<T> parameters (**

**(T) 0.1, // uMax (Vitesse maximum d'écoulement)**

**(T) 10000., // Re (nombre de Reynolds)**

**1000., // N (Nombre de discrétisations)**

**0,791, // lx (longueur du domaine d'écoulement dans la direction x)**

**0,06 // ly (longueur du domaine d'écoulement dans la direction y)**

**// 0. // lz (longueur du domaine d'écoulement dans la direction z) ) ;**

L'introduction des paramètres hydrauliques ou plus précisément les paramètres de l'écoulement incompressible (IncomprFlowParam) se fait de la manière suivante :

- **Vitesse**

Nous avons pris comme vitesse caractéristique ou vitesse maximale une valeur de 0,1m/s. C'est cette vitesse qui nous assurera une bonne convergence de notre programme sans risque de divergence au cas où les vitesses s'accentueraient.

- **Densité**

La densité prise en considération est la densité de l'eau est égale à $10^3 kg/m^3$.

- **Nombre de Reynolds**

Après le choix de tous les paramètres, le calcul du nombre de Reynolds se fait par le biais de la formule suivante :

$$R_e = \frac{DV}{\upsilon}$$

Dans le programme que nous avons élaboré nous, avons changé le nombre de Reynolds en fonction des cas d'étude et selon la nature du régime d'écoulement. Le programme que nous proposerons permettra de faire varie le nombre de Reynolds afin de couvrir le plus de cas possibles, et d'assurer un passage graduel d'un régime d'écoulement à un autre.

- **Longueurs et largeurs**

Pour notre cas, le modèle physique d'une conduite prismatique de dimensions 0,04*0,08*0,791m, nous avons choisi les largeurs suivante : à l'entrée de chaque divergent nous avons choisi $E_m$=0,04m, et à l'entrée de chaque convergent $E_M$=2$E_m$=0,08m, et comme notre conduite est prismatique, nous avons calculé la langueur caractéristique longitudinale équivalente qu'est : $E = \frac{E_m + E_M}{2}$ =0 ,06m.

Et en longueur totale de la conduite de L=0,791m.

## II.4 Introduction du nombre de discrétisations selon x et y

**Const plint nx= parameters.getNx () ;**

**Const plint ny= parameters.getNy () ;**

Dans cette section le nombre de discrétisations selon les deux axes X et Y seront introduits de la façon suivante :

**Nx= lx*N.**

**Ny =ly*N.**

### III.5 Initialisation une frontière de pression à une densité constante

Dans cette section la densité est initialisée à une valeur constante à l'aide d'une classe constant Density.

**constant density**

```
template<typename T>
class ConstantDensity {
public:
  ConstantDensity(T density_)
    : density(density_)
  { }
  T operator()(plint iX, plint iY) const {
    return density;
}
private:
  T density;
};
```

### II.6 Création des conditions initiales

Les conditions initiales sont très importantes et présentent une très grande difficulté lors de la construction du programme. En effet, les conditions initiales permettent à l'écoulement de s'établir beaucoup plus rapidement, et aux résultats de converger plus vite. Une manière très répandue d'établir ces conditions initiales consiste à imposer un champ de vitesse pour t = 0. Dans notre cas, nous établirons cette condition initiale en imposant un profil parabolique de vitesse sur le domaine d'écoulement à l'instant initial à l'aide d'une expression analytique.

Dans notre programme, les conditions aux limites sont imposées à partir d'une fonction :

**classe Poiseuille Velocity And Density**.

### II.7 Définition du domaine d'étude

Nous avons adopté la géométrie du suivante

```
double r =ny/2;
double a =nx/14;
double b =ny/6;
```

```
DotList2D cylinderShape;
for (plint iX=0; iX<nx; ++iX) {
for (plint iY=0; iY<ny; ++iY) {
// Création des motifs de la partie inférieur
if ((iX<a)and iY<(b/a)*iX)
cylinderShape.addDot(Dot2D(iX,iY));
else if ((iX>a)and iY<(-b/a)*(iX-2*a))
cylinderShape.addDot(Dot2D(iX,iY));
else if ((iX>2*a) and (iX<3*a)and iY<(b/a)*(iX-2*a))
cylinderShape.addDot(Dot2D(iX,iY));
else if ((iX>3*a)and (iX<4*a) and iY<b*(4-(iX/a)))
cylinderShape.addDot(Dot2D(iX,iY));
else if ((iX>4*a)and (iX<5*a)and iY<(b/a)*iX-4*b)
cylinderShape.addDot(Dot2D(iX,iY));
else if ((iX>5*a)and (iX<6*a)and iY<(-b/a)*iX+6*b)
cylinderShape.addDot(Dot2D(iX,iY));
else if ((iX>6*a)and (iX<7*a)and iY<(b/a)*iX+7*b)
cylinderShape.addDot(Dot2D(iX,iY));
else if ((iX>7*a)and (iX<8*a)and iY<(-b/a)*iX-7*b)
cylinderShape.addDot(Dot2D(iX,iY));
else if ((iX>8*a)and (iX<9*a)and iY<(b/a)*iX+9*b)
cylinderShape.addDot(Dot2D(iX,iY));
else if ((iX>9*a)and (iX<10*a)and iY<(-b/a)*iX-9*b)
cylinderShape.addDot(Dot2D(iX,iY));
else if ((iX>10*a)and (iX<11*a)and iY<(b/a)*iX+11*b)
cylinderShape.addDot(Dot2D(iX,iY));
else if ((iX>11*a)and (iX<12*a)and iY<(-b/a)*iX-11*b)
cylinderShape.addDot(Dot2D(iX,iY));
else if ((iX>12*a)and (iX<13*a)and iY<(b/a)*iX+13*b)
```

```
cylinderShape.addDot(Dot2D(iX,iY));
else if ((iX>13*a)and (iX<14*a)and iY<(-b/a)*iX-13*b)
cylinderShape.addDot(Dot2D(iX,iY));
// Création des motifs de la partie supérieur
else if ((iX<a)and iY>((-b/a)*iX+2*r))
cylinderShape.addDot(Dot2D(iX,iY));
else if ((iX>a)and iY>(b/a)*iX+2*(r-b))
cylinderShape.addDot(Dot2D(iX,iY));
else if ((iX>2*a)and (iX<3*a)and iY>(-b/a)*iX+2*(r+b))
cylinderShape.addDot(Dot2D(iX,iY));
else if ((iX>3*a)and (iX<4*a)and iY>(b/a)*iX+2*(r-2*b))
cylinderShape.addDot(Dot2D(iX,iY));
else if ((iX>4*a)and (iX<5*a)and iY>(-b/a)*iX+2*(r+2*b))
cylinderShape.addDot(Dot2D(iX,iY));
else if ((iX>5*a)and (iX<6*a)and iY>(b/a)*iX+2*(r-3*b))
cylinderShape.addDot(Dot2D(iX,iY));
else if ((iX>6*a)and (iX<7*a)and iY>(-b/a)*iX+(2*r-7*b))
cylinderShape.addDot(Dot2D(iX,iY));
else if ((iX>7*a)and (iX<8*a)and iY>(b/a)*iX+(2*r+7*b))
cylinderShape.addDot(Dot2D(iX,iY));
else if ((iX>8*a)and (iX<9*a)and iY>(-b/a)*iX+(2*r-9*b))
cylinderShape.addDot(Dot2D(iX,iY));
else if ((iX>9*a)and (iX<10*a)and iY>(b/a)*iX+(2*r+9*b))
cylinderShape.addDot(Dot2D(iX,iY));
else if ((iX>10*a)and (iX<11*a)and iY>(-b/a)*iX+(2*r-11*b))
cylinderShape.addDot(Dot2D(iX,iY));
else if ((iX>11*a)and (iX<12*a)and iY>(b/a)*iX+(2*r+11*b))
cylinderShape.addDot(Dot2D(iX,iY));
else if ((iX>12*a)and (iX<13*a)and iY>(-b/a)*iX+(2*r-13*b))
```

```
cylinderShape.addDot(Dot2D(iX,iY));
else if ((iX>13*a)and (iX<14*a)and iY>(b/a)*iX+(2*r+13*b))
cylinderShape.addDot(Dot2D(iX,iY));
```

