Revisiting grid refinement algorithms for the lattice Boltzmann method

Lagrava Sandoval, Daniel Walter

UNIVERSITÉ DE GENÈVE
Département d'Informatique

FACULTÉ DE SCIENCES
Professeur Bastien Chopard
Dr. Jonas Latt
Dr. Orestis Malaspinas

Revisiting Grid Refinement Algorithms For The Lattice Boltzmann Method

THÈSE

présentée à la Faculté des Sciences de l'Université de Genève
pour obtenir le grade de Docteur ès sciences, mention informatique

par

Walter Daniel Lagrava Sandoval

de Sucre (Bolivia)

Thèse N^{o}4520

GENÈVE
Atélier d'impression ReproMail
2012

UNIVERSITÉ
DE GENÈVE

FACULTÉ DES SCIENCES

Doctorat ès sciences
Mention informatique

Thèse de *Monsieur Daniel Walter LAGRAVA SANDOVAL*

intitulée :

"Revisiting Grid Refinement Algorithms for the Lattice Boltzmann Method"

La Faculté des sciences, sur le préavis de Messieurs B. CHOPARD, professeur ordinaire et codirecteur de thèse (Département d'informatique), J. LATT, docteur et codirecteur de thèse (Département d'informatique), O. MALASPINAS, docteur et codirecteur de thèse (Département d'informatique), A. CAIAZZO, docteur (Department of Civil and Environmental Engineering, University of Trento, Italy & Research Group Numerical Mathematics and Scientific Computing, Weierstrass Institute for Applied Analysis and Stochastics, Berlin, Germany) et R. KRAUSE, professeur (Institute of Computational Science, University of Lugano, Switzerland), autorise l'impression de la présente thèse, sans exprimer d'opinion sur les propositions qui y sont énoncées.

Genève, le 18 décembre 2012

Thèse - 4520 -

Le Doyen, **Jean-Marc TRISCONE**

N.B. - La thèse doit porter la déclaration précédente et remplir les conditions énumérées dans les "Informations relatives aux thèses de doctorat à l'Université de Genève".

Remerciements

Dans ces lignes je vais essayer de remercier toutes les personnes qui ont contribué à la reussite de ma thèse. Premièrement, le Prof. Bastien Chopard qui m'a donné l'opportunité de me lancer dans le monde de la recherche au sein de son groupe. J'aimerais aussi remercier les membres externes qui composaient le jury: le Prof. Rolf Krause et le Dr. Alfonso Caiazzo.

Cette thèse n'aurait pas été possible sans l'aide de deux personnes. Jonas Latt m'a aidé depuis le début, en m'apprenant le LBM avec sa patience et sa méthode toujours très pédagogique. Un grand merci pour avoir répondu à toutes mes questions et d'avoir reussi à me supporter pendant des années. Mon "maître" et grand ami, Orestis Malaspinas, a dû se montrer très estoïque pendant le temps qu'il m'a encadré. C'est lui qui m'a poussé dans les moments les plus durs de la thèse pour que celle–ci aboutisse à un résultat. En plus, le pauvre a eu le courage de faire la relecture de ce manuscript en entier au moins deux fois.

J'ai toujours pensé que je ne mérite pas les super copains que j'ai, en tout cas un grand merci à vous. J'essaie de vous passer en revue rapidement: Andrea, Christophe, Cristian, David, Diana, Fede, Pablo et Patrick. Grâce à votre présence, votre aide et votre amitié, cette période passée à Genève à été meilleure.

Il ne faut surtout pas oublier tous les amis et les collegues de Battelle: Alexis, Federico, Mohamed, Ranaivo, Jean Luc et Xavier. Merci pour la compagnie, l'amitié, les bons moments, toutes les pauses café, les mots croisés. Cela a été un vrai plaisir de partager ces années avec vous.

Para mi familia, las personas mas importantes del mundo. Gracias por el amor y el apoyo a Susy, mi madre, Juan José, mi padre y a mis dos hermanos, Juan Pablo y Laura, sepan que siempre han estado en mi corazón.

J'ai la chance d'avoir une famille en Suisse aussi. Cela serait une faute grave de ma part de les oublier. Donc, merci à vous Carmen, José, Mathias et Andreas, chez vous j'ai toujours eu un endroit où me sentir à la maison.

Finalement, Je me dois de remercier Fabiola pour tellement de choses vécues ensemble. Avec tout mon amour pour toi.

Contents

List of symbols

Acronyms & abbreviations

1D	One dimensional
2D	Two dimensional
3D	Three dimensional
BGK	Bhatnagar, Gross, Krook
CFD	Computational Fluid Dynamics
L.H.S.	Left Hand Side
LBM	Lattice Boltzmann Method
LGA	Lattice–Gaz Automata
NS	Navier–Stokes
R.H.S.	Right hand side

Adimensional numbers

D	Number of physical dimensions
Kn	Knudsen number
Ma	Mach number
Re	Reynolds number

Greek letters

$\boldsymbol{\xi}$	Particle velocity
Δ	Filter size
$\boldsymbol{\xi}_i$	Discrete particle velocity
μ_s	Dynamic viscosity
ν	Kinematic viscosity

ρ Density

τ Relaxation time

Others

δt Temporal discretization

δx Spatial discretization

Roman letters

C Set containing all the coarse sites $\boldsymbol{x}$

C_d Drag coefficient

C_l Lift coefficient

C_s Smagorinsky constant

c_s Sound speed

f^{eq} Equilibrium distribution function

f_i^{eq} Discrete equilibrium distribution function

f_i^{neq} Discrete non–equilibrium velocity distribution function

F Set containing all the fine sites $\boldsymbol{x}$

f Particle density distribution function

F_d Drag force

f_i Discrete particle density distribution function

F_l Lift force

G_i Grid of level i

p Pressure

$\boldsymbol{S}$ Strain rate tensor

t Time

$\boldsymbol{u}$ Macroscopic velocity field

$\boldsymbol{x}$ Coordinate

Résumé

La simulation de fluides est un problème multi-échelle. Si nous pensons à un écoulement turbulent par exemple, le nombre d'échelles qui y apparaissent s'étend sur plusieurs ordres de grandeur. Ceci devient critique lorsque nous voulons simuler les écoulements de fluides via une méthode numérique, car la quantité de points nécessaires pour représenter et résoudre toutes les échelles peut dépasser largement les capacités de calcul des plus puissantes machines actuelles.

Une solution qui a été proposée pour remédier à ce problème est l'utilisation de grilles non–uniformes. L'idée est d'essayer de garder le temps d'exécution raisonnable en augmentant la précision uniquement dans les endroits critiques de la simulation. Le problème des grilles non–uniformes a été étudié dans toutes les méthodes numériques dites traditionnelles.

Dans cette thèse, nous nous intéressons à une méthode numérique en particulier, la méthode de Boltzmann sur réseau. Dans sa formulation originale, cette méthode utilise des grilles uniformes et régulières. Si nous y introduisons des grilles non–uniformes, il y a des discontiniuités sur les quantités physiques qui apparaissent. La communauté a proposé un nombre d'algorithmes afin d'éliminer ces discontinuités sur les frontières. Malgré toute la littérature existante, nous sommes convaincus qu'il y a encore des zones d'ombre tant dans les bases théoriques, que dans les implémentations.

Parmi tous les algorithmes présentés par la communauté, nous avons choisi celui qui nous paraît le plus général [18]. Après avoir présenté les concepts les plus basiques du raffinement de grille, cet algorithme est décortiqué. Il s'en suit tous les détails théoriques nécessaires à la mise en place du couplage entre grilles de résolution différente. Une des nouveautés de cette thèse est d'avoir trouvé que ces opérations doivent posséder certaines charactéristiques minimales pour être consistantes.

La théorie a été appliquée à une implémentation générique du raffinement de grille. Nous avons décidé d'utiliser la librairie multi–physique Palabos. Cette librairie possède des implémentations vérifiées d'un grand nombre de modèles de Boltzmann sur réseau. En plus, Palabos propose une exécution en parallèle transparante pour l'utilisateur, qui a été testée sur des machines avec plusieurs milliers de processeurs. Finalement, un autre avantage de Palabos est qu'il s'agit d'une librarie open–source, donc il est possible de rendre notre code accessible aux personnes intéressées.

Pour témoigner de la fiabilité de Palabos, nous l'avons comparé à un logi-

ciel commercial sur un écoulement de flux sanguin dans une géométrie artificielle représentant un anévrisme. Nous avons mesuré un certain nombre de quantités physiques sur les deux logiciels, trouvant qu'elles sont en accord.

Notre code a été testé sur des problèmes à deux et trois dimensions. Nous avons pris soin de choisir des simulations intéressantes présentant des difficultées, comme des gradients de vitesse élevés qui traversent les frontières de raffinement. Les résultats obtenus prouvent que nos opérations de couplage entre les grilles sont robustes et précises. Nous avons notamment prouvé numériquement que le schéma garde le deuxième ordre de précision en temps et en espace.

Pour finir cette thèse, nous proposons deux sujets en relation avec le raffinement de grille. Premièrement, un nouvel algorithme de couplage pour des grilles avec des résolutions différentes. Ensuite, nous avons développé un critère algorithmique permettant d'identifier les regions de l'espace qui ont besoin d'avoir un raffinement local. Nous remarquons que ce type de critères sont nécessaires pour bien profiter des grilles non–uniformes et que la littérature dans le sujet est assez réduite.

Abstract

Fluid motion is an intrinsically multi-scale phenomenon. For instance, in the turbulent case, the range of excited scales of motion spans over many orders of magnitude. This becomes of great concern when trying to simulate fluids numerically. One solution to this problem is the usage of non-uniform grids or grid refinement. The main idea is to keep the computational time reasonable, while simulating all or most of the scales present in the problem. In particular, in this thesis, we are interested in grid refinement for the lattice Boltzmann method.

Although the topic has been already studied by the community, we are convinced that there is room for improvements, both in the theoretical and practical explanations provided so far.

We present thoroughly all the theory of the grid refinement for the lattice Boltzmann. As an interpolation operation is necessary in the selected approach, the minimal order of interpolation is explained. Further, we introduce a new filtering operation that is mandatory to study turbulent flows. Our generic implementation, which was developed on top of the Palabos open-source library, is explained in detail. Our approaches are tested on several two and three-dimensional benchmarks. We have specially chosen challenging cases, where, among others, large gradients cross the grid-refinement frontiers. Finally, we end with two other subjects related to grid refinement, namely a grid refinement algorithm that does not use an overlapping zone between the grids and a criterion to choose zones where local refinement is necessary.

Keywords : computational fluid dynamics, lattice Boltzmann method, multi–scale simulations, grid refinement, Palabos.

Notations

We introduce here the mathematical notations that are used throughout this thesis.

The scalar variables are noted with lower case characters. The vectors are represented in boldface lower case and the tensors in boldface upper case characters. A dot "$\cdot$" between two vectors is their scalar product, whereas the full index contraction is denoted by a colon ":". The tensor product of two vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ is denoted $\boldsymbol{ab}$.

The nabla operator, $\boldsymbol{\nabla}$, is the vector containing the space derivatives in each spatial direction if not specified otherwise. If there is an possible ambiguity the quantity on which the derivative depends is underscored.

Finally, we use $\langle\cdot\rangle$ to define a temporal mean of a quantity.

In the table below the reader will find a summary of the notations, with some examples.

Object	**Example**
Scalar	ρ, u
Vector	$\boldsymbol{u}$, $\boldsymbol{a}$
Tensor	$\boldsymbol{\Pi}$, $\boldsymbol{Q}$
Scalar product	$\boldsymbol{\xi} \cdot \boldsymbol{u}$
Full tensor contraction	$\boldsymbol{\Pi} : \boldsymbol{S}$
Tensor product	$\boldsymbol{c\xi}$
Nabla	$\boldsymbol{\nabla}_{\boldsymbol{x}}$, $\boldsymbol{\nabla}_{\boldsymbol{r}}$
Temporal mean	$\langle\rho\rangle$, $\langle\boldsymbol{u}\rangle$

Chapter 1

Introduction

The governing equations for fluids dynamics are solved with numerical methods, because they are notoriously intractable analytically. The continuum problem may be approximated at a set of discrete times, over a discrete number of spatial regions known as a mesh.

Depending on the particular problem to be solved, it might be desirable to locally assign more points to a certain region, while keeping the rest of the mesh untouched. Such an operation has been studied in all the traditional methods for computational fluid dynamics, in the so–callled non–uniform meshes or grids.

In this thesis, we work with a particular numerical method: the lattice Boltzmann method. Even though this method is relatively recent with respect to traditional numerical methods, it has proven to be a useful tool in many fields of application, such as complex multi–phase fluids [51] and bio–engineering [1].

In the original formulation, the lattice Boltzmann method is used over an evenly spaced grid. The introduction of non–uniform grids or grid refinement implies the creation of discontinuities of the physical quantities on the boundaries between grids with different refinement level. As grid refinement is a very important feature for a numerical method, the lattice Boltzmann community has tried to find a solution to this problem almost since the invention of the method.

The main goal of this introductory chapter is to provide the motivations for the current thesis. In the second part, we present the outline of the chapters that compose this document.

1.1 Motivations for this thesis

As we have already mentioned, the grid refinement is not a new subject in the lattice Boltzmann community. We find however, that there are some problems with what has been done so far.

First, we must note that there are many methods available for the grid refinement in the lattice Boltzmann framework. They exploit different properties of the method or use even different formulations. This can be seen as a good sign, because of the interest of the community to solve the problem. In opposition, we

find that this variety of approaches and methods can also be seen as a weakness, as there is clearly not one method that has gained the acceptance of the whole community.

Another problem that we have found in the literature is that, in many cases, not all the details needed to implement a particular grid refinement are provided.

Then, we must note that very often, the codes for each of the different grid refinement methods are not available to the public. For instance, it would be very interesting to benchmark them one against the other for a given problem, so that clear results on performances and precision can be assessed.

This thesis tries to settle some of these issues. After studying the literature on grid refinement, we have chosen a particular algorithm, which we considered to be general and an appropriate candidate for our generic implementation.

In the framework of this algorithm, we study in detail all the operations needed for the coupling of grids with several refinement levels. We have found that two coupling operations in this framework, which are the interpolation and the decimation, need to meet some minimal requirements to allow a consistent communication of all the important physical quantities.

All these concepts are applied into an implementation that uses an open–source library. We have tried to develop a generic code that hides the complexity of the problem to the user.

The fact that we have chosen to use an open–source library is motivated by the desire to share the code, so that people can test it and experienced users can find all the algorithmic details. We have performed a number of tests ourselves, to ensure that both the theoretical concepts and the implementation are actually sound.

1.2 Outline of this thesis

The chapters of this manuscript are organized as follows:

- The second chapter gives a short introduction to hydrodynamics and an introduction to the lattice Boltzmann method, which is the numerical method used through this thesis.

- The third chapter is devoted to the Palabos open–source library, which is the building block for the generic grid refinement that we have implemented. In this chapter we also present a benchmark between Palabos and a commercial software for a blood flow problem.

- In the fourth chapter we present the basics of grid refinement. We have tried to summarize as much as possible all the necessary concepts. We emphasize

in the couplings between the grids, which are explained in great detail. This chapter ends with a numerical proof of the importance of the order of the interpolation step.

- The details of the generic grid refinement and its implementation come in the fifth chapter. We test our implementation on several 2D and 3D examples, in order to prove the correctness of all the operations described in the fourth chapter.

- The sixth chapter proposes two new solutions for specific aspects of the grid refinement. First we explain an algorithm to couple two grids of different level with no overlapping zone between them. Then, we present a criterion that permits to detect zones where more precision is needed.

- In the last chapter, we give the conclusions of our work and we propose some perspectives for the future.

Chapter 2

The Lattice Boltzmann Method

The first part of this chapter is dedicated to a brief overview of the hydrodynamics field. The rest is devoted to the introduction and study of several aspects concerning the lattice Boltzmann method (LBM) [68, 12, 77], which is the numerical method that is used on the rest of this thesis.

2.1 Short introduction to hydrodynamics

In this section we first present the properties that describe a fluid and the equations that govern its motion. After that, we continue with the introduction of computational fluid dynamics (CFD). Finally, we discuss the different natures of the flows present in nature and define the Reynolds number as a classifier.

2.1.1 The basic equations

Let us start presenting the hydrodynamics, a field of physics that deals with the motion of a fluid.

Fluids have several macroscopic quantities that describe their state:

Density (ρ) which relates the quantity of mass per volume of the fluid.

Velocity ($\boldsymbol{u}$) that specifies the sense and rapidity of movement of the flow.

Pressure (p) is the force per unit of area applied to the fluid.

Viscosity (μ) or dynamic viscosity, that quantifies the resistance of a fluid to flow. It is also common to consider the kinematic viscosity ν, which is defined as $\nu = \mu/\rho$.

When a fluid has a constant kinematic viscosity ν, it is called a Newtonian fluid. Otherwise, in the case of non-Newtonian flows, the viscosity can depend both on space and on time. In the same manner, if the density of the fluid does not change over space or time, then the fluid is called incompressible.

In this thesis, we are interested in the study of incompressible Newtonian fluids. In this particular case, the basic equations that represent the evolution of the observable properties of the fluid are given by the continuity equation

$$\nabla \cdot \boldsymbol{u} = 0, \tag{2.1}$$

and the conservation of the momentum equation

$$\partial_t \boldsymbol{u} + (\boldsymbol{u} \cdot \nabla)\boldsymbol{u} = -\frac{1}{\rho}\nabla p + \nu \nabla^2 \boldsymbol{u}, \tag{2.2}$$

in absence of any external force.

These equations form a set of equations called the Navier–Stokes (NS) equations. The NS equations play a fundamental role in representing fluid motion, as they allow us to study the evolution of the properties that describe the fluid.

For further information about NS, the reader is invited to read the following references: Landau [37] and Ryhming [62] (only in French).

2.1.2 Computational fluid dynamics

Even though there exist analytical work on the NS equations in particular cases, when dealing with complex setups or turbulent flows they are intractable analytically. To solve them in practice, one needs to solve them with a numerical method in order to obtain an approximated solution to our problem. This field is known as CFD.

There exist several traditional numerical methods that are used in CFD. We are not using any of these methods, in consequence, they are not treated on this thesis. The interested reader is referred to Griebel [26] for an introduction to the finite differences, Versteeg [76] which deals with the finite volumes and Zienkiewicz [81], for the finite element method applied to fluid dynamics.

A rather different level of description was used with the lattice gas automata (LGA). We simulate fictitious particles that travel inside a uniform grid according to a fixed set of velocities. The particles have very simple collision rules, but these simple microscopical interactions produce a rich and complex macroscopic behaviour.

Notable and well-known examples of the LGA are the HPP (Hardy, Pomeau & de Pazzis) and the FHP (Frisch Hasslacher & Pomeau) LGA. Both of them and their properties are studied in detail in [12]. We must note that FHP was able to recover the NS equations correctly. However, there were many problems with these approaches. For instance, finding a 3D version for the LGA was a difficult issue which resulted in solutions that would use hypercubes and the projection of a four–dimensional version of the NS equations (see [16]).

The lattice Boltzmann method was proposed as a solution to the problems plaguing the LGA. Even though in its original formulation, it was conceived as a LGA, the method can also be obtained as the discrete version of another equation: the Boltzmann equation.

2.1.3 Nature of flows

Let us think about honey flowing slowly through a pipe and the flow caused by a spherical object moving very rapidly in the water. Both flows have a very different nature.

The first case is an example of a laminar behaviour. When a flow is laminar, the fluid moves like parallel layers. In real life however, most of the flows are of turbulent nature. In the turbulent case, the aforementioned layers are broken and the movement of the fluid becomes more chaotic. The flow, even following deterministic equations, becomes somehow random, and its evolution becomes difficult to predict.

Turbulent flows have the particularity to present phenomena in a vast number of space and time scales, which are often represented by the Kolmogorov scales (see [53]).

Consider Eq. 2.2. Let us define a characteristic velocity of the fluid U_0 and a reference length L, which is for instance important, like the diameter of the pipe or the diameter of the object in our examples. The non–linear term on the left hand side (L.H.S.) $(\boldsymbol{u}\cdot\nabla)\boldsymbol{u}$ can be approximated by U_0^2/L. In the same way, the viscous term on the right hand side (R.H.S.) $\nu\nabla^2\boldsymbol{u}$ can be approximated by $\nu U_0/L^2$. The ratio of this two quantities is the dimensionless number, called the Reynolds number Re

$$\mathrm{Re} = \frac{U_0 L}{\nu}. \tag{2.3}$$

The Reynolds number is one of the common ways that exist to define the nature of the flow. For small Re, the flow is laminar because the viscous term $\nu\nabla^2\boldsymbol{u}$ is significantly more important, implying that the viscous forces dominate the movement. Then there is a critical Re_c which delimits the creation of turbulence at higher values. A flow with Re values high above the critical value is turbulent. In this case, it is the non–linear term $(\boldsymbol{u}\cdot\nabla)\boldsymbol{u}$ which defines the behaviour of the flow. We note that this is the same exact term that is responsible for the difficulty of the NS equations.

2.2 The Boltzmann equation

The NS equations make a description of the fluid based on observable macroscopic properties. The kinetic theory of gases proposes a different approach. As the fluid is composed of particles, it is also possible to use classical mechanics to describe their motion and interactions, which will in turn dictate the macroscopic properties of the fluid. However, the number of particles contained in small quantities of fluids are huge. For instance, consider simply 1 cm^3 of water at ambient temperature. It contains approximately $3.3 \cdot 10^{22}$ molecules. To represent the detailed microscopic state and interactions of such system of particles is an intractable problem.

The statistical mechanics present yet another level of description. In this case, we do not track single particles, but we are interested in large numbers of particles and their collisions. In a region of the space $\boldsymbol{x} \in \mathbb{R}^3$ with a given velocity $\boldsymbol{\xi} \in \mathbb{R}^3$ at time t we associate a particle mass distribution function $f(\boldsymbol{x}, \boldsymbol{\xi}, t)$ to these particles. For the sake of simplification we note $f \equiv f(\boldsymbol{x}, \boldsymbol{\xi}, t)$ when no confusion is possible.

The number of particles found in a small volume d^3x near $\boldsymbol{x}$ with velocity $d^3\xi$ around $\boldsymbol{\xi}$ is computed by:

$$f d^3x d^3\xi \tag{2.4}$$

The Boltzmann equation describes the time evolution of f. The balance equation for a gas without any external force is given by

$$(\partial_t + \boldsymbol{\xi} \cdot \nabla_{\boldsymbol{x}}) f = \Omega(f) \tag{2.5}$$

where $\Omega = Q(f, f)$ is a term representing the internal collisions of pairs of particles that conform the gas.

The two first moments of f give us the macroscopic density:

$$\rho(\boldsymbol{x}, t) = \int f d\boldsymbol{\xi} \tag{2.6}$$

and momentum $\boldsymbol{j} = \rho \boldsymbol{u}$:

$$\boldsymbol{j}(\boldsymbol{x}, t) = \int \boldsymbol{\xi} f d\boldsymbol{\xi} \tag{2.7}$$

where both integrations are taken over the whole microscopic velocity space.

The collision term Ω is very difficult to evaluate (see for instance [77] for the complete expression). Therefore, there exist a number of models $\Omega \sim \bar{\Omega}$ that try to simplify it. A very well known approximation is given by the BGK model (for Bhatnagar, Gross, Krook [6]), that states that the distribution functions are driven towards an equilibrium state f^{eq} with relaxation time τ

$$\bar{\Omega} = \frac{1}{\tau}(f(\boldsymbol{x}, \boldsymbol{\xi}, t) - f^{eq}(\boldsymbol{x}, \boldsymbol{\xi}, t)). \tag{2.8}$$

In order for $\bar{\Omega}$ to have the same collision invariants as Ω, the equilibrium distribution can be of the Maxwell-Boltzmann distribution type:

$$f^{eq}(\boldsymbol{x},\boldsymbol{\xi},t) = \frac{\rho(\boldsymbol{x},t)}{(2\pi\theta(\boldsymbol{x},t))^{D/2}} \exp\left(-\frac{(\boldsymbol{u}(\boldsymbol{x},t)-\boldsymbol{\xi})^2}{2\theta(\boldsymbol{x},t))}\right), \tag{2.9}$$

where D is the number of dimensions of the system, $\theta = k_b T/m$, where k_B is the Boltzmann constant, m the particles mass and T is the temperature.

When f^{eq} is chosen to have this form, then its moments are the same as the ones for the particle distribution function f

$$\rho(\boldsymbol{x},t) = \int f^{eq} d\boldsymbol{\xi} \tag{2.10}$$

$$\boldsymbol{j}(\boldsymbol{x},t) = \int \boldsymbol{\xi} f^{eq} d\boldsymbol{\xi} \tag{2.11}$$

In Wolf–Gladrow [77], we find a nice proof that shows that under an asymptotic Chapman–Enskog multi-scale analysis, the incompressible NS equations can be obtained from Eq. 2.5 with the BGK approximation.

2.3 Discretization of the Boltzmann equation

Although we have mentioned that the lattice Boltzmann method (LBM) was developed in the line of thinking of the LGA, it can be obtained entirely from a clever discretization of the Boltzmann equation. There are many properties of the method that are a consequence of the discretization, so we have found interesting to present some of the details.

The first part of the discretization restrains the Boltzmann equation to a finite set of velocities. This is a rather tedious process with heavy calculations using the projection of the particle density functions over the Hermite polynomials, according to the ideas presented in Shan et *al.* [66]. As the whole calculations exceed the scope of this thesis, we only reproduce the most important results here. Nevertheless, the interested reader can find the complete calculation details in Malaspinas [46].

The corollary of the velocity discretization is that it is possible to define a discrete set of q velocities $\boldsymbol{\xi}_i$, $i = 0, ...q-1$ for which the Eq. 2.5 is still valid, but now reads

$$(\partial_t + \boldsymbol{\xi}_i \cdot \nabla_{\boldsymbol{x}}) f_i = -\frac{1}{\tau}(f_i(\boldsymbol{x},\boldsymbol{\xi}_i,t) - f_i^{eq}(\boldsymbol{x},\boldsymbol{\xi}_i,t)) \tag{2.12}$$

for each of the finite velocities $\boldsymbol{\xi}_i$. The f_i and f_i^{eq} are the discrete version of the particle distribution functions and equilibrium distribution function respectively.

The moments become discrete as well, giving us

$$\rho = \sum_{i=0}^{q-1} f_i = \sum_{i=0}^{q-1} f_i^{eq}, \tag{2.13}$$

$$\rho \boldsymbol{u} = \sum_{i=0}^{q-1} \boldsymbol{\xi}_i f_i = \sum_{i=0}^{q-1} \boldsymbol{\xi}_i f_i^{eq}, \tag{2.14}$$

The next step is a temporal and spatial discretization. Let us start by associating the partial differential operator in the L.H.S. of Eq. 2.12 with a total derivative with respect to the time

$$\partial_t + \boldsymbol{\xi}_i \cdot \nabla_{\boldsymbol{x}} \equiv \frac{\mathrm{d}}{\mathrm{dt}} \tag{2.15}$$

We can then integrate Eq. 2.12 with the BGK approximation along characteristics over the interval $[0, \delta t]$ obtaining

$$\begin{aligned} f_i(\boldsymbol{x} + \delta t \boldsymbol{\xi}_i, \boldsymbol{\xi}_i, t + \delta t) - f_i(\boldsymbol{x}, \boldsymbol{\xi}_i, t) = \\ - \frac{1}{\tau} \int_0^{\delta t} [f_i(\boldsymbol{x} + \boldsymbol{\xi}_i s, \boldsymbol{\xi}_i, t + s) + f_i^{eq}(\boldsymbol{x} + \boldsymbol{\xi}_i s, \boldsymbol{\xi}_i, t + s)] \mathrm{d}s \end{aligned} \tag{2.16}$$

We recall that the trapezoidal method to approximate integrals is given by the following formula

$$\int_x^{x+\delta x} f(x) dx = \frac{\delta x}{2} [f(x + \delta x) + f(x)] + O(\delta x^3) \tag{2.17}$$

We apply the trapezoidal method to Eq. 2.16 in order to compute the two unknown integrals on the right hand side. We obtain then:

$$\begin{aligned} f_i(\boldsymbol{x} + \delta t \boldsymbol{\xi}_i, \boldsymbol{\xi}_i, t + \delta t) - f_i(\boldsymbol{x}, \boldsymbol{\xi}_i, t) = \\ - \frac{\delta t}{2\tau} \{ f_i(\boldsymbol{x} + \delta t \boldsymbol{\xi}_i, \boldsymbol{\xi}_i, t + \delta t) + f_i(\boldsymbol{x}, \boldsymbol{\xi}_i, t) \\ - f_i^{eq}(\boldsymbol{x} + \delta t \boldsymbol{\xi}_i, \boldsymbol{\xi}_i, t + \delta t) - f_i^{eq}(\boldsymbol{x}, \boldsymbol{\xi}_i, t) \} \end{aligned} \tag{2.18}$$

obtaining an implicit equation (we do not know how to compute $f^{eq}(\boldsymbol{x} + \delta t \boldsymbol{\xi}_i, \boldsymbol{\xi}_i, t + \delta t)$). One can get rid of this problem with the following variable change:

$$\bar{f}_i = f_i + \frac{\delta t}{2\tau} (f_i - f_i^{eq}) \tag{2.19}$$

One finally obtains the explicit BGK discrete equation:

$$\bar{f}_i(\boldsymbol{x} + \delta t \boldsymbol{\xi}_i, \boldsymbol{\xi}_i, t + \delta t) - \bar{f}_i(\boldsymbol{x}, \boldsymbol{\xi}_i, t) = -\frac{1}{\bar{\tau}} [\bar{f}_i(\boldsymbol{x}, \boldsymbol{\xi}_i, t) - f_i^{eq}(\boldsymbol{x}, \boldsymbol{\xi}_i, t)] \tag{2.20}$$

where $\bar{\tau} = \tau + \frac{1}{2}$ is the modified relaxation time of the model.

