UNIVERSITAT ROVIRA I VIRGILI

NUMERICAL STUDY OF THE HEAT AND MASS TRANSFER PROCESSES WITH THE LATTICE BOLTZMANN METHOD: LAMINAR MIXED CONVECTION IN A SQUARE OPEN C

Javier Burgos Vergara

Doctoral Thesis

Numerical study of the heat and mass transfer processes with the Lattice Boltzmann method: Laminar Mixed Convection in a Square Open Cavity

Javier Burgos Vergara

Supervised by:

Dr. Ildefonso Cuesta

Dr. Clara Salueña

Universitat Rovira i Virgili

Department of Mechanical Engineering

Tarragona - 2016

UNIVERSITAT
ROVIRA i VIRGILI

I STATE that the present study, entitled:

Numerical study of the heat and mass transfer processes with Lattice Boltzmann Method: Laminar Mixed Convection in a Square Open Cavity.

Presented by **Javier Burgos Vergara** for the award of the degree of Doctor, has been carried out under our supervision at the Department of Mechanical Engineering of this University.

Tarragona, 14 June 2016

Supervised by:

Dr. Ildefonso Cuesta **Dr. Clara Salueña**

Associate Professors of Fluid Mechanics Department of Mechanical Engineering of Universitat Rovira i Virgili. Research group ECOMMFIT - Experimentació, Computació i Modelització en Mecànica de Fluids i Turbulència.

Abstract

The Lattice Boltzmann LBM method has become in tool for studying hydrodynamically and thermally developing, and also chemical reaction and fluid structure interaction among other. Several researchers have aroused great interest of the LBM in all fields of computational fluid dynamic CFD. Due to his easy implementation, variety of the numerical methods could be found in many literature to solver the Navier-Stoker equation. The topic of this thesis is the study of the LBM and implementation in a fluid flow, temperature and concentrations fields, we focus on boundary conditions and accuracy of the method. In order to apply the method, a developed code of Lattice Boltzmann was built. The code was developed using an open-source LBM code and modified for the purpose to simulate the heat and mass transfer phenomena. This thesis is divided in two sections.

(I) Various simulations were carried out using LBM for the different approaches of the method, for two-dimensional numerical simulation case, using the D2Q9-model Multiple Relaxation Time model (MRT), this model was implemented for the fluid flow, and temperature concentration field. Three different case for laminar flow were studied in order to validate the method, first stage a Hagen-Poiseuille flow in a channel and a pipe were simulate, the results were compared with the exact solution of the Poiseuille-flow equations. As well as, the Lid-driven cavity for a incompressible laminar flow was executed for a set of Reynolds number, Re = 100, 1000 and 7500, obtained good result by comparing with other numerical method, such as U.Ghia et al [1] at 1982 presented a benchmark of the Lid-driven cavity for High Reynolds number using a multi-grid method. Moreover, convection-diffusion problem of a Gaussian pulse, Natural convection in a closed cavity and mass flux rate in a flat plate were performed with the objective to analyze the boundary condition of LBM implement for passive scalar concentration, these results were compared with previous research [2, 3, 4, 5, 6].

(II) Second stage, the main objective of this thesis was the study of the steady and unsteady laminar flow in a channel with open cavity and heated bottom wall. A two-dimensional simulation has been carried out for the mixed convective flow, using a D2Q9 model for the flow and temperature fields. MRT-model with a Boussinesq approximation equation were applied, and it obtained a good accuracy and stability. LBM is compared against results obtained by ANSY-Fluent software for validation. Temperature, velocity and Nusselt number, calculated with TLBM presented very well agreement in the range of Reynolds and Richardson numbers studied, i.e. 50$\leqslant$Re$\leqslant$1000 and 0.01$\leqslant$Ri$\leqslant$10. The observations indicate that the effect of the buoyancy force is negligible for Ri$\leqslant$0.1, for all values of the Reynolds number considered. For Ri$\geqslant$4 and Re>200, buoyancy effects are important causing the development of the upstream secondary vortex and the stratification of the flow into two main recirculating cells. For high enough Ri, the recirculation is no longer encapsulated, therefore the flow becomes unsteady, and an oscillatory instability develops. According to the simulation results, observed from Re = 500, Ri = 10. The analysis of the unsteady regime reveals a very rich phenomenology where the geometry of the problem couples with the oscillatory thermal instability. This regime is characterized by the periodic emission of pairs of vortices generated from the upper downstream vertex of the square cavity, and pseudo-periodic variations of the Nusselt number which persist at least up to Re = 1500, while the two main vortices remain in the cavity. The observations extend previous studies and shed a new light on the characteristics of the oscillatory instability and the role of the Reynolds and Richardson numbers.

Acknowledgements

This study was financially supported the Spanish Ministry of Science and Technology under project reference CTQ2013-46799, Spain. I would like to thank my advisors Ildefonso Cuesta and Clara Salueña for their guidance during these years and for their understanding and comprehension, thank you so much for all the helpful information and knowledge on the field of numerical methods. I have learned many things during these long years. I would also like to thank Professor Gunther Brenner and my co-worker Ma Yong of the TU-Clausthal University of Technology in Germany, for their useful comments and valuable suggestions about the method during my research stay in the spring of 2014.

A mis compañeros del grupo, Albert, Jorge, Jonathan, David, Miguel, en especial a Xavier Inglès y Manuel Martínez por todas las sugerencias y conocimientos en ANSYS-FLUENT, a los compañeros nuevos de grupo, Jordi y Tony, a Carlos Moreno por esos cafelitos y conversaciones por la mañana para levantar el ánimo y a todos mis amigos que he conocido en Tarragona en todos estos años.

A mi familia que aunque se encontraba lejos siempre estaban animándome y dándome fuerza para seguir adelante y poder concluir mi período en Tarragona.