Nous avons obtenu la forme de la conduite prismatique à 07 motifs périodiques suivante:

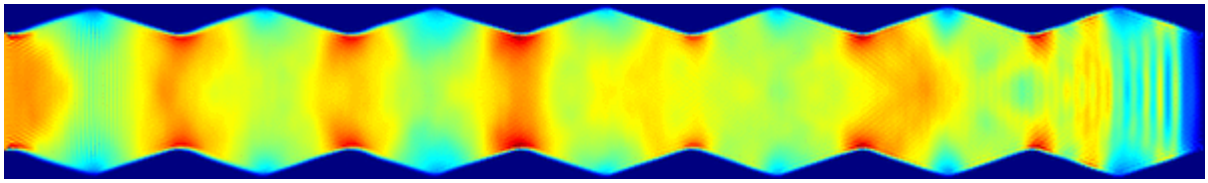

**Figure II.1:** conduite à motifs périodiques

### II.8 Définition du domaine d'application des conditions aux limites

```
void cavitySetup( MultiBlockLattice2D<T,DESCRIPTOR>& lattice,
                  IncomprFlowParam<T> const& parameters,
                  OnLatticeBoundaryCondition2D<T,DESCRIPTOR>& boundaryCondition )
{
    const plint nx = parameters.getNx();
    const plint ny = parameters.getNy();
    Box2D topLid = Box2D(0, nx-1, ny-1, ny-1);
    Box2D input  = Box2D(0,0,0,ny-1);
    Box2D inlet(0,0,1,ny-2);
    Box2D inletOutlet(0,0,1,ny-2);
    Box2D everythingButTopLid = Box2D(0, nx-1, 0, ny-2);
    Box2D topWall(0, nx-1, ny-1, ny-1);
    Box2D bottomWall(0, nx-1, 0, 0);
    Box2D outlet(nx-1, nx-1, 2, ny-2);
boundaryCondition.setVelocityConditionOnBlockBoundaries(lattice);
```

Nous définissons le domaine d'application des conditions aux limites à travers les fonctions **boundaryCondition.setVelocityConditionOnBlockBoundaries** suivie de l'étendu du domaine d'application par exemple :

```
boundaryCondition.setVelocityConditionOnBlockBoundaries ( lattice, Box2D (0, nx-1, 0, 0))//
```

Fait référence à l'application des conditions aux limites pour X variant de 0 à nx-1 et pour Y=0.

```
SArray<T,2> v;
v[0]=0.175*(dt1/dx1);//pourtransfirmerlavitesseàl'unité lattice
v[1]=0.0;
SetBoundaryVelocity (lattice, inlet, v ) ;       // vitesse appliquée à l'entrée
SetBoundaryDensity ( lattice, outlet,ConstantDensity<T>(1.) )
```

Concernant la vitesse, nous avons appliqué la vitesse adoptée dans la partie expérimentale.

On a lancé l'exécution pour 5 débits différents $Q_1$=0,63*l/s*, $Q_2$=0,62*l/s*, $Q_3$=0,59*l/s*, $Q_4$=0,54*l/s* *et* $Q_5$=0,47*l/s* qui correspondent respectivement aux vitesses suivantes :

```
V[0]=0.175*(dt1/dx1);
V[0]=0.172*(dt1/dx1);
V[0]=0.16*(dt1/dx1);
V[0]=0.15*(dt1/dx1);
V[0]=0.13*(dt1/dx1);
```

### II.9 formulation de la densité du fluide pour l'écoulement turbulent

```
void randomIniCondition (plint iX, plint iY, T& rho, Array<T,2>& velocity) {
T randomValue = (T) rand () / (T)RAND_MAX;
velocity.resetToZero () ;
rho = (T) 1 + 1.e-2*randomValue ;}
```

### II.10 Introduction des paramètres de Smagorinsky pour la viscosité turbulente

```
Void cavitySetup (MultiBlockLattice2D<T, DESCRIPTOR>& lattice,
IncomprFlowParam<T> const& parameters,
OnLatticeBoundaryCondition2D<T, DESCRIPTOR>& boundaryCondition)
{
Box2D topLid = Box2D (0, nx-1, ny-1, ny-1) ;
Box2D everythingButTopLid = Box2D (0, nx-1, 0, ny-2) ;
T cSmago = 0.14
```

Introduction de la constante Csmago dont l'ordre de grandeur varie de 0,1 à 0,2 et qui sera introduite dans l'expression de la viscosité turbulente. Dans notre programme, la valeur 0,14 est attribuée comme une constante Csmago.

```
boundaryCondition.setVelocityConditionOnBlockBoundaries (lattice) ;
T u = sqrt((T)2)/(T)2 * parameters.getLatticeU() ;
```

Introduction des modifications dans le calcul de la vitesse Lattice et prise en compte de l'effet de la viscosité turbulente.

```
InitializeAtEquilibrium (lattice, everythingButTopLid, randomIniCondition) ;
SetBoundaryVelocity (lattice, topLid, Array<T,2>(u,0.) ) ;
lattice.initialize () ;
```

### II.11 Fonction Write GIF

```
Void writeGif(MultiBlockLattice2D<T,DESCRIPTOR>& lattice, plint iter)
{
ImageWriter<T> imageWriter ("leeloo") ;
imageWriter.writeScaledGif (createFileName ("u", iter, 6),
*computeVelocityNorm (lattice) ) ;
}
```

C'est la fonction qui génère des images **GIF** suivant la grandeur voulue et la perspective désirée. Dans notre cas, il s'agit de **VelocityNorm** (la norme de la vitesse) mais il est possible de l'adapter à la grandeur qu'on veut représenter.