2.4 Lattice Boltzmann Method

In this section we detail more about the LBM, introducing examples of the discrete velocities and the most important results of the Chapman–Enskog multi–scale expansion. Then, we present an algorithm for the implementation of the LBM.

The interested reader can find further informations in Succi [68], Wolf–Gladrow [77] and Chopard et *al.* [12].

2.4.1 Basic definitions

In the LBM, we are going to simulate the numerical scheme presented in Eq. 2.20. Because of this, we can omit the $\bar{\cdot}$ operator on the concerned quantities with no further considerations. At this stage, one obtains the kinetic equation of the LBM for the BGK model. For the sake of simplicity, we suppose that $\delta t = 1$, then the final equation reads

$$f_i(\boldsymbol{x} + \boldsymbol{\xi}_i, t+1) = f_i(\boldsymbol{x}, t) - \frac{1}{\tau}\left(f_i(\boldsymbol{x}, t) - f_i^{eq}(\boldsymbol{x}, t)\right). \tag{2.21}$$

Here f_i^{eq} is the discrete equilibrium distribution function, τ the modified relaxation time, and δt the discrete time step.

The relaxation frequency ω is defined by

$$\omega = \frac{1}{\tau} \tag{2.22}$$

The discrete equilibrium distribution is defined by a truncated version of the Maxwell–Boltzmann distribution

$$f_i^{eq} = w_i \rho \left(1 + \frac{\boldsymbol{\xi}_i \cdot \boldsymbol{u}}{c_s^2} + \frac{1}{2c_s^4}\boldsymbol{Q}_i : \boldsymbol{u}\boldsymbol{u}\right), \tag{2.23}$$

where ρ is the density, $\boldsymbol{u}$ is the macroscopic velocity field, $\boldsymbol{Q}_i = \boldsymbol{\xi}_i\boldsymbol{\xi}_i - c_s^2\boldsymbol{I}$, c_s and w_i the lattice speed of sound and the lattice weights respectively. We note that ρ and $\boldsymbol{u}$ are computed by the distribution function through the relations presented in Eqs. 2.13 and 2.14 respectively.

When the discrete equilibrium has this form, then the following relations hold

$$\rho = \sum_{i=0}^{q-1} f_i^{eq}, \tag{2.24}$$

$$\rho\boldsymbol{u} = \sum_{i=0}^{q-1} \boldsymbol{\xi}_i f_i^{eq}. \tag{2.25}$$

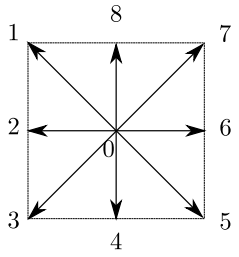


Fig. 2.1: The D2Q9 lattice with the vectors representing the microscopic velocity set $\boldsymbol{\xi}_i$. A rest velocity $\boldsymbol{\xi}_0 = (0, 0)$ is added to this set.

The pressure p is defined by the perfect gas law relation

$$p = c_s^2 \rho \tag{2.26}$$

and the kinematic viscosity ν follows

$$\nu = c_s^2 (1/\tau - 1/2), \tag{2.27}$$

We have a set of q velocities in a D dimensional space which results into a DDQq model. For instance, the D2Q9 lattice (see Fig. 2.1) which is given by:

$$\begin{gathered} c_s^2 = 1/3,\ w_0 = 4/9,\ w_j = 1/9,\ j = \{2,4,6,8\},\ w_k = 1/36,\ k = \{1,3,5,7\} \\ \{\boldsymbol{\xi}_i\}_{i=0}^{8} = \{(0,0),(-1,1),(-1,0),(-1,-1),(0,-1), \\ (1,-1),(1,0),(1,1),(0,1)\}. \end{gathered} \tag{2.28}$$

By performing a multi-scale Chapman–Enskog (CE) expansion (see [8, 12] for more details) one can show that the LBM BGK discretization is asymptotically equivalent to the incompressible Navier–Stokes equations (Eqs. 2.1 and 2.2).

The CE expansion is done under the assumption that f_i is given by a small perturbation of the equilibrium distribution

$$f_i = f_i^{eq} + \varepsilon f_i^{(1)} + \mathcal{O}(\varepsilon^2), \tag{2.29}$$

where $\varepsilon \ll 1$ can be identified with the Knudsen number (see [33]).

During the CE expansion, one finds that $f_i^{(1)}$ is given by

$$\varepsilon f_i^{(1)} = \frac{w_i}{2c_s^4} \boldsymbol{Q}_i : \boldsymbol{\Pi}^{(1)}, \tag{2.30}$$

where the tensor $\boldsymbol{\Pi}^{(1)} \equiv \sum_i \boldsymbol{\xi}_i \boldsymbol{\xi}_i \varepsilon f_i^{(1)}$ is related to the strain rate tensor $\boldsymbol{S}$ through the relation

$$\boldsymbol{\Pi}^{(1)} = -2c_s^2 \rho \tau \boldsymbol{S}, \tag{2.31}$$

where $\boldsymbol{S}$ is defined by

$$\boldsymbol{S} = \frac{1}{2}\left(\boldsymbol{\nabla u} + (\boldsymbol{\nabla u})^{\mathrm{T}}\right). \tag{2.32}$$

The f_i can be approximated by

$$f_i = f_i^{eq} + f_i^{neq}, \tag{2.33}$$

where f^{neq} is the non–equilibrium part of the particle distribution function. According to the assumptions of the CE expansion f_i^{neq} is proportional to $f^{(1)}$

$$f^{neq} \sim \varepsilon f^{(1)} \tag{2.34}$$

This allows us to express the f_i depending only on macroscopic values ρ, $\boldsymbol{u}$ and $\boldsymbol{S}$

$$f_i = w_i\rho\left(1 + \frac{\boldsymbol{\xi}_i \cdot \boldsymbol{u}}{c_s^2} + \frac{1}{2c_s^4}\boldsymbol{Q}_i : \boldsymbol{uu}\right) - \frac{w_i\rho\tau}{c_s^2}\boldsymbol{Q}_i : \boldsymbol{S}. \tag{2.35}$$

We end this section by noting that the CE expansion comes from the physics community. For a more numerical approach, the reader is invited to read the papers dealing with the asymptotical analysis (AA) (see [35]).

2.4.2 The algorithm

In this part, we enumerate the operations to implement the LBM BGK algorithm.

We start with the decomposition of a time step into two parts that are applied successively on the whole computational domain.

1. The **collision**, which modifies *locally* the value of the populations according to
$$f_i^{\mathrm{out}}(\boldsymbol{x}, t) = f_i(\boldsymbol{x}, t) - \frac{1}{\tau}\left(f_i(\boldsymbol{x}, t) - f_i^{eq}(\boldsymbol{x}, t)\right). \tag{2.36}$$
2. The **streaming**, which moves the populations to their neighbours according to their microscopic velocity
$$f_i(\boldsymbol{x} + \boldsymbol{\xi}_i, t+1) = f_i^{\mathrm{out}}(\boldsymbol{x}, t). \tag{2.37}$$

In Fig. 2.2 both steps are executed over a site $\boldsymbol{x}$. The f_i become f_i^{out} and then become f_i again but at the neighbouring site $\boldsymbol{x} + \boldsymbol{\xi}_i$.

We note that the collide and stream steps can be executed in a single operation, which we call the "collide-and-stream" operation. This conversion is very advantageous for the implementation as it allows better performances and a save of memory (see [52]).

The complete algorithm is given by:

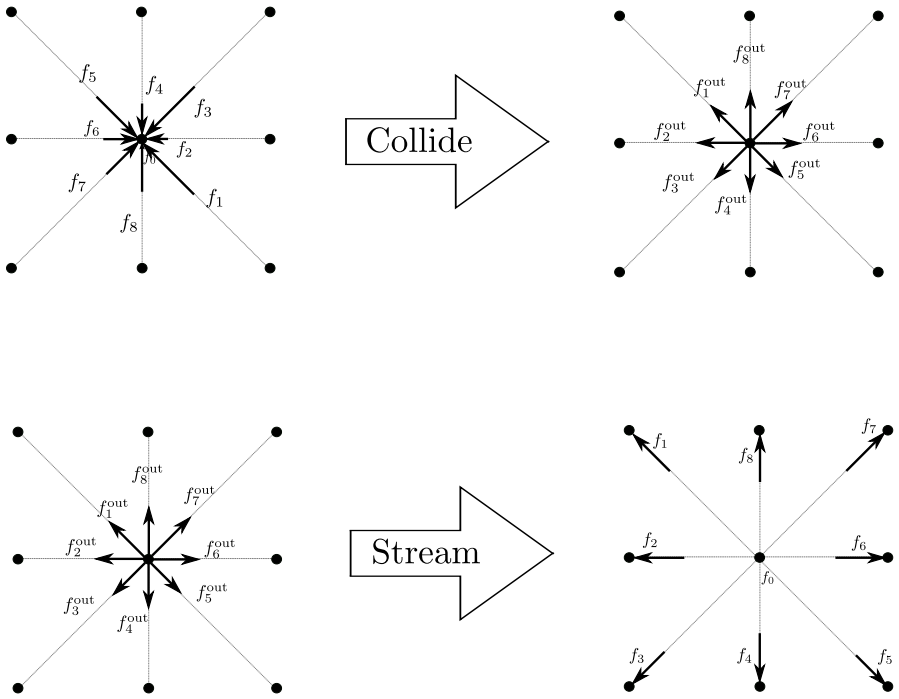


Fig. 2.2: A depiction of a collide and a stream operations on a $D2Q9$ model.

1. Define and compute all the necessary variables for the simulation, for example Re, so that our fluid has the desired behaviour.

2. Initialize the f_i to some values. One usually defines a starting set of values for ρ and $\boldsymbol{u}$, which serve to compute $f_i^{eq}(\rho, \boldsymbol{u})$. Then we simply start by supposing that $f_i = f_i^{eq}$.

3. Compute the macroscopic variables ρ and $\boldsymbol{u}$ in every site $\boldsymbol{x}$.

4. With this macroscopic variables we compute f_i^{eq}

5. We apply the collide and stream operations to modify the f_i.

6. At this point one whole iteration of the simulation has been performed. We can verify if some stopping criteria has been met, otherwise we return to point 3.

2.5 Estimation of error for the LBM

In this section, we show that the error of the LBM consists of three ingredients: spatial error, temporal error and a compressibility error. We must note that this third term appears only when we recover the incompressible NS equations.

The values for the spatial and temporal errors are a consequence of the usage of the trapezoidal schema when performing the integration of the discrete Boltzmann equation, which is shown above. We note that even though the trapeze is $O(\delta t^3)$, this holds only for a time step. For all the time steps between $0, \ldots, t$ the order is reduced to $O(\delta t^2)$.

We note that in the paper by He et *al.* [30], there is an integration of the discrete equation that incorrectly proves that the LBM is only first order accurate. The problem with their approach is the integration method used.

The error for a LBM BGK schema with time step δt and space step δx obeys the equation:

$$E(\delta x, \delta t) = O(\delta x^2) + O(\delta t^2) + O(\mathrm{Ma}^2). \tag{2.38}$$

The schema is second order in time and space. When simulating a system with the LBM, in order to decrease the error, it is necessary to decrease both δx and δt. There is however a supplementary constraint which relates these discretizations in the form of the third term $O(\mathrm{Ma}^2)$.

The term $O(\mathrm{Ma}^2)$ originates by the compressibility error, as we assume that our fluid is not compressible or slightly compressible during the Chapman–Enskog expansion. This term is proven to behave like $O(\delta t^2/\delta x^2)$ in Latt [38]. The existence of this relation obliges temporal and spatial discretizations to be tied together, so that an overall order for the schema is preserved. There exist two popular types of scaling that control the dependence between spatial and temporal discretizations.

The first one is the so-called convective rescaling of quantities, in which we state that $\delta t \sim \delta x$. This means that when dividing δx by a factor n to obtain a better precision, the same operation must be applied to δt. When analysing the error evolution, we see that error will be second order in space. However, there is a drawback with the last term, which becomes a constant term $O(1)$. This means that whatever the discretization step we use, we end with a given constant error. Nevertheless, this rescaling is faster, as both δx and δt vary linearly, in opposition to the rescaling that follows.

The second popular rescaling is the diffusive rescaling. In this case, we impose that $\delta t \sim \delta x^2$. In this case, when replacing this values in Eq. 2.38, we obtain again a method of second order in space. What we have achieved is to make the incompressibility error disappear, as it becomes $O(\delta x^2)$. This rescaling is thus used to get rid of the compressibility error. Nevertheless, it is clear that it is more expensive in computational terms. When dividing the spatial discretization δx by

a factor n, it is mandatory to divide δt by a factor n^2.

We note that rescalings become relevant and worth mentioning for the following chapters. The reason being that more than one spatial and temporal discretizations exist at once in the presence of non–uniform grids.

2.6 Summary

In this chapter, we have made a small introduction to the hydrodynamics and the numerical method used in this manuscript.

We have started by introducing the macroscopic properties that describe a fluid, as well as the NS equations, which govern their variations. Then, we talk about CFD, citing several traditional methods, but also presenting the LGA, as an alternative for the discretization of the NS equations. We also explain that the LBM was conceived as an improvement of the LGA. Next, we classify the flows of different fluids according to their nature: laminar and turbulent. We introduce the Reynolds number as a classifier of the nature of a flow.

A more analytical fashion to obtain the LBM is the discretization of the Boltzmann equation. We start by briefly presenting this equation. We mention that it has been proved to be asymptotically equivalent to the NS equations. Then, we move to the discretization of the Boltzmann equation. The discretization of the velocity space for the Boltzmann equation is quite long, so we have chosen to detail only the time and space discretization.

Once we have found the discrete BGK equation, we present the basic ingredients of the LBM, namely the relation between the most important physical quantities involved. We also enumerate the most important results obtained from the CE expansion of the LBM kinetic equation. To end this section, we show the basic algorithm of the LBM and define the collide and stream operations.

Finally, we analyse the error of the LBM. As a consequence of the integration used for the discrete Boltzmann equation, the method is second order accurate in time and space. There is however a supplementary constraint when recovering the incompressible NS equations in the form of the so–called compressibility error, which ties the time and space discretizations. We then present the two popular rescalings that exist to relate space and time: the convective and the diffusive rescalings.

Chapter 3

The Palabos open-source library

This chapter is devoted to the description of the library used to develop the generic grid refinement, the Palabos open source library [39]. The first section presents the basic parts and most basic data–structures of Palabos. Then, we discuss some of the more advanced models that have been included and implemented in Palabos. In the next section, we detail a benchmark to compare Palabos and the commercial software CFX for steady blood flow in a geometry representing an aneurysm, a pathological malformation of a blood vessel.

3.1 Features of Palabos

Palabos is an open source library for fluid simulations conceived by Jonas Latt. In order to provide both a good performance and maintainable code, the language of choice is C++. Currently, there exist Java and Python bindings that allow to use the features of the library from these languages as well.

The idea behind Palabos is to simplify the work required to perform a simulation, by providing a simplified syntax and by hiding the complexity of the numerical model from the user. Another important feature is to provide facilities to an advanced user that wishes to implement a particular model into the library.

A great amount of models and algorithms are already implemented in Palabos. In the following we describe some of the featured ones in Palabos.

We begin with the boundary conditions. We have not presented them before, but they are a necessary ingredient to any simulation to have a well–posed problem. First, we cite the straight wall boundary conditions. We have for instance:

- the Skordos boundary condition [67]
- the Zou/He boundary condition [82]
- the Regularized boundary condition [40]

The reader is invited to read Latt et *al.* [42] for more on straight wall boundary conditions.

Often, we need to represent arbitrary geometries. For this purpose, it is possible to use another boundary condition, called bounce–back. In this case, a lattice site can be either fluid or solid. The only drawback of this approach is that complex curvy objects are represented by a staircase approximation.

An improvement of this staircase approximation is possible with the off–lattice boundary conditions. This state–of–the–art boundary conditions represent solid walls with more accuracy, as we suppose that the solid wall does not necessarily need to correspond exactly to a lattice site. The Guo boundary condition [27], which exists in Palabos, is a good example of this kind of boundary condition.

We have presented the BGK collision model for the Boltzmann equation. Nevertheless, there exist a number of different collision operators. The interest of this models is to provide increased numerical stability or different physical behaviour. Let us for example cite the following models implemented in Palabos:

- the multi–relaxation time model [15].
- the entropic lattice Boltzmann [7].

Besides the details that we have presented, Palabos also implements a number of complex models to solve multi–physics problems. We can cite:

- Multi–phase models, like the Shan/Chen [65] and Lee [43] models, where we simulate the interaction of two or more immiscible fluids with different physical properties, like density or viscosity. The model computes the surface tension between the interfaces of the liquid in the contact zone. We note that this models are restricted to small density ratios.
- Free–surface flows, which is a model for immiscible multi–phase systems formed a liquid and a gas, where the ratio of densities is very important. Normally the gas is supposed to be inviscid and is not simulated. Instead, we apply a surface tension to the liquid to simulate the effects of the gas. The algorithm is based on [36].
- Lagrangian particles, which can be used to trace the stream–lines of the flow for instance. But this is not the only use one can make of particles. The addition of this model allows to define particles which have their own characteristics and can interact between them and with the fluid. This can be used to model red blood cells for example (see [48]). They can also be used to model two miscible fluids, allowing to avoid the diffusive non–physical term that appears with standard multi–phase models and give more importance to the molecular level (see [49]).

Thanks to a constantly growing user community, all the models in Palabos, which extends over several thousand lines of code, have the opportunity to be constantly tested. This has allowed to correct many bugs and enhance the capabilities of the library over the time, making it more reliable.

As we have mentioned in the beginning of this section, Palabos offers many advantages for advanced users to implement other models using the framework. We present two of these features, which are very important when implementing the generic grid refinement.

In Palabos, the simplest lattice where one can perform a simulation is called a **block**. A much more useful data–structure called **multi–block**, which is a logical group of several of the aforementioned blocks. It ensures the communication among the internal blocks, while presenting the behaviour of a single unit to the user.

The multi–block approach provides many advantages such as:

- Sparse representation: only the needed parts of the domain are allocated, thus saving space when we deal with domains that have holes or other non allocated regions inside. For instance, see Fig. 3.1 where we use different number of blocks to cover the same geometry. In certain geometries, we have achieved an economy of almost 45% of the fluid nodes by using a sparse representation.

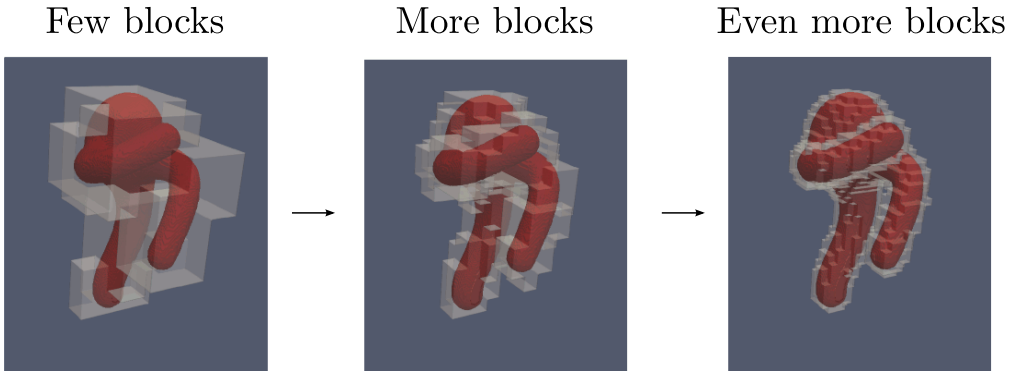


Fig. 3.1: Sparse representation of the same geometry: depending on the quantity of blocks, less fluid nodes are allocated.

- Parallelism: it is possible to divide a domain in several blocks. Latter this blocks are attributed to different processors. The simulation can then run effectively in parallel thanks to communications using the MPI library. This feature has been tested in huge clusters containing several thousand cores with a really good scaling.

This possibility of having distributed multi–blocks raises and additional question: how to apply operations over them in a transparent way to the user? The solution that has been proposed to this problem are the **processing functionals**. The user must define the operations to be executed over a single block. When needed, every processor identifies the local blocks which he contains and applies the operation to them. Finally, in order to provide more flexibility, the user has the choice to either include them permanently on a multi–block, so that they are executed automatically or apply them at a precise moment.

3.2 Benchmark of Palabos and CFX for blood flow

We have performed a quantitative comparison between Palabos and the ANSYS CFX numerical solver. The target problem was the blood flow in a synthetic geometry representing a blood vessel attained of a deformation of the arterial walls, called an aneurysm.

The Palabos simulations were performed by the Scientific and Parallel Computing (SPC) group at the university of Geneva. In order to have someone with a broad experience with the CFX software, we contacted the Laboratory of Hemodynamics and Cardiovascular Technology (LHCT) from EPFL, directed by Prof. N. Stergiopulos. This laboratory has a solid experience with this software, which they have used for many years. The simulations were performed by Dr. Dimitrios Kontaxakis.

3.2.1 Motivation

The motivation to benchmark both softwares is to give a complete quantitative comparison of the LBM with the finite volumes method which is implemented in ANSYS CFX, assessing both the advantages and problems with each of the methods. We know that the LBM is an intrinsically time-dependant method, so the question is how does it compare with the finite volumes for a steady problem.

Not only the simulation is in itself compelling, but also the study of the complete pipeline of operations needed to achieve the results in both cases, in order to better understand the forces and weaknesses of each approach.

This is an excellent opportunity to prove not only that the physical values for both softwares are the same, but also to verify if it is possible to compute them up to an arbitrary number of significant digits.

Finally, it is very interesting to compare this two softwares, Palabos being an open–source software and CFX a commercial product.

3.2.2 Geometry and simulation set-up

We have chosen to use a synthetic geometry for the simulation. To generate it, we have used the open-source software shapes [5], which allows us to create smooth 3D geometries composed from XML files. Shapes produces an STL (STandard Litography) file. This is a standard form of representing for surfaces in 3D.

The interest of using an artificial geometry resides in the fact that it is not subject to copyright or legal issues, as it is very often the case with real data taken from patients. The artificial geometry can be distributed to other people with no further considerations, so that they are able to reproduce the benchmark. For instance, the benchmark geometry in STL format can be found in the current Palabos distribution.

The geometry is shown in Fig. 3.2. We have defined one inlet which is found on the bottom and two outlets. For the inlet we apply a Poiseuille velocity profile. We use a constant pressure for both outlets. As boundary condition, we have chosen to use off–lattice boundary conditions, because of their increased accuracy when compared to bounce–back.

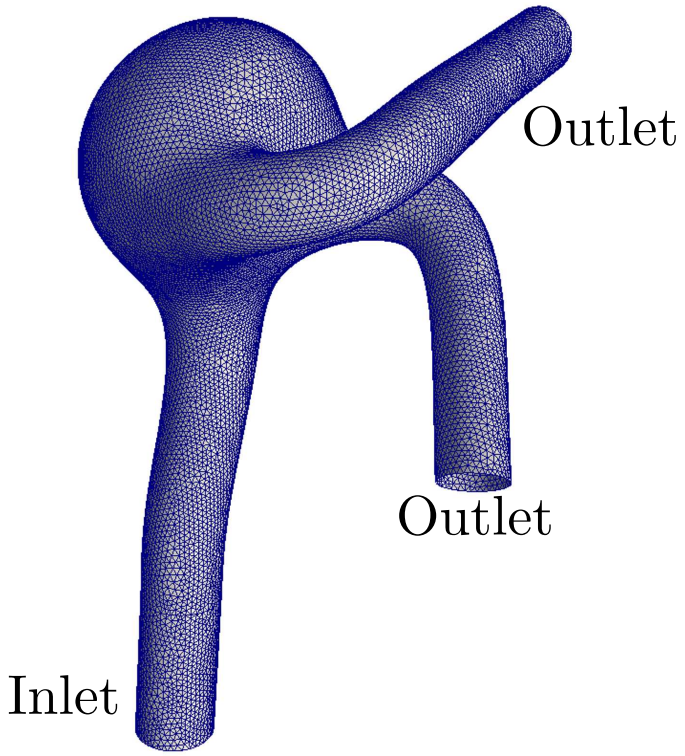


Fig. 3.2: Synthetic geometry used for the simulations.

We compute the mean Reynolds number using $\mathrm{Re} = U_{\mathrm{inlet}} L/\nu$, with $U_{\mathrm{inlet}} = 0.071176$ the mean velocity in the inlet (m/s), $L = 0.012$ the diameter of the inlet (m) and $\nu = 3.7037e-06$ the kinematic viscosity of the blood (m/s^2). One obtains a value of $\mathrm{Re} = 230.61$, which corresponds to the regime of the carotid artery (see [58]). All these quantities are then converted to lattice units for the simulation.

3.2.3 Methodology: Palabos

To accelerate the convergence for Palabos, we have decided to use a multi–grid technique[1]. The idea is to solve the problem over a coarse grid G_0 first. This grid is not able to correctly represent all the small scales of the problem, only the big ones, but the convergence is fast. One can use the steady state of G_0 as a starting condition for a simulation over a finer grid G_1. This avoids the need to propagate all the big scales over G_1 entirely, which only has to solve the smaller scales. In this fashion, even finer grids can be added to the pipeline, in order to further reduce the total computation time, with respect to use only one fine grid.

We have decided to use three grids for our problem: G_0, with only almost 7 lattice sites for the inlet, G_1 which is twice finer than G_0 and G_2, that is four times finer than G_0. In order to decide the convergence of a grid we use the mean of the kinetic energy $U_k = |\boldsymbol{u}|/2$ over all the fluid nodes of the domain. All the simulations are executed in Intel Xeon E5620 cores.

The results are shown in Fig. 3.3. There we analyse the complete evolution of the mean energy over each grid. We start with G_0, which has converged in two minutes and 12 dimensionless time units, so taking almost 10 seconds per dimensionless time. Then, we use the steady state found with G_0 as initial condition for the G_1, and in converges in 11 minutes. G_1 needs 9 dimensionless time units to converge, meaning that a dimensionless time unit of G_1 takes more than a minute. Finally, the finest grid G_2 uses the steady state of G_1 as starting condition. It converges in only three dimensionless time units, which nevertheless take 52 minutes of computational time, or almost 17 minutes per dimensionless time.

In Tab. 3.1, we summarize the values of mean kinetic energy U_k and mean vorticity[2] ω obtained for each of the three grids for the simulation.

The conclusion is that the total time of computation is almost 64 minutes for the three grids. In opposition, we suppose that we use only G_2 and that the system needs 10 dimensionless units to converge. This would be equal to 170 minutes of computation in total. This proves that the multi–grid methods are useful to reduce the computational time.

[1] Here, multi–grid refers to a method to solve the problem over several grids with different resolutions. See for instance [71].

[2] The vorticity is defined by $\omega = \partial_x u_y - \partial_y u_x$

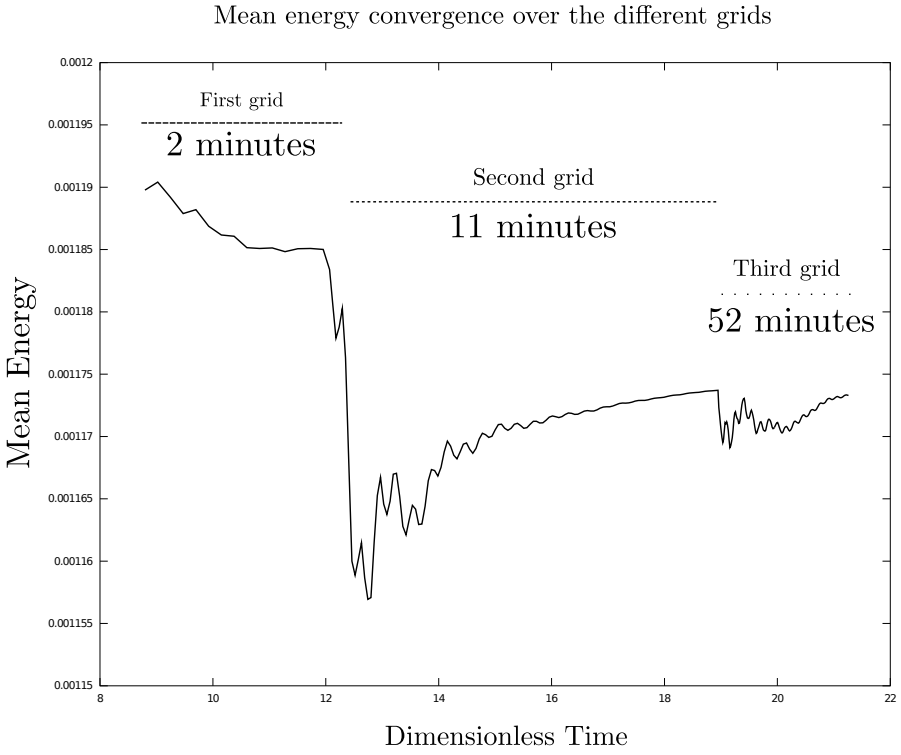


Fig. 3.3: Convergence of the multi–grid system in dimensionless time units.

Table 3.1: Acceleration of the convergence using multiple grids.

	G_0	G_1	G_2 (finest)
Inlet radius (lattice nodes)	6.9	13.8	27.5
Site number	87360	698515	5586266
Computation time (min.)	2	11	52
Mean energy	0.00118506	0.00117371	0.0011733
Mean vorticity	0.000524386	0.000542149	0.000548843

We note that the mean value for the energy is not completely wrong for the coarsest grid when compared to the one from the finest value (error of 1% only). The value of the mean vorticity has an error of 4%. In Fig. 3.4 we plot several streamlines over the same plane across the aneurysm for the three grids at the steady state. In this figure, we confirm that G_0 provides an already acceptable behaviour with respect to the two other finer grids.