Symbols

f_i	Density distribution function
f_i^{eq}	Equilibrium density distribution function
g_i	Internal energy distribution function
g_i^{eq}	Equilibrium internal energy distribution function
A_i^{σ}	Concentration distribution function of species
$A_i^{\sigma eq}$	Equilibrium Concentration distribution function of species
A_i^{A}	Concentration distribution function of species A
$A_i^{A,\ eq}$	Equilibrium Concentration distribution function of species A
A_i^{B}	Concentration distribution function of species B
$A_i^{B,\ eq}$	Equilibrium Concentration distribution function of species B
τ	Dimensionless relaxation time for momentum
τ_g	Dimensionless relaxation time for energy
τ_σ	Dimensionless relaxation time for concentration of species
τ_A	Dimensionless relaxation time for concentration of species A
τ_B	Dimensionless relaxation time for concentration of species B
c_s	Lattice speed of sound,$=c/\sqrt{3}$
$\mathbf{c}_i$	Discrete velocity directions
ω_i	Weights of LBM,($i = 0 - 8$; D2Q9), ($i = 0 - 18$; D3Q19), lu
c	Speed on the Lattice, $=\Delta x/\Delta t$, Lattice space and time step size
t_{lbm}	Characteristic time in lattice, lu
ν_{lbm}	Kinematic viscosity, Lattice units, lu
α_{lbm}	Thermal diffusivity, Lattice units, lu
u_{lbm}	Lattice characteristic velocity, Lattice units, lu

C^{σ}	Lattice concentration of species, Lattice units, lu
C^{A}	Lattice concentration of species A, Lattice units, lu
C^{B}	Lattice concentration of species B, Lattice units, lu
$\mathcal{D}^{\sigma}$	Lattice mass diffusivity of species, Lattice units, lu
$\mathcal{D}^{A}$	Lattice mass diffusivity of species A, Lattice units, lu
$\mathcal{D}^{B}$	Lattice mass diffusivity of species B, Lattice units, lu
Δx	Discrete space, Lattice units, lu
Δt	Discrete time, Lattice units, lu
N	Number of nodes
$\mathbf{u}$	Macroscopic velocity, lu
T_{lbm}	Local temperature, Lattice units, lu
$T_{\infty,lbm}$	Reference temperature, Lattice units, lu
$T_{h,lbm}$	hot temperature, Lattice units, lu
$T_{c,lbm}$	cold temperature, Lattice units, lu
$\mathbf{\Omega}$	Collision matrix operator, MRT model
$\mathbf{S}$	Matrix S, MRT model
$\mathbf{M}$	Matrix M, MRT model
$\mathbf{m}$	components of the momentum vector, MRT model
ϕ	Scalar variables (energy, mass and momentum)
l_0	Characteristic length, m, cm
t_0	Characteristic time, s
u_0	Characteristic velocity, $cm/s - m/s$
ν	kinematic viscosity, m^2/s
α	Thermal diffusivity, m^2/s
T_0	Local temperature, K
T_{∞}	Reference temperature, K
$T_{0,cold}$	cold temperature, K
$T_{0,hot}$	hot temperature, K
D_f	Diffusion coefficient, m^2/s
Re	Reynolds number, $= u_0 l_0/\nu$

Ra	Rayleigh number, $=Gr \cdot Pr{=}g\beta(Th - T_\infty)l_0^3/\nu\alpha$
Pr	Prandtl number, $=\nu/\alpha$
Gr	Grashof number, $= g\beta(Th - T_\infty)l_0^3/\nu^2$
Ri	Richardson number, $= Gr/Re^2 = g\beta(Th - T_\infty)l_0/u_0^2$
Nu	Local Nusselt number, $= -\frac{L}{\Delta T}\frac{\partial T}{\partial y}\mid_{wall}$
$\overline{\mathrm{Nu}}$	Average Nusselt number, $= 1/L\int_0^L Nudx$
F_i	External force, $= 3\rho\omega_i g\beta(T_0 - T_\infty)$
D	Dimensionless diameter
u_d	Dimensionless characteristic velocity
T	Dimensionless temperature
L	Dimensionless length of the domain
θ	Dimensionless temperature in the mid plane of the cavity
$\boldsymbol{D}$	Dimensionless diffusion coefficient
C	Dimensionless concentration
U_x	Dimensionless x-comp. of the velocity - vertical mid plane
U_y	Dimensionless y-comp. of the velocity - horizontal mid plane

Contents

List of Figures

List of Tables

A mi familia y amigos, los de siempre y los que he conocido en estos largos años…

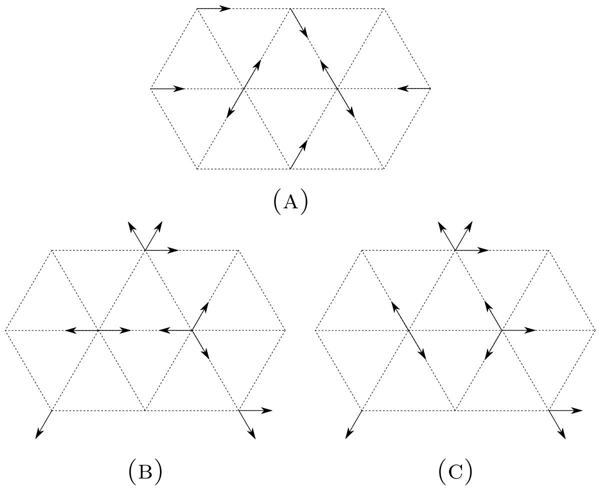
Chapter 1

Introduction

In the last decade the Lattice Boltzmann Method (LBM) has become a powerful alternative to finite element and finite volume methods, for solving different problems and applications in various engineering fields such as different fluid flow cases, transport problems for single and multiphase flow, heat and mass transfer, compressible flows, porous media, magnetodynamics, chemical reaction and biomedical simulation [8].

The LBM evolved from the lattice gas automata (LGA) [9, 10], and was first proposed as a model by von Neumann in the late 1940's as cellular automata (CA) [11]. LGA was developed as a simple, fully discrete microscopic model for a fluid flow based on "fictitious" particles occupying a regular lattice point in space, which can collide with similar neighbors. Afterwards, certain mass- and momentum-conserving collision rules are applied in the cell, updating the velocities of the colliding particles. The important feature of the LGA is that mass and momentum are explicitly conserved, unlike the CA. This proved to be useful to simulate real physical problems. It can be shown that the summation of the microdynamical mass and momentum equations are asymptotically equivalent to the Navier-Stokes equations for incompressible flows [12].