### II.12 Fonction Write VTK

```
void writeVTK (MultiBlockLattice2D<T, DESCRIPTOR>& lattice,
IncomprFlowParam<T> const& parameters, plint iter)
{
T dx = parameters.getDeltaX () ;
T dt = parameters.getDeltaT () ;
VtkImageOutput2D<T> vtkOut (createFileName("vtk", iter, 6), dx) ;
vtkOut.writeData<float>(*computeVelocityNorm(lattice), "velocityNorm", dx/dt) ;
vtkOut.writeData<2, float>(*computeVelocity(lattice), "velocity", dx/dt) ;
```

**}**

C'est la fonction qui permet d'enregistrer les résultats (données de sortie) sous format **VTK** et avoir à la fin une image de l'écoulement en deux ou trois dimensions.

Les fichiers **VTK** peuvent fournir des informations sur la vitesse, la densité et la grandeur **VelocityNorm** : la norme de la vitesse. Cette dernière donne une idée sur la distribution de la vitesse moyenne du fluide dans la structure.

La fonction a comme paramètre la Lattice (réseau), les paramètres de l'écoulement Incompressible et l'itération.

A noter que les résultats affichés en fichier **VTK** sont automatiquement transformés en unités physiques.

### II.13 Corps principale du programme

**int main (int argc, char* argv[]) {**

**plbInit (&argc, &argv) ; //**

La fonction plbInit doit être appelée au début du programme pour garantir une consistance entre les programmes exécutés en parallèle et en série.

**global :directories ().setOutputDir("./tmp/") ;**

**tmp est le fichier de sortie il permet de stocker les images GIF et les fichiers VTK.**

**const T logT = (T) 0.4;// pas d'enregistrement des résultats.**

**const T imSave =(T) 0.06;// pas d'enregistrement des images gif.**

**const T vtkSave = (T) 10 ; // pas de génération des fichiers VTK de sortie.**

**const T maxT = (T) 50 ; // temps maximum de simulation.**

**Pcout <_< "nx=" <_< nx <_< "ny=" <_< ny <_< endl ;**

Fonction classique d'affichage de nx et ny respectivement.

### II.14 Appel des fonctions principales constituant le programme

**WriteLogFile (parameters, "Poiseuille flow") ;**

**MultiBlockLattice2D<T, DESCRIPTOR> lattice (**

**parameters.getNx (), parameters.getNy (),**

**new BGKdynamics<T,DESCRIPTOR>(parameters.getOmega()) ) ;**

Durant la création du multiblock lattice (réseau a plusieurs blocs) la nature de la collision que subissent les cellules est définie ici dans new BGKdynamics, de sorte que nous appliquerons le modèle BGK à un seul temps de relaxation omega.

```
OnLatticeBoundaryCondition2D<T, DESCRIPTOR>*
BoundaryCondition = createLocalBoundaryCondition2D<T, DESCRIPTOR>() ;
CylinderSetup (lattice, parameters, *boundaryCondition) ;
```

Le type LocalBoundaryCondition2D implémente les conditions aux limites qui sont locales La fonction CylinderSetup servira de son coté à introduire ces conditions aux limites à la géométrie définie précédemment.

### II.15 Boucle d'iteration

La fonction principale qui suit est la fonction qui permet le calcul des différents paramètres de l'écoulement. Il s'agit de l'évolution au fil du temps de ces paramètres. Cette boucle complexe permet de calculer la vitesse, la densité, l'énergie, la pression et plein d'autres variables en tout point du domaine. L'écriture des modules des vitesses est faite à partir d'une sous structure intégrée à cette boucle, et ce pour n'importe quel pas de temps. Une autre structure génère des captures d'images du domaine d'écoulement durant le déroulement de la simulation, en affichant les vitesses et la pression pour chaque pas de temps.

```
For (plint iT=0 ; iT*parameters.getDeltaT () <maxT; ++iT) {
T u0 [2] ;
for (plint iY=1 ; iY<= 100-1 ;++iY) {
```

### II.16 Génération des images GIF

```
if (iT%parameters.nStep(imSave)==0) {
Pcout <_< "Saving Gif ..." <_< endl
WriteGif (lattice, iT) ; }
```

Grâce à la fonction **WriteGif**, les données de simulation sont sauvegardées sous forme d'images **Gif**. Il est à noter que le choix du pas de temps nous revient et qu'il est possible de le changer si nécessaire.

### II.17 Generation des fichiers VTK

```
if (iT%parameters.nStep(vtkSave)==0 && iT>0) {
Pcout <_< "Saving VTK file ..." <_< endl ;
WriteVTK (lattice, parameters, iT) ;
```

**}**

La fonction **WriteVTK**, nous permet elle aussi de sauvegarder les données de simulation sous forme de fichiers **VTK**, cela nous permettra de visualiser l'écoulement.

### II.18 Déroulement de la phase de propagation et de collision

```
if (iT%parameters.nStep(logT)==0) {
Pcout <_< "step» <_< iT <_< " ; t=" <_< iT*parameters.getDeltaT () ;
}
lattice.collideAndStream () ;
```

Cette partie du programme permettra le déroulement de deux phases importantes de l'algorithme LB, la collision et la propagation :

**Collision :** cette étape est locale et implique uniquement les liens qui arrivent au même noeud. Parmi les configurations possibles se trouve la collision frontale de deux particules qui peuvent subir une variation $\pi/2$ de la direction de leurs vitesses.

**Propagation :** les particules présentes en chaque lien sont transportées vers les neufs plus proches voisines selon leurs vitesses respectives. On note que les mouvements sont synchronisés de sorte qu'après advection toutes les particules suivent l'équation d'évolution de la forme : $n_j(x_i + c_j, t+1) = n_j(x_i, t) + \Omega(nk)$

### II.19 Affichage de la valeur numérique de l'énergie

```
Pcout <_< " ; av energy =" <_< Setprecision (10) <_< getStoredAverageEnergy<T> (lattice)
```

La fonction **getStoredAverageEnergy** nous permettra d'afficher une valeur moyennée de l'énergie d'où le préfixe av pour averaged.

### II.20 Affichage de la valeur numérique de la densité

```
<_< " ; av rho ="
<_< GetStoredAverageDensity<T>(lattice) <_< endl ;
```

Tout comme le cas précédent, la fonction **GetStoredAverageDensity** nous affichera une valeur moyennée de la densité.