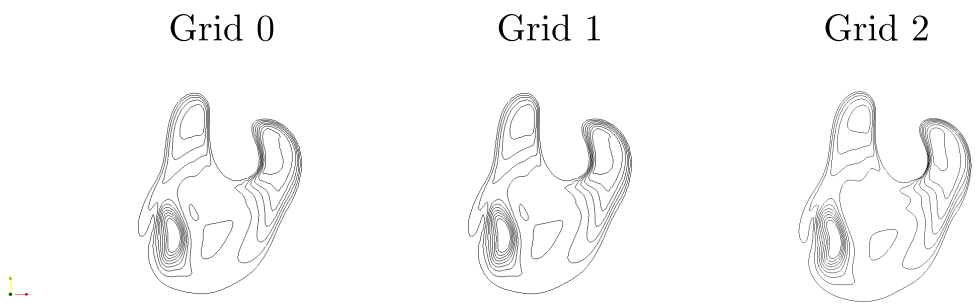


Fig. 3.4: Stream lines on a plane for the three grids.

3.2.4 Results

In order to compare the solutions provided by both solvers, we define several planes and points of interest. The idea is to compare physical quantities point-wise and to compare the iso-surfaces of velocity on the planes. Several integral mean quantities are also computed over the whole domain. In Fig. 3.5, we depict these six points and three planes.

In Tab. 3.2, we find the coordinates of the six points used for the comparison. We note that these coordinates are given with respect to the STL file reference. In the same way, we define the three planes that interest us, which are described on Tab. 3.3.

Table 3.2: Coordinates of the points for the comparison.

Point	**Coordinates**
P_1	$(0.022046, 0.015072, 0.044152)$
P_2	$(0.027132, 0.049947, 0.095012)$
P_3	$(0.027132, 0.049947, 0.095012)$
P_4	$(0.031492, 0.025971, 0.084113)$
P_5	$(0.025679, 0.025971, 0.091379)$
P_6	$(0.018413, 0.011439, 0.076848)$

Table 3.3: Coordinates of the planes for the comparison.

Plane	**Coordinates**
Plane 1	$x = 0.016960$
Plane 2	$y = 0.017978$
Plane 3	$z = 0.084113$

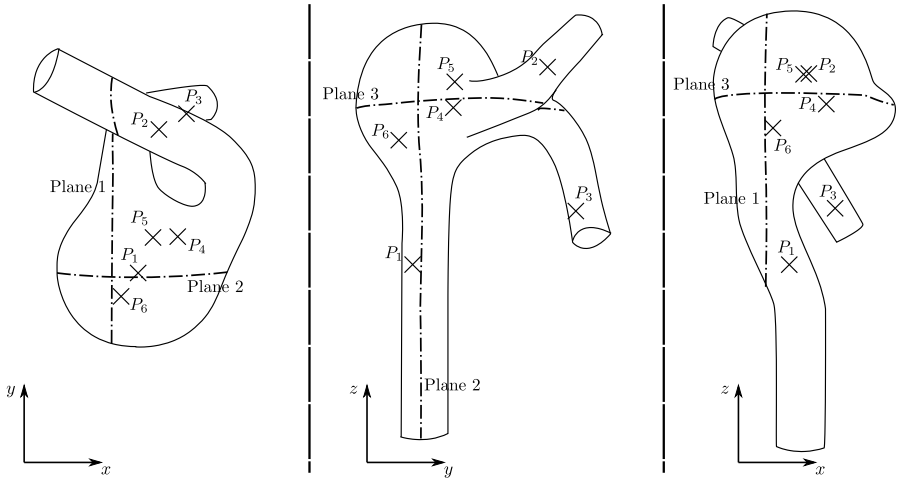


Fig. 3.5: Planes and points of interest.

For each solver we have defined two grids. The first one is the benchmark grid, which is the one we use for comparison with each other. The other one is the reference grid, which is a grid with many more points, that we define as the reference solution. For Palabos, the benchmark grid contains 700000 points, while the reference solution contains up to 6 million points. In the case of CFX, the benchmark grid has up to 1.2 million points and the reference one, 7 million.

Let us start analysing the results for mean quantities when the steady state has been achieved. The results can be found in Tab. 3.4. We evidence that the kinetic energies are very close for all the set-ups. The same happens with the enstrophy, which this time is close to two significant digits. The pressure error is only of almost 6% with respect to the value for CFX. For the reference grids, the error is even smaller, their difference being only of 3%.

Table 3.4: Mean physical quantities.

	Kinetic Energy	**Enstrophy**	**Pressure**
CFX (Benchmark)	5.531e-8	1.252e-2	10.28
CFX (Reference)	5.569e-8	1.290e-2	10.22
Palabos (Benchmark)	5.668e-8	1.216e-2	9.673
Palabos (Reference)	5.674e-8	1.252e-2	9.944

Then we compare the velocity norm at each of the chosen points. All the values for the four different grids are found in Tab 3.5. We can assess that the results of both softwares are in agreement in this case as well. The only point that needs a special explanation is P_5. We must note that it is located inside the aneurysm.

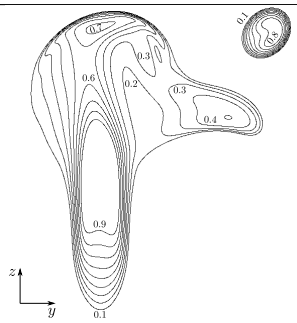
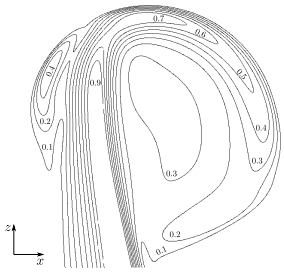
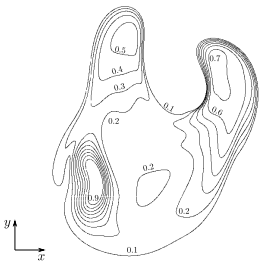
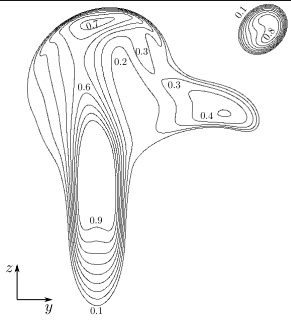
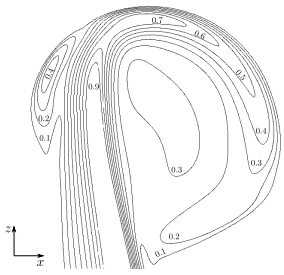
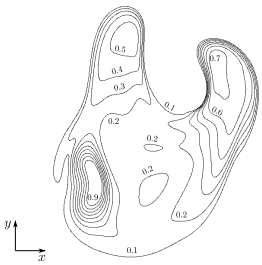
From the contour-plots, we can observe that there are velocity gradients that travel inside the aneurysm. Therefore, it is possible that the benchmark grid for Palabos does not have enough resolution for P_5. We observe that this value is improved on the reference solution in Palabos, obtaining a value closer to the reference solution in CFX (10% error).

Table 3.5: Velocity at the chosen points.

	CFX (Bench.)	**CFX (Ref.)**	**Palabos (Bench.)**	**Palabos (Ref.)**
P_1	1.108e-1	1.120e-1	1.116e-1	1.116e-1
P_2	5.836e-2	5.901e-2	5.860e-2	5.850e-2
P_3	7.096e-2	6.890e-2	6.930e-2	6.901e-2
P_4	1.829e-2	1.771e-2	1.740e-2	1.843e-2
P_5	1.908e-2	1.883e-2	1.434e-2	1.665e-2
P_6	2.796e-2	2.715e-2	3.006e-2	2.846e-2

We take several contour-plots of the velocity norm for each of the planes to see if they are the same. The images corresponding to Palabos and CFX can be found in Tab. 3.6. Each contour-plot is labelled with the velocity that corresponds to it.

Table 3.6: Comparative contour-plots over the different planes.

	Plane 1	**Plane 2**	**Plane 3**
Palabos			
CFX			

Let us discuss more about the performances of the codes. As proposed, we want to measure and compare the time needed for the whole pipeline. In the case of Palabos, all the steps are performed by the same software, including the mesh creation, which does take only some seconds of the total time. The mesh in the case of the LBM is regular, there is no additional complexity when generating it. In addition, Palabos can process the voxelisation algorithm in parallel, being able to voxelize enormous STL files in few seconds.

In opposition, CFX uses an external software to generate the grid. The process of generating a good meshing is more complex and it is also very important, as the results depend strongly on the final mesh. We have preferred to except this step from the considered pipeline for CFX.

Both codes have been executed on a parallel machine containing up to 64 Intel Xeon E5620 cores. The performances for 1, 8, 16, 32 and 64 cores are presented in Tab. 3.7. We see that CFX has a big advantage over a single core, but that over 64 cores, both softwares solve the problem in the same order of time.

Table 3.7: Computational times.

Cores	**Palabos**	**CFX**
1	8h. 13 min.	3h 32 min.
8	84 min.	48 min.
16	48 min.	31 min.
32	26 min.	19 min.
64	16 min.	12 min.

Finally, one result that is worth mentioning is the agreement of the wall–shear stress[3] (WSS) computed by the formula

$$\tau_w = \mu \left. \frac{\partial u}{\partial n} \right|_{n=0}, \tag{3.1}$$

where μ is the dynamic viscosity, u is the velocity tangential to the wall and n is the normal direction to the wall. On Fig. 3.6, we show the coloured version of WSS for the reference simulations of Palabos and CFX. We see clearly that they are in agreement in both cases. To our knowledge this kind of results for the WSS have not been presented for blood flow.

[3]One can think of the wall–shear stress as the force exerted by wall over the fluid in a direction perpendicular to the wall.

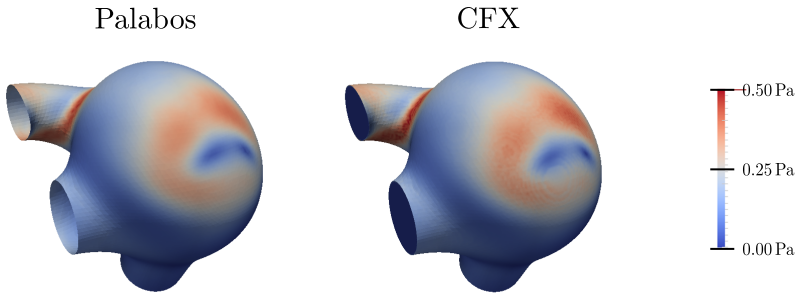


Fig. 3.6: Agreement of the wall–shear stress for both softwares.

3.3 Summary

This chapter introduces the open–source library Palabos, which is used latter for our grid–refinement implementation.

The novelty of this chapter is a benchmark of Palabos with the commercial software ANSYS CFX based on the finite volumes method. The methodology and main results are summarized in the following.

- The benchmark is performed over a synthetic geometry generated by an open–source software, in order to guarantee repeatability.
- The methodology in Palabos includes a multi–grid technique to accelerate the convergence to the steady state.
- For both softwares we have performed two simulations: a reference simulation with a large number of grid points and a benchmark simulation with a reasonable number of points, which is the one that are effectively compared.
- The results show a good agreement in the mean quantities like pressure and energy, as well as in the point–wise quantities like velocity.
- We have obtained an interesting accordance for the wall–shear stress in the reference simulations.

Chapter 4

Grid Refinement

In this chapter, we will present the key concepts of the grid refinement for the LBM. First, we give the intuitive motivation for the use of non–uniform meshes. Then, we explain all the concepts of grid refinement, from the most basic to the most advanced.

4.1 Motivation

As shown in the previous chapter, the LBM error depends on the temporal and spatial discretization steps (Eq. 2.38). In this section we intend to show the interest of grid refinement to avoid the increase of computational time that is associated with a uniform grid.

Let us illustrate this growth in a simple 2D example. In what follows we study the cost of a hypothetical simulation over three uniform grids as depicted in Fig. 4.1. In order to simplify the computations, we consider a square geometry of side $L = 1$.

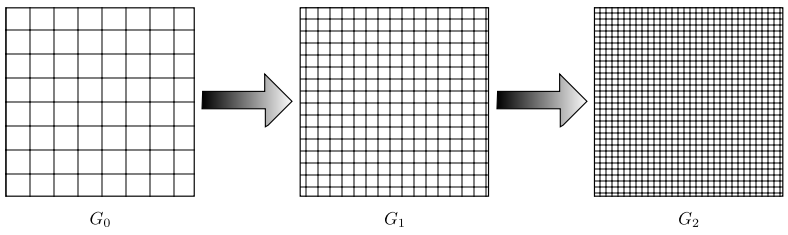


Fig. 4.1: Schema of finer grids to solve a problem.

Let us suppose that G_0 has $\delta x_0 = 1/N$. It uses N^2 points to solve the problem. The second grid G_1 is defined to be twice as finer as G_0 ($\delta x_1 = \delta x_0/2$), thus containing a total of $4N^2$ points. Finally, G_2 is four times finer than G_0, which results in $16N^2$ points. From these numbers, one concludes that using a grid twice as fine to solve a problem means four times more points in 2D (eight times in 3D).

However, there is another cost that needs to be taken into account. Because of reasons of stability (see [26]), when the spatial step is decreased, it is also necessary to decrease the temporal step. In the following, we compute the cost for one dimensionless time unit. Let us consider the convective rescaling (see Sec. 2.5), which is the cheaper one from the two possible rescalings presented. We recall that it specifies that when we divide the spatial discretization by two, we must do the same with the temporal discretization. For the incompressible NS equations, this produces a $O(1)$ error, but we expect this error to be small enough to be neglected.

We go back to our example. Let us suppose that the temporal step of G_0 is simply $\delta t = 1/T$, T and integer value. To achieve one dimensionless time unit, we need to do T iterations. In the case of G_1, which has a temporal step $\delta t/2$, to achieve one dimensionless time unit, we need $2T$ iterations. Finally, following the same logic, G_2 needs $4T$ iterations to achieve one dimensionless time unit.

In summary, the total cost for T dimensionless times for G_0 is N^2T. For G_1 it becomes $8N^2T$ and for G_2 $64N^2T$. If we suppose that we need a grid G_2 to solve our problem within an acceptable level of accuracy, our simulation is 64 times more expensive than one that uses only G_0. In 3D the cost of using G_2 over G_0 is 128 times.

Let us note at this point that for highly multi–scale turbulent flows, the number of points needed to represent the smallest scales might become intractable. Another important observation is that in many physical problems, the most complicated physics might happen only in small portions of the domain. For example, close to boundaries.

This suggests the idea, as in most traditional numerical methods, to try to avoid this global refinement by inserting local finer grids only in selected areas where the important physics happen. A schematic view is shown in figure 4.2 if we suppose that the important region of the simulation is found in the middle of the domain.

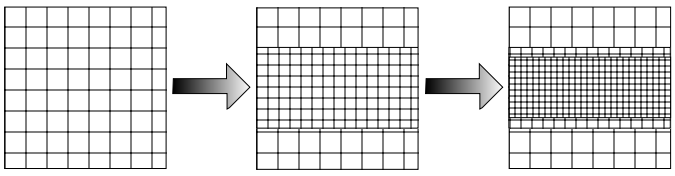


Fig. 4.2: Local refinements to increase accuracy

Let us suppose that we have one half of the original grid G_0 by inserting a grid of type G_1 and one third of the original domain with a grid of type G_2. We can

estimate the cost of a dimensionless time unit for this hybrid grid. In this case[1] has ideally a cost proportional to $39N^2T/2$, which is almost three times smaller as the cost of having G_2 over the whole domain.

This comparison shows the benefits of the non–uniform grids. Of course, it holds, as long as the refined simulation has the same precision (up to a small percentage of error) with the precision of the simulation that uses only G_2.

4.2 State of the art

In order to summarize as best as possible the literature for the grid refinement, we have chosen to present the following five most used algorithms for non–uniform grids in the LBM. In what follows, each of these algorithms is presented, followed by the articles that use it.

Finite volume lattice Boltzmann equation (FVLBE).

This algorithm discretizes the Boltzmann equation (Eq. 2.5) over finite volumes. The original idea is presented in Nannelli et *al.* [50]. Once this has been done, one can implement non–uniform meshes exactly as in the finite volume method.

The literature in this case treats specially the control elements that have been chosen to project the Boltzmann equation in order to preserve to be consistent with the CE expansion. We can cite Xi et *al.* [78], where the authors use quadrilateral control volumes for 2D simulations. Xi et *al.* [79] extend the model to trilinear hexahedral elements in the case of a 3D simulation. In Ubertini et *al.* [73], further studies about the method are presented like the usage of triangular control volumes and boundary condition for this algorithm.

Interpolation–supplemented lattice Boltzmann equation (ISLBE).

This method from He et *al.* [31] proposes to define a non–uniform computational grid that is unrelated to the uniform lattice where the LBM is solved. An algorithmic step after the collide and the stream operations is added, where the populations on the computational grid that do not exist in the uniform grid are interpolated from their neighbours in the uniform lattice. He [28] shows that the order of the interpolation supplemented lattice Boltzmann method is highly dependent on the interpolation scheme used.

Lu et *al.* [44] simulate a turbulent fluid inside a stirred tank. They propose to modify the order of the operations, so that the interpolation is performed avant the propagation step. Dixit et *al.* [17] simulate a thermal model using the ISLBE

[1] We are neglecting the cost of the communications intra grids to simplify the discussion.

for high Rayleigh numbers[2]. Prenmath et *al.* [54] use this model for the simulation turbulent jets coupled with a sub–grid model. We note that all these articles use a second order interpolation.

Rescaling of f^{neq} post–collision (FH).

This method was presented for the first time in Filippova et *al.* [20]. We apply the collision step to the particle distribution functions f_i obtaining f_i^{out}. In order to ensure the continuity of the velocity and the pressure between grids of different levels, a convective rescaling is chosen (see Sec. 2.5). The continuity of these quantities also implies that the f^{eq} part is the same in the coarse and fine grids. The authors find that a constant relates the post–collision non–equilibrium part f^{neq} of a coarse grid and a fine grid. The relation for this quantity, considering a coarse grid with lattice size δx_c and a fine one with lattice size δx_f, is given by

$$f_{i,c}^{neq,\text{out}} = \frac{\delta x_c(\tau_c - 1)}{\delta x_f(\tau_f - 1)} f_{i,f}^{neq,\text{out}} \tag{4.1}$$

where subindex c and f are used for coarse and fine grid respectively.

The authors of the algorithm have used it to solve turbulent flows [22]. Another application for small Mach numbers compared to the commercial solver FLUENT [21] is also performed by the authors. The paper of Yu et *al.* [80] presents clearly the ideas of the FH and implement a code to solve the cavity 2D and the flow past a cylinder. It is interesting to note that they use a cubic symmetric spline function to interpolate the unknown values.

Let us note that all the following articles use the paper [80] as underlying algorithm for the grid refinement. Crouse et *al.* [14] propose the usage of quadtrees type refinements in 2D to cover the domain in a recursive fashion. This paper contains an adaptative flow simulation in two–dimensions. The quadtree data–structure is converted in octrees for three dimensional simulations. An application of these is presented in Geller et *al.* [24], where a lattice Boltzmann solver with grid refinement is compared to the finite elements and the finite volumes methods. In Toelke et *al.* [69] the authors achieve a multi–phase flow simulation with adaptative grids. More recently, Geller et *al.* [25] have performed a turbulent jet simulation using the cascaded lattice Boltzmann model, which is a new kind of multiple time relaxation operator (see [23]), which allows a direct numerical simulation of the problem. The solution is compared to a sub-grid model for the turbulence.

[2]The Rayleigh number is a dimensionless quantity that expresses the ratio of conduction and convection for heat transfer

Rescaling of f^{neq} pre–collision (DC).

The idea of Dupuis et *al.* [18] is to apply the rescaling of the non–equilibrium part to the pre–collision populations. For more details, this method is explained in the next section. Let us note that this is the method that we have chosen for our generic implementation.

For this family of algorithms, we can cite the treatment of the reaction–diffusion problem with the lattice Boltzmann (see [3]) with grid refinement in order to improve the performances by Alemani et *al.* [4]. Alemani et *al.* [2] also study the influence of the different possible rescalings of time and space between the grids of different level affect the solution to 1D reaction–diffusion.

Volumetric grid refinement (VGR).

This method is based on the volumetric description of the lattice Boltzmann method by Chen [9]. Using this formulation, Chen et *al.* [10] propose a method that relies in the independence of the volumetric quantities from the resolution of the grids. The grid refinement uses two special operations on coarse and fine cites that do not have neighbours of the resolution. The Coalesce procedure sums the particle populations on the fine nodes to obtain the particle populations for the corresponding coarse nodes. In opposition, the Explode step redistributes the populations on the coarse node to the surrounding fine nodes. The importance of this method relies on the fact that it is implemented in the commercial software PowerFlow [19].

The paper by Rohde et *al.* [61] proposes a modified version of the original algorithm with uniform redistribution of particles in order to complete the missing informations in the places where coarse and fine grids co–exist. We can also cite Premnath et *al.* [55], which use the model to simulate the turbulent flow in a channel with an underlying sub-grid model.

For the sake of clarity, all the aforementioned articles are classified in the following table according to the algorithm used. We specify each time the algorithm used and the methods to increase and decrease the information between grids of different resolution.

Algorithm	Publication	Increase information	Decrease information
FVLBME	Xi et *al.* [78]	-	-
	Xi et *al.* [79]	-	-
	Ubertini et *al.* [73]	-	-
	Ubertini et *al.* [74]	-	-
ISLBE	Lu et *al.* [44]	Asymmetric second order interpolation	-
	Dixit et *al.* [17]	Asymmetric second order interpolation	-
	Prenmath et *al.* [54]	Asymmetric second order interpolation	-
	Chen et *al.* [11]	Linear interpolation	-
FH	Filippova et *al.* [22]	Second order	-
	Filippova et *al.* [21]	Second order	-
	Yu et *al.* [80]	Cubic symmetric spline	-
DC	Alemani et *al.* [4]	Not specified	-
	Alemani et *al.* [2]	Not needed (1D problem)	-
VGR	Rohde et *al.* [61]	Recombination	Uniform distribution
	Prenmath et *al.* [55]	Explode	Coalesce

Some comments on the table are necessary. Among the methods that need an interpolation, we can see that the only reference where a cubic interpolation is proposed is Yu et *al.* [80]. Let us note however, that they are using a cubic spline which uses further information over the points, such as the derivatives of the function.

The only methods which propose a decrease in the information that is passed from a fine grid to a coarse grid are the volumetric ones, because it is a step included in the algorithm. We show latter in this thesis, that a special decrease operation is necessary for high Reynolds numbers flows.

We end this review by noting that grid refinement has been lately implemented in GPUs. See for instance the work by Schoenherr et *al.* [64].

4.3 Basics of grid refinement

In this section we go through the basic concepts of grid refinement. To keep the discussion simple the approach is explained over a two dimensional example, but can be straightforwardly generalized in three dimensions. We start with geometrical considerations. Then we explain the rescaling of units between the grids. The rescaling of the populations f_i comes afterwards. We end this section by briefly describing the couplings between the grids that are needed. These couplings are then treated in more detail in the next section.

4.3.1 Geometrical considerations

The following classification is closely related the one presented in reference [63]. There exists two grid refinement techniques, the multi-grid and multi-domain. In the first case, the coarse grid is present all over the simulation domain, even in the places where there exist refined patches (see Fig 4.3). On the other hand, when defining a multi-domain refinement, the regions where refined patches are inserted are taken off the coarse grid (see Fig. 4.4).

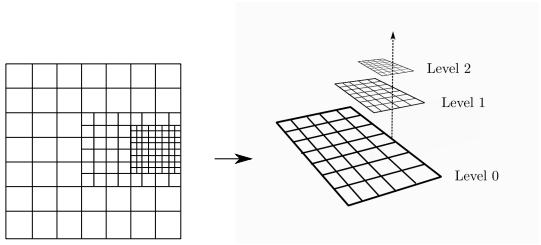


Fig. 4.3: A schematic view the multi-grid approach.

Each of these techniques has its advantages and disadvantages. For instance the multi–domain approach has better CPU performances and higher memory savings. However the actual implementation and the coupling between grids for this approach may be more complex than for the multi–grid as will be explained later.

The geometry management is of course easier in the multi–grid case, where we have the coarse grids that are present over the whole domain. On the opposite, there is need to realize geometrical operations to generate the domains in the case of the multi–domain geometry. For instance, think of the necessity to exclude a refined patch to be inserted and only keep the original domain excluding the patch.

The communication of information for the multi–domain must be done through the common boundaries of the different domains, as there is not another possibility.

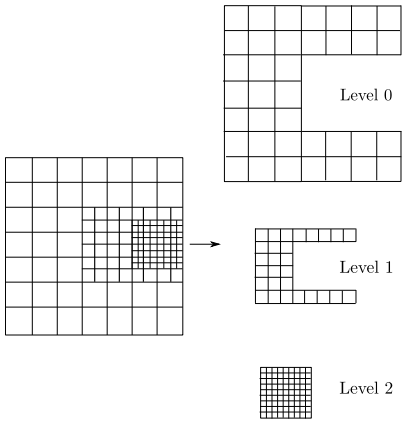


Fig. 4.4: A sketch of the multi-domain approach.

In the multi–grid case, it is possible to choose a between a one-way coupling or a two-way coupling. In the one-way case the information only travels from coarser grids to finer grids but there is no feed–back from the fine to the coarse grids. Although this is simpler from an algorithmic point of view, the coarse grid might send insufficient information to the fine grid for the system to correctly develop and solve the smaller scales represented by the finer grid. In the other case, when having a complete feed–back between the grids, this problem might be avoided. Let us note that the communication schema becomes the same as in the multi–domain approach.

Finally, the division in parallel regions might easier for the multi–grid than for the multi–domain, as the domains are more regular, thus facilitating a distribution. In any case, the load balancing for non–uniform meshes is a very difficult problem.

4.3.2 Unit conversion

Each resolution level possesses its own "lattice units". This change of scales induces a need for a rescaling of the physical quantities between the grids. In the following, we work in lattice units, the c subscript stands for coarse grid units, while f for fine grid units.

In this thesis we chose to refine the grids by a factor two only. If we define δx_c and δx_f to be the spatial discretization of the coarse and fine grids respectively, then they are related by

$$\delta x_f = \delta x_c/2. \tag{4.2}$$

Once the spatial refinement has been chosen, one still has the freedom to chose the temporal refinement. However, as shown in the error equation, one must either

choose between the convective and diffusive scaling. It is the first one that is used in general in grid refinement algorithms.

The convective scaling has the consequence that the ratio of the spatial and temporal discretizations is a constant

$$\delta t_f/\delta x_f = \delta t_c/\delta x_c = \text{const.} \tag{4.3}$$

Therefore the temporal loop in the fine grid must do twice more iterations than the coarse one.

An important consequence of the convective scaling, is that the velocity and the pressure (and also the density, see Eq. (2.26)) in lattice units are continuous fields on the grid transition while the viscosity must be rescaled as discussed now. Let us recall the definition of the Reynolds number related to the coarse or fine grid as

$$\text{Re}_n = U_n L_n/\nu_n, \tag{4.4}$$

where n stands for c or f, U_n, L_n and ν_n being respectively the characteristic velocity, the characteristic length-scale and the viscosity of the n grid. Writing the U_c, L_c, U_f and L_f in terms of the physical characteristic velocity and length-scale, U and L, one has

$$U_n = U\delta t_n/\delta x_n, \quad L_n = L/\delta x_n, \tag{4.5}$$

and imposing that the Reynolds number is independent of the grid one gets

$$\text{Re}_c = \text{Re}_f \Leftrightarrow \frac{UL\delta t_c}{\delta x_c^2 \nu_c} = \frac{UL\delta t_f}{\delta x_f^2 \nu_f}. \tag{4.6}$$

Finally remembering Eq. (4.3) one finds for the rescaling of the viscosity

$$\nu_f = \frac{\delta x_c}{\delta x_f}\nu_c = 2\nu_c. \tag{4.7}$$

As a consequence, by using the relation between the relaxation time and the viscosity of Eq. (2.27), one finds that τ_f is related to τ_c by

$$\tau_f = \frac{2\tau_c\delta x_c + \delta x_f + \delta x_c}{2\delta x_f}, \tag{4.8}$$

$$\tau_f = \frac{4\tau_c + 3}{2}. \tag{4.9}$$

4.3.3 Algorithm for the rescaling of populations

In this section we present in detail the algorithm proposed by Dupuis et *al.* [18], as it is the one we use for our generic implementation in Chap. 5

It must be noted that this algorithm is the same as the algorithm of Filippova et *al.* [20], with the exception that the rescaling is applied before the collision. This fact avoids the restriction one clearly sees in Eq. 4.1 over the value of the relaxation time on both coarse and fine grids, that the FH algorithm imposes.

The algorithm proposes a rescaling of the distribution function f_i when passing between grids of different refinement level. We consider that the convective scaling is used.

The basic ideas of the algorithm are explained in the following. As noted in Eq. (2.33) each $f_{i,n}$ can be written as

$$\begin{aligned} f_{i,n} &= f_i^{eq}(\rho_n, \boldsymbol{u}_n) + f_{i,n}^{neq}(\boldsymbol{\nabla u}), \\ f_{i,n} &= f_i^{eq}(\rho, \boldsymbol{u}) + f_{i,n}^{neq}(\boldsymbol{\nabla u}), \end{aligned} \tag{4.10}$$

where n stands again for c or f. In the second line we used that $\rho_f = \rho_c = \rho$ and $\boldsymbol{u}_f = \boldsymbol{u}_c = \boldsymbol{u}$ are expressed in the same units independently if computed from $f_{i,f}$ or $f_{i,c}$. Since f_i^{eq} only depends on ρ and $\boldsymbol{u}$ (see Eq. 2.23) it is continuous between the grids and does not need any rescaling.