Let us consider fictitious particles inside of a hexagonal lattice, see Fig. 1.1, moving together with a determined velocity and particular position, Fig. 1.1(A). In each generation the particles move in only one direction in the lattice, Fig. 1.1(B); the dispersion of the particles caused by motion makes that these particles change the direction in the lattice Fig. 1.1(C), if these particles arrived to the same site, they can collide with each other, which can disperse the particles and change their velocity. However, the collisions cannot change the total number of particles and mass and momentum are conserved.


(A)

(B)

(C)

FIGURE 1.1: 6-Speed hexagonal lattice, FHP model

Henceforth, the LGA is constructed as a simplified, fictitious molecular dynamics in which space, time and particle velocities are all discrete. Composed of Boolean variable $n_i(\vec{x},t)$, where $i = 0, ..., M$ represent occupation of the the particles, M means the number of directions of the particles velocities at each node of the lattice. The evolution of the equation can be described for the LGA in the following form,

$$n_i(x + \mathbf{c}_i\delta t, t+1) = n_i(x,t) + \Omega_i(n_i(x,t)), \qquad i = 0, ..., M \tag{1.1}$$

where $n_i(x,t) = 0$ or 1 representing the number of particles moving with discrete velocity $\mathbf{c}_i$ at node x at time t, δt is the time step. Ω_i is the collision operator representing the influence of the particle collisions. For simplicity, no more that one particle is allowed at a given time and node with a given velocity [13]. For LBM, the distinctive feature is to replace the particles n_i by a single particle distribution function $f_i = \langle n_i \rangle$, where $\langle n_i \rangle$ means an ensemble average [14]. In LBM, the primitive variables are the averaged particle distributions, which are now mesoscopic variables, being the kinetics still the same as in the LGA, and the advantages of locality in the kinetic approach are retained.

In this thesis, a LBM code is written using a well-stablished methodology in order to study numerically heat and mass transfer processes inside a cavity in laminar mixed convection regime, in two and three dimensions, and the results are compared with the traditional methodology used in computational fluid dynamics. In the last decade, several authors have implemented the LBM to simulate engineering problems, such as heat exchange, catalysis reaction, multiphase flow, advection-diffusion problems among others [15, 16, 17]. Particularly, G. Barrios et al. [3] and Kuen-Hau Lin et al. [18], implemented the LBM to study the heat transfer in a cavity with a partially heated wall under natural convection and a complex geometry. Moreover, Almalowi et at. [15] carried out a study on the laminar developing flow in a channel, obtaining excellent results for the momentum and thermal field. These results were compared with the ANSYS-FLUENT software, obtaining a good agreement, indicating that the LBM is an effective computational fluid dynamic tool to study non-isothermal flow problems.

In order to demonstrate the applicability of our LBM code for non-isothermal and isothermal flows, different benchmark cases will be analyzed to validate the code for momentum, heat and mass transfer phenomena. These results will be compared with those obtained with different methods (e.g., Finite Volume method, Finite difference method, etc). For some cases, the exact solution is available, or found in the literature [2, 19, 20]. Finally, a study of laminar mixed convection in an open cavity, in both steady and unsteady state will be carried out. The results obtained with our LBM code are compared with those obtained using the ANSYS-FLUENT software. They complement the knowledge available in the literature of such problems with new data, like the development of secondary vortices with and the onset of unsteady flow and the accompanying mechanism of vortex ejection [21].

In the following sections, all the physical problems chosen for the validation cases and their numerical impementation will be described. The details of the LBM implementation are also described: in the LBM, different boundary condition schemes can be implemented depending on the geometry of the problem. Likewise, different approaches can be applied to the force term (i.e. the buoyancy force term used in natural and mixed convection) [22]. The results obtained will be compared with those available in the literature, and their validation will be analyzed in detail.

1.1 Objectives of the thesis

As pointed out above, the main objective of this thesis is the design and implementation of the Lattice Boltzmann code to study the steady and unsteady laminar flow in a channel with an open square cavity heated at the bottom. The effect of the buoyancy force in combination with the external flow in the channel reveals the generation of interesting structures. In order to achieve this goal, some important tasks need to be examined before the implementation. These are the following,

- Development of a Fortran code for the implementation of the LB equations for momentum, heat and mass transfer processes.
- Implementation of the LBM and its different approaches: appropriate boundary condition schemes, and most relevant geometrical configurations in two and three dimensions.
- Analysis of the incompressible regime for laminar fluid flow.
- Adjustment and testing of the accuracy and stability of the solver for simulation of problems with heat and mass transfer processes.
- Generation of the numerical mesh and the boundaries for each case studied.
- Validation and comparison of the results obtained in each case with bibliographic data.

1.2 Document Structure

The present document is organized as follows:

1. Chapter 2 introduces the Lattice Boltzmann Method and its basic concepts, models, boundary schemes and stability conditions, and the most relevant equations.

2. Chapter 3 shows an analysis and implementation of the LBM equation for the SRT (Single Relaxation Time) and MRT (Multiple Relaxation Time) models. Also, the stability of the method is analyzed for the three following cases: Two dimensional Hagen–Poiseuille flow in a channel, Lid-Driven cavity flow and three dimensional Poiseuille flow, which is used to validate the LBM for incompressible fluid flow.

3. In Chapter 4 the LBM is applied in three different problems in order to validate the model used for temperature and concentration fields. First, a Gaussian pulse is used to simulate and observe the advection-diffusion effect. Then, natural convection in a cavity is considered and their results compared with those in the literature. Finally, a mass transfer problem on a flat plate is implemented (Léveque problem).

4. Chapter 5 shows the application of the Thermal Lattice Boltzmann method to a laminar mixed convection problem. The results obtained were compared with ANSYS-FLUENT software, analyzing the agreement obtained for the different variables.

5. Chapter 6 present a general conclusion for the different problems used for validation, and a detailed discussion of the main case of study completes the thesis.