### II.21 Affichage de la valeur numérique de la vitesse

```
Array<T, 2> velocity
lattice.get (nx/2, ny/2).computeVelocity (velocity) ;
```

**Pcout <_< "Velocity in the middle of the lattice : (" <_< velocity [0] <_< "," <_< velocity [1] <_< ")" <_< endl ;**

Le vecteur vitesse sera affiché grâce à la déclaration d'un vecteur de deux composantes, cela se traduit dans le programme par la déclaration **Array<T, 2>**.

Le point de contrôle ou checkpoint est indiqué par la fonction **lattice.get(nx/2,ny/2)**, ce qui voudra dire que les composantes de la vitesse seront calculées dans un point précis qui est **(nx/2,ny/2)**.

**pcout <_< "Velocity norm in the middle of the lattice : " <_< endl ;**

**Box2D line (nx/2, nx/2 , ny/2, ny/2) ;**

**pcout <_< setprecision(3) <_< *computeVelocityNorm(*extractSubDomain(lattice, line)) <_< endl ;**

Avec la méthode **extractSubDomain** un sous domaine est extrait du domaine Lattice, c'est la où la vitesse va être calculée et affichée.

**Plb_ofstream ofile ("profile.dat") ;**

**Ofile <_< setprecision (3) <_< *computeVelocityNorm (*extractSubDomain (lattice, line)) <_<**

**endl ;**

Dans ce cas, les résultats de la compilation seront affichés dans un fichier indépendant du dossier contenant le programme.

### II.22 Commandes d'exécution du parallèle

```
date >>date_enr
mpirum-mpich2 –np 1 ./smagorinskyCavity3D
echo np=1  >> date_enr
date >>date_enr
mpirum-mpich2 –np 2 ./smagorinskyCavity3D
echo np=2  >> date_enr
date >>date_enr
mpirum-mpich2 –np 3 ./smagorinskyCavity3D
echo np=3 >> date_enr
date >>date_enr
mpirum-mpich2 –np 4 ./smagorinskyCavity3D
echo np=4 >> date_enr
```

**date >>date_enr**

**mpirum-mpich2 –np 5 ./smagorinskyCavity3D**

**echo np=5 >> date_enr**

**date >>date_enr**

**mpirum-mpich2 –np 6 ./smagorinskyCavity3D**

**echo np=6 >> date_enr**

**Conclusion**

Apres avoir introduit les modifications dans le code Palabos, fait la transformation des unités et introduite les conditions de stabilité, ainsi d'autres commandes d'exécution du parallèle nous allons passer à l'application de programme sur un conduit à sept motifs périodiques, où nous allons effectuer une étude de performance pour la réduire le temps de calcul.

**Chapitre III**

# Performances parallèles

Les calculateurs les plus répandus actuellement sont de type MIMD (Multiple Instruction, Multiple Data) où chaque processeur exécute son propre flux d'instructions appliqué à son flux de données. Les micro-ordinateurs actuels qui disposent souvent de plusieurs cœurs appartiennent à cette catégorie.

Pour comprendre les spécificités des ordinateurs parallèles, il faut entrer dans l'architecture mémoire de ces machines.

Essentiellement, on rencontre deux grandes classes d'architectures parallèles :

- les multiprocesseurs à mémoire partagée (SMP, symmetric multiprocessors) ont une mémoire commune partagée par plusieurs nouds (figure III.1) ;

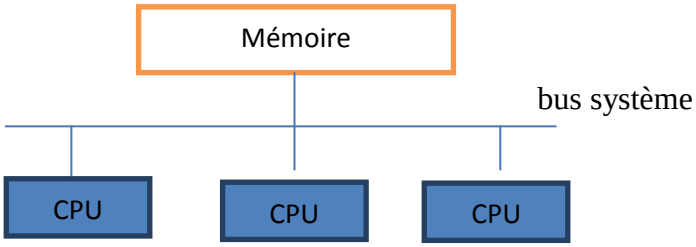


**Figure III. 1:** Multiprocesseur à mémoire partagée.

- les machines à mémoire distribuée sont caractérisées par des bancs de mémoire accessibles par un seul processeur (figure III.2). C'est le cas des *clusters* (ou «grappes» de machines), qui présentent des intérêts considérables pour les utilisateurs du calcul parallèle : elles sont de moindre coût, leur conception matérielle est plus aisée et elles sont plus facilement extensibles que les SMP.

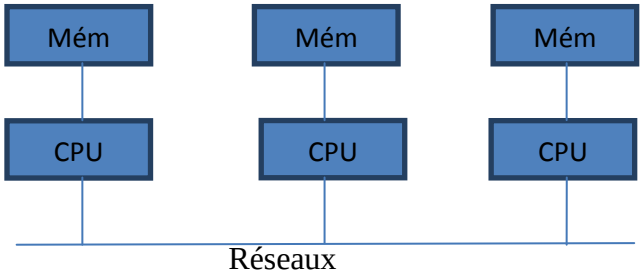


**Figure III. 2 :** Architecture à mémoire distribuée

La première est la plus simple d'emploi, tous les processeurs utilisent le même espace d'adressage, toute modification de la mémoire est immédiatement visible de tous les

processeurs. Le principal inconvénient est la difficulté et le coût de construction, mais aussi la limitation en nombre de processeurs connectables.

L'architecture à mémoire distribuée est plus facile à réaliser, chaque processeur adresse directement sa mémoire, comme dans le cas d'un processeur unique et communique avec les autres processeurs par un réseau d'interconnexion.

Chaque processeur gère seul la mémoire qui lui est connectée. Le temps d'accès à la mémoire dépend de sa localisation. Pour la mémoire directement connectée au processeur, le temps d'accès est très inférieur à la (µs), il peut se compter en (ms) ou centaines de (ms) dans le cas d'un accès par réseau Internet. Les espaces de mémoire étant disjoints la programmation est plus difficile.

Dans le cas des architectures distribuées, deux paramètres sont importants, le nombre de processeurs par nœud, et le réseau d'interconnexion.

### III.1. Modèles de programmation

Le mode de programmation le plus répandu des machines à mémoire distribuée est l'échange de messages, solidement établi depuis la mise au point et l'acceptation générale de la norme MPI[1] (Message-Passing Interface). Cette technique de programmation est assez contraignante, car c'est le programmeur qui doit gérer explicitement la répartition des données dans les mémoires distribuées. De plus, la communication de structures de données complexes (avec des pointeurs, par exemple) est très délicate avec MPI.

En revanche, les SMP se programment très efficacement en utilisant la mémoire partagée au lieu du passage de messages. Dans ce modèle de programmation, toutes les données partagées sont accessibles par tous les processeurs, et la cohérence de la mémoire vue par les différents sites est assurée par le matériel.