On the other hand, the non-equilibrium part $f_i^{neq} = f_i - f_i^{eq}$ is proportional to the gradient of the velocity, it is therefore necessary to rescale it when communicating it between grids with different resolution. To determine this scaling, let us note by $f_{i,c}^{neq}$ the non-equilibrium part of the coarse grid and $f_{i,f}^{neq}$ the one from the fine grid. The continuity of the f_i^{neq} quantities read

$$f_{i,f}^{neq} = \alpha f_{i,c}^{neq}, \tag{4.11}$$

where α is the factor to be determined to impose the continuity of the non-equilibrium distribution functions. At the leading order one has $f_i^{neq} \cong \epsilon f_i^{(1)}$ and using Eq. (2.30) and (2.31) in the previous equation one has

$$\begin{aligned} \tau_f \boldsymbol{Q}_i : \boldsymbol{S}_f &= \alpha \tau_c \boldsymbol{Q}_i : \boldsymbol{S}_c \\ \frac{\tau_f}{\delta t_f} \boldsymbol{Q}_i : \boldsymbol{S} &= \alpha \frac{\tau_c}{\delta t_c} \boldsymbol{Q}_i : \boldsymbol{S} \\ \alpha &= \frac{\delta t_c}{\delta t_f} \frac{\tau_f}{\tau_c}, \end{aligned} \tag{4.12}$$

where $\boldsymbol{S}$ is the strain rate tensor in physical units, while $\boldsymbol{S}_c$ and $\boldsymbol{S}_f$ are the same tensor, but in coarse and fine lattice units respectively. Finally we find the following relation

$$f_{i,c}^{neq} = \frac{\delta t_f \tau_c}{\delta t_c \tau_f} f_{i,f}^{neq} = \frac{2\tau_c}{\tau_f} f_{i,f}^{neq}, \tag{4.13}$$

where we used that $\delta t_f = \delta t_c/2$.

Finally, in order to reconstruct the fine and coarse populations from their corresponding partners, we can use the following equations

$$f_{i,f} = f_i^{eq} + \frac{\tau_f}{2\tau_c} f_{i,c}^{neq} \tag{4.14}$$

and

$$f_{i,c} = f_i^{eq} + \frac{2\tau_c}{\tau_f} f_{i,f}^{neq}. \tag{4.15}$$

This rescaling between the grids allows for a continuous transition of the physical quantities at the grids interface.

4.3.4 Communication between grids

When using multi-resolution approaches, a communication between the grids is needed. In the case of multi-domain methods the communication is done on the boundaries connecting the grids. The coupling is made in two directions: from coarse to fine and from fine to coarse grids. In this subsection we intend to pose some definitions that are going to be used hereafter.

On the boundaries of each refinement level, after a "collide-and-stream" operation (see Sec. 2.4) there will be some missing information (some populations f_i are unknown on the coarse and on the fine grids) that one needs to reconstruct. The details on this algorithmic part are given latter in the Sec. 4.4.

For the sake of clarity, let us call C the ensemble of coarse sites and F the ensemble of all fine sites. Let us now define $\boldsymbol{x}_{f\to c}$ the fine sites that are contained in F and C where the coupling from fine to coarse is performed and $\boldsymbol{x}_{c\to f}$ all the sites contained in F and C where the coupling goes from coarse to fine (see Fig. 4.5). Let us also define:

- $\boldsymbol{x}^c_{f\to c} = \{\boldsymbol{x} | \boldsymbol{x} \in \boldsymbol{x}_{f\to c}$ and $\boldsymbol{x} \notin F\}$, the set of coarse sites where we perform the copy from fine to coarse.
- $\boldsymbol{x}^c_{c\to f} = \{\boldsymbol{x} | \boldsymbol{x} \in \boldsymbol{x}_{c\to f}$ and $\boldsymbol{x} \in C\}$, the set that contains only the coarse sites where we perform the copy from coarse to fine.
- $\boldsymbol{x}^f_{c\to f} = \{\boldsymbol{x} | \boldsymbol{x} \in \boldsymbol{x}_{c\to f}$ and $\boldsymbol{x} \notin \boldsymbol{x}^c_{c\to f}\}$, the set of fine sites where we perform the copy from coarse to fine, which do not have a corresponding coarse site.

The coupling proposed in this thesis requires the grids to overlap themselves by a domain of at least one coarse cell width, as shown in the one-dimensional example in Fig. 4.6. Let us now analyse what is happening on Fig. 4.6. After a coarse grid collide-and-stream operation, all the sites have the necessary information, except

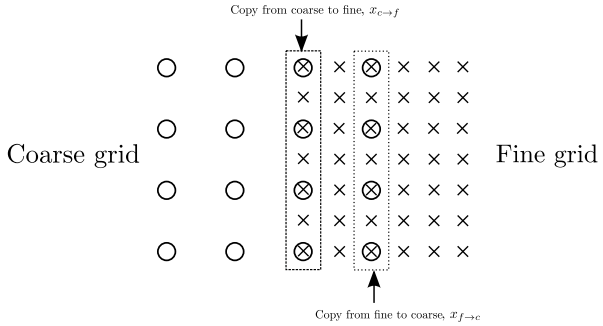


Fig. 4.5: Complete sketch of the places where the copies are performed.

for the last site of the grid (labelled "unknown value on Fig. 4.6), where the are missing populations. It is therefore impossible to perform the coupling on this coarse site to the fine grid site. However, all the other sites have all the needed information, thus becoming good candidates to provide the information to the fine grid. If we apply the same reasoning to the fine grid, it becomes clear that it is necessary to implement a redundant area between the grids, so that the copies are performed over complete sites. The complete two dimensional example is found in Fig. 4.5

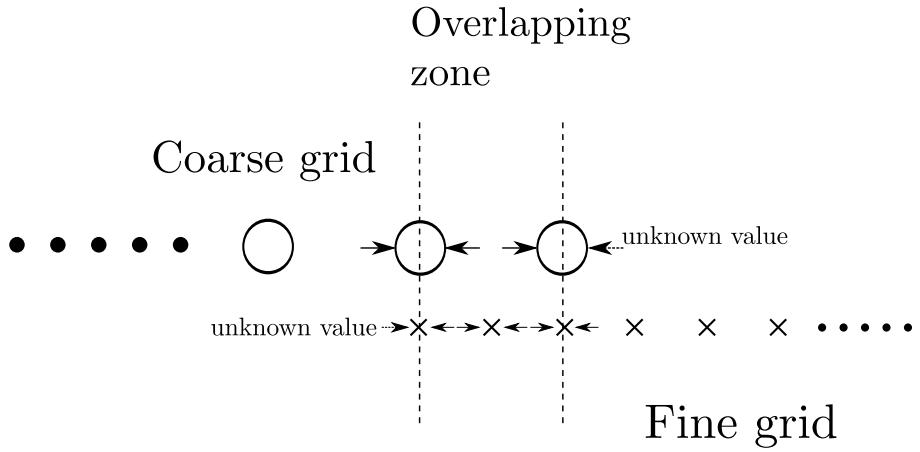


Fig. 4.6: Overlapping zone between coarse and fine grids

We have chosen to make this overlapping area as small as possible, namely one coarse site, or equivalently, two fine sites. However, when dealing with turbulent flows, it might be desirable to have a more important area between the grids, in order to allow a gentler transition of the information.

We note that in the final chapter, we propose a new refinement algorithm which

does not need of a buffering or overlapping zone between the grids.

4.4 The couplings in detail

The two-way coupling between a coarse and a fine grid involves two fundamental operations that are the decimation and the interpolation steps. When going from the fine to the coarse grid the amount of information represented in the fine grid must be reduced, while it must be "increased" when going from the coarse to the fine grid.

In this section we treat both of these operations thoroughly. We start by studying the decimation operation, which is intended to reduce the information from the fine to the coarse grid in a consistent way. Then, we explain more about the interpolation operation used to increase the information passing from the coarse to the fine grid.

4.4.1 Coupling from fine to coarse grid

As the fine grid has more sites than the coarse one (see Fig. 4.5), it is necessary to decrease the information to be transferred. The necessary steps are: restrict the values, rescale them and copy them to the coarse grid. The restriction operation can be a simple copy from the corresponding site, or something more complicated, as a low-pass filter. This latter can be justified by the fact that the fine grid contains information about scales that cannot be resolved by the coarse grid and thus must be eliminated.

The proposed coupling is over the sites marked as $\boldsymbol{x}_{f\to c}$ is expressed by the following equation

$$f_{i,c}(\boldsymbol{x}^c_{f\to c}, t) = f_i^{eq}\left(\rho_f(\boldsymbol{x}^c_{f\to c}, t), \boldsymbol{u}_f(\boldsymbol{x}^c_{f\to c}, t)\right) + \frac{2\tau_c}{\tau_f}\overline{f}^{neq}_{i,f}(\boldsymbol{x}^c_{f\to c}, t). \tag{4.16}$$

where $\rho_f = \sum_i f_{i,f}$ and $\boldsymbol{u}_f = \sum_i \boldsymbol{\xi}_i f_{i,f}$, and $\overline{f}^{neq}_{i,f}(\boldsymbol{x}^c_{f\to c}, t)$ is the result of applying the restriction to the incoming fine grid values. It must be noted that ρ and $\boldsymbol{u}$ are calculated over the fine grid and by continuity of this fields due to the convective rescaling, no other operation over them needs to be performed.

If we perform a simple decimation, then it is clear that the restriction operation is given by

$$\overline{f}^{neq}_{i,f}(\boldsymbol{x}^c_{f\to c}, t) = f^{neq}_{i,f}(\boldsymbol{x}^c_{f\to c}, t) \tag{4.17}$$

However, as we will show later, this is not enough. We propose a filtering operation that is inspired by [56]. The aim of this operation is to remove the

scales of the fine grid that cannot be resolved and treated by the coarse grid. We use a filter width corresponding to the coarse grid resolution. A simple box filter is used here but more complex filters can also be applied.

There exists many ways of applying a filter in the lattice Boltzmann framework. One can filter the complete distribution function or only ρ and $\boldsymbol{u}$ (see [60]). In our case, we chose to apply the filter only on the non-equilibrium part of the populations f_i^{neq}, because when the filtering was done on f_i, or only on ρ and $\boldsymbol{u}$ there was a too strong dissipation added by the filter and therefore we noticed a decrease of the accuracy. In practice one simply does an averaging over all the q lattice directions, thus obtaining the following restriction

$$\overline{f}_{i,f}^{neq}(\boldsymbol{x}_{f\to c}^{c}, t) = \frac{1}{q}\sum_{i=0}^{q-1} f_{i,f}^{neq}(\boldsymbol{x}_{f\to c}^{c} + \boldsymbol{\xi}_i, t) \tag{4.18}$$

As a final remark, we have chosen not to filter ρ and $\boldsymbol{u}$. The main reason for this being that filtering these quantities results in an artificial increase of viscosity around the refinement interface. This can, of course, have undesired results such as loss of accuracy and modification of the expected behaviour of the system.

4.4.2 Coupling from coarse to fine grid

As shown in Fig. 4.5, the fine grid possesses sites that do not have a corresponding site in the coarse one. Thus when performing the copy from the coarse to the fine grid, it is necessary to estimate the missing information in such sites. We have chosen to apply an interpolation in order to complete the missing informations. The details of the interpolation are explained further in this document, as we have found that it is a vital part of the algorithm.

The coupling over the sites $\boldsymbol{x}_{c\to f}$ is given by two different operations. If a point has a corresponding coarse site in $\boldsymbol{x}_{c\to f}$ (i.e. if a computational node has both a coarse and a fine site, or in a mathematical notation if $\boldsymbol{x}_f \in \boldsymbol{x}_{c\to f}^{c}$) then

$$f_{i,f}(\boldsymbol{x}_{c\to f}^{c}) = f_i^{eq}(\rho_c(\boldsymbol{x}_{c\to f}^{c}), \boldsymbol{u}_c(\boldsymbol{x}_{c\to f}^{c})) + \frac{\tau_f}{2\tau_c} f_{i,c}^{neq}(\boldsymbol{x}_{c\to f}^{c}) \tag{4.19}$$

where $\rho_c = \sum_i f_{i,c}$, $\boldsymbol{u}_c = \sum_i \boldsymbol{\xi}_i f_{i,c}$, and $f_{i,c}^{neq}$ are computed from the populations of the coarse grid.

However, if the fine site does not correspond to a coarse site in $\boldsymbol{x}_{c\to f}$ (i.e. the computational node contains only a site of the fine domain), we will use the following formula

$$f_{i,f}(\boldsymbol{x}_{c\to f}^{f}) = f_i^{eq}(\overline{\rho}_c, \overline{\boldsymbol{u}}_c) + \frac{\tau_f}{2\tau_c}\overline{f_{i,c}^{neq}} \tag{4.20}$$

where $\overline{\rho}_c$, $\overline{\boldsymbol{u}}_c$ and $\overline{f_{i,c}^{neq}}$ are interpolated from the values where the fine and coarse sites are coincident.

The fine grid can resolve more scales than the coarse one, it might be necessary to try recreate these smaller scales when transferring the information from the coarse to the fine grid. In order to solve this issue, we implemented the approximate deconvolution approach proposed in [56]. However in the benchmarks we performed we did not notice a significant gain when using this method and therefore for the sake of clarity we will not present it.

4.4.3 The complete algorithm

Here we present the complete algorithm proposed for the grid refinement.

Let us suppose that the system is at time t. Both grids are complete, i.e. all information needed on every site is given. A complete time iteration, for the convective scheme explained above, consists of one iteration of the coarse grid and two iterations of the fine grid. The details of these iterations are given now.

1. A "collide-and-stream" operation is performed on the coarse grid bringing it to time $t + \delta t_c$. At this point the populations at $\boldsymbol{x}_{f\to c}$ that were supposed to be streamed from the fine grid are unknown.

2. A "collide-and-stream" cycle is performed on the fine grid bringing it at time $t + \delta t_c/2$. The grid lacks information on the grid refinement boundary sites $\boldsymbol{x}_{c\to f}$. One then performs a double interpolation, one in time and one in space. First the values of ρ_c, $\boldsymbol{u}_c$ and $f_{i,c}^{neq}$ of the coarse sites at $\boldsymbol{x}_{c\to f}$ and are interpolated at time $t + \delta t_c/2$, by a linear scheme (which is of second order accuracy at $\delta t_c/2$). Then the values $\rho_c(t+\delta t/2)$, $\boldsymbol{u}_c(t+\delta t/2)$ and $f_{i,c}^{neq}(t+\delta t/2)$ are interpolated in space according to the discussion of Subsec. 4.4.1 by using a local cubic scheme. All the populations at $\boldsymbol{x}_{c\to f}$ (and not only the missing ones) are reconstructed following Eqs. (4.19) and (4.20). At this point all the fine sites are complete.

3. A second "collide-and-stream" operation is performed on the fine grid, bringing it to time $t + \delta t_c$. At this point we have the information from the coarse grid to complete the fine grid at $\boldsymbol{x}_{c\to f}$, and therefore no time interpolation is necessary. However, a space interpolation must be performed for $\rho_c(t + \delta t_c)$, $\boldsymbol{u}_c(t + \delta t_c)$, and $f_{i,c}^{neq}(t + \delta t_c)$ as in the previous step. Then the populations of the fine grid are all (and not only the missing ones) replaced at position $\boldsymbol{x}_{c\to f}$ according to Eqs. (4.19) and (4.20) again.

4. All the populations of the coarse grid at $\boldsymbol{x}_{f\to c}$ are replaced following Eqs. (4.16)

and (4.18)

$$f_{i,c}(\boldsymbol{x}_{f\to c},t) = f_i^{eq}\left(\rho_f(\boldsymbol{x}_{f\to c},t),\boldsymbol{u}_f(\boldsymbol{x}_{f\to c},t)\right) + \frac{2\tau_c}{\tau_f}\frac{1}{q}\sum_i \overline{f}_{i,f}^{neq}(\boldsymbol{x}_{f\to c}+\boldsymbol{\xi}_i,t). \tag{4.21}$$

At this point both grids are presently at time $t+\delta t_c$, are complete and ready for a new iteration.

4.4.4 Interpolation in detail

Here we discuss the proposed scheme of interpolation needed to reconstruct the information from the coarse to the fine grid. The interested reader can find a different approach in [70].

As pointed out in the preceding subsection, the values of ρ_c, $\boldsymbol{u}_c$ and $f_{i,c}^{neq}$ on nodes that do not contain both fine and coarse grids sites must be interpolated, and are noted by $\overline{\rho}_c$, $\overline{\boldsymbol{u}}_c$ and $\overline{f^{neq}}_{i,c}$.

4.4.4.1 2D schema

In 2D, this interpolation is performed over a line, thus over sites parallel to the refinement interface. The problem becomes then a 1D interpolation of a function g which is known at the coarse sites. In the following, we show our interpolation schemes for g.

The two easiest symmetric ways of computing the interpolation are using two or four neighbours as shown on Fig. 4.7. If we use two points, it is sufficient to

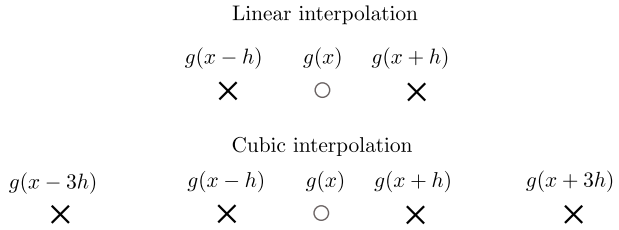


Fig. 4.7: Possible interpolation schemes. Order one (top) and order three (bottom).

perform a mean of them:

$$g(x) = \frac{g(x+h)+g(x-h)}{2} \tag{4.22}$$

As the point where we interpolate is exactly in the middle of the two others, an analysis of the order of interpolation shows us that this is in fact a second order method in h.

On the other hand, by using four points, we find the following equation:

$$g(x) = \frac{9}{16}(g(x+h) + g(x-h)) - \frac{1}{16}(g(x+3h) + g(x-3h)) \tag{4.23}$$

By using Taylor expansions for each term, this interpolation can be proved to be of fourth order.

The only drawback for this interpolation is the cost in parallel execution. Every point must ensure access to two neighbours, this means that when the code is executed in parallel, more communications have to be done between the distributed blocks.

In addition, as the first and last site do not have as many neighbours as needed, we need to use a non-centred third order scheme as depicted is Fig. 4.8 and given by the following formula

$$g(x) = \frac{3}{8}g(x-h) + \frac{3}{4}g(x+h) - \frac{1}{8}g(x+3h). \tag{4.24}$$

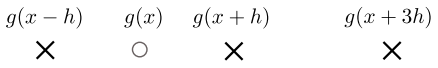


Fig. 4.8: Asymmetric second order interpolation for sites that do not have enough neighbours.

Finally, let us note that there are no other interpolation cases in our two dimensional implementation, because the outermost fine grid points always have corresponding coarse sites. In particular, cases like the one depicted in Fig. 4.9 are forbidden.

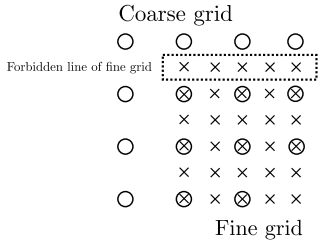


Fig. 4.9: Forbidden case to avoid complicated interpolations.

4.4.4.2 3D schema

In the 3D case, the contact between two grids is made over a plane. The interpolation problem becomes then a 2D interpolation or the interpolation of a function $g(x, y)$.

Based on what we have learned from the 2D approach, for the bulk interpolation, we have decided to use an extended neighbourhood counting 16 points to perform a higher-order interpolation. In order to explain our approach let us suppose that we know a function $g(x, y)$ at the points that are shown in Fig. 4.10. According to these coordinates system, we are interested in finding the values at the points $g(0, 0)$, $g(h, 0)$, $g(0, h)$, $g(-h, 0)$ and $g(0, -h)$.

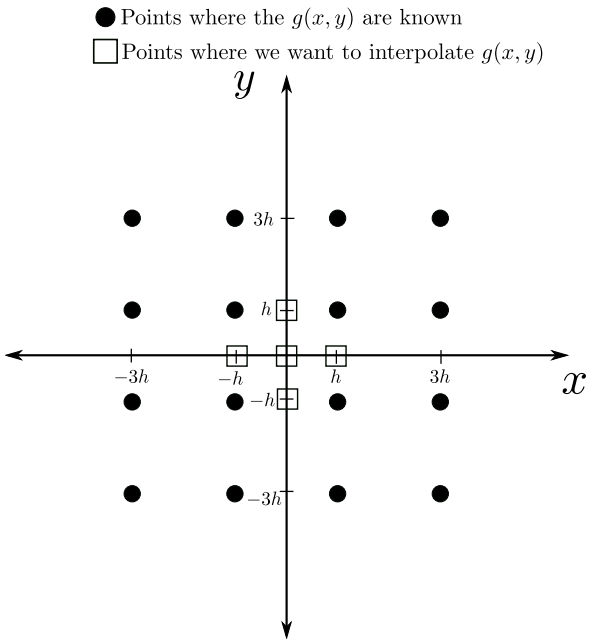


Fig. 4.10: Points where we want to interpolate the function $g(x, y)$.

One standard manner to perform the interpolation is to take the polynomial of order three in x and y:

$$P(x, y) = \sum_{i=0}^{3} \sum_{j=0}^{3} a_{i,j} x^i y^j \tag{4.25}$$

replace the known values, obtaining a system of 16 equations, which allows us to compute all the unknowns $a_{i,j}$. When this is done, all we have to do is replace the desired points in the polynomial to obtain the interpolated values. By construction, the method is of order three in both variables.

In order to speed things up, we have discovered that this interpolation (for all the points excepting $g(0,0)$ is equivalent to the 1D cubic interpolation presented above, using the four neighbouring points in the parallel direction. This gives us the following formulas:

$$g(h,0) = \frac{9}{16}(g(h,h) + g(h,-h)) - \frac{1}{16}(g(h,3h) + g(h,-3h)) \tag{4.26}$$

$$g(-h,0) = \frac{9}{16}(g(-h,h) + g(-h,-h)) - \frac{1}{16}(g(-h,3h) + g(-h,-3h)) \tag{4.27}$$

$$g(0,h) = \frac{9}{16}(g(h,h) + g(-h,h)) - \frac{1}{16}(g(3h,h) + g(3h,h)) \tag{4.28}$$

$$g(0,-h) = \frac{9}{16}(g(-h,-h) + g(h,-h)) - \frac{1}{16}(g(3h,-h) + g(-3h,-h)) \tag{4.29}$$

$$\tag{4.30}$$

We can also find another explicit formula for $g(0,0)$ which this time depends on the 16 points. It is given by:

$$\begin{aligned} g(0,0) = \frac{81}{256}&(g(h,h) + g(h,-h) + g(-h,h) + g(-h,-h)) \\ &- \frac{9}{256}(g(-h,3h) + g(-h,-3h) + g(-3h,h) + g(-3h,-h)) \\ &- \frac{9}{256}(g(h,3h) + g(h,-3h) + g(3h,-h) + g(3h,h)) \\ &+ \frac{1}{256}(g(-3h,3h) + g(-3h,-3h) + g(3h,-3h) + g(3h,3h)) \end{aligned} \tag{4.31}$$

Of course, the borders need a special treatment. In this case, we use the asymmetric schema depicted in Fig. 4.11. In this case, the points perpendicular to the wall use an asymmetric interpolation schema, using the 1D asymmetric formulas proposed above. In opposition, the site which is parallel to the wall still uses four neighbours for a third–order 1D interpolation. The only value computed differently again is $g(0,0)$, for which we use the formula:

$$\begin{aligned} g(0,0) = \frac{27}{128}&(g(-h,-h) + g(h,-h)) + \frac{27}{64}(g(-h,h) + g(h,h)) \\ &- \frac{3}{128}(g(-3h,-h) + g(3h,-h)) - \frac{3}{64}(g(-3h,h) + g(3h,h)) \\ &+ \frac{1}{128}(g(-3h,3h) + g(3h,3h)) \end{aligned} \tag{4.32}$$

Notice that the schema is the same for all the borders of the plane, only a rotation is needed to have all the four cases.

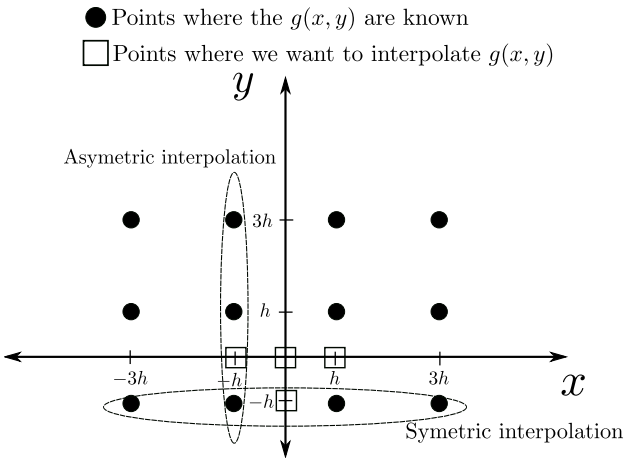


Fig. 4.11: Asymmetric interpolation for the borders.

Finally, the corners are restricted to interpolations using 9 neighbours. The case is presented in Fig. 4.12. In this case, the formulas again are simply given by asymmetric 1D interpolations for everyone. Once again, the only different schema is for the origin $g(0, 0)$ that is computed via:

$$g(0,0) = \frac{9}{32}(g(-h,h) + g(h,-h)) + \frac{9}{64}g(-h,-h) + \frac{9}{16}g(h,h) + \frac{1}{64}g(3h,3h) - \frac{3}{64}(g(3h,-h) + g(-h,3h)) - \frac{3}{32}(g(3h,h) + g(h,3h)) \tag{4.33}$$

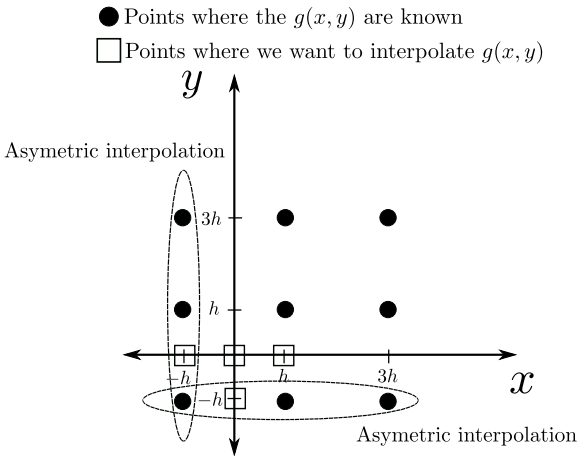


Fig. 4.12: Asymmetric interpolation for a corner.

Once again for parallelism purposes, it is necessary for every site to have two neighbours to be able to perform its interpolations, thus increasing the cost of the parallel time execution.

4.5 Is second order enough?

Interpolation is a key part of grid refinement. In this section, we show that even for a simple Poiseuille flow, the second order interpolation does not conserve the mass.

We fix a flow at $Re = u_{\max}N/\nu = 100$, with $u_{\max} = 0.01$ the maximum value of the velocity in lattice units, N the width of the channel in coarse lattice units and ν the kinematic viscosity. The length of the channel is $4N$ (see Fig. 4.13). We set the reference length to have $N = 30$ lattice sites. We prescribe the analytical solution of the velocity on both the inlet and the outlet and the horizontal walls have a zero velocity boundary condition. This set-up is depicted in Fig. 4.13. The channel is

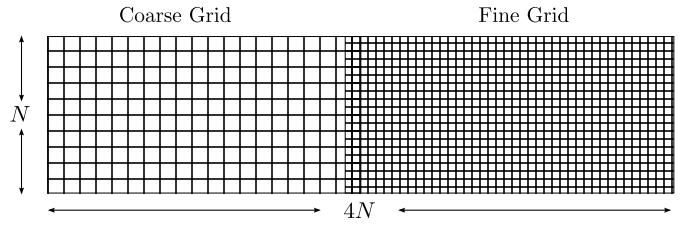


Fig. 4.13: Set-up for the channel flow

divided in two equal parts. The first one is covered by a coarse grid and the second part is covered by a grid two times finer. A linear pressure drop is expected in the direction of the flow. As can be seen in Fig. 4.14 a clear discontinuity of the pressure appears on the refinement interface in the case of second order interpolation, while it remains completely smooth in the cubic interpolation case.

Even though the pressure slope is correct, the mass loss is clearly visible in the pressure jump right where the interface is located which leads, in more complicated cases, to numerical instabilities and to the introduction of spurious error terms. In Fig. 4.15 we plotted the value of this pressure jump with respect to the resolution. One can see that the magnitude of the jump is of order one in space.

This error can be explained in the following way. On the one hand, the linear interpolation is locally of order two in space (since the interpolated point lies exactly in the middle of the two known points, see Fig. 4.7). This error gives rise to an order one error globally as errors are summed over the entire line. On the other hand, the LBM time-space integration is third order accurate locally, and second

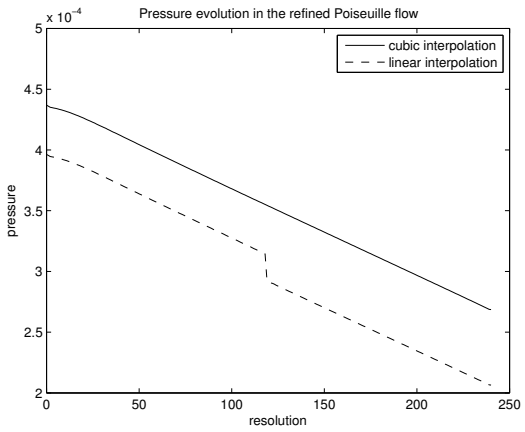


Fig. 4.14: Pressure plot along a horizontal line for the proposed refined Poiseuille flow.