Chapter 2

Lattice Boltzmann Method - LBM

As explained in Chapter 1, LBM is being used more frequently to simulate complex fluid flow problems. It was first developed in the 90's as the lattice gas automaton to solve the Navier-Stokes Equation [9, 10]. Currently, it is one of the most applied methods in the field of computational fluid dynamics (CFD). The LBM is focused on the average macroscopic behaviour. Solving the kinetic equation provides many of the advantages of molecular dynamics (MD), including clear physical pictures, easy implementation of boundary conditions, and fully parallel algorithms (OpenMP)[23].

Furthermore, there are different methods currently used for transport equations (e.g., heat, mass, momentum) that can be simulated on different scales. On a macroscopic scale, partial differential equations (PDE) like Navier-Stokes equations (NS) are used. These equations are difficult to solve analytically due to non linearity, complex geometry and boundary conditions. With the help of numerical schemes like the finite difference method (FDM), the finite volume method (FVM), the finite element method (FEM), the system of PDEs is converted into a system of algebraic equations.These equations are used to solve iteratively until convergence is achieved of the results.

On the other side, the Hamilton's equations is applied in MD on small particles (e.g., atoms, molecules) where these particles collide with each other, the problem of this method is to track the location and velocity of each particle (i.e., its trajectory), as well as the amount of data required to compute macroscopic variables (i.e., temperature, pressure, thermophysical properties). The main idea of LBM is to bridge the gap between micro-scale and macro-scale by not considering the individual details of each one of the particles, but the behavior of the collection of particles as a unit [24], in Fig. 2.1 we can observe the different scales applied in the different methods: LBM is located in the meso-scale as mentioned above.

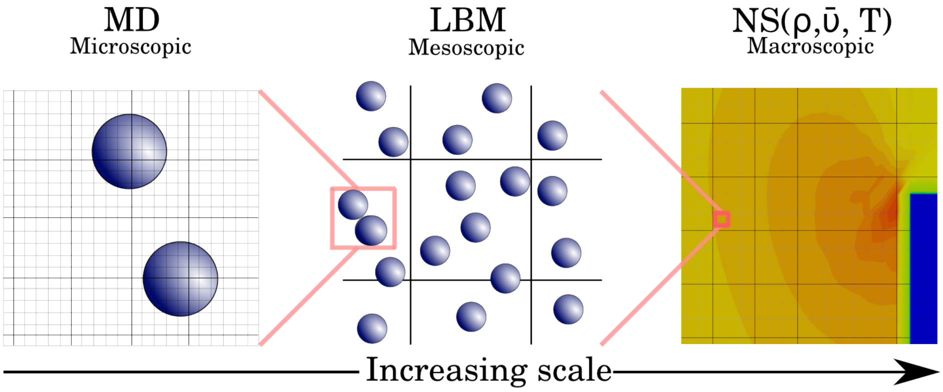


FIGURE 2.1: Illustration of open cavity with boundary conditions.

2.1 Lattice Boltzmann Equation

The LBM is based on the discretization of the Boltzmann equation in time and space, so that the set of discretized velocity distribution functions $f_i(x,t)$ evolves propagating from node to node in every single time step. Specifically, LBM is performed in two main steps: during the propagation step, the particles are forced to move along some prescribed lattice directions, while the collision step imposes thermalization with some characteristic time scale after one mean free path.

McNamara and Zanetti [25] used the Lattice Boltzmann method for the first time as an alternative to LGA in 1988. Later, other researchers made different contributions with the same goal: to simplify the collision term.

Higuera and Jimenez et. al. [26], made in 1989 an important simplification, linearizing the collision operator by assuming that the distribution function is close to the local equilibrium state.

2.1.1 The collision operator

A simple linearized version of the collision operator, which relaxes towards the local equilibrium using a Single Relaxation Time, is the Bhatnagar-Gross-Krook collision operator (BGK) [27], subsequently used by several authors [28, 29]. This has become the most commonly form used in LBM at present [8, 25, 30, 31].

The collision operator is as follows,

$$\Omega = \omega(f^{eq} - f) = \frac{1}{\tau}(f^{eq} - f) \tag{2.1}$$

where $\omega = 1/\tau$ is the collision frequency, τ is the relaxation time and f^{eq} is an approximation of the Maxwellian equilibrium distribution function.

The BGK model with Single Relaxation Time (SRT), which is commonly used in the LBM, is given by

$$\underbrace{f_i(x + \mathbf{c}_i\Delta t, t + \Delta t) - f_i(x,t)}_{\text{Streaming}} = \underbrace{\frac{1}{\tau}\left[f_i^{eq}(x,t) - f_i(x,t)\right]}_{\text{Collision}} \tag{2.2}$$

where f_i ($i = 0, ...8$, in the case of a two-dimensional model such as the D2Q9 that will be explained in more details on the next subsection) denotes the discrete velocity distribution function, and $\mathbf{c}_i$ the corresponding discrete velocities. Some of these models will be explained in the next paragraph.

Equation (2.2) shows a linear relaxation equation where the right side represents the collision term of the LBM, whereas the left side accounts for advection or streaming:

- Streaming: each particle moves to the nearest node in the direction of its velocity

- Collision: particles arrive at a node and interact by changing their velocity directions according to the scattering rules mentioned in the Chapter. 1.

2.2 Lattice Models

There are different velocity discretizations for the LBM, responding to different configuration models for the lattice domain. For one dimensional simulations, D1Q2, D1Q3 and D1Q5 are frequently used. For two-dimensional simulations D2Q5 and D2Q9 are the most common models [24]. In this work we consider D2Q9 for momentum and temperature and concentration grids. For three dimensional simulations, different models such as LBM, D3Q15, D3Q19 and D3Q27 are commonly applied. We opted for D3Q19 for this work. In the next subsections, the models used for the different simulations are explained.

2.2.1 Two Dimensional

2.2.1.1 D2Q9-model

The main two-dimensional model for LBM, D2Q9 has eight velocity vector issued from the central nodes. This choice is very common in the LBM, used to solve problems of fluid flow and heat transfer. The number of velocity vectors (the choice of discretization) imposes different weights in the discrete velocity distribution functions at equilibrium, coming from the discretization and expansion of the Maxwellian solution $f_i^{eq}(x,t)$.