1. MPI est une interface de programmation par échange de messages, où un processeur n'a accès qu'à sa propre mémoire, et où les communications avec d'autres processus se font par envois explicites de messages.

## III.2 Parallélisation

L'implémentation parallèle du code a deux objectifs :

- Obtenir des temps de calculs raisonnables. Nous pouvons espérer idéalement diviser le temps réel par le nombre de processeurs utilisés.
- Pouvoir traiter des simulations réelles exigeant une très importante place mémoire, en répartissant le stockage des données dans la mémoire de chaque processeur.

Après une présentation de la machine utilisée pour faire les simulations, on présente en détail l'algorithme parallèle. Une analyse des performances est ensuite donnée.

### III.2.1 Présentation de la machine WORKSTATION HP Z800 HP

La **WORKSTATION HP Z800 HP** est une des plateformes les plus rapides fournies par Hewlett-Packard (HP). On a utilisé l'une des deux stations de calcule qui se trouve au niveau du **Laboratoire de Recherche Sciences de l'Eau (LRSE),** c'est une machine M.I.M.D (multiples instructions, multiples données) et a une architecture de mémoire partagé.

### III.2.2 Configuration interne de la station de calcul

La WORKSTATION HP Z800 HP à 6 processeurs de type Intel® Xeon® Six-Core Processor X5650 avec une fréquence de 2.66 GHz, et un disque dure de 1500GO. Les processeurs et le matériel associés forment ce qui s'appelle généralement le nœud. Le nœud utilise une conception symétrique du multiprocesseur (SMP) qui peut exploiter le parallélisme à grain fin.

La machine est synchronisée par plusieurs unités de commande :

- La barre transversale d'hyperplan de HP qui est composée de quatre contrôleurs de connexion de routage. Elle permet à tous les processeurs d'accéder à toute la mémoire disponible.
- Les processeurs sont installés sur les agents contrôleurs des processeurs, qui permettent au processeur et au sous-ensemble d'Entrée/Sortie (le contrôleur d'interface) l'accès à la barre transversale d'hyperplan.
- Également reliés à la barre transversale d'hyperplan sont les contrôleurs d'accès mémoire.
- Les entrées/sorties se relient au système qui est reliée aux agents de processeur.
- L'utilitaire de noyau dans le nœud (généralement appelé panneau d'utilitaires) contient une section du matériel (hardware) appelée logique de noyau. Elle fournit des interruptions à tous les processeurs du système par le bus de logique de noyau qui se relie à chaque agent de processeur.

Le processeur et la mémoire sont reliés par un commutateur à barres croisées de sorte que chaque processeur soit relié à toute la mémoire.

### III.3.3 Organisation de la mémoire

Mémoire Physique : La mémoire physique ou la RAM (mémoire à accès sélectif) sur WORKSTATION HP Z800 HP est de 8GB de stockage répartis sur 64 banques de mémoire de 64 MB. La division de la mémoire (RAM) dans 64 banques de mémoire permet l'accès simultané de la mémoire vers les différentes banques, ce qui ne serait pas possible si la mémoire entière était traitée en tant qu'une unité de mémoire.

La mémoire physique contient les programmes à exécuter et les données (appelés "mémoire logique"), mais contient également beaucoup d'informations "système".

Mémoire virtuelle : La mémoire virtuelle est disponible par l'utilisation d'unités de mémoire appelées pages. Elle permet, par l'accès à des segments de page de mémoire, l'exécution de beaucoup plus de programmes que ne peut la mémoire physique. Le compilateur produit des adresses de la mémoire virtuelle beaucoup plus grandes, mais seules les pages s'exécutant sont traduites en mémoire physique. La WORKSTATION HP Z800 HP traduit des adresses de mémoire virtuelles et physiques de 32 bits et 64 bits en adresses de 64 bits.

Cache : Chaque processeur possède 6 MB de mémoire cachée pour les données ainsi que 6 MB de mémoire cache pour les instructions, soit un total 12 MB. La mémoire cache est généralement plus petite, plus rapide et plus proche de l'unité centrale de traitement que la mémoire physique.

## III.4 Programmation parallèle

Le système d'exploitation de la WORKSTATION HP Z800 HP. Sur le plan des logiciels, la plupart des langages de programmation sont accessibles sur cette machine.

Pour la programmation parallèle, le HP MPI (Message Passing Interface) est utilisé. Ce dernier fournit un éventail de dispositifs facilitant le développement des applications parallèles. Ces dispositifs incluent

- La conformité à la norme de la version 1.2 de MPI.
- Des styles de programmation SPMD (programme simple à multiples données) et MPMD (programmes multiple à multiples données).
- Un outil (XMPI) permettant de tracer et de surveiller les applications et le fonctionnement des processeurs.
- Un outil (MPIVIEW) permettant de visualiser et d'analyser les performances de chaque processeur.

### III.4.1 But de la parallélisation

*Qu'est-ce que la parallélisation?*

Ensemble de techniques logicielles et matérielles permettant l'exécution simultanée de séquences d'instructions indépendantes, sur des processeurs différents

*Bénéfice de la parallélisation ?*

- Exécution plus rapide du programme (gain en temps de restitution) en distribuant le travail.
- Résolution de problèmes plus gros (plus de ressource matérielle accessible, notamment la mémoire)

### III.4.2 Optimisation de la parallélisation

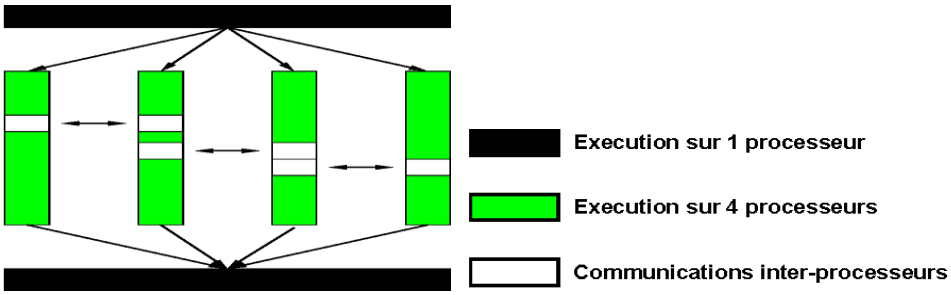


**Figure III. 3 :** la communication entre les processeurs

#### III.4.2.1 ***Accélération et efficacité***

- Les deux sont une mesure de la qualité de la parallélisation
- Soit T(p) le temps d'exécution sur p processeurs
- L'Accélération **A(p)** et l'Efficacité **E(p)** sont définies comme étant :

$A(p) = T(l) / T(p)$ (p=l, 2, 3, ...)