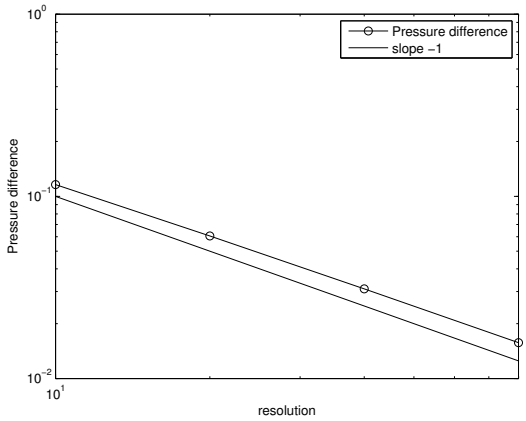


Fig. 4.15: The pressure jump magnitude versus the resolution of the coarse grid.

order globally (it is obtained through a trapezoidal integration). Therefore the linear interpolation gives rise to error terms incompatible with the error expected from the LBM.

4.6 Summary

In this chapter we introduced the grid refinement for the LBM.

Besides the thorough presentation of all the details, we would like to remark the following novelties in this chapter:

Filtering operation. (see Subsec. 4.4.2) The fine grid solves smaller scales than the coarse grid. When passing from the fine to the coarse grid, we propose a filtering operation based on traditional numerical methods, in order to remove high frequencies that the coarse grid is not able to handle. In the next chapter, we show that this operation is mandatory when dealing with turbulent flows.

Cubic interpolation. (see Subsec. 4.4.4) For the opposite operation, namely the copy from the coarse to the fine grid, we need to increase the information. This increase is based on an interpolation. We propose a symmetrical cubic interpolation to complete the fine sites which do not have a corresponding coarse site.

Importance of interpolation. (see Sec. 4.5) Numerical illustration of the importance of the interpolation order used in the increase of information. We simulate a simple Poiseuille flow and we show that when using a second order interpolation, we have a loss of mass in the interface. We then use a cubic interpolation. This operations avoids the problem.

Chapter 5

Implementation of grid refinement

In this chapter, we present the steps that have been followed in order to implement our generic grid refinement in the Palabos library. Our implementation is then tested over laminar and turbulent problems.

5.1 Main implementation ideas

In order to implement the grid refinement, we need two basic ingredients. First, we need to create the geometry that represents the non–uniform grid system we want simulate. Then, we must add the couplings in the interfaces between grids of different levels, where the communication of the informations take place. This section presents both aspects, as well as some other details important to the implementation.

In the following, we fix the number of grid levels that can co-exist in our system to M. We define the grid G_0 to be the coarsest grid and to serve as the reference level. Accordingly, the finest grid is the grid G_{M-1}. When considering two grids G_i and G_j, if $j > i$ then G_j is finer than G_i.

5.1.1 Geometrical considerations

We have decided to use the multi–domain approach presented in Subsec. 4.3.1. To represent a grid of a certain level, we use a multi–block that contains only the parts corresponding to this level. The use of a multi–block allows the level to have a global knowledge of the positions of the other blocks. In Fig. 5.1, we show an example of a multi–domain consisting of three levels and how it is represented in reality.

We have defined two functions with the purpose of constructing the refined grids: the **refine** operation, which adds a finer grid over a coarse one and the **coarsen** operation, that replaces a region of a fine grid with a coarser grid. In

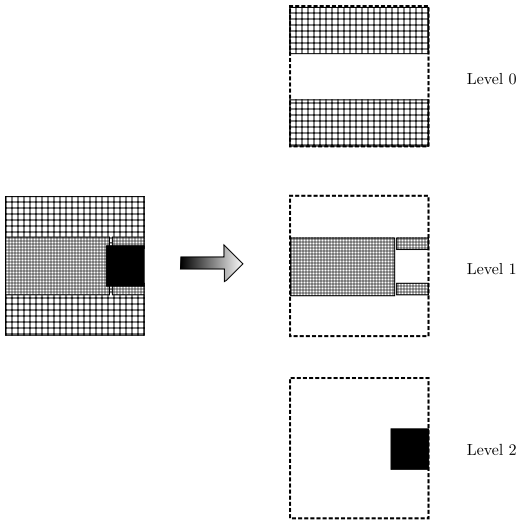


Fig. 5.1: A multi–domain of three levels is actually three multi–blocks.

what follows, both operations are explained for a 1D example, but the 2D and 3D generalizations are straightforward.

Let us note that while performing the geometrical operations, the regions for the couplings between grids appear naturally. In order to express them, we re-use the notations of Subsec. 4.3.4. We recall that $\boldsymbol{x}_{f\to c}$ are the sites where we perform the copy of informations from the fine grid to the coarse one. Equally, $\boldsymbol{x}_{c\to f}$ defines the set of coordinates where we perform the other direction: from the coarse to the fine grid.

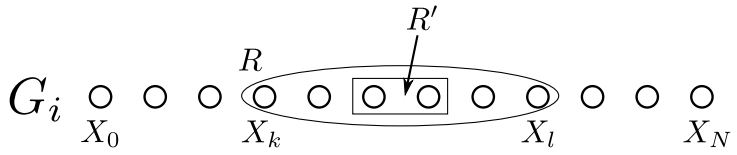
First, we describe the **refine** operation. All the necessary steps are shown in Fig. 5.2. We note that the grid G_{M-1}, by definition, cannot be refined.

1. Our starting point is a coarse grid G_i, represented by an ordered set of sites $G_i = \{X_i | i \in [0, \ldots, N]\}$. Let us define a sub-set $R \subseteq I$, represented by $R = \{X_i | i \in [k, \ldots, l]\}$. We wish to insert a refined grid on R. We define $R' = \{X_i | i \in [k-2, \ldots, l+2]\}$, which is the "decreased" version of the set R, because we have taken two sites at left and at right of the original.

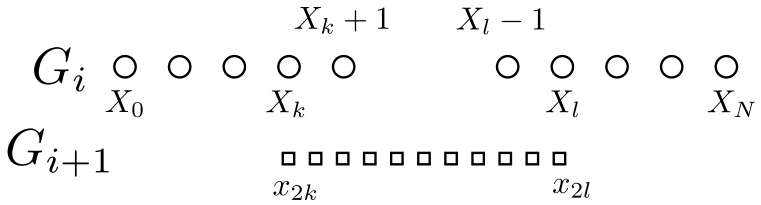
2. We perform the exception operation $G_i \setminus R' = \{X_i | i \in [X_0, \ldots, X_k - 1] \cup [X_l + 1, \ldots, X_N]\}$ and define G_i as being equal to this difference. In opposition, the finer grid G_{i+1} takes R and converts it with twice as many elements, obtaining $r = \{x_i | i \in [2k, \ldots, 2l]\}$. We redefine G_{i+1} as $G_{i+1} \cup r$.

3. Finally, we can extract both sets of coordinates where the couplings must be done: $\boldsymbol{x}_{f\to c} = \{x_{2k}, x_{2l}\}$ and $\boldsymbol{x}_{c\to f} = \{X_k + 1, X_l - 1\}$.

1. Definition of R and R'



2. Exclusion of R' from G_i
   Addition of r to G_{i+1}



3. Definition of the zones for the couplings

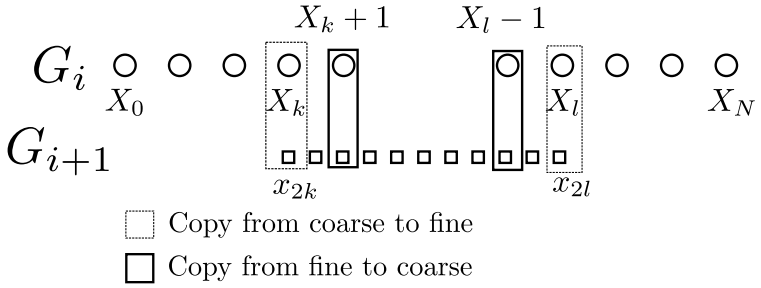


Fig. 5.2: 1D dimensional example of a refine operation of grid G_i over the domain R.

The **coarsen** operation can be explained in a similar way. All the steps are depicted in Fig. 5.3. We forbid the coarsen operation to G_0.

1. Let us begin with a fine grid $G_i = \{x_i | i \in [0, \ldots, N]\}$. We define a set of sites $r = \{x_i | i \in [2k, \ldots, 2l]\}$ where we want a coarser resolution. We must ensure that the first and last sites in r have corresponding sites over the coarse resolution, this is why we have chosen them as being even number. We decrease the set r to obtain $r' = \{x_i | i \in [2k + 1, \ldots, 2l - 1]\}$.

2. We exclude the set r' from G_i, obtaining $G_i \setminus r' = \{x_i | i \in [0, \ldots, 2k] \cup [2l, \ldots, N]\}$, which takes the place of G_i. The set r is converted to coarse coordinates of grid G_{i-1}. This results into a set $R = \{X_i | i \in [k, \ldots, l]\}$. We then increase this set obtaining $R' = \{X_i | i \in [k - 1, \ldots, l + 1]\}$. It is this set R' which is added to G_{i-1}.

3. To end the explanation, we can once again easily extract the sets of coordinates where the couplings need to be added. One recovers the sets $\boldsymbol{x}_{c\to f} = \{x_{2k}, x_{2l}\}$ and $\boldsymbol{x}_{f\to c} = \{X_k - 1, X_l + 1\}$.

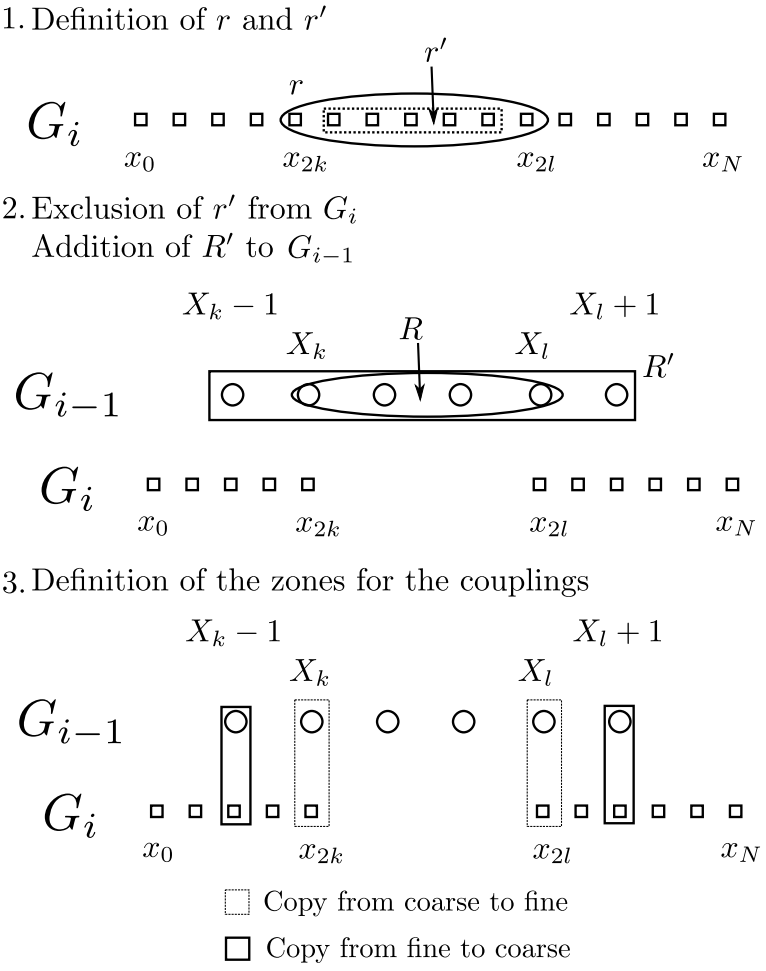


Fig. 5.3: 1D dimensional example of a coarsen operation of grid G_i over a domain r.

5.1.2 Couplings

The couplings presented in Sec. 4.4 must be implemented. As we are using a multi–block per level, the natural mechanism in Palabos to implement such operations are the processing functionals (see Chap. 3).

We have defined two different processing functionals, one for the copy from coarse to fine and one for the copy from fine to coarse. We remark that these processing functionals must be applied only over the corresponding interfaces defined by the sets $\boldsymbol{x}_{c\to f}$ and $\boldsymbol{x}_{f\to c}$.

The first processing functional copies the information from the coarse to the fine grid. This operation has a double behaviour: copy the information to the fine sites that correspond to a coarse neighbour and interpolate over fine sites which do not have a corresponding coarse site. As we have shown in Subsec. 4.4.4 there are different interpolations to be performed according to the available neighbours. We must then create a specific processing functional for each case. For instance, in Fig. 5.4, we show that two kinds of processing functionals exist, according to the type of interpolation they carry. We recall that in the case of a 2D example, the interpolation is performed over a line.

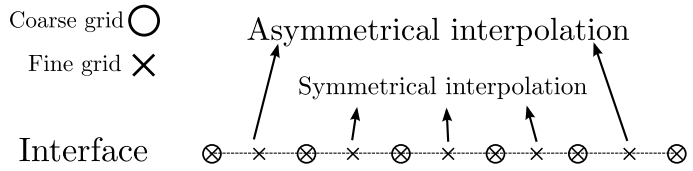


Fig. 5.4: In 2D the interpolations are carried over a line. Here we show which kind of interpolation is used over each site.

The second processing functional performs the filtering operation (see Subsec. 4.4.2) prior to the copy from the fine to the coarse grid. This data processor needs to keep track of the current time, so that it is executed every two collide–and–stream steps of the fine grid. For the sake of completeness, we have also implemented the copy operation without any kind of filtering. This has allowed us to test both approaches over the examples.

There is one more detail to note in the couplings. As a result of the relation between the temporal step on the coarse and fine grids $\delta t_c = 2\delta t_f$ we must perform a temporal interpolation.

We depict this situation in Fig. 5.5 for one collide–and–stream step. Let us suppose that we are at a time t_0. When a collide–and–stream operation is applied over the coarse grid, it goes to time $t_1 = t_0 + \delta t_c$. In the case of the fine grid, two collide–and–stream operations are needed to arrive to the same $t_1 = t_0 + 2\delta t_f$. It is therefore necessary to complete the fine grid at time $t_{1/2} = t_0 + \delta t_f$. The easy solution is to use the values from the coarse grid at t_0 and t_1 and perform a linear interpolation. This interpolation has forced us to store the populations at time t_0 and t_1 over every site contained on the set $\boldsymbol{x}_{c\to f}$.

The set-up of the couplings is a challenging operation because of the technical

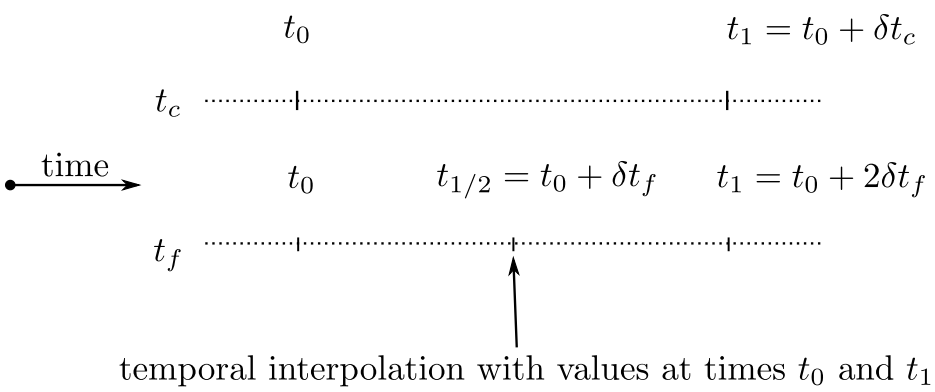


Fig. 5.5: Graphic representation of the temporal evolution of the coarse and the fine grids to show the need of a temporal interpolation.

difficulties that appear. One must, at this point, ensure that all the processing functionals are applied in a synchronous fashion, so that the couplings are executed at the right moment. This becomes specially challenging when considering that up to M grid levels can co-exist in our system.

5.1.3 Other considerations

Let us present other details that are not related with the geometrical management or the couplings.

First of all, in order to implement the whole system, we have created a master object to make abstraction for the M different grids that compose our non–uniform grid. As we have chosen to use a multi–block per level, this object uses the results of the geometrical calculations to create the multi–blocks for each level. At the creation, it also sets up all the necessary operations and special dynamics to ensure the coupling between the different levels. One call to the collide–and–stream method of the multi–grid object implies that the coarsest grid of level 0 advances exactly one time step. However, this hides the complexity behind the fact that all the M present levels also advance the number of time steps corresponding to this one time step. Thanks to the processing functionals, all the communications intra–grid are performed when needed.

The second remark that we need to do deals with the computation of statistics on the refined grids. To extract information for specific regions of the domain, if the region is contained entirely in one level of resolution, then we can proceed directly with no further problems. In opposition, it is possible to need statistics over regions that span on several grid levels. In this case, we start by choosing a level k, such that $0 \leq k \leq M$. All the grids which are coarser interpolate their

values up to the resolution of the level k. In opposition, the grids that are finer apply a restriction to their values up to k.

Finally, we need to treat the problem of parallel execution of our code. This becomes important, because to solve almost any interesting problem a single processor is not sufficient. We have decided to implement a simple manual division. Once the user has placed the refinements, he proceeds to select a number of divisions in each coordinate: N_x blocks in the x direction, N_y blocks in the y direction and N_z blocks in the z direction. He must then execute the code over $N_x \cdot N_y \cdot N_z$ processors. For instance, in Fig. 5.6, we show and example of a 2D dimensional domain in which $N_x = 3$ and $N_y = 1$. We note that when using such a simple division algorithm, the quality of the correct distribution of the workload depends on the regularity of the domains.

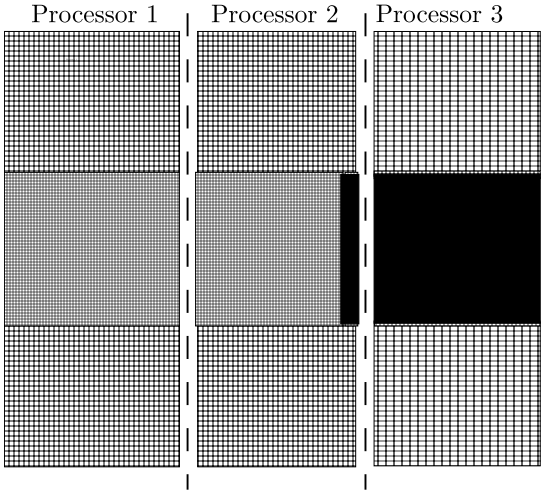


Fig. 5.6: A three–level grid system which has been divided in three sections for parallel execution.

5.2 Examples and validation

In this section, we validate the implementation of the grid refinement that is presented above. We make the difference between the two-dimensional and the three-dimensional versions. For each of them, we start with a problem which is relatively laminar. The more complex problems come afterwards, where we use higher Reynolds numbers to simulate turbulent flows.

The physical quantities of interest in each problem are computed and shown to be in agreement in both the uniform and non–uniform grids. As it is one of

the main interests for non-uniform grids, we also discuss the performances of our implementation compared to the use of a uniform grid.

5.2.1 2D examples

First, we study the unsteady flow past a circular cylinder at Re = 100 which is a well documented problem (see [72] for example). Then, we simulate the case of the dipole–wall collision, which is a 2D turbulent problem that has been already studied and benchmarked for a LBM code in [41].

5.2.1.1 Flow past a cylinder

This flow has been extensively studied. In particular we find a set of tests in [72] where a set of values have been obtained using a wide range of methods. Here we are interested in the steady flow past a circular cylinder. We fix the Reynolds number $\mathrm{Re} = u_{\mathrm{mean}} N/\nu = 100$, with u_{mean} the mean velocity of the inlet flow, N the diameter of the cylinder and ν the kinematic viscosity. We are interested in comparing our result for the maximum drag C_d and lift C_l coefficients

$$C_d = \frac{2F_d}{\rho u_{\mathrm{mean}}^2 N} \ , \ C_l = \frac{2F_l}{\rho u_{\mathrm{mean}}^2 N}, \tag{5.1}$$

F_d and F_l are the drag and lift forces respectively, which are given by the equations

$$F_d = \int_S (\rho\nu \frac{\partial \boldsymbol{v_t}}{\partial \boldsymbol{n}} n_y - p n_x) dS \ , \ F_l = \int_S (\rho\nu \frac{\partial \boldsymbol{v_t}}{\partial \boldsymbol{n}} n_x + p n_y) dS, \tag{5.2}$$

where S is the cylinder surface, $\boldsymbol{n} = (n_x, ny)$ is the normal vector of the surface S, and $\boldsymbol{v}_t$ is the tangential velocity on S. In the LBM, this forces can be easily computed by the momentum exchange method as presented in [47].

The details of the simulation geometry are shown in Fig. 5.7. We note that all the quantities are expressed with respect to the diameter of the cylinder N. At the inlet we impose a Poiseuille profile and a zero velocity gradient at the outlet. Furthermore near the outlet we filtered the solution using the method proposed in [60] to increase the numerical stability of the boundary condition. The horizontal walls have zero velocity. We use a lattice velocity $u_{\mathrm{mean}} = 0.00333$ and the cylinder diameter is fixed to have N lattice sites. First, we used a uniform single-level grid to find a resolution at which we find values for the maximum of the drag and lift coefficients that are close enough to the values in the reference [72]. We found that $N = 80$ was enough.

Based on this, we implemented a four level refined grid where the finest resolution has $N_f = 80$. In this case the coarsest grid will only have $N_c = 10$. For

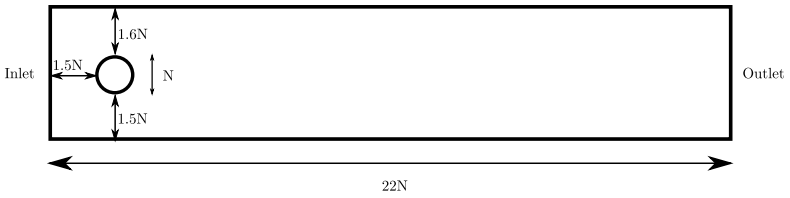


Fig. 5.7: Geometry for the flow past a cylinder.

accuracy reasons in this benchmark, we have decided to use the incompressible BGK model [29]. The only difference introduced by this model is the computation of the equilibrium distribution which is now given by

$$f_i^{eq} = w_i \left(\rho + \rho_0 \left(\frac{\boldsymbol{x}_i \cdot \boldsymbol{u}}{c_s^2} + \frac{1}{2c_s^4} \boldsymbol{Q}_i : \boldsymbol{u}\boldsymbol{u} \right) \right), \tag{5.3}$$

where here ρ_0 is a constant and was chosen to be equal to one here. Since the discussion we performed in Chap. 4 is generic and does not really depend on the exact form of the equilibrium distribution there is no change in the refinement strategy.

The refinement used for our simulation is depicted in Fig. 5.8. In this picture, N refers to the total diameter of the cylinder in coarse units. The results obtained

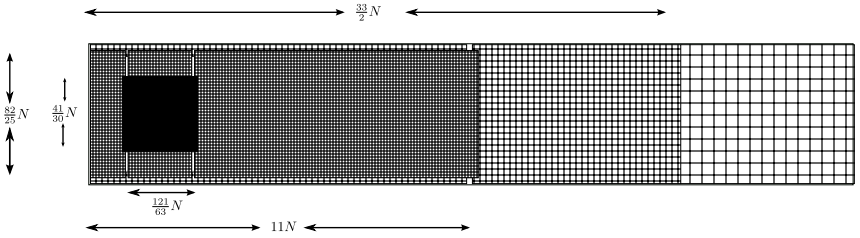


Fig. 5.8: Proposed refinement for the 2D flow past a cylinder.

for the drag and lift for the refined case match the accuracy with three significant digits (which is also the level of accuracy given in Ref. [72])

$$C_d = 3.24, \quad C_l = 0.982. \tag{5.4}$$

Furthermore, these values are in good agreement with those found in [72], where $C_d = 3.22 - 3.24$ and $C_l = 0.99 - 1.01$. The results for the uniform grid are obtained with about $5900N^2$ points, whereas there are only about $1000N^2$ points in the refined case, representing roughly five time less points. We did not perform

any advanced technique to find the optimal position of our refined grids, it is therefore possible that one can get similar results with even less points.

In order to compare the CPU performances, we used a single CPU to simplify the comparison procedure. When benchmarking the uniform grid against the refined one for several values of N, we find that a mean speed-up of 5, thus proving that the overhead introduced by the coupling between the different grid levels does not have a significant impact in the overall performances.

5.2.1.2 Dipole–wall collision

This benchmark, based on Refs. [13] and [41], analyses the time evolution of a self-propelled dipole confined within a 2D box. The geometry of the box is a square domain $[-1,1]\times[-1,1]$ and is surrounded by no-slip walls. The initial condition describes two counter-rotating monopoles, one with positive core vorticity at the position (x_1,y_1) and the other one with negative core vorticity at (x_2,y_2). This is obtained with an initial velocity field $\boldsymbol{u}_0=(u_x,u_y)$ which reads as follows in dimensionless variables :

$$u_x=-\frac{1}{2}\left\|\omega_e\right\|(y-y_1)\mathrm{e}^{-(r_1/r_0)^2}+\frac{1}{2}\left\|\omega_e\right\|(y-y_2)\mathrm{e}^{-(r_2/r_0)^2}, \tag{5.5}$$

$$u_y=+\frac{1}{2}\left\|\omega_e\right\|(x-x_1)\mathrm{e}^{-(r_1/r_0)^2}-\frac{1}{2}\left\|\omega_e\right\|(x-x_2)\mathrm{e}^{-(r_2/r_0)^2}. \tag{5.6}$$

Here, $r_i=\sqrt{(x-x_i)^2+(y-y_i)^2}$, defines the distance to the monopole centres. The parameter r_0 labels the diameter of a monopole and ω_e its core vorticity.

The average kinetic energy of this system at a given time is defined by the expression

$$\langle E\rangle(t)=\frac{1}{2}\int_{-1}^{1}\int_{-1}^{1}\left\|\boldsymbol{u}\right\|^2(\boldsymbol{x},t)d^2x, \tag{5.7}$$

and the average enstrophy by

$$\langle\Omega\rangle(t)=\frac{1}{2}\int_{-1}^{1}\int_{-1}^{1}\omega^2(\boldsymbol{x},t)d^2x, \tag{5.8}$$

where $\omega=\partial_x u_y-\partial_y u_x$ is the flow vorticity.

The dipole described by Eqs. (5.5) and (5.6), under the actions of viscous forces, develops a net momentum in the positive x-direction and is self-propelled towards the right wall. When the dipole collides with the wall a maxima of enstrophy is achieved.

In order to obtain really accurate results, the pressure field must be initialized from the velocity field, using the Poisson equation

$$\boldsymbol{\nabla}^2 p=(\boldsymbol{\nabla}\boldsymbol{u}):(\boldsymbol{\nabla}\boldsymbol{u})^T \tag{5.9}$$

where the column symbol stands for the full index contraction. This equation is solved with a Gauss-Seidel iterative method.

In this test-case, we are aiming at obtaining the maximum values of the enstrophy which have been computed in [13] with a spectral method. We intend to use local refinement in the areas where we know the important physics happen, namely close to the wall where the small structures are formed. It is expected to obtain accurate values comparable to those obtained with a uniform grid. We study two different Reynolds numbers , $\mathrm{Re} = 625$ and $\mathrm{Re} = 5000$.

For $\mathrm{Re} = 625$, we use three levels of refinement as shown in the Fig. 5.9. The pathway of the dipole is refined. However, we know where it will collide, so this part is further refined. We find that this strategy agrees with the physical phenomenon.

This example is particularly challenging from the numerical point of view since strong velocity gradients are crossing the different refinement interfaces. We can

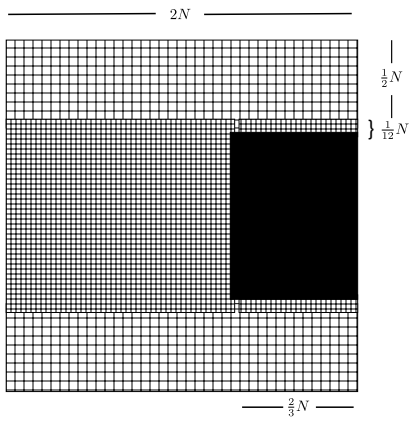


Fig. 5.9: Refinement for the dipole-wall collision

compute the number of points in the refined domain and compare it against the number of points in the non-refined grid. Let us call N the resolution of the coarsest grid. The finer grid would therefore have a resolution of $4N$ or a total amount of points of $16N^2$. When using a local refinement strategy as the one proposed in this benchmark, one obtains $\frac{9}{2}N^2$ points. For any length N we observe that we are using almost four times less points in the refined case.

When performing a one CPU benchmark between a uniform grid and our refined version for several values of N, we find a mean speed-up of roughly three. Once again it is interesting to see that the performance gain is close to the memory saving that has been computed.

When analysing the error for our refined grids, we compare the values obtained with the most accurate value of the enstrophy found by a spectral method, which

is 933.6. When decreasing N and simulating again, we find that the solutions tends to this value with second order slope. This can be seen in Fig. 5.10. This is really an interesting fact, as we are preserving the second order accuracy of the LBM with the grid refinement technique. It is important to note that the case

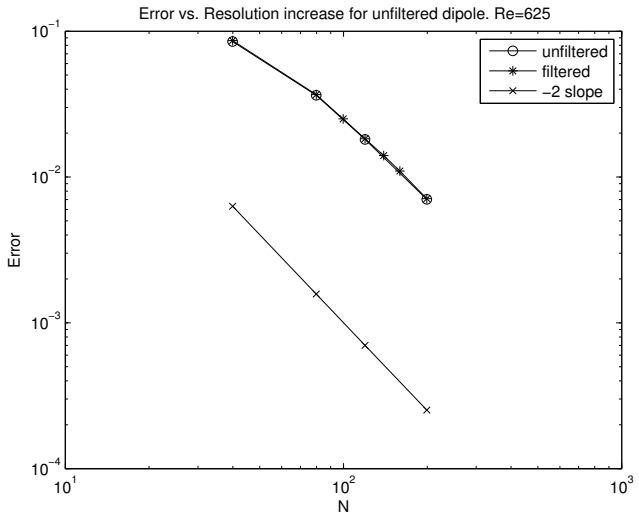


Fig. 5.10: Evolution of error for the filtered and unfiltered simulations.