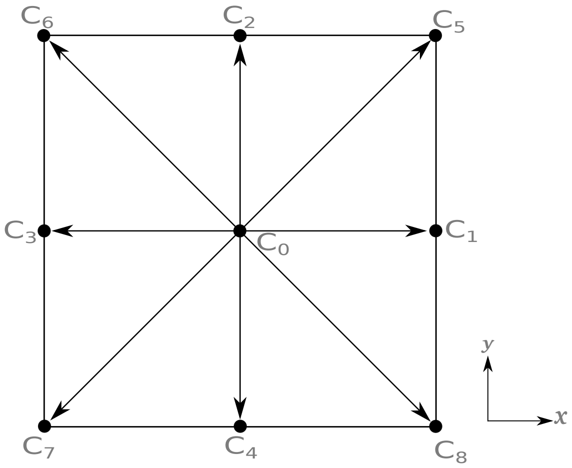


FIGURE 2.2: D2Q9, Lattice velocities

Figure 2.2, shows the geometry of the D2Q9 lattice, with the set of velocity vectors $\mathbf{c}_i$ used for the discretization, the components of which are shown in Table. 2.1.

TABLE 2.1: Lattice velocities for the D2Q9 model, $\mathbf{c}_i$, $i = 0, 8$

	$\mathbf{c}_0$	$\mathbf{c}_1$	$\mathbf{c}_2$	$\mathbf{c}_3$	$\mathbf{c}_4$	$\mathbf{c}_5$	$\mathbf{c}_6$	$\mathbf{c}_7$	$\mathbf{c}_8$
u_x	0	1	0	-1	0	1	-1	-1	1
u_y	0	0	1	0	-1	1	1	-1	-1

The local equilibrium distribution function f^{eq} is the key element of the method as the f_i relax towards f_i^{eq} according to Eq. (2.2). The derivation of the local equilibrium distribution f^{eq} of the standard D2Q9 model can be found in [10], which leads to the following D2Q9 discretized probability densities [12].

$$f_i^{eq} = \omega_i \rho \left[1 + 3 \frac{\mathbf{c}_i \cdot \mathbf{u}}{c^2} + \frac{9}{2} \frac{(\mathbf{c}_i \cdot \mathbf{u})^2}{c^4} - \frac{3}{2} \frac{\mathbf{u} \cdot \mathbf{u}}{c^2} \right], \tag{2.3}$$

where ρ is the local material density, $\mathbf{u}$ is the macroscopic velocity, and $c = \Delta x / \Delta t$, being Δx and Δt the lattice space and the lattice time step size, which are set to unity. The weights ω_i and the discrete velocities $\mathbf{c}_i$ are given by

$$\omega_i = \begin{cases} 4/9, & i = 0 \\ 1/9, & i = 1,2,3,4 \\ 1/36, & i = 5,6,7,8 \end{cases} \tag{2.4}$$

2.2.2 Three-Dimensional

For three-dimensional problems, the most common models used are D3Q15, D3Q19 and D3Q27. From these models, the D3Q15 model entails a faster calculation. However, this advantage impacts the precision in the result. For simulations in which there is no need of a high accuracy, the D3Q15 can be used for quick implementation to simulate 3-D problems [32].

Figure 2.3 shows the speeds of the D3Q15 model, and Table 2.2 their respective weight factors for the local equilibrium distribution function f^{eq}.

2.2.2.1 D3Q15-model

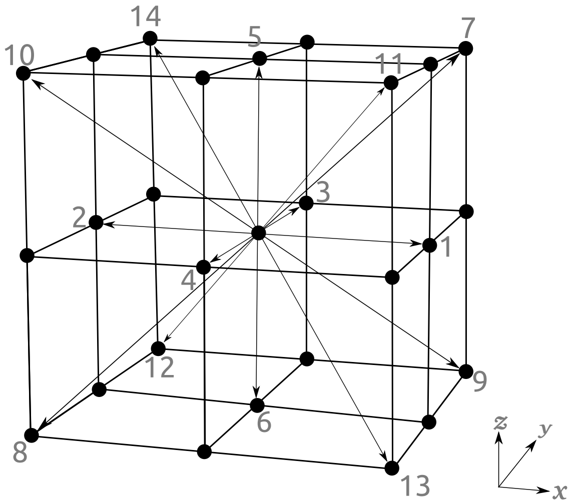


FIGURE 2.3: D3Q15- Lattice velocities vectors

TABLE 2.2: Lattice velocities for D3Q15 model, $\mathbf{c}_i$, $i = 0, 14$ with their respective speeds for 3-D simulations

	$\mathbf{c}_0$	$\mathbf{c}_1$	$\mathbf{c}_2$	$\mathbf{c}_3$	$\mathbf{c}_4$	$\mathbf{c}_5$	$\mathbf{c}_6$	$\mathbf{c}_7$	$\mathbf{c}_8$	$\mathbf{c}_9$	$\mathbf{c}_{10}$	$\mathbf{c}_{11}$	$\mathbf{c}_{12}$	$\mathbf{c}_{13}$	$\mathbf{c}_{14}$
u_x	0	1	-1	0	0	0	0	1	-1	1	-1	1	-1	1	-1
u_y	0	0	0	1	-1	0	0	1	-1	1	1	-1	1	-1	1
u_z	0	0	0	0	0	1	-1	1	-1	-1	1	1	-1	-1	1

$$\omega_i = \begin{cases} 16/72, & i = 0 \\ 8/72, & i = 1-6 \\ 1/72, & i = 7-14 \end{cases} \tag{2.5}$$

2.2.2.2 D3Q19-model

D3Q19 has 19 velocity vectors, with speed zero on the central vector. Table 2.3 shows the different weights of the model and the velocity components for each direction.