$E(p) = A(p) / p$

- Pour une accélération parallèle parfaite on obtient :

$A(p) = T(l) / T(p) = T(l) / ( T(l) / p) = p$

$E(p) = A(p) / p = p / p = 100\%$

#### III.4.2.2 Stabilité

- Propriété d'une application à être exécutée efficacement sur un très grand nombre de processeurs.
- Raisons possibles d'une faible stabilité :
  - La "machine" parallèle employée : architecture inadaptée, charge, ...

- Le programme parallélisé : analyser le comportement, améliorer les performances
- un peu des deux ...

## III.5 Simulation numérique direct (DNS[2])

La simulation numérique d'un écoulement turbulent nécessite des ressources très importantes en temps de calcul et en place de mémoire. En effet, la simulation numérique directe résout les équations de Navier-Stokes sans faire d'approximations autres que celles dues à la discrétisation numérique (erreur numérique par rapport à la solution des équations aux dérivées partielles). On doit donc utiliser des maillages suffisamment fins pour capter toutes les structures turbulentes y compris les plus petites. On montre que si L est la taille de l'échelle intégrale et λ celle de l'échelle de Kolmogorov, alors L/ λ est de l'ordre de $R_e^{3/4}$, où $R_e$ est le nombre de Reynolds basé sur les grandeurs turbulentes caractéristiques de l'écoulement. Ainsi, si l'on s'intéresse à un écoulement dans un domaine de taille caractéristique L, alors le nombre de nœuds N dans chaque direction nécessaire à la simulation de phénomènes de l'ordre de l'échelle de Kolmogorov est proportionnel à $R_e^{3/4}$. Ainsi, même pour un nombre de Reynolds faible, $R_e = 10^4$, une simulation directe nécessite $10^3$ points dans chaque direction, ce qui engendre un milliard de nœuds au total. Les écoulements dans des configurations industrielles ont des nombres de Reynolds basés sur l'écoulement moyen de l'ordre de $10^6$ à $10^8$ (ce qui correspond à un nombre de Reynolds turbulent de l'ordre de $10^4$ à $10^6$), il est donc impossible et pour une longue période encore d'effectuer des DNS sur des configurations industrielles malgré le progrès exponentiel des moyens de calculs. Il est donc nécessaire de mettre en œuvre des modèles de turbulence qui permettent de modéliser les plus petites ou la totalité des structures turbulentes et non plus de les simuler par le calcul**.**

Pour ces raisons, on s'est restreint au cas 2D. En effet, pour $R_e = 10000$ le nombre de nœuds nécessaire pour faire une simulation directe de la turbulence est (dans le cas 2D) : $(R_e^{3/4})^2 = (10000^{3/4})^2 != 10^6$ un million de nœuds, ce qui est faisable par la machine qu'on dispose au niveau de laboratoire.

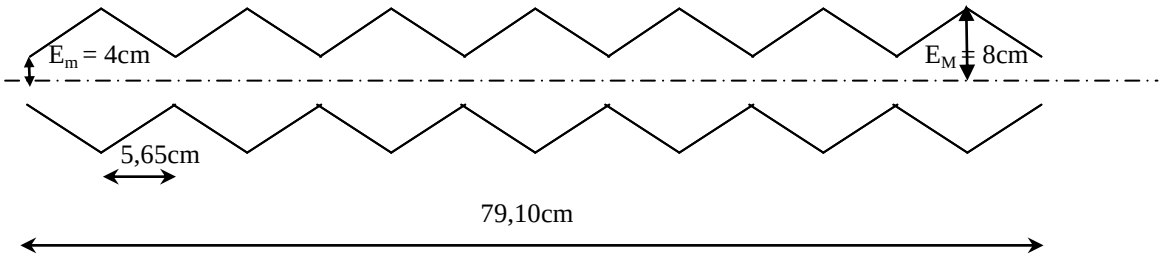


**Figure III. 4** : conduite à 07 motifs périodiques

2. DNS (Domain Name System) tel que le DNS sert à transmettre et connaitre l'association entre un nom de domaine et une adresse IP

Dans notre cas, nous avons une conduite de longueur 79,10cm et d'une largeur 2 x 8 = 16cm, on calcule le nombre de nœuds pour $N_x$ = Ny*79,10/16 = 1000* 79,10/16 = 4943, donc le nombre de nœuds est : 4943x1000= 4,943x$10^6$>$(Re^{3/4})^2$ =$(10000^{3/4})^2$=$10^6$ ce qui justifie que nous sommes dans les normes, et nous pouvons effectuer des DNS.

## III.6 Présentation des cas tests et Performance

### III.6.1 Outils de performance

Les résultats des tests seront analysés à travers les trois types de courbes suivants :

les courbes de temps qui nous informent sur l'évolution du temps d'exécution du programme en fonction du nombre de processeurs ainsi que sur l'importance du coût dû à la communication.

- les courbes d'accélération (Speed-Up).

En fait, c'est à travers les courbes de Speed-Up qu'on pourra apprécier l'efficacité d'un programme parallèle. Si le programme séquentiel résout un problème de taille n en un temps $T_1(n)$ et si l'algorithme parallèle le résout en un temps $T_p(n)$ avec p processeurs, l'accélération (Speed-Up) est donnée par le nombre suivant :

$$A_p(n) = \frac{T_1(n)}{T_p(n)}$$

Ce terme peut être inférieur à 1 si le programme parallèle est pénalisé par le surcoût lié à la communication entre les processeurs dû à l'utilisation des routines de communication. On distingue deux parties dans le temps de communication

- l'overhead stable qui représente le coût des send/receive,
- l'overhead instable qui mesure le temps des attentes réseau. Celui-ci est important lors de l'utilisation d'une machine parallèle virtuelle, mais reste incontrôlable puisqu'il est intimement lié à l'état du réseau (charge, bande passante,...).

Par la suite, on considère le temps comme la somme du temps de calcul et celui de communication (sans oublier qu'il dépend étroitement de la charge de la machine). Le temps de communication ou overhead comprend le temps de préparation (latence + surcoût) et le temps de transfert.

- Pour avoir un pourcentage de la réduction du temps de calcul en présente la courbe d'efficacité

### III.6.2 Présentation de cas test

Le tableau suivant montre les résultats de simulation de l'accélération et l'efficacité :

**Tableau III. 1 :** les résultats de la simulation

| p | T(p) en (min) | A(p) en ($m/s^2$) | E(p) en % |
|---|---|---|---|
| 1 | 50 | 1 | 1 |
| 2 | 26 | 1,92 | 0,96 |
| 3 | 23 | 2,17 | 0,72 |
| 4 | 17 | 2,94 | 0,73 |
| 5 | 14 | 3,57 | 0,71 |
| 6 | 13 | 3,84 | 0,64 |

### III.6.3 Analyse des performances

La figure III.5 présente la variation du temps de simulation en fonction du nombre de processeurs.