$Re = 625$ can be done without the filtering method presented in Subsec. 4.4.2. However, when dealing with the case $Re = 5000$, it is imperative to use the filtering operation, as the small structures from the fine grid pollute the coarser grids, triggering numerical instability. This can be seen in Fig. 5.11, where, on the left, the non-filtered case shows signs of numerical instability (and finally diverges) while the filtered one remains stable.

For the case Re = 5000 the most accurate value obtained for the enstrophy in [13] (with an spectral method code) is 5536. When using six levels of refinement (Fig. 5.12) and setting $N = 100$ for the coarsest grid we have found 5500 for the maximum of the enstrophy. When using a uniform grid, this same result would require a resolution of approximately $N = 3200$, which makes the solution of the problem really difficult to achieve, thus proving the importance of grid refinement.

5.2.2 3D examples

5.2.2.1 Flow past a sphere

We consider a channel flow of an incompressible viscous fluid. We put a sphere in the channel and simulate the flow. In particular, as it had been done for the 2D

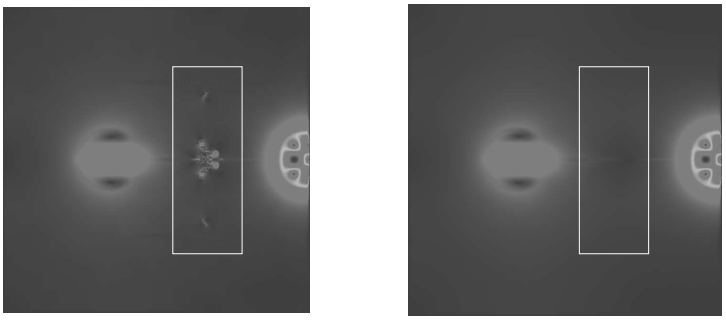


Fig. 5.11: Problems at $Re = 5000$ for non-filtered simulation on the left. On the right the filtered version, which shows no perturbation.

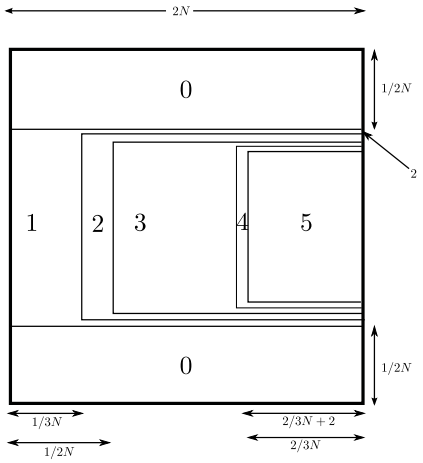


Fig. 5.12: Six level grid refinement to achieve a good enstrophy value.

cylinder problem, we are interested in studying the force exerted over the sphere in the case of laminar steady flows.

The set-up for the simulation is presented in Fig. 5.13. Let N be the diameter of the sphere. The bounding box has coordinates $[0, 29N] \times [0, 9N] \times [0, 9N]$. We have chosen to place the sphere at the point $\boldsymbol{c} = (4.5N, 4.55N, 4.55N)$. In the yz plane, the sphere is not perfectly in the centre in order to break the symmetry of the problem.

We have chosen the following Reynolds numbers: 50, 100 and 150. Exactly as for the 2D case, the Reynolds number is defined as $\mathrm{Re} = u_{\mathrm{mean}} N/\nu$, where $u_{\mathrm{mean}} = 0.1$ is the mean velocity in lattice units, N is the diameter of the sphere and ν is the kinematic viscosity. We have chosen a relatively important velocity u_{mean} to accelerate the convergence of the system to its steady state.

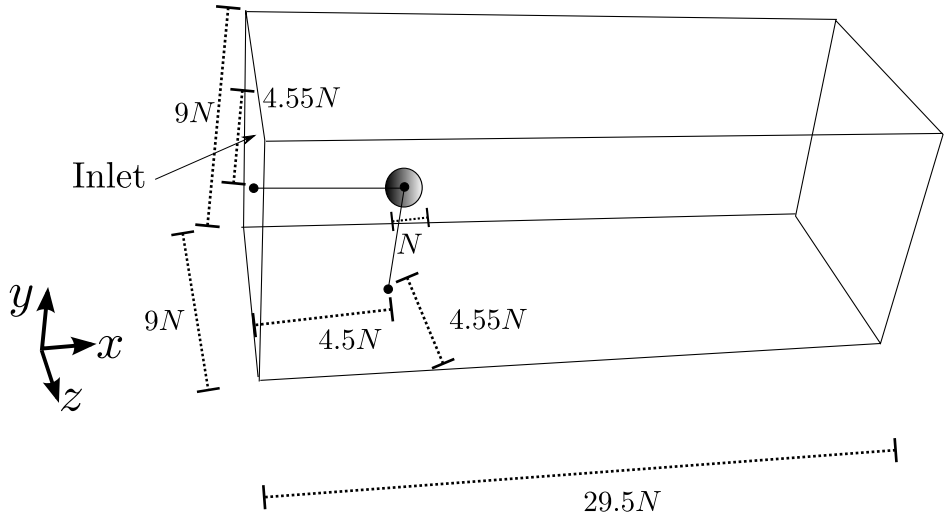


Fig. 5.13: Schema of the set-up for the simulation of the flow around a sphere.

All the walls have a velocity boundary condition. The inlet has velocity $\boldsymbol{u}$ fixed to $(u_{\text{mean}}, 0, 0)$. The rest of the walls have null velocity.

We must note that we represent the sphere with bounce–back nodes as it is simpler than using a more sophisticated boundary condition, such as an interpolated boundary condition.

The computation of the drag coefficient C_d is similar to the 2D case, excepting that this time we need to consider the surface of the sphere as dividing factor

$$C_d = \frac{F_d}{2\rho u_{\text{mean}}^2 \pi N^2}, \tag{5.10}$$

where once again F_d is the drag force as before, which is given by the equation

$$F_d = \int_S (\rho\nu \frac{\partial \boldsymbol{v_t}}{\partial \boldsymbol{n}} n_y - p n_x) dS. \tag{5.11}$$

This time S is the sphere surface, $\boldsymbol{n} = (n_x, n_y, n_z)$ is the normal vector of the surface S, and $\boldsymbol{v}_t$ is the tangential velocity on S. This forces are once again computed using the momentum exchange algorithm.

The drag coefficient C_d obtained are compared to the values found in Johnson et *al.* [34]. By using a uniform grid with $N = 32$ we have found a good agreement between our results and the results presented in the reference.

For the refined grid, we have used four grid levels. The refinements are placed near the sphere. For reasons of consistency, the sphere is completely contained in the finest level. We have chosen to impose that the sphere has the same diameter

as in the best result for the uniform grid, so our finest level has $N_f = 32$. With four levels of refinements, this implies that $N_c = 4$ for the coarsest level. The refinements are depicted in Fig. 5.14.

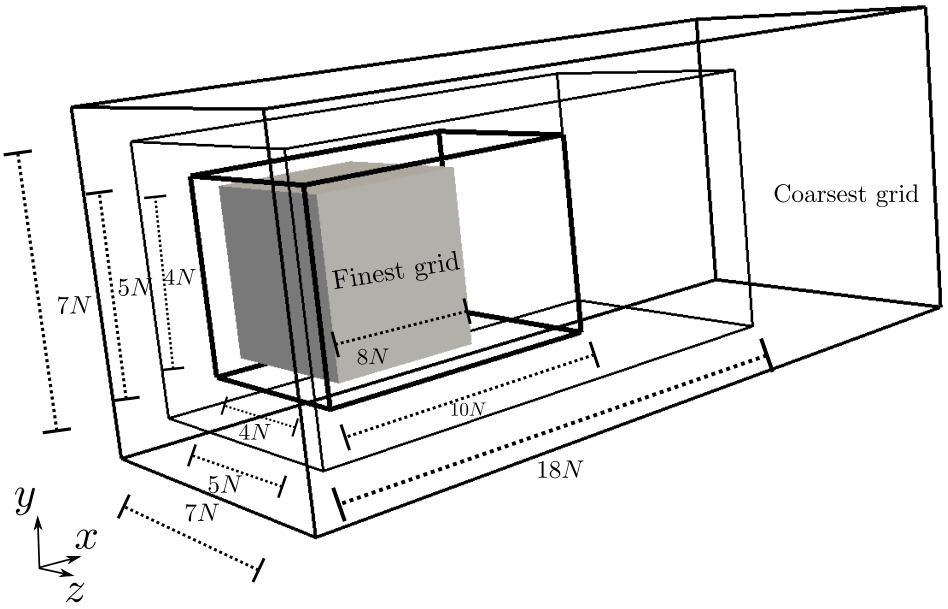


Fig. 5.14: The four levels of refinement proposed to solve the flow around a sphere.

Let us start by comparing the number of points in both the uniform and the refined grids. We express all the quantities in function of the resolution of the coarsest grid N_c. For the uniform grid, this implies that $N = 8N_c$. In function of this quantity, the uniform grid counts with $740096N_c^3$ mesh points, which gives approximately 47 million of points. For the refined grid, doing the calculation for each level and summing the number of points, we conclude that it has $89717.5N_c^3$ points or almost 5.7 million of points. The economy is considerable, as we have almost eight times less points in the refined grid.

We then proceed to compare the uniform grid and a refined grid for all the different Reynolds numbers cited above. The values obtained are found in Tab. 5.1. We observe that the error between the uniform grid and the refined grid are always smaller than 3%.

Table 5.1: Comparison of the C_d values for the uniform and refined grids.

Re	Uniform grid	Refined grid
50	1.504	1.49647
100	0.979818	0.95673
150	0.84463	0.8297

Concerning the computational time, the uniform grid was executed in 120 Xeon Intel processors. For one dimensionless time unit we require 3240.3 s. in mean. The refined grid was tested on 60 of the same processors. This time, the dimensionless time unit was achieved in 1021.4 s. thus being three times faster than the uniform grid on twice times less processors. We can safely say that our simulation is six times faster per processor.

Finally, we want to ensure that the second order of the scheme is preserved when using the refined grid. We fix Re = 50 and we simulate the system for resolutions $N_c = 1$, 2, 4, and 8 in order to perform a convergence test. The values obtained are compared with the drag coefficient for a uniform grid with $N = 64$, which is approximately 1.54. The result is plotted against the -2 slope as seen in Fig. 5.15.

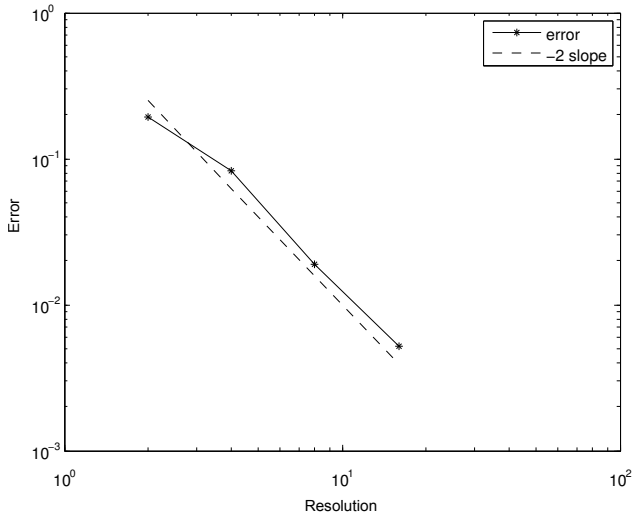


Fig. 5.15: Convergence of the drag coefficient C_d on the refined grid for Re = 50.

As it was the case with the 2D cylinder example, in the case of the steady flow around the sphere in 3D, adding or not the filtering operation did not change the results.

5.2.2.2 Circular turbulent jet

The next problem that we have treated is the circular turbulent jet. We consider a liquid contained in a box. The same liquid is injected at high speed through a circular opening of diameter d. For the interested reader, the complete theoretical background of the problem and the experimental results can be found in [53].

Our set-up for the simulation domain to solve this problem is shown in Fig. 5.16. The diameter d serves to define the whole domain. the width and height of the domain are defined to be equal to $7d$. The aspect ratio of the jet is chosen to be $x/d = 30$ thus giving us the depth of the box. In the reference book, the authors study jets at much higher Re, asserting that there would be a need to have a aspect ratio x/d of at least 50 units. We must however note that for the Re that we have defined, our aspect ratio is sufficient.

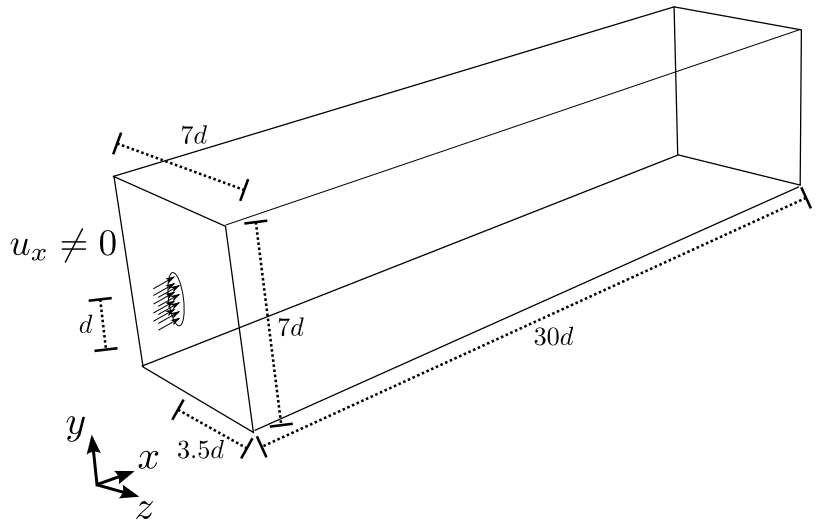


Fig. 5.16: Set-up for the turbulent jet.

In the literature, we have found that the critical Re for a non–buoyant turbulent jet is 1500 (see [75]). We have chosen a Reynolds number Re of 2000. It is defined by $\mathrm{Re} = u_{\mathrm{mean}} d/\nu$, where u_{mean} is the mean velocity at the inlet, d is the diameter of the opening and ν is the kinematic viscosity.

We have fixed the jet to have a fixed velocity $\boldsymbol{u} = (u_{\mathrm{mean}}, 0, 0)$ (in lattice units) in the inlet. The rest of the walls have null velocity.

In order to perform a direct numerical simulation of the proposed problem, considering the Kolmogorov scales (see [53]), we would need at least $\mathrm{Re}^{3/4} \cong 300$ lattice points for the diameter of the jet to solve the smallest scales. The size of the domain would then become enormous. In order to have a domain size more tractable, we need to use a model for the turbulence.

We have used a large eddy simulation model for the smallest scales not represented in reason of the restricted on the spatial discretization. In particular, Sub–grid models try to model the scales smaller than the size of a filter Δ by locally adapting the viscosity.

The Smagorinsky model is implemented in Palabos (see [32]). It is based on the usage of a modified effective relaxation time τ_{eff} which is defined by

$$\tau_{\text{eff}} = \tau + \tau_{\text{sgs}}, \tag{5.12}$$

where τ is the usual relaxation time and τ_{sgs} is the sub–grid relaxation time. τ_{sgs} is defined by the following relation

$$\tau_{\text{sgs}} = \frac{C_s \Delta}{c_s^2} ||S||, \ ||S|| = \sqrt{2\boldsymbol{S} : \boldsymbol{S}} \tag{5.13}$$

with C_s the Smagorinsky constant, Δ the filter size and $\boldsymbol{S}$ is the strain–rate tensor. This tensor can be computed from finite–differences approach using Eq. 2.32 for example. The LBM with the BGK approximation becomes then:

$$f_i(\boldsymbol{x} + \boldsymbol{\xi}_i, t+1) = f_i(\boldsymbol{x}, t) - \frac{1}{\tau_{\text{eff}}} \left(f_i(\boldsymbol{x}, t) - f_i^{eq}(\boldsymbol{x}, t) \right). \tag{5.14}$$

To find more about the complete theory of the sub–grid models for the LBM, the interested reader is invited to read Malaspinas 2012 [45].

In our code, we use a Smagorinsky constant C of 0.14 on the bulk of the domain, which are common values in the literature (see [32]). Nevertheless, we have defined a higher constant ($C = 0.5$) for d sites before the outlet. The effect of this zone is to damp the high frequencies that arrive to the outlet boundary condition as much as possible, to avoid that the simulation becomes numerically unstable.

The utilisation of the Smagorinsky sub–grid model permits the usage of only 40 lattice sites to represent the diameter of the jet and still obtain results that agree with the ones found in the corresponding literature.

Next, we detail the quantities that are compared with the experimental results. Let us define the velocity $\boldsymbol{u}(\boldsymbol{x}) = (u_x, u_y, u_z)$ for a point $\boldsymbol{x} = (x, y, z)$. Let the centre of the jet be $\boldsymbol{c} = (c_x, c_y, c_z)$, then, the centreline velocity can be defined as

$$U_0(x) = \langle u_x(x, c_y, c_z) \rangle, \tag{5.15}$$

where the operator $\langle \cdot \rangle$ stands for a temporal mean.

The jet half-width $r_{1/2}(x)$ is defined as the distance where the mean velocity equals half of the centreline velocity. Formally the relation is

$$\langle u_x(x, r_{1/2}(x), c_z) \rangle = \frac{1}{2} U_0(x). \tag{5.16}$$

It is important to remark that the value of $U_0(x)$ must decrease when x increases. Other important observation is that $r_{1/2}(x)$ must increase with x. There exists a x_0 value for which $r_{1/2}(x)$ behaves linearly with slope s, for instance like

$$r_{1/2}(x) = s(x - x_0), \tag{5.17}$$

the slope for different Re numbers ranging from ten thousand to one hundred thousand lies around $s = 0.01$. As this value is valid for such a huge range of values, we compare our values of $r_{1/2}(x)$ with this slope as well.

Turbulent flows present some characteristic quantities that are self–similar. Two quantities at different scales are self similar if they behave exactly. In particular, for the turbulent jet, the quantity $\langle U \rangle / U_0(x)$ plotted against $r/r_{1/2}(x)$ give the same exact curve after some critical aspect ratio x/d.

Let us note that we have performed a mean of u_x over 1000 dimensional units after the development of the turbulence for the following computations.

First, we show that $r_{1/2}(x)$ has a linear behaviour, which corresponds to the slope $s = 0.01$ in a certain region. The reader can find the plot in Fig. 5.17. We note that near the exit, this law is not valid. This is due to the fact that we have used a dampening zone as it was mentioned earlier.

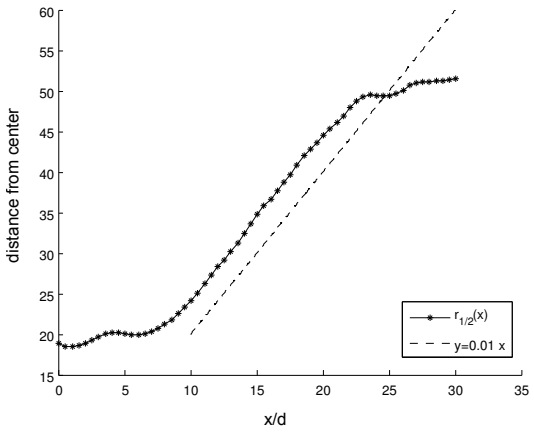


Fig. 5.17: plot of $r_{1/2}(x)$ against x.

The self–similarity of the flow is clearly presented in Fig. 5.18 for the aspects ratio $x/d = 10$, $x/d = 15$ and $x/d = 20$.

In conclusion, our numerical results agree very well with the expected behaviour from experimental data that is found on the aforementioned reference.

The next step is to simulate the same problem with a refined grid and verify that the results for the physical quantities that interest are in agreement for both set-ups.

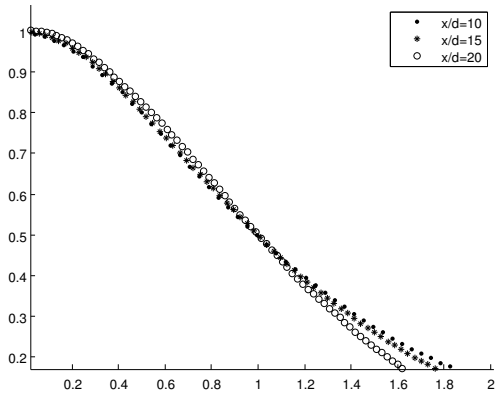


Fig. 5.18: Self–similarity of the quantity $\langle U \rangle / U_0(x)$ plotted against $r/r_{1/2}(x)$.

We have chosen to use only three levels of refinement. We have not used more levels, because in a turbulent problem, we are limited by the relaxation time in the coarsest grid. Let us recall Eq. 2.27. In this equation, we have a relationship between the relaxation time and the viscosity. In order to have a positive viscosity, the relaxation parameter must be smaller than two. Numerical instabilities are known to arrive when τ is too close to two. As we have fixed the value of the diameter d for the finest grid, this does not allow for more than two coarser grids before arriving to a value of $\tau = 1.9946$.

The diameter d on the finest grid is defined to be 40, to match the diameter used for the results on the uniform grid. As a result, the coarsest grid has $d_c = 10$ and the intermediate grid $d = 20$. The refinements used and their positions are depicted in Fig. 5.19. Let us count the grid points for the uniform and non–uniform grids, by using the coarsest grid diameter d_c as reference. We find $94080 d_c$ points for the uniform grid and $23550 d_c$ for the refined grid. In conclusion, we have four times less points in the refined case.

Because of the symmetry of the refined domain, we can have a good load balancing very simply. We have this time obtained a speed–up of almost 3.5 between the refined and the uniform grids when executed over the same number of processors. This shows that the overhead introduced because of the interpolation and filtering operations is not important.

In order to perform the analysis of the results in the refined case, as in the uniform case, the data used for the following results is a temporal mean of the u_x velocity over 1000 dimensionless time units, after the turbulence has fully developed.

In Fig. 5.20, we can find the velocity norm on the system after 1000 dimensionless time units. We present this figure to show that the jet is indeed turbulent

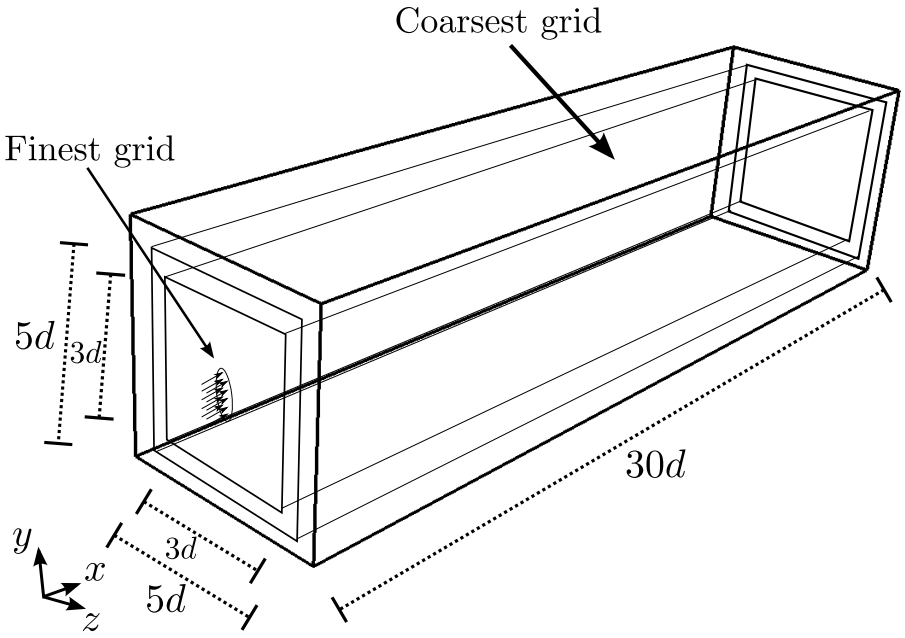


Fig. 5.19: The three level refinement that we have defined for the turbulent jet.

and that no visible artefacts between the different grid levels are present.

When simulating this proposed refined grid, one finds that $r_{1/2}(x)$ behaves in the same way it did for the uniform grid. The evolution of $r_{1/2}(x)$ is shown on Fig. 5.21.

Next, we analyse the self–similarity. The results are in agreement with the behaviour obtained for the uniform grid. The curves corresponding to the same aspect ratios that we presented for the uniform grid are shown in Fig. 5.22.

We present the mean velocity u_x velocity obtained with the refined grid. In Fig. 5.23, we show a cut of the mean velocity over the xy plane in the middle of the domain.

Finally, we have tested the same set-up, but this time we have not performed the filtering operation when copying informations from the fine to the coarse grids. This results on bogus structures that appear through the interfaces of the grids. This structures false completely the simulation of the data and eventually provoke the divergence of the whole system. We find a clear example of the problem in Fig. 5.24 where we compare the density gradient in the filtered (left) and unfiltered (right) versions in a plane parallel to the yz plane, found in the middle of the simulation domain at $t = 500$ dimensionless time units. In the unfiltered version, we clearly see bogus structures over the interfaces of the different grids.

Once again, this confirm the hypothesis that a filtering operation is mandatory

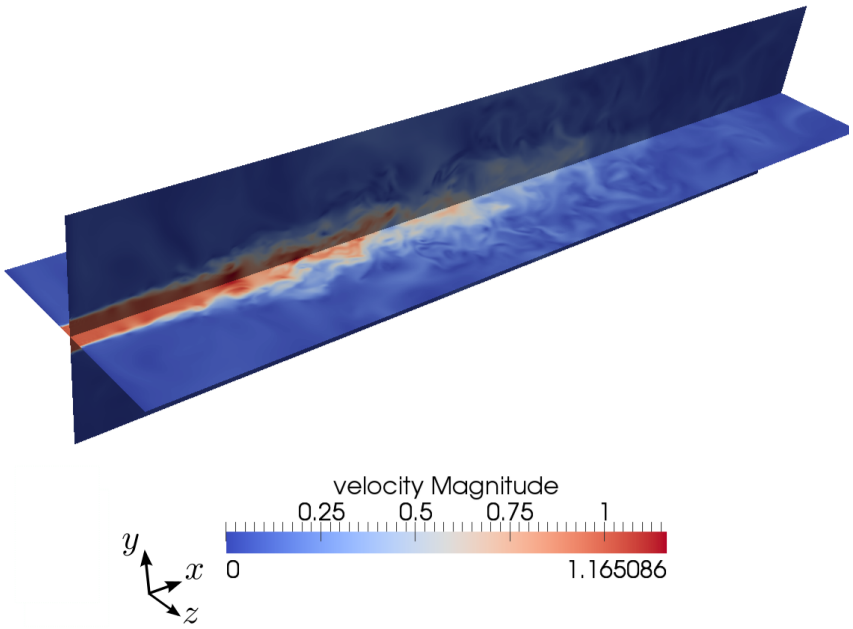


Fig. 5.20: Instant velocity norm of the developed turbulent jet at time $t = 1000$.

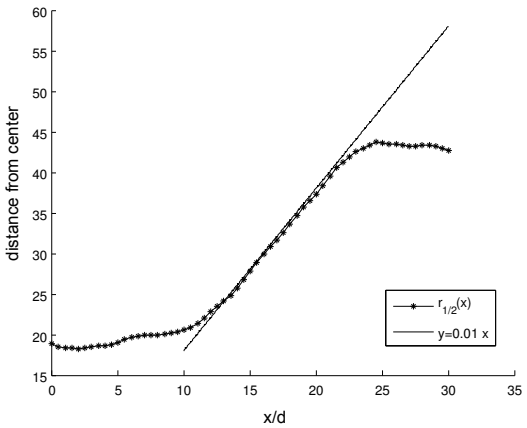


Fig. 5.21: plot of $r_{1/2}(x)$ against x for the refined grid.

when passing information from fine grids to coarser ones.

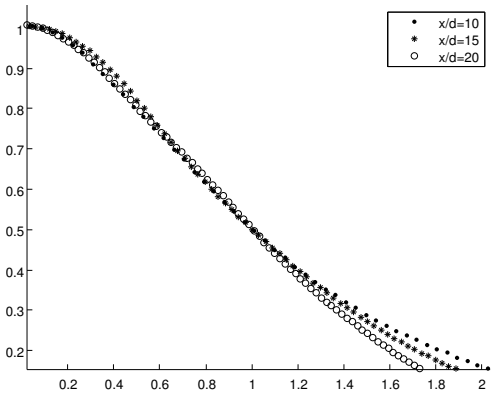


Fig. 5.22: Self–similarity of the quantity $\langle U \rangle / U_0(x)$ plotted against $r/r_{1/2}(x)$.

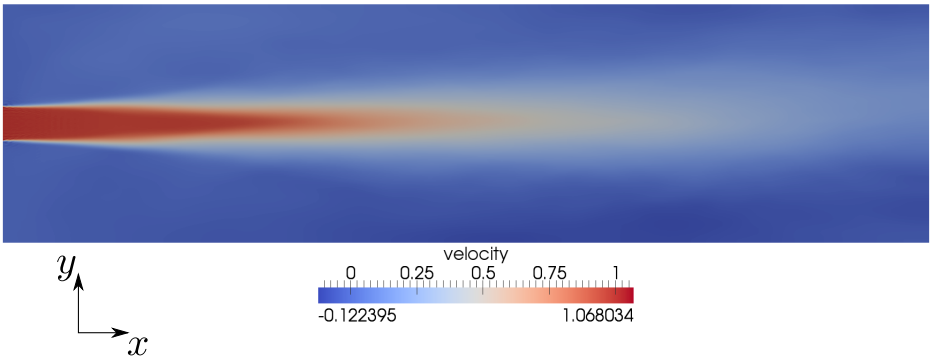


Fig. 5.23: Mean u_x velocity for the refined grid.

5.3 Summary

In this chapter, we have presented our implementation of the generic grid refinement on the Palabos library.