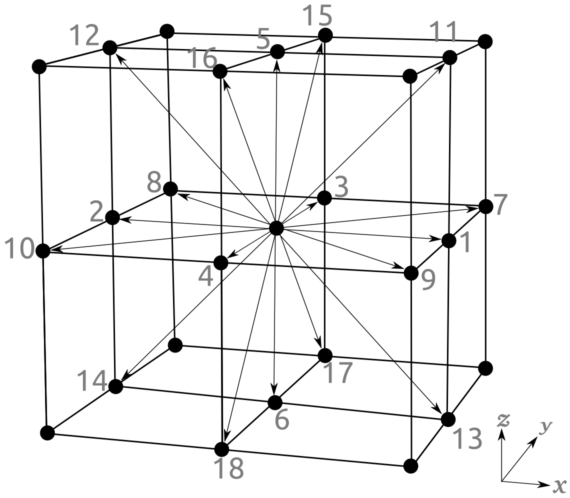


FIGURE 2.4: D3Q19- Lattice velocities vectors

TABLE 2.3: The lattice velocity for D3Q19 model, $\mathbf{c}_i$, $i = 0, 18$

	$\mathbf{c}_0$	$\mathbf{c}_1$	$\mathbf{c}_2$	$\mathbf{c}_3$	$\mathbf{c}_4$	$\mathbf{c}_5$	$\mathbf{c}_6$	$\mathbf{c}_7$	$\mathbf{c}_8$	$\mathbf{c}_9$	$\mathbf{c}_{10}$	$\mathbf{c}_{11}$	$\mathbf{c}_{12}$	$\mathbf{c}_{13}$	$\mathbf{c}_{14}$	$\mathbf{c}_{15}$	$\mathbf{c}_{16}$	$\mathbf{c}_{17}$	$\mathbf{c}_{18}$
u_x	0	1	-1	0	0	0	0	1	1	-1	-1	1	-1	1	-1	0	0	0	0
u_y	0	0	0	1	-1	0	0	1	-1	1	-1	0	0	0	0	1	1	-1	-1
u_z	0	0	0	0	0	1	-1	0	0	0	0	1	1	-1	-1	1	-1	1	-1

$$\omega_i = \begin{cases} 12/36, & i = 0 \\ 2/36, & i = 1-6 \\ 1/36, & i = 7-18 \end{cases} \tag{2.6}$$

2.2.2.3 D3Q27-model

Together with D3Q15 and D3Q19, the D3Q27 model is one of the most commonly applied LBM models in three dimensions [33]. This model is characterized by a higher precision with respect to the other models, since more degrees of freedom are incorporated as the number of velocity directions is increased. However, this has a direct effect on the computational time, total number of iterations and the increasing complexity of the coding. Figure 2.5 and Table 2.4 show the configuration, velocity vectors and the weight coefficients of the equilibrioum distribution for this model.

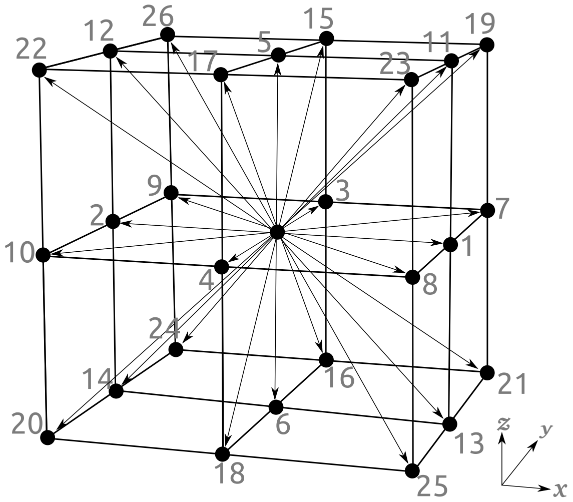


FIGURE 2.5: D3Q27- Lattice velocity vectors

TABLE 2.4: The lattice velocity for D3Q27 model, $\mathbf{c}_i$, $i = 0, 26$

	$\mathbf{c}_0$	$\mathbf{c}_1$	$\mathbf{c}_2$	$\mathbf{c}_3$	$\mathbf{c}_4$	$\mathbf{c}_5$	$\mathbf{c}_6$	$\mathbf{c}_7$	$\mathbf{c}_8$	$\mathbf{c}_9$	$\mathbf{c}_{10}$	$\mathbf{c}_{11}$	$\mathbf{c}_{12}$	$\mathbf{c}_{13}$	$\mathbf{c}_{14}$
u_x	0	1	-1	0	0	0	0	1	1	-1	-1	1	-1	1	-1
u_y	0	0	0	1	-1	0	0	1	-1	1	-1	0	0	0	0
u_z	0	0	0	0	0	1	-1	0	0	0	0	1	1	-1	-1

$\mathbf{c}_{15}$	$\mathbf{c}_{16}$	$\mathbf{c}_{17}$	$\mathbf{c}_{18}$	$\mathbf{c}_{19}$	$\mathbf{c}_{20}$	$\mathbf{c}_{21}$	$\mathbf{c}_{22}$	$\mathbf{c}_{23}$	$\mathbf{c}_{24}$	$\mathbf{c}_{25}$	$\mathbf{c}_{26}$
0	0	0	0	1	-1	1	-1	1	-1	1	-1
1	1	-1	-1	1	-1	1	-1	1	-1	-1	1

$$\omega_i = \begin{cases} 8/327, & i = 0 \\ 2/27, & i = 1 - 6 \\ 1/216, & i = 7 - 14 \\ 1/54, & i = 15 - 26 \end{cases} \tag{2.7}$$

In this Ph.D thesis, the D2Q9 model is used for two-dimensional problems for momentum, temperature and concentration fields: this model is the most commonly applied in these problems, due to its simple implementation as well as a quick convergence. For three-dimensional problems the

D3Q19 model has been chosen due to its good accuracy with less computational cost with respect to D3Q15 and D3Q27, respectively. Also, the D3Q19 has been proven to provide good results.

2.3 Macroscopic Properties

For a better understanding, let us imagine a regular uniform spatial grid (i.e., a Lattice) as shown in Fig. 2.6. At each node, a number of distribution functions $f_i(x,t)$ is defined, each associated to a given velocity vector c_i. These represent the density of particles with velocity c_i at the location x, at time t. These distribution functions have all the information on the macroscopic properties (e.g., density, velocity, temperature, concentration).