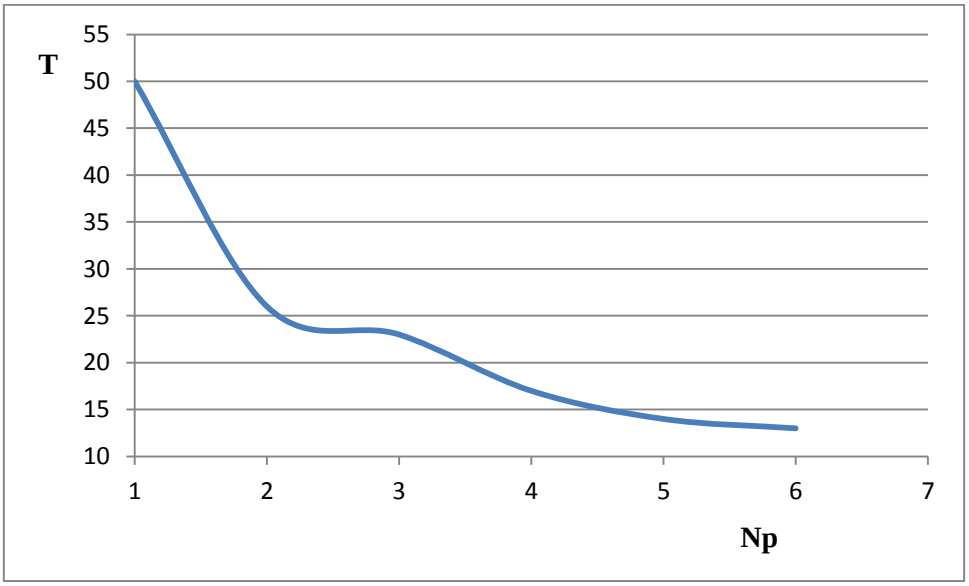


**Figure III. 5** : Variation du temps de calcul

D'après la figure III.5, on remarque que le temps de simulation suit une tendance décroissante dès que le nombre de processeurs augmente, donc chaque fois qu'on augmente le nombre de processeurs le temps de calcul se réduit, ce qui signifie un gain de temps important.

Ce gain en temps est proportionnel au cas traité qui peut se justifier par le temps de communication entre les processeurs qui devient plus important quand on augmente le nombre de processeurs.

Les figures III.6 et III.7 montrent les évolutions de l'accélération et l'efficacité en fonction du nombre des processeurs.

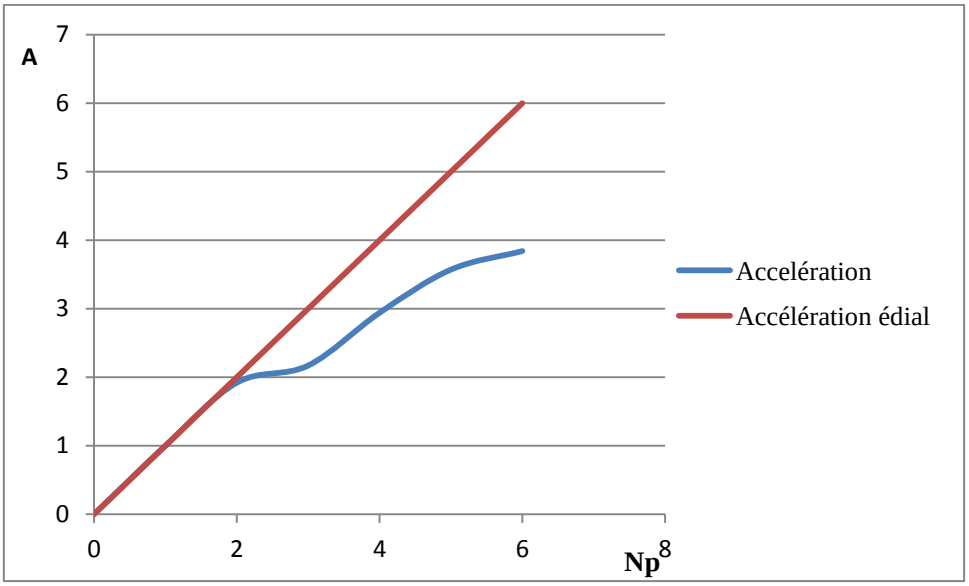


**Figure III. 6 :** Variation de l'accélération

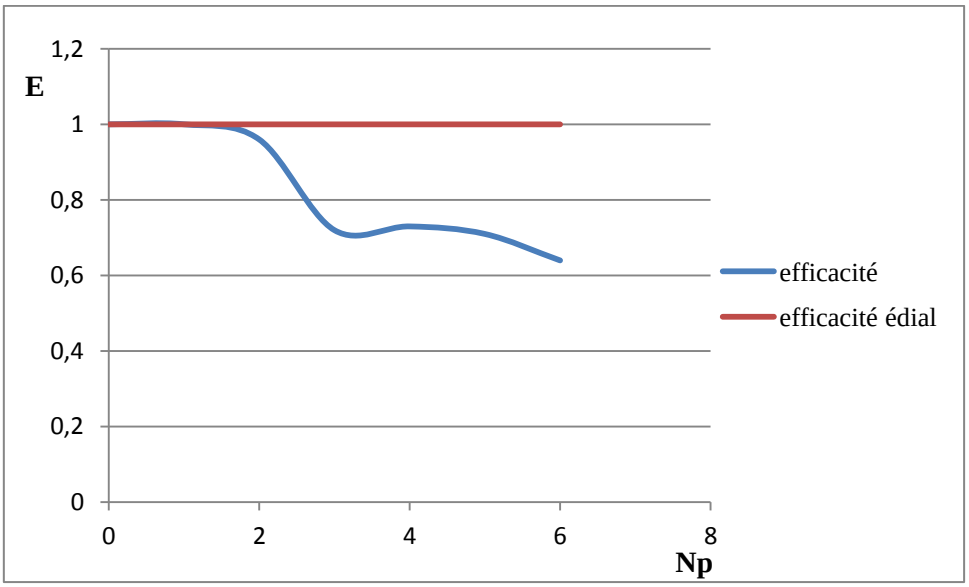


**Figure III. 7** : Variation de l'efficacité

Dans notre cas, nous remarquons qu'à partir d'un certain nombre de processeurs, une sous linéarité se présente par l'efficacité qui décroît à cause du temps de communication qui devient important. Néanmoins, cette décroissance peut expliquer par le temps perdu dans l'échange d'informations entre les processeurs.

On remarque aussi que la réduction du temps de calcul est de l'ordre de 62% qui est proportionnelle aux nombre de processeur utilisés.

**Conclusion**

Le but cette étude était de répondre aux problèmes de place mémoire et de temps de calcul posés par la simulation des écoulements turbulents dans les conduits à géométrie complexe.