We start with the geometrical considerations, which are the base of grid refinement. We explain in detail two operations: refine and coarsen. The first one allows us to replace a region of a coarse grid by a finer one. The coarsen operation replaces a region of a fine grid by a coarser grid. Then, we briefly comment on the implementation of the couplings between the grids of different levels. For this operations, we have used processing functionals in Palabos. Finally, we discuss other details about our implementation, namely the representation of the system and the simple choices for parallel execution that we have made.

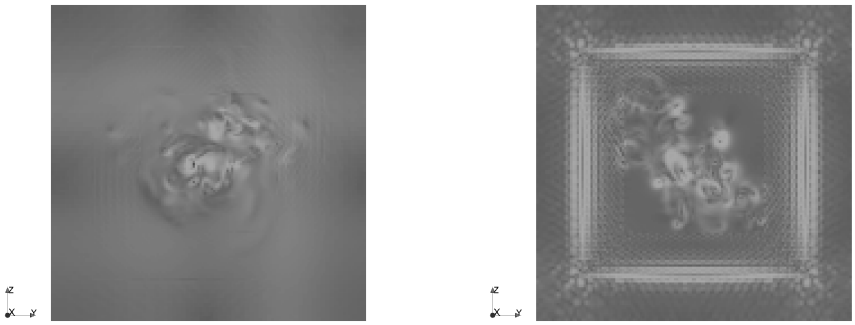


Fig. 5.24: Comparison of the density gradient in the middle of the domain for filtered and unfiltered versions.

Afterwards, we pass to the validation of our implementation. We have divided our tests in 2D and 3D cases.

In 2D we have solved two problems: the flow around a cylinder at relatively low Reynolds number and the more complicated dipole collision against a solid wall. In the first case, we have found the drag and lift coefficients. On the second test case we are interested in the enstrophy values when the dipole collides with the wall. For the dipole collision, we have simulated a high Reynolds number and found the filtering to be necessary for the stability and consistency of the simulation. In both cases we have compared the results for a uniform grid and a refined grid and found that the results match.

In 3D, we have first dealt with the flow around a sphere represented with bounce–back nodes. In this case, we study the evolution of the drag coefficient for several Reynolds numbers. We have found that the results of the refined grid are in conformity with the values obtained with the uniform grid.

The final test–case that we have simulated is the turbulent jet at a Reynolds number of 2000. A sub–grid model had to be used, because a direct numerical simulation was computationally too expensive. We were interested in two experimental results found in the literature: the behaviour of $r_{1/2}(x)$ and the self–similarity of the flow. Once again, we obtain results in agreement for the uniform and refined grids. Finally, we have remarked the importance of the filtering operation in this precise case, because not using it results in numerical instability.

For all the problems that we have simulated, we have shown that the LBM remains second order for our refined grids.

To end this summary, we can safely assert that the proposed coupling operations and our implementation are sound.

Chapter 6

Further directions of research for grid refinement

Here we present two topics related to grid refinement, but since they are not directly related to our generic implementation, we have chosen to make a separate chapter.

We start by introducing a new algorithm for grid coupling without an overlapping region between the grids, in opposition to the algorithm used in our generic implementation. Then, we suggest an algorithmic criterion to find regions susceptible to need a local grid refinement.

6.1 Grid refinement algorithm with no overlapping region

As we have shown previously in Chap. 4, the approach that we use for the generic grid refinement, requires a buffering redundant region between two grids of different resolutions. In this way, we can ensure that the copy of information is performed from sites having no missing information.

It is however possible to consider a different case, where no overlapping region exists between the coarse and fine grids. One needs to complete the missing populations in the coarse and fine grids consistently, similarly to a boundary condition algorithm. The goal of such idea is to eliminate the redundant sites that the buffering region introduces.

In this section, we start by showing the bases of the proposed algorithm. The algorithm is then implemented in the Palabos library. The approach is finally validated on a two–dimensional Poiseuille flow.

The sub–indexes c and f are used for, respectively, coarse and fine grids, as it has been done throughout this thesis.

To our knowledge, the only paper where an algorithm with no overlapping zone is Rheinländer [59]. This algorithm is fundamentally different to ours as it is based on the asymptotic analysis and it uses a diffusive scaling.

6.1.1 The algorithm

We consider a site $\boldsymbol{x}$ which belongs to the interface between a coarse and a fine grid. Let us divide the set of indexes $\{0, \ldots, q-1\}$ of the populations f_i in three sets I, II and III. We demand that the following properties are respected

$$I \cup II \cup III = \{0, \ldots, q-1\} \tag{6.1}$$

$$I \cap II = II \cap III = I \cap III = \emptyset \tag{6.2}$$

I represents the indexes of the unknown populations only on the fine grid. Similarly, II contains all the populations unknown only on the coarse grid. Finally, III is formed by indexes which are known in both grids.

In Fig. 6.1, we present an example of division of indexes for the D2Q9 lattice with the numbering of velocities proposed in Eq. 2.28.

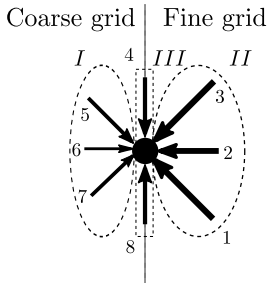


Fig. 6.1: Division of the indexes I, II and III for the D2Q9 when we have a coarse grid on the left and a fine one on the right.

The coupling between the two grids resides in the reconstruction of the missing distributions $f_{II,c}$ and $f_{I,f}$ at $\boldsymbol{x}$. To this end, we consider the following conditions that must be satisfied by the f_i^{neq}:

$$\sum_i f_i^{neq} = 0 \tag{6.3}$$

$$\sum_i \boldsymbol{\xi}_i f_i^{neq} = 0 \tag{6.4}$$

or in other words, for each of the grids:

$$0 = \sum_{i \in I} f_i^{neq} + \sum_{i \in II} f_i^{neq} + \sum_{i \in III} f_i^{neq} \tag{6.5}$$

$$0 = \sum_{i \in I} \boldsymbol{\xi}_i f_i^{neq} + \sum_{i \in II} \boldsymbol{\xi}_i f_i^{neq} + \sum_{i \in III} \boldsymbol{\xi}_i f_i^{neq} \tag{6.6}$$

We replace each f_i^{neq} by its definition $f_i^{neq} = f_i - f_i^{eq}(\rho, \boldsymbol{u})$. In this case f_i^{eq} is the standard equilibrium distribution, but ρ and $\boldsymbol{u}$ are unknowns yet to be defined.

As we are using a convective scaling, then it follows that

$$f_{i,f}^{eq} = f_{i,c}^{eq} = f_i^{eq}, \tag{6.7}$$

because the density and velocity are continuous through the different grids.

We also recall the result of Sec. 4.3.3 that states that non–equilibrium distributions in the fine and coarse grids are related by a factor b:

$$f_{i,f}^{neq} = b f_{i,c}^{neq} \tag{6.8}$$

which is given

$$b = \frac{\tau_f}{\tau_c}\frac{\delta t_f}{\delta t_c} = \frac{\tau_f}{2\tau_c}. \tag{6.9}$$

We rewrite Eqs. 6.5 and 6.6 for the fine grid. We replace $f_{i,f}^{neq}$ by $b f_{i,c}^{neq}$ for the indexes in I obtaining

$$0 = \sum_{i\in I} b(f_{i,c} - f_i^{eq}) + \sum_{i\in II}(f_{i,f} - f_i^{eq}) + \sum_{i\in III}(f_{i,f} - f_i^{eq}) \tag{6.10}$$

$$0 = \sum_{i\in I} b\boldsymbol{\xi}_i(f_{i,c} - f_i^{eq}) + \sum_{i\in II}\boldsymbol{\xi}_i(f_{i,f} - f_i^{eq}) + \sum_{i\in III}\boldsymbol{\xi}_i(f_{i,f} - f_i^{eq}). \tag{6.11}$$

We remark that the only unknowns in these equations are the $f_i^{eq} = f_i^{eq}(\rho, \boldsymbol{u})$. Please note that we have chosen to use the fine values for the region labelled III.

Let us remember that $\sum_i f_i^{eq} = \rho$ and $\sum_i \boldsymbol{\xi}_i f_i^{eq} = \rho\boldsymbol{u}$. This allows us to transform Eqs. 6.10 and 6.11 into

$$\rho = (1-b)\sum_{i\in I} f_i^{eq}(\rho, \boldsymbol{u}) + b\sum_{i\in I} f_{i,c} + \sum_{i\in II\cup III} f_{i,f} \tag{6.12}$$

$$\rho\boldsymbol{u} = (1-b)\sum_{i\in I} \boldsymbol{\xi}_i f_i^{eq}(\rho, \boldsymbol{u}) + b\sum_{i\in I} \boldsymbol{\xi}_i f_{i,c} + \sum_{i\in II\cup III} \boldsymbol{\xi}_i f_{i,f} \tag{6.13}$$

These equations can be further simplified in order to be solved. Let us recall the form of f^{eq} in Eq. 2.23. Let us write it as

$$f_i^{eq}(\rho, \boldsymbol{u}) = \rho h_i(\boldsymbol{u}) \tag{6.14}$$

with $h_i(\boldsymbol{u}) = w_i\left(1 + \frac{\boldsymbol{\xi}_i\cdot\boldsymbol{u}}{c_s^2} + \frac{1}{2c_s^4}\boldsymbol{Q}_i : \boldsymbol{u}\boldsymbol{u}\right)$.

If we replace this value in Eq. 6.12, we obtain

$$\rho = \frac{K}{1 - (1-b)\sum_{i\in I} h_i(\boldsymbol{u})} \tag{6.15}$$

with

$$K = b\sum_{i\in I} f_{i,c} + \sum_{i\in II\cup III} f_{i,f}. \tag{6.16}$$

Then, if we replace this equation in Eq. 6.13, we find one equation where the only unknown is $\boldsymbol{u}$:

$$\frac{K}{1-(1-b)\sum_{i\in I} h_i(\boldsymbol{u})}\boldsymbol{u} = (1-b)\sum_{i\in I}\boldsymbol{\xi}_i\frac{K}{1-(1-b)\sum_{i\in I} h_i(\boldsymbol{u})}h_i(\boldsymbol{u}) + \boldsymbol{L} \tag{6.17}$$

where

$$\boldsymbol{L} = b\sum_{i\in I}\boldsymbol{\xi}_i f_{i,c} + \sum_{i\in II\cup III}\boldsymbol{\xi}_i f_{i,f}. \tag{6.18}$$

After some manipulations we finally arrive to

$$K\boldsymbol{u} = K(1-b)\sum_{i\in I}\boldsymbol{\xi}_i h_i(\boldsymbol{u}) + \boldsymbol{L} - (1-b)\boldsymbol{L}\sum_{i\in I} h_i(\boldsymbol{u}) \tag{6.19}$$

which depends on the form of the function $h_i(\boldsymbol{u})$. When using the standard form of the equilibrium function (see Eq. 2.23), h_i is a second degree equation on $\boldsymbol{u}$.

We note that in the D2Q9 case, Eqs. 6.15 and 6.17 are quite simple to solve, as it is shown in the next section. In opposition, the same equations for the D3Q27 model form a system of non–linear equations with no analytical solution. It may be solved effectively using an iterative method, as the Newton method.

Once we solve these equations, we are in measure to compute $f_i^{eq}(\rho, \boldsymbol{u})$ which is noted simply f_i^{eq} in the next equations.

Finally, we reconstruct the missing populations in both coarse and fine grids using the following formulas

$$f_{I,f} = f_I^{eq} + b(f_{I,c} - f_I^{eq}) = (1-b)f_I^{eq} + bf_{I,c} \tag{6.20}$$

$$f_{II,c} = f_{II}^{eq} + \frac{1}{b}(f_{II,f} - f_{II}^{eq}) = (1-\frac{1}{b})f_{II}^{eq} + \frac{1}{b}f_{II,f} \tag{6.21}$$

$$f_{III,c} = f_{III}^{eq} + \frac{1}{b}(f_{III,f} - f_{III}^{eq}) = (1-\frac{1}{b})f_{III}^{eq} + \frac{1}{b}f_{III,f} \tag{6.22}$$

6.1.2 Implementation and validation

In this section, we rework the example of the 2D Poiseuille flow with Re = 100 presented in Sec. 4.5. We use the same set-up, namely a coarse grid on the left connected to a fine grid in the right. This time we ensure that there is no overlapping region between our grids. The set-up is shown in Fig. 6.2.

One interface node is represented as in Fig. 6.1. We obtain the following three sets that divide the different indexes

$$I = \{1, 2, 3\}$$

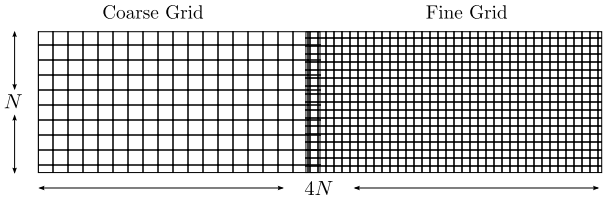


Fig. 6.2: Set-up for the 2D Poiseuille flow.

$$\begin{aligned} II &= \{5, 6, 7\} \\ III &= \{0, 4, 8\}. \end{aligned} \tag{6.23}$$

We rewrite the system presented in Eqs. 6.12 and 6.13 for the D2Q9 model. Let us write $\boldsymbol{u} = (u_x, u_y)$ and $\boldsymbol{L} = (L_x, L_y)$, then we obtain

$$\rho = K + (1-b)(\frac{1}{6}\rho - \frac{1}{2}u_x\rho + \frac{1}{2}\rho u_x^2) \tag{6.24}$$

$$u_x\rho = L_x + (1-b)(-\frac{1}{6}\rho + \frac{1}{2}u_x\rho - \frac{1}{2}\rho u_x^2) \tag{6.25}$$

$$u_y\rho = L_y + (1-b)(\frac{1}{6}u_y\rho - \frac{1}{2}\rho u_y u_x), \tag{6.26}$$

where K has the exact same form as in Eq. 6.16 and $\boldsymbol{L}$ the one in Eq. 6.18.

It is possible to compute u_x by combining Eqs 6.24 and 6.26. One obtains

$$0 = L_x(5+b) - K(b-1) + 3(L_x(b-1) - K(b+1))u_x + 3(b-1)(L_x - K)u_x^2. \tag{6.27}$$

We conclude that u_x is one of the roots of the second order polynomial

$$u_x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}. \tag{6.28}$$

with

$$a = 3(b-1)(L_x - K) \tag{6.29}$$

$$b = 3(L_x(b-1) - K(b+1)) \tag{6.30}$$

$$c = L_x(5+b) - K(b-1). \tag{6.31}$$

We note that given the form of $h_i(\boldsymbol{u})$, our system provides two roots. In our particular example, we know that the Poiseuille velocity is positive, so we have chosen the positive root of Eq. 6.28. However, in a more general set-up, we must choose the root that ensures the continuity of the velocity. In order to remain completely local, we can use the velocity at the preceding time step.

Once we have u_x, we replace it in Eq. 6.24 to compute ρ. Finally replacing both values in Eq. 6.26, we obtain the value of u_y.

We have to note that a spatial interpolation is required in this algorithm. However, the quantities to be interpolated are not the same as in the algorithm presented in Chap. 4, which were ρ, $\boldsymbol{u}$ and all the non–equilibrium distributions f_i^{neq}. In this particular algorithm, in a fine grid node which does not have a corresponding coarse node, there are no f_I populations. Therefore, we need to interpolate these populations from the neighbours, obtaining $\bar{f}_I$. Then, we use these $\bar{f}_I$ populations to solve Eqs. 6.15 and 6.17 for this particular site.

We depict an example of this situation for the population f_5 in Fig. 6.3. Once again, as it was discussed in Subsec. 4.4.1, we can choose between the two possible spatial 1D interpolations.

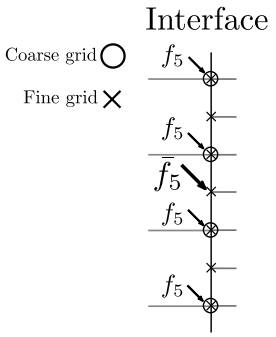


Fig. 6.3: $\bar{f}_5$ may be interpolated from the known f_5 on the coarse grid.

As we are using a convective scaling for the physical quantities of the two grids, it is necessary to perform a temporal interpolation as well. Let us name the spatial resolution of the coarse grid δt_c and the one on the fine grid $\delta t_f = \delta t_c/2$. The populations $f_{I,c}(\boldsymbol{x}, t + \delta t_f)$ are unknown, so in order to calculate it we use the known populations at time t and time $t + \delta t_c$ in a linear interpolation

$$f_{I,c}(\boldsymbol{x}, t + \delta t_f) = \frac{1}{2}\left[f_{I,c}(\boldsymbol{x}, t) + f_{I,c}(\boldsymbol{x}, t + \delta t_c)\right]. \tag{6.32}$$

The algorithm has been implemented in Palabos. The basic iteration of the system consists of the following parts

1. at time t_0 both grids are complete
2. collide and stream on the coarse grid, save populations at $t_1 = t_0 + \delta t_c$
3. spatial interpolation of populations $f_{I,c}$ on the coarse grid at times t_0 and t_1

4. temporal interpolation of the populations $f_{I,c}$ at time $t_{1/2} = t_0 + \delta t_f$
5. collide and stream on the fine grid
6. complete the I populations on the fine grid. For the populations coming from the coarse grid use populations at time $t_{1/2}$
7. collide and stream on the fine grid
8. complete the I populations on the fine grid using coarse populations at time t_1
9. complete the II and III populations on the coarse grid

Let us analyse the results of the simulation. If we plot the norm of the numerical velocity and compare it to the analytical one, we see a perfect match. We remark that the velocity norm has been measured over a column in the middle of the fine grid (see Fig. 6.2). This result is depicted in Fig. 6.4.

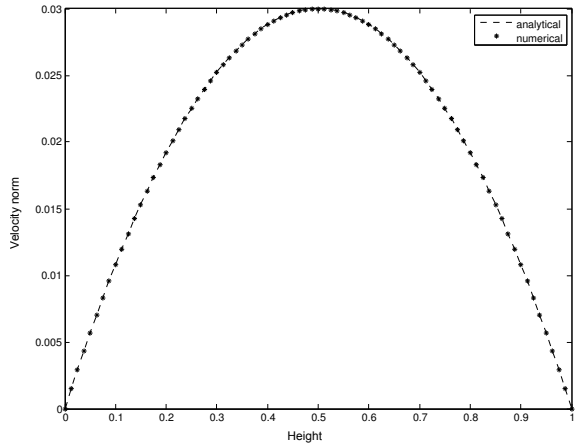


Fig. 6.4: Comparison of the analytical velocity norm and the one on the fine grid.

It is always very important to study the order of convergence of the method for the problem when using the refinement. We observe that the method remains second order accurate as it can be seen on Fig. 6.5 when using the cubic interpolation. In opposition, the linear interpolation does not guarantee that the method remains second order.

Our final result is that the mass remains perfectly constant in the grid transition, as it can be seen in Fig. 6.6, where we plot the density across the channel.

It is interesting to notice that even when using a simple linear interpolation the density seems to behave correctly. We think that the reason of this odd

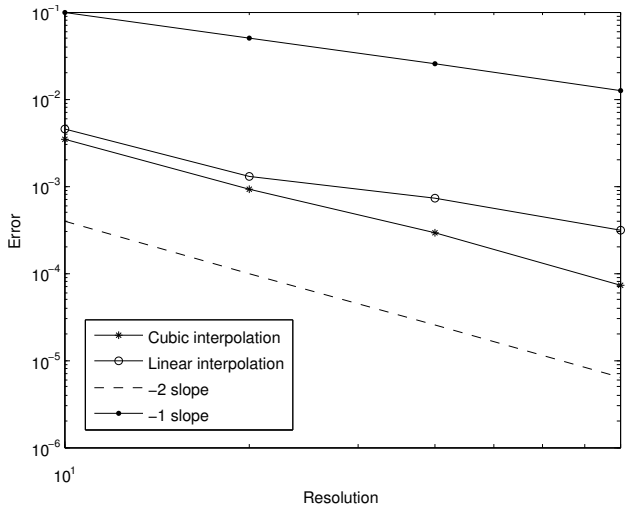


Fig. 6.5: Convergence of the proposed refinement algorithm with linear and cubic interpolation.

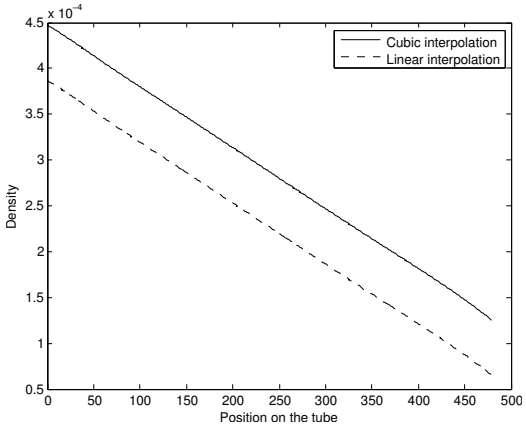


Fig. 6.6: Density for the Poiseuille flow, for cubic and linear interpolation, as a function of position along the tube.

behaviour, compared to the one in Fig. 4.14, may be related to the quantities being interpolated: f_i for the current algorithm, against f_i^{neq} in our generic grid refinement. This topic needs to be clarified in the future. However, as shown before, the spatial linear interpolation remains a bad choice, because it diminishes the order of the method.

6.1.3 Comments

It is necessary to clarify some aspects of our algorithm. First of all, it is interesting to note that if we use our algorithm to couple two grids of the same resolution, which is equivalent to fixing $b = 1$, we obtain the well–known relations for the density and the velocity. For instance, when replacing $b = 1$ in Eq. 6.12, it becomes

$$\rho = \sum_{i \in I} f_i + \sum_{i \in II \cup III} f_i = \sum_{i=0}^{q-1} f_i. \tag{6.33}$$

In the same way, Eq. 6.13 becomes

$$\rho \boldsymbol{u} = \sum_{i \in I} \boldsymbol{\xi}_i f_i + \sum_{i \in II \cup III} \boldsymbol{\xi}_i f_i = \sum_{i=0}^{q-1} \boldsymbol{\xi}_i f_i. \tag{6.34}$$

Given the simplicity of the problem proposed, we have not implemented the filtering operation. It is however possible to include this operation to our algorithm quite easily. The filtering operation should be applied before completing the II and III populations of the coarse grid (step 9. in the proposed algorithm). For the filtering of this populations, we could for instance use a 1D symmetric neighbourhood as shown in Fig. 6.7. The filtering is done over the region marked as F. The non–equilibrium filtered population $\overline{f}_i^{neq}$ can be then expressed by

$$\overline{f}_i^{neq} = \frac{1}{|F|} \sum_{\boldsymbol{x} \in F} f_i^{neq}(\boldsymbol{x}), \tag{6.35}$$

where $|F|$ is the cardinality of the F region.

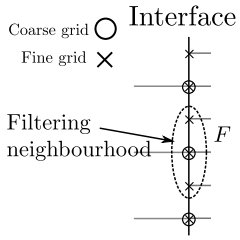


Fig. 6.7: Definition of the region F for the filtering.

The filtering would only be applied over the non–equilibrium parts. We note that we have avoided to filter ρ and $\boldsymbol{u}$ (see Subsec. 4.4.1) because it provokes an artificial local increase of the viscosity. Thanks to this detail, the filtering operation does not change the computation of both these quantities with the system presented in Eq. 6.12 and Eq. 6.13.

Finally, we note that our problem does not have corners. Nevertheless, we remark that the indexes I, II and III do not have a fixed form in the postulation of the algorithm. Therefore, they may be arbitrary, as in the case of a corner.

6.2 A criterion to detect areas to be refined

The problem of locating areas that would require a local finer grid in an algorithmic way can be difficult. We present an automatic process to select portions of the simulation space that are likely to need more grid points.

We believe that an analytical approach is necessary. For instance in all our examples of Chap. 5, we have used previous knowledge of the simulations, because they are well–known examples. In general, this is not something that is known a–priori.

In this section, we start by presenting the theory that supports our criterion. Then we discuss some numerical examples, in order to prove the feasibility of the approach.

6.2.1 Bases and algorithm

We use the fact that when considering the Boltzmann equation and doing a multi-scale Chapman–Enskog analysis (see Chap. 2), we find that each particle density function is composed of an equilibrium and an off-equilibrium part $f = f^{eq} + f^{neq}$. We have supposed that $f^{eq} \gg f^{neq}$. We have also shown that they are related by the Knudsen number Kn so we can safely assert that the following relation holds:

$$\frac{f^{neq}}{f^{eq}} \sim \mathrm{Kn}. \tag{6.36}$$

We recall that the Knudsen number can be computed as

$$\mathrm{Kn} = \frac{\mathrm{Ma}}{\mathrm{Re}}. \tag{6.37}$$

Let us consider a grid G_c with spatial spacing δx_c. We also define a finer grid G_f with spatial step δx_f, defined by the relationship $\delta x_c = \delta x_f / n$. We perform the same simulation over both grids, thus keeping the Reynolds number Re fixed. We recall that $\mathrm{Re} = u_{lb} N / \nu_{lb}$, where both velocity u_{lb} and viscosity ν_{lb} are given in lattice units.

Let us the recall the convective rescale ($\delta t \sim \delta x$). To simulate the same Re in G_c and G_f, u_{lb} is constant and we rescale the viscosity ν_{lb}. As a consequence, the Mach number Ma remains constant for any n. We can conclude that because of

relation 6.37, Kn is also constant for any n. In the following, we are going to work with this particular property.

We propose to use Eq. 6.36 as a mean to measure the quality of the results of a simulation according to a given resolution. For the sake of the automation of the whole process and measures, we postulate the following algorithm

1. Choose a spatial step δx
2. Divide the simulation domain R in m–by–n sub–domains $R_{i,j}$.
3. Simulate R for some time.
4. Compute the Kn number from the relation 6.37.
5. for each sub–domain $R_{i,j}$ of R compute the mean value of the quantity

$$C_{i,j} = \frac{1}{|R_{i,j}|} \sum_{x \in R_{i,j}} \sum_{i=0}^{q-1} \frac{f_i^{neq}}{f_i^{eq}}, \tag{6.38}$$

   where $|R_{i,j}|$ is the number of grid points inside the $R_{i,j}$ region.
6. Compare $C_{i,j}$ to Kn. If $C_{i,j} > \alpha$Kn, for some fixed α, then we deduce that sub–domain $R_{i,j}$ might need a finer grid.

We note that our algorithm depends on the number of regions (m and n) and on the parameter α.

The value of the parameter α is discussed in the numerical results section, but let us note in advance that the quantity $C_{i,j}$ behaves as the velocity gradient, because it is proportional to f^{neq}. Because the LBM has a second order of accuracy, the gradient decreases linearly with the increase of resolution. The direct consequence of this fact is that if we want to insert a grid G_n which is n times finer than the current one on a region $R_{i,j}$, then $C_{i,j}$ is at most n times smaller on $R_{i,j}$ for G_n. This implies that $\alpha \geq \frac{1}{n}$, because we cannot do better with G_n with respect to the original grid.

6.2.2 Numerical results

In this subsection we show the results of the numerical tests that we have performed in order to that for fine grids f^{neq}/f^{eq} gets arbitrarily closer to Kn than in coarse grids.

The first problem that we use is the cavity 2D simulation, a bounded domain, where a velocity is applied over the top lid of the domain. In our particular case, the Reynolds number Re is fixed to 100. The velocity of the top lid of the cavity

is fixed to 0.01 in lattice units. All the other borders of the domain are set to have null velocity. The whole set-up is depicted in Fig. 6.8. In this numerical example we can compute Kn = 0.00017.

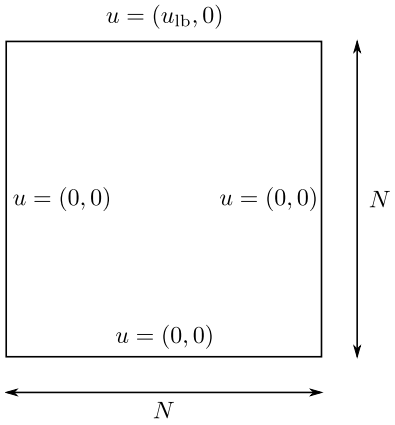


Fig. 6.8: Set-up for the cavity 2D example.

We divide the domain in several regions $R_{i,j}$. For instance for 5 lines and 5 columns, which is our test case, these regions are depicted in Fig. 6.9. We perform the measure of the quantity $C_{i,j}$ for $N = 30, 60, 120$ and 240. For each N, we save the biggest $C_{i,j}$, which corresponds to the region where we have theoretically the worst resolution. We observe that the biggest $C_{i,j}$ converges to the computed value of Kn. This fact is shown in Fig. 6.10.

Top lid

$R_{0,0}$	$R_{0,1}$	$R_{0,2}$	$R_{0,3}$	$R_{0,4}$
$R_{1,0}$	$R_{1,1}$	$R_{1,2}$	$R_{1,3}$	$R_{1,4}$
$R_{2,0}$	$R_{2,1}$	$R_{2,2}$	$R_{2,3}$	$R_{2,4}$
$R_{3,0}$	$R_{3,1}$	$R_{3,2}$	$R_{3,3}$	$R_{3,4}$
$R_{4,0}$	$R_{4,1}$	$R_{4,2}$	$R_{4,3}$	$R_{4,4}$

Fig. 6.9: Division of the simulation domain in several regions.