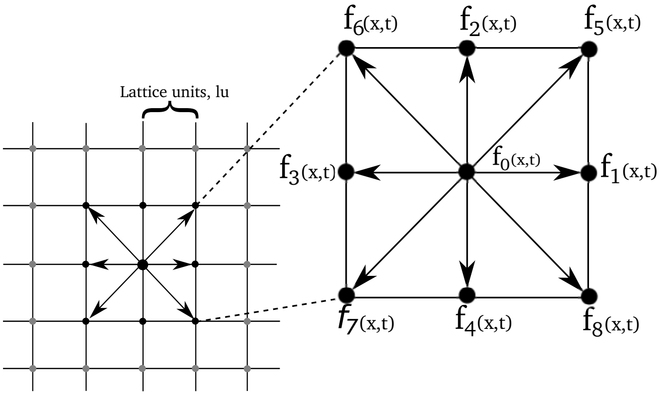


FIGURE 2.6: Scheme of the lattice grid, example of D2Q9 model

Therefore, $f_i(x,t)$ can be used to compute all the macroscopic variables, such as density ρ and velocity $\mathbf{u}$, calculated as follows:

$$\rho(x,t) = \sum_{i=0}^{8} f_i(x,t), \tag{2.8}$$

$$\rho\mathbf{u}(x,t) = \sum_{i=0}^{8} \mathbf{c}_i f_i(x,t), \tag{2.9}$$

Just as the distribution function of the momentum, energy and mass transfer have their respective distribution functions, g_i and A_i

$$g_i(x + \mathbf{c}_i\Delta t, t + \Delta t) = g_i(x,t) + \frac{1}{\tau_g}\left[g_i^{eq}(x,t) - g_i(x,t)\right], \tag{2.10}$$

$$A_i^\sigma(x + \mathbf{c}_i\Delta t, t + \Delta t) = A_i^\sigma(x,t) + \frac{1}{\tau_\sigma}\left[A_i^{\sigma eq}(x,t) - A_i^\sigma(x,t)\right], \tag{2.11}$$

Also, the temperature T and concentration C^σ are defined through the averages with the corresponding discrete distributions, g_i and A_i,

$$T_{lbm}(x,t) = \sum_{i=0}^{8} g_i(x,t), \tag{2.12}$$

$$C^\sigma(x,t) = \sum_{i=0}^{8} A_i^\sigma(x,t). \qquad \sigma = (a, b, c..), \tag{2.13}$$

where the index σ refers to the different species. Similarly to Eq. (2.3), a set of discrete temperature distribution functions are defined, for which their local equilibrium counterparts read

$$g_i^{eq} = \omega_i T_{lbm}\left[1 + \frac{\mathbf{c}_i\cdot\,\mathbf{u}}{c^2}\right]. \tag{2.14}$$

For the concentration of the species σ, the local equilibrium distribution functions $A_i^{\sigma eq}$ are defined by

$$A_i^{\sigma eq} = \omega_i C^\sigma\left[1 + \frac{\mathbf{c}_i\cdot\,\mathbf{u}}{c^2}\right], \tag{2.15}$$

where the weights ω_i and discrete velocities $\mathbf{c}_i$ are given as presented in Sec. 2.2, for the different two and three dimensional models discussed.

The relaxation times τ control the rates approaching equilibrium for the momentum, temperature and concentration fields. The kinematic viscosity ν, the thermal diffusivity α and the mass diffusivity $\mathcal{D}^\sigma$ are directly related to the relaxation time parameters τ, τ_g, τ_σ for each lattice grid. These parameters control the stability of the method and are defined in terms of their respective relaxation times [3, 18]. The macroscopic viscosity, thermal diffusion coefficient and mass difussivity are calculated as:

$$\nu_{lbm} = \left(\tau - \frac{1}{2}\right) c_s^2 \Delta t, \tag{2.16}$$

$$\alpha_{lbm} = \left(\tau_g - \frac{1}{2}\right) c_s^2 \Delta t, \tag{2.17}$$

$$\mathcal{D}^{\sigma} = \left(\tau_{\sigma} - \frac{1}{2}\right) c_s^2 \Delta t. \qquad \sigma = (a, b, c..), \tag{2.18}$$

where $c_s = c/\sqrt{3}$, is the lattice speed of sound.

2.4 Stability Conditions

The main parameter to ensure stability in the LBM is the value of the relaxation time τ, which plays a important role in the collision step. This parameter is fixed during the initialization process and is directly linked to the lattice kinematic viscosity ν_{lbm}, as explained in Section. 2.3. Therefore, one has to be careful while discretizing the system because the value of τ cannot be changed once fixed [34].

Two more parameters are important to ensure a correct accuracy of the method, which are the Mach number Ma $= u_0/c_s$ and the characteristic field velocity u_0. The Lattice Boltzmann Method simulates incompressible flow under low Mach numbers, so that the Mach number should be set within a small enough range (i.e., Ma$\leqslant$ 0.3) to maintain the incompressibility condition of the flow and ensure both an accurate solution and stability. Therefore the characteristic velocity u, must range between $u_0 \leqslant 0.1$ and $u_0 \geqslant 0$. The Reynolds number Re in dimensional/dimensionless system units has to be equal to the Reynolds number in lattice units Re$_{lbm}$. Therefore, Re $=u_0 \cdot l_0/\nu_0 = \text{Re}_{lbm} = u_{lbm} \cdot N/\nu_{lbm}$, where u_0 is the characteristic velocity of the system in dimensional units, and l_0 is the characteristic length of the macroscopic scale and ν_0 is the kinematic viscosity of the fluid. Then, u_{lbm} is the velocity in lattice, N is the number of node in the characteristic length or height of the computational domain and ν is the macroscopic viscosity in lattice. The following section presents an example to clarify the discretization of the dimensional system in a real case.