La stratégie de parallélisation adopté s'est révélée payante. En effet, les tests numériques effectués sur un cas d'application classique montrent des résultats de plus en plus convenable en terme de l'efficacité. Du point de vue temps de calcul, on observe des gains très importants du code parallèle, ces gains étant fonction du nombre de processeurs utilisés.

# Conclusion générale

Dans le cadre de ce mémoire, nous avons développé un outil de simulation numérique d'un écoulement en charge, pour le traitement des écoulements turbulents. Vu les difficultés liées à la taille du problème et la complexité de la modélisation des écoulements dans des conduits à géométrie complexe, le but de cette étude est de présenter un modèle simplifié et abordable numériquement et de de faire une application de parallélisme sur le code.

D'abord une première approche, consiste à étudier l'écoulement de l'eau par le biais des équations de Navier-Stokes et de tenir compte de l'effet de prendre des condition aux limites assez convenable, ensuite, pour la deuxième approche, nous avons utilisé la modélisation de la turbulence pour écrire un modèle plus complet qui concerne les écoulements turbulents dans des conduits à géométrie complexe.

Sur le plein numérique, nous avons développé un code de calcul résolvant les problèmes liés à la complexité de la géométrie issus des schémas de Boltzmann sur réseau utilisés pour notre modèle.

Les résultats des tests numériques obtenus sur une coupe 2D d'un domaine réel se sont révélés encourageants et ont montré que le modèle est intéressant puisqu'il nous permet d'avoir des résultats en un temps assez petit.

Pour réduire le coût très important pour la simulation numérique sur des domaines réels, nous avons présenté une parallélisation d'un code numérique traitant le modèle BGK pour les écoulements turbulent. L'effort a été porté sur une parallélisation qui représente environ 62% du temps calcul du code total. Nous avons effectivement adoptés un gain important en temps.

Par la suite, vu les résultats encourageants obtenus en terme de performance, nous envisageons de généraliser la parallélisation au modèle corrigé afin de vérifier l'importance de le nombre de discrétisations. Pour augmenter plus les performances du code parallèle, il serait intéressant de prendre des conditions adéquates et qui devient importante quand on augmente le nombre de processeurs.

Enfin, pour répondre aux problèmes du temps de calcul posés par la simulation des écoulements turbulents dans les conduits à géométrie complexe, une solution consiste à utiliser toujours le calcul parallèle.

**Références bibliographie**


1. Abdelwahed, M., (2002). “Modélisation et simulation numérique d’écoulement diphasique“, " thèse de doctorat d’état en mathématiques appliquées. 165page.
2. Belakroum, R. & Khadja, M. & Zibouche, H. (2007) "Simulation numérique du phénomène d’éclatement tourbillonnaire dans la zone de sillage d’un obstacle de section circulaire". International Conférence on Energetics and pollution constantine. Vol 08, pp11-13.
3. Benmamar, S. (2006) "Etude des écoulements dans les conduits à motifs périodiques – Application aux évacuateurs de crues" thèse de doctorat d’état en hydraulique de l’Ecole Nationale Polytechnique d’Alger. ENP. 197page.
1. Bouffenech, R. & Djamai, Z. I, (2012). “ Simulation d’un écoulement pulse à motifs périodiques par la méthode lattice Boltzmann “. Ingénieur d’état en hydraulique à l’école nationale polytechnique d’Alger, 162 pages.
2. Chetibi, M., (2013). “ Simulation numérique d’un écoulement à surface libre dans des canaux à géométrie variable via PALABOS“. Ingénieur d’état en hydraulique à l’école nationale polytechnique d’Alger, 138 pages.
3. Cheng, J-Y., (2006). “Bending moment distribution along swimming fish“ Journal of Fluids and Structures VOL 23 PP 207–226.2006.
4. Chopard Droz, M., (2005). “Cellular automata modeling of physical systems“. Cambridge University Press, 18 pages.
5. Dupuis, A., (2002). "From a lattice Boltzmann model to a parallel and reusable implementation of a virtual river," PhD, Faculté des sciences de l'université de Genève, Université de Genève, 210 pages.
6. Goncalves, E., (2005). "Méthodes, analyse et calculs numériques "mémoire à institut polytechnique de Grenoble, 99 pages
7. Khabbouchi, I., & Guellouz, M.S. (2008) “Ecoulement autour d’un cylindre circulaire proche d’une paroi : Effet de l’écoulement type jet “. Laboratoire d’Etude des Systèmes Thermiques et Energétiques Ecole Nationale d’Ingénieurs de Monastir Rue Ibn El Jazzar 5019 Monastir, Tunisie. 145 Pages
8. Lam, K & Zou, L., (2009) “Experimental study and large eddy simulation for the turbulent flow around four cylinders in an in-line square configuration“. International Journal of Heat and Fluid Flow. Vol 30 PP 276–285.

9. Masselot, (2000). “A new numerical approach to snow transport and deposition by wind: a parallel lattice gas model“. These d’obtention de grade PhD, university of Geneva, 185 pages.

10. Oughlissi, M. & Zouggagh, M., (2011). “Simulation d’un écoulement hémodynamique dans un anévrisme par la méthode de Lattice Boltzmann “ Ingénieur d’état en hydraulique à l’école nationale polytechnique d’Alger, 78 pages.

11. Peixinho, J., (2012). "Ecoulement dans un tube faiblement divergent transition laminaire-turbulent" Laboratoire ondes et milieux complexes université de Havre. 157Page.

12. Robinet, j.C., (2010). "effet et modélisation de la turbulence" UEE d'Arts et Métiers ParisTech. 132page.

13. Saleh, K., (2008). "Simulation d'écoulements turbulents solution analytique de l'équation de la chaleur avec des conditions aux limites particulières". Projet de fin d’études département mécanique des fluides, énergie et environnement de l’école national des ponts et chaussées. 74pages

14. Sayoud, S., (2004). "Simulation numérique d'un écoulement turbulent dans une conduite à géométrie complexe Modèle K-ε" projet de fin d’étude de l’Ecole Nationale Polytechnique d’Alger. ENP. 61page.

15. Shan, X. Yuan, X.-F., & Chen, H., (2006). “Kinetic theory representation of hydrodynamics: A way beyond the Navies-Stokes equation “. Journal of fluid mechanics, vol. 550, pp. 413-441.

16. Takafumi, Y., (2006). “Flow around a circular cylinder in linear shear flows At subcritical Reynolds number“. Journal of Wind Engineering and Industrial Aerodynamics Vol 96 PP 961–973.

17. Taoussi, N., (2010). “Validation de Poiseuille d’un Modèle Lattice Boltzmann Hémodynamique “. Magister en sciences de l’eau à l’école nationale polytechnique d’Alger. 92 pages.