An interesting result is to take the biggest, the mean and the smallest values of $C_{i,j}$ and divide them by Kn. If we analyse the behaviour of these quantities,

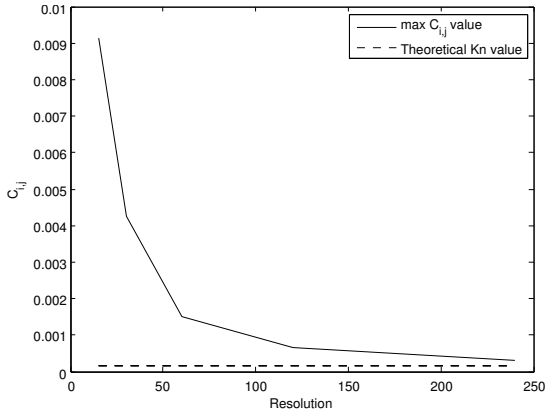


Fig. 6.10: Convergence to Kn for the biggest $C_{i,j}$.

we find out that they behave linearly, as it can be seen in Fig. 6.11. So when doubling the resolution, we expect at most that we are going to reduce each $C_{i,j}$ at most by a half. This gives a clear message for the value of the parameter α in our algorithm. We safely assert that $1/2 \leq \alpha \leq 1$ is the most useful range for this parameter.

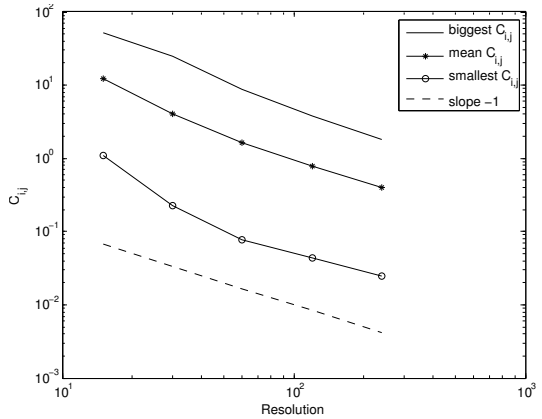


Fig. 6.11: Convergence of several $C_{i,j}$ values divided by Kn.

Without any loss of generality we have chosen to use $\alpha = 1$ in our tests. After the steady state of the cavity 2D is achieved, we save the quantities $C_{i,j}$ for each resolution and we compare them to Kn. The results for the different N can be found in Fig. 6.12, where a white colour means that the region $R_{i,j}$ must be refined ($C_{i,j} >$ Kn) and gray means that the resolution is sufficient. In this case, we see that $N = 15$ is not sufficient anywhere. Then, for bigger values of N we observe

that the white regions decrease, until only the corners are left.

This behaviour is expected, as we know that there is a discontinuity at the corners of the domain in the cavity 2D problem, because of a passage from a non–null velocity to 0 abruptly in just one lattice site. The bigger values of $C_{i,j}$ are found around these areas, as our tests confirm.

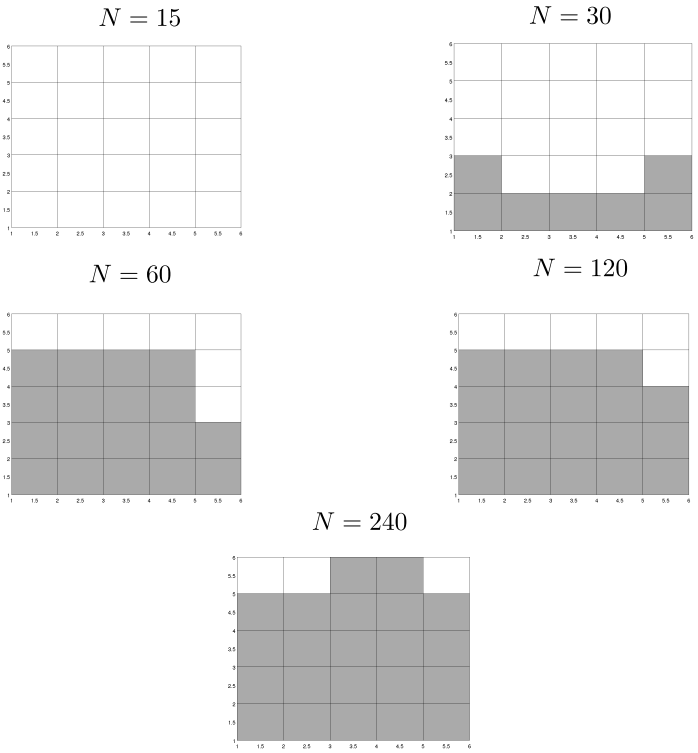


Fig. 6.12: Zones to refine per resolution N according to a 5–by–5 division. White means refine (i.e. $C_{i,j} > \mathrm{Kn}$), while gray stands for the fact that the resolution is sufficient.

There is another utilisation for the fact that $C_{i,j}$ decreases linearly. Let us imagine that we simulate a resolution N, then we compute $C_{i,j}$ over each region. We compute $C_{i,j}/\mathrm{Kn}$ and this actually gives us a numerical value, which, when rounded, can be safely associated to the number of times a grid must be refined to correctly solve a region $R_{i,j}$. If we fix $N = 30$ and we compute this value we obtain the values present in Fig. 6.13. Once again, the biggest values are found in the corners, then in the top lid.

We must end this section by noting that we have only dealt with a steady laminar case to prove that the approach is feasible. The approach should be tested against more difficult cases to have further confirmation of its utility.

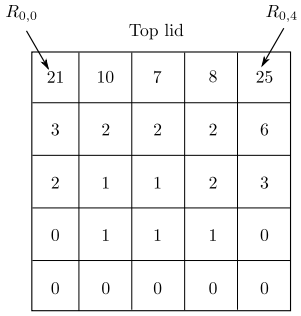


Fig. 6.13: Fixing $N = 30$, how many times each region $R_{i,j}$ must be refined, according to our criterion, in order to provide a good solution.

6.3 Summary

In this chapter we have treated two subjects in direct relation with grid refinement, but that are not used within our generic grid refinement implementation.

The first subject proposes an algorithm to couple two grids of different resolutions with no overlapping zone. This coupling can be solved as if it was a boundary condition. We divide all the populations in three groups, the unknown populations only on the fine grid (I), the unknown populations only on the coarse grid (II) and populations which are known on both grids (III). We stress the fact that this sets do not have a particular form.

We note that we use the rescaling of the off–equilibrium part given by:

$$f_{i,f}^{neq} = b f_{i,c}^{neq}$$
$$b = \frac{\tau_f}{\tau_c}\frac{\delta t_f}{\delta t_c} = \frac{\tau_f}{2\tau_c}$$

After computations, we find that the only unknowns we need to compute at the interface are ρ and $\boldsymbol{u}$ via the non–linear system (Eqs. 6.12 and 6.13)

$$\begin{aligned}\rho =& (1-b)\sum_{i\in I} f_i^{eq}(\rho,\boldsymbol{u}) + b\sum_{i\in I} f_{i,c} + \sum_{i\in II\cup III} f_{i,f}\\ \rho\boldsymbol{u} =& (1-b)\sum_{i\in I} \boldsymbol{\xi}_i f_i^{eq}(\rho,\boldsymbol{u}) + b\sum_{i\in I} \boldsymbol{\xi}_i f_{i,c} + \sum_{i\in II\cup III} \boldsymbol{\xi}_i f_{i,f}.\end{aligned}$$

Once we solve for ρ and $\boldsymbol{u}$, we are able to compute $f_i^{eq}(\rho, \boldsymbol{u})$, which is the same for both grids because of the convective scaling.

We finally proceed to complete the populations using Eqs. 6.20, 6.21 and 6.22

$$
\begin{aligned}
f_{I,f} &= f_I^{eq} + b(f_{I,c} - f_I^{eq}) = (1-b)f_I^{eq} + bf_{I,c} \\
f_{II,c} &= f_{II}^{eq} + \frac{1}{b}(f_{II,f} - f_{II}^{eq}) = (1-\frac{1}{b})f_{II}^{eq} + \frac{1}{b}f_{II,f} \\
f_{III,c} &= f_{III}^{eq} + \frac{1}{b}(f_{III,f} - f_{III}^{eq}) = (1-\frac{1}{b})f_{III}^{eq} + \frac{1}{b}f_{III,f}
\end{aligned}
$$

The second subject in this chapter deals with a criterion to detect regions where we need more resolution. Our criterion is based on the property

$$
f_i^{neq}/f_i^{eq} \sim \mathrm{Kn},
$$

where Kn is the Knudsen number.

We propose an algorithm to divide the domain in several rectangular blocks $R_{i,j}$ where we compute the quantity

$$
C_{i,j} = \frac{1}{|R_{i,j}|} \sum_{x \in R_{i,j}} \sum_{i=0}^{q-1} \frac{f_i^{neq}}{f_i^{eq}},
$$

If $C_{i,j} >$ Kn then we conclude that a refinement is necessary. This criterion is validated on the cavity 2D problem.

Chapter 7

Conclusion and future perspectives

In this thesis we have revisited the theoretical concepts of the grid refinement for the LBM. We have studied and presented in detail all the necessary ingredients for a successful implementation of a multi–domain based grid refinement. We have taken special care in the study of the coupling operations between grids of different levels, which are the core part of the grid refinement. In particular, we have found that they must have some minimal requirements in order ensure numerical stability and preserve the order of accuracy of the method.

First, we have studied the passage from a coarse to a fine grid, where the information needs to be increased for the fine grid sites which do not correspond to a coarse site. In this case, we have shown that a second order interpolation to find the missing values is not sufficient as it results in a mass loss, even in very simple cases, like laminar flows. We have shown that a four-point symmetric cubic interpolation is mandatory, as it ensures that the method remains second order accurate. We note that this type of cubic interpolation was not found in literature. The only reference to cubical interpolation is presented in [80], but using a cubic spline, which requires the first and second derivatives of the quantity to interpolate.

The second novelty that we have proposed concerns the communication from the fine grid to the coarse one. In this case, we must decrease the amount of information communicated. The simplest solution is to perform a copy from the corresponding fine site to the coarse grid. However, we have shown that this direct copy is incompatible with high Reynolds number flow simulations, causing numerical noise to appear on the boundaries between grids. The reason of this behaviour is that the fine grid is able to resolve smaller scales, which the coarse grid cannot handle. The solution that we have proposed is to use a filtering operation, as it is done in traditional numerical methods. The filtering operation is a low pass filter that removes the high frequencies. In practice, we have chosen to perform a mean operation on the non–equilibrium part of the particle density functions in a fixed neighbourhood on the fine grid, prior to the copy of these quantities to the coarse grid. After testing, we have found that this filtering should not be applied

to the density and velocity, as it results into an artificial viscosity increase, which is far too important and provokes unwanted behaviours.

In order to apply the theoretical concepts and prove their accuracy, we have made a generic implementation of our grid refinement. We have chosen the Palabos open–source library as building block for our code.

Our implementation has been tested over a number of 2D and 3D problems. For every test, we have compared the relevant results of a uniform grid with a refined grid. We have evidenced a good agreement between the results on both grids. We have also tried to compare the reduction of computational cost.

Among the four examples that we have simulated, we must specially emphasise the dipole problem in 2D and the turbulent jet in 3D. Both of these problems are challenging, as high velocity gradients go through the interfaces. Nevertheless, our implementation has shown to be robust and accurate on both cases. These set-ups have also aided us to underline the importance of the filtering operation, as the simulations were numerically unstable when it was not used.

To end the thesis, we have presented two subjects related with grid refinement. First, an algorithm to couple two grids without an overlapping zone. Then, we have proposed a criterion to detect zones of a simulation which might need a local finer grid. Both subjects have been tested over simple set-ups. We have studied them aside of the generic implementation, but we are persuaded that they might be very promising for the future.

The grid refinement is far from being a closed chapter in the LBM. In what follows, we present some future directions for the research concerning the grid refinement.

In our opinion, the next step for the generic grid refinement is to find a more automatic way to detect zones for refinement, in order to create the simulation domains. This step is currently performed manually on our code. We think that in this direction, our criterion for the detection of non–resolved areas might be useful. It should be extended and generalized for more complex problems.

When executing a code in parallel over a big quantity of computational cores, we wish that all of them have an equivalent workload. In computer science, this is known as the load balancing problem. The load balancing is known to be an intractable problem, but there exist some heuristic methods to find solutions to this problem. A non–uniform grid introduces additional constraints to the load balancing problem, namely the difference of workload for different refinement levels and the difference of temporal discretization between the levels. The heuristic methods used before need to be adapted to take into account those new constraints. This is another future direction for the research on grid refinement for the LBM, as performance is one of the capital interests behind non–uniform grids. One of the possible answers is to use solutions for methods such as the finite differences

(see [57] for example).

Bibliography

[1] *The Thrombus project.*

[2] D. Alemani, J. Buffle, B. Chopard, and J. Galceran, *Study of three grid refinement methods in the lattice boltzmann framework for reactive–diffusive processes*, International Journal of Modern Physics C, 18 (2007), pp. 722–731.

[3] D. Alemani, B. Chopard, J. Galceran, and J. Buffle, *LBGK method coupled to time splitting technique for solving reaction-diffusion processes in complex systems*, Physical Chemistry Chemical Physics, 7 (2005), pp. 3331–3341.

[4] D. Alemani, B. Chopard, J. Galceran, and J. Buffle, *Time splitting and grid refinement methods in the lattice boltzmann framework for solving a reaction-diffusion process*, in Computational Science - Iccs 2006, Pt 2, Proceedings, V. N. Alexandrov, G. D. VanAlbada, and P. M. A. Sloot, eds., vol. 3992, Springer-Verlag Berlin, Berlin, 2006, pp. 70–77.

[5] F. Beekhof and G. Berti, *A framework for geometric modeling based on physical fields*, tech. rep., University of Geneva and CMM, 2011.

[6] P. L. Bhatnagar, E. P. Gross, and M. Krook, *A model for collision processes in gases. i. small amplitude processes in charged and neutral one-component systems*, Phys. Rev., 94 (1954), pp. 511–525.

[7] B. M. Boghosian, P. J. Love, P. V. Coveney, I. V. Karlin, S. Succi, and J. Yepez, *Galilean-invariant lattice-boltzmann models with h theorem*, Physical Review E, 68 (2003), p. 025103.

[8] S. Chapman and T. G. Cowling, *The mathematical theory of nonuniform gases*, Cambridge University Press, Cambridge, 1960.

[9] H. Chen, *Volumetric formulation of the lattice boltzmann method for fluid dynamics: Basic concept*, Physical Review E, 58 (1998), pp. 3955–3963.

[10] H. Chen, O. Filippova, J. Hoch, K. Molvig, R. Shock, C. Teixeira, and R. Zhang, *Grid refinement in lattice boltzmann methods based on volumetric formulation*, Physica A: Statistical Mechanics and its Applications, 362 (2006), pp. 158–167.

[11] Y. Chen, Q. Kang, Q. Cai, and D. Zhang, *Lattice boltzmann method on quadtree grids*, Physical Review E, 83 (2011), p. 026707.

[12] B. Chopard and M. Droz, *Cellular automata modeling of physical systems*, Cambridge University Press, June 2005.

[13] H. Clercx and C. Bruneau, *The normal and oblique collision of a dipole with a no-slip boundary*, Computers & Fluids, 35 (2006), pp. 245–279.

[14] B. Crouse, E. Rank, M. Krafczyk, and J. Tölke, *A LB-based approach for adaptive flow simulations*, International Journal of Modern Physics B, 17 (2003), pp. 109–112.

[15] D. d'Humières, *Multiplerelaxationtime lattice boltzmann models in three dimensions*, Philosophical Transactions of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences, 360 (2002), pp. 437–451.

[16] D. d'Humières, P. Lallemand, and U. Frisch, *Lattice gas models for 3D hydrodynamics*, Europhysics Letters (EPL), 2 (1986), pp. 291–297.

[17] H. Dixit and V. Babu, *Simulation of high rayleigh number natural convection in a square cavity using the lattice boltzmann method*, International Journal of Heat and Mass Transfer, 49 (2006), pp. 727–739.

[18] A. Dupuis and B. Chopard, *Theory and applications of an alternative lattice boltzmann grid refinement algorithm*, Physical Review E, 67 (2003), p. 066707.

[19] EXA, *powerflow*, 2012.

[20] O. Filippova and D. Hänel, *Grid refinement for Lattice-BGK models*, Journal of Computational Physics, 147 (1998), pp. 219–228.

[21] O. Filippova, B. Schwade, and D. Hänel, *Multiscale lattice boltzmann schemes for low mach number flows*, Philosophical Transactions of the Royal Society of London Series a-Mathematical Physical and Engineering Sciences, 360 (2002), pp. 467–476.

[22] O. Filippova, S. Succi, F. Mazzocco, C. Arrighetti, G. Bella, and D. Hänel, *Multiscale lattice boltzmann schemes with turbulence modeling*, Journal of Computational Physics, 170 (2001), pp. 812–829.

[23] M. Geier, A. Greiner, and J. Korvink, *Cascaded digital lattice boltzmann automata for high reynolds number flow*, Physical Review E, 73 (2006).

[24] S. Geller, M. Krafczyk, J. Tölke, S. Turek, and J. Hron, *Benchmark computations based on lattice-Boltzmann, finite element and finite volume methods for laminar flows*, Computers & Fluids, 35, pp. 888–897.

[25] S. Geller, S. Uphoff, and M. Krafczyk, *Turbulent jet computations based on MRT and cascaded lattice boltzmann models*, arXiv:1206.0389, (2012).

[26] M. Griebel, T. Dornsheifer, and T. Neunhoeffer, *Numerical Simulation in Fluid Dynamics: A Practical Introduction*, Society for Industrial and Applied Mathematics, illustrated edition ed., Dec. 1997.

[27] Z. Guo, C. Zheng, and B. Shi, *An extrapolation method for boundary conditions in lattice Boltzmann method*, Phys. Fluids, 14 (2002), pp. 2007–2010.

[28] X. He, *Error analysis for the interpolation-supplemented lattice-boltzmann equation scheme*, International Journal of Modern Physics C, 08 (1997), pp. 737–745.

[29] X. He and L.-S. Luo, *Lattice Boltzmann model for the Incompressible Navier–Stokes Equation*, J. Stat. Phys., 88 (1997), pp. 927–944. 10.1023/B:JOSS.0000015179.12689.e4.

[30] X. He and L.-S. Luo, *Theory of the lattice boltzmann method: From the boltzmann equation to the lattice boltzmann equation*, Physical Review E, 56 (1997), pp. 6811–6817.

[31] X. He, L.-S. Luo, and M. Dembo, *Some progress in lattice boltzmann method. part i. nonuniform mesh grids*, Journal of Computational Physics, 129 (1996), pp. 357–363.

[32] S. Hou, J. Sterling, S. Chen, and G. D. Doolen, *A lattice boltzmann subgrid model for high reynolds number flows*, arXiv:comp-gas/9401004, (1994).

[33] K. Huang, *Statistical Mechanics*, J. Wiley, New York, 1987.

[34] T. A. Johnson and V. C. Patel, *Flow past a sphere up to a reynolds number of 300*, Journal of Fluid Mechanics, 378 (1999), pp. 19–70.

[35] M. Junk, A. Klar, and L.-S. Luo, *Asymptotic analysis of the lattice boltzmann equation*, Journal of Computational Physics, 210 (2005), pp. 676–704.

[36] C. Körner, M. Thies, T. Hofmann, N. Thürey, and U. Rüde, *Lattice boltzmann model for free surface flow for modeling foaming*, Journal of Statistical Physics, 121 (2005), pp. 179–196.

[37] L. D. Landau, E. M. Lifshitz, and E. M. Lifshits, *Fluid mechanics*, Butterworth-Heinemann, Jan. 1987.

[38] J. Latt, *Hydrodynamic limit of lattice boltzmann equations*, (2007).

[39] J. Latt, *The Palabos project*, June 2010.

[40] J. Latt and B. Chopard, *Lattice boltzmann method with regularized non-equilibrium distribution functions*, arXiv:physics/0506157, (2005).

[41] ——, *A benchmark case for lattice Boltzmann: Turbulent dipole-wall collision*, International Journal of Modern Physics C, 18 (2007), p. 619.

[42] J. Latt, B. Chopard, O. Malaspinas, M. Deville, and A. Michler, *Straight velocity boundaries in the lattice boltzmann method*, Physical Review E, 77 (2008), p. 056703.

[43] T. Lee and C.-L. Lin, *A stable discretization of the lattice boltzmann equation for simulation of incompressible two-phase flows at high density ratio*, Journal of Computational Physics, 206 (2005), pp. 16–47.

[44] Z. Lu, Y. Liao, D. Qian, J. McLaughlin, J. Derksen, and K. Kontomaris, *Large eddy simulations of a stirred tank using the lattice boltzmann method on a nonuniform grid*, Journal of Computational Physics, 181 (2002), pp. 675–704.

[45] O. Malaspinas and P. Sagaut, *Consistent subgrid scale modelling for lattice boltzmann methods*, Journal of Fluid Mechanics, 700 (2012), pp. 514–542.

[46] O. P. Malaspinas, *Lattice Boltzmann method for the simulation of viscoelastic fluid flows*, EPFL, Lausanne, 2009.

[47] R. Mei, D. Yu, W. Shyy, and L. Luo, *Force evaluation in the lattice Boltzmann method involving curved geometry*, Physical Review E, 65 (2002), p. 041203.

[48] S. Melchionna, *A model for red blood cells in simulations of large-scale blood flows*, Macromolecular Theory and Simulations, 20 (2011), p. 548561.

[49] F. Muggli, L. Chatagny, and J. Latt, *Lattice boltzmann method for the simulation of laminar mixers*, tech. rep., 2012.

[50] F. Nannelli and S. Succi, *The lattice boltzmann equation on irregular lattices*, Journal of Statistical Physics, 68 (1992), pp. 401–407.

[51] A. Parmigiani, *Lattice Boltzmann calculations of reactive multiphase flows in porous media*, University of Geneva, Geneva, 2011.

[52] T. Pohl, M. Kowarschik, J. Wilke, K. Iglberger, and U. Ruede, *Optimization and profiling of the cache performance of parallel lattice boltzmann codes*, Parallel Processing Letters, 13 (2003), pp. 549–560.

[53] S. B. Pope, *Turbulent Flows*, Cambridge University Press, Aug. 2000.

[54] K. N. Premnath and J. Abraham, *Discrete lattice BGK boltzmann equation computations of transient incompressible turbulent jets*, arXiv:physics/0503208, (2005). Int. J. Mod. Phys. 15, 699-719 (2004).

[55] K. N. Premnath, M. J. Pattison, and S. Banerjee, *Dynamic subgrid scale modeling of turbulent flows using Lattice-Boltzmann method*, Physica A, (2009), p. 318.

[56] P. Quéméré, P. Sagaut, and V. Couailler, *A new multidomain/multiresolution method for largeeddy simulation*, International Journal for Numerical Methods in Fluids, 36 (2001), pp. 391–416.

[57] C. A. Rendleman, V. E. Beckner, M. Lijewski, W. Crutchfield, and J. B. Bell, *Parallelization of structured, hierarchical adaptive mesh refinement algorithms*, Computing and Visualization in Science, 3 (2000), pp. 147–157.

[58] P. Reymond, F. Merenda, F. Perren, D. Rüfenacht, and N. Stergiopulos, *Validation of a one-dimensional model of the systemic arterial tree*, American Journal of Physiology - Heart and Circulatory Physiology, 297 (2009), pp. H208–H222.

[59] M. Rheinländer, *A consistent grid coupling method for lattice-boltzmann schemes*, Journal of Statistical Physics, 121 (2005), pp. 49–74.

[60] D. Ricot, S. Marié, P. Sagaut, and C. Bailly, *Lattice Boltzmann method with selective viscosity filter*, J. Comp. Phys., 228 (2009), pp. 4478–4490.

[61] M. Rohde, D. Kandhai, J. J. Derksen, and H. E. A. van den Akker, *A generic, mass conservative local grid refinement technique for lattice-Boltzmann schemes*, International Journal for Numerical Methods in Fluids, 51 (2006), pp. 439–468.

[62] I. L. Ryhming, *Dynamique des fluides: Un cours de base du deuxième cycle universitaire*, PPUR presses polytechniques, 2004.

[63] P. Sagaut, S. Deck, and M. Terracol, *Multiscale And Multiresolution Approaches in Turbulence*, Imperial College Press, June 2006.

[64] M. Schönherr, K. Kucher, M. Geier, M. Stiebler, S. Freudiger, and M. Krafczyk, *Multi-thread implementations of the lattice boltzmann method on non-uniform grids for CPUs and GPUs*, Computers & Mathematics with Applications, 61 (2011), pp. 3730–3743.

[65] X. Shan and H. Chen, *Lattice boltzmann model for simulating flows with multiple phases and components*, Physical Review E, 47 (1993), pp. 1815–1819.

[66] X. Shan, X.-F. Yuan, and H. Chen, *Kinetic theory representation of hydrodynamics: a way beyond the NavierStokes equation*, Journal of Fluid Mechanics, 550 (2006), pp. 413–441.

[67] P. A. Skordos, *Initial and boundary conditions for the lattice boltzmann method*, Physical Review E, 48 (1993), pp. 4823–4842.

[68] S. Succi, *The lattice Boltzmann equation for fluid dynamics and beyond*, Oxford University Press, Oxford, 2001.

[69] J. Tölke, S. Freudiger, and M. Krafczyk, *An adaptive scheme using hierarchical grids for lattice boltzmann multi-phase flow simulations*, Computers & Fluids, 35 (2006), pp. 820–830.

[70] J. Tölke and M. Krafczyk, *Second order interpolation of the flow field in the lattice boltzmann method*, Computers and Mathematics with Applications, 58 (2009).

[71] J. Tölke, M. Krafczyk, and E. Rank, *A multigrid-solver for the discrete boltzmann equation*, Journal of Statistical Physics, 107 (2002), pp. 573–591.

[72] S. Turek and M. Schaefer, *Recent benchmark computations of laminar flow around a cylinder*, (1996).

[73] S. Ubertini, G. Bella, and S. Succi, *Lattice boltzmann method on unstructured grids: Further developments*, Physical Review E, 68 (2003), p. 016701.

[74] S. Ubertini and S. Succi, *Recent advances of lattice boltzmann techniques on unstructured grids*, Progress in Computational Fluid Dynamics, an International Journal, 5 (2005), pp. 85–96.

[75] C. D. Ungate, D. R. F. Harleman, and G. H. Jirka, *Stability and mixing of submerged turbulent jets at low reynolds numbers*, technical report, MIT Energy Lab, 1975. Originally presented as the first author's thesis (M.S.), Temperature reduction in a submerged vertical jet in the laminar-turbulent transition, M.I.T. Dept. of Civil Engineering.

[76] H. K. Versteeg and W. Malalasekera, *An Introduction to Computational Fluid Dynamics: The Finite Volume Method*, Pearson Education, Feb. 2007.

[77] D. A. Wolf-Gladrow, *Lattice-gas cellular automata and lattice Boltzmann models: an introduction*, Springer, 2000.

[78] H. Xi, G. Peng, and S. H. Chou, *Finite-volume lattice boltzmann method*, Physical review. E, Statistical physics, plasmas, fluids, and related interdisciplinary topics, 59 (1999), pp. 6202–6205. PMID: 11969609.

[79] ——, *Finite-volume lattice boltzmann schemes in two and three dimensions*, Physical review. E, Statistical physics, plasmas, fluids, and related interdisciplinary topics, 60 (1999), pp. 3380–3388. PMID: 11970153.

[80] D. Yu, R. Mei, L. Luo, and W. Shyy, *Viscous flow computations with the method of lattice boltzmann equation*, Progress in Aerospace Sciences, 39 (2003), pp. 329–367.

[81] O. C. Zienkiewicz, R. L. Taylor, and P. Nithiarasu, *The Finite Element Method for Fluid Dynamics*, Butterworth-Heinemann, Dec. 2005.

[82] Q. Zou and X. He, *On pressure and velocity boundary conditions for the lattice boltzmann BGK model*, Physics of Fluids, 9 (1997), p. 1591.

Walter Daniel Lagrava Sandoval

9, rue Sismondi
CH-1201 Genève
Suisse

Tel.: +41 22 731 48 32 (home)
+41 77 488 51 79 (mobile)

e-mail: daniel.lagrava@unige.ch

Date of birth : 4 April 1981
Nationality : Bolivian
Languages : Spanish, French, English, German

Education

11/2006 Master in Computer Science – University of Geneva, CH
12/1998 High School Diploma – Col. Pestalozzi, Sucre, Bolivia

Employments

08/08 - 07/12 Teaching Assistant
University of Geneva, CH

02/07 - 12/07 Teaching Possition
Universidad Catolica San Pablo, Cochabamba, Bolivia

02/06 - 11/06 Internship
CERN, Geneva, CH

Teaching experience

Autumn 2008-11	Introduction aux Algorithmes University of Geneva, CH
Spring 2008	Technologie des Ordinateurs University of Geneva, CH
Spring 2010	Algorithmes Probabilistes University of Geneva, CH
Spring 2009-11	Algorithmes Parallèles University of Geneva, CH

Peer Review

America Physical Society, Journal of Computational Physics.

Publications

Research Articles

[1] Bastien Chopard, Daniel Lagrava: A Cellular Automata Model for Species Competition and Evolution. ACRI 2006: 277-286.

[2] Bastien Chopard, Daniel Lagrava, Jonas Latt, Orestis Malapinas and Rafik Ouared: A Lattice Boltzmann Modeling Of Bloodflow In Cerebral Aneurysms, ECCOMAS 2010, Lisbon.

[3] Daniel Lagrava, O. Malaspinas, Jonas Latt, Bastien Chopard: Advances in multi-domain lattice Boltzmann grid refinement. J. Comput. Physics 231(14): 4808-4822 (2012)

Theses

[1] Daniel Lagrava, Access to Satellite Image Metadata on the Grid, Master Thesis, University of Geneva, 2006

Conference Presentations

Summer 2011 — International Conference for Mesoscopic Methods in Engineering and Science (ICMMES)
Lyon, France

Summer 2010 — Discrete Simulation of Fluid Dynamics (DSFD)
Rome, Italy

Software

- Involved in the development of modules of Palabos, Open-Source C++ Parallel Lattice Boltzmann Solver, `http://www.palabos.org`.

Programming languages and scientific software

C++, C, Java, Python, Fortran, Maple, Matlab, Paraview, SQL, PHP.

Other

09/98 — Bronze medal in the Iberoamerican Chemistry Olympiades
Bogota, Colombia