As mentioned above, the relaxation time parameter τ is the most important parameter and must be carefully set in the optimal range to keep a good stability and accuracy of the simulations. Commonly, this value is set based on the range $0.53 \leqslant \tau \leqslant 1.80$. This is due to the fact that for Single-Relaxation-Time models, the method becomes unstable when $\tau \leqslant 0.53$. One way to increase this value is to, for example, modify the parameters related to the viscosity of the lattice ν_{lbm} (e.g., Reynolds number, $\mathrm{Re}_{lbm} = u_{lbm} \cdot N/\nu_{lbm}$). Modifying some of the parameters one can increase the value of ν_{lbm} and respectively the value of τ because these tw parameters are directly related considering that $c_s = c/\sqrt{3}$, $c = \Delta x/\Delta t$ where $\Delta x = 1$ and $\Delta t = 1$. Therefore, Eq. (2.16) can be reformulated as $\nu_{lbm} = (\tau - 1/2) \cdot 1/3$. So, by isolating τ from the former equation, we have:

$$\tau = 3 \cdot \nu_{lbm} + \frac{1}{2} \tag{2.19}$$

Another alternative could be used when we find these ranges of relaxation time under $\tau \leqslant 0.53$. Other relaxation time models can be implemented in the Lattice Boltzmann equation. In this thesis it is implemented the Multiple Relaxation time model (MRT) [35, 16, 36, 37], ensure a good accuracy and stability of the method in this range of τ. This model is further explained in the next section.

2.5 MRT-Multiple Relaxation Time

2.5.1 MRT-D2Q9 model

The Single Relaxation Time (SRT-BGK) is the most common scheme used in LBM for two and three-dimensional simulations due to its simplicity, but this simplicity leads to the two problems: a possible lack of acuracy in some cases when the correct implementation of boundary conditions is used, and the inability to simulate problems at a high Prandtl number, Reynolds, Richardson, Rayleigh and Schmidt numbers among others. In this case, the LBM can overcome these deficiencies using the Multiple Relaxation Time (MRT) approach [37]. The collision operator is thus modeled by a linear relaxation process with multiple relaxation parameters, which is carried out in the moment space as first proposed by d'Humières. Therefore we have opted here for the Multiple-Relaxation Lattice Boltzmann scheme [24, 37, 16], where Eq. (2.2) is used as follows,

$$f_i(x + \mathbf{c}_i \Delta t, t + \Delta t) - f_i(x, t) = -\mathbf{\Omega} \left[f_i^{eq}(x, t) - f_i(x, t) \right], \qquad (2.20)$$

where $\mathbf{\Omega}$ is the collision matrix[24]. In the D2Q9 model, the matrix operator $\mathbf{\Omega}$ can be expressed through the diagonal matrix $\mathbf{S}$ of the inverse of the relaxation times for the following 9-component moment vector,

$$\mathbf{m} = (\rho, e, \epsilon, j_x, q_x, j_y, q_y, p_{xx}, p_{xy})^T \qquad (2.21)$$

which conform a moment space of macroscopic quantities,

- Density (0^th order) : ρ
- Energy (2^th order) : e
- Energy-square (4^th order) : ϵ
- Momentum in x and y direction (1^th order) : j_x, j_y
- Heat flow in x and y direction (3^th order) : q_x, q_y
- Diagonal and off-diagonal stress(2^th order) : $p_{xx,xy}$

obtained as averages with the velocity distribution functions which serve as a basis for the linear transformation,

$$\mathbf{m} = \mathbf{M} \cdot \mathbf{f}, \qquad \mathbf{f} = \mathbf{M}^{-1} \cdot \mathbf{m}, \qquad (2.22)$$

leading to the MRT-D2Q9 LBM model,

$$f_i(x + \mathbf{c}_i \Delta t, t + \Delta t) - f_i(x, t) = -\mathbf{M}^{-1} \cdot \mathbf{S} \cdot \left[\mathbf{m}(x, t) - \mathbf{m}^{eq}(x, t) \right] \qquad (2.23)$$

where the nine components of the equilibrium moment vector $\mathbf{m}^{eq}$ read

$$\begin{aligned}
m_0^{eq} &= \rho \\
m_1^{eq} &= -2\rho + 3\left(j_x^2 + j_y^2\right) \\
m_2^{eq} &= \rho - 3\left(j_x^2 + j_y^2\right) \\
m_3^{eq} &= j_x \\
m_4^{eq} &= -j_x \\
m_5^{eq} &= j_y \\
m_6^{eq} &= -j_y \\
m_7^{eq} &= \left(j_x^2 + j_y^2\right) \\
m_8^{eq} &= j_x + j_y
\end{aligned} \tag{2.24}$$

Here j_x and j_y are the components of the linear momentum

$$j_x = \rho \mathbf{u}_x = \sum_{i=0}^{8} f_i^{eq} \mathbf{c}_{ix} \qquad j_y = \rho \mathbf{u}_y = \sum_{i=0}^{8} f_i^{eq} \mathbf{c}_{iy} \tag{2.25}$$

Finally, the matrix $\mathbf{M}$ and its inverse are

$$\mathbf{M} = \begin{pmatrix}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
-4 & -1 & -1 & -1 & -1 & 2 & 2 & 2 & 2 \\
4 & -2 & -2 & -2 & -2 & 1 & 1 & 1 & 1 \\
0 & 1 & 0 & -1 & 0 & 1 & -1 & -1 & 1 \\
0 & -2 & 0 & 2 & 0 & 1 & -1 & -1 & 1 \\
0 & 0 & 1 & 0 & -1 & 1 & 1 & -1 & -1 \\
0 & 0 & -2 & 0 & -2 & 1 & 1 & -1 & -1 \\
0 & 1 & -1 & 1 & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & -1 & 1 & -1
\end{pmatrix}. \tag{2.26}$$

and

$$\mathbf{M}^{-1} = a \begin{pmatrix} 4 & -4 & 4 & 0 & 0 & 0 & 0 & 0 & 0 \\ 4 & -1 & -2 & 6 & -6 & 0 & 0 & 9 & 0 \\ 4 & -1 & -2 & 0 & 0 & 6 & -6 & -9 & 0 \\ 4 & -1 & -2 & -6 & 6 & 0 & 0 & 9 & 0 \\ 4 & -1 & -2 & 0 & 0 & -6 & 6 & -9 & 0 \\ 4 & 2 & 1 & 6 & 3 & 6 & 3 & 0 & 9 \\ 4 & 2 & 1 & -6 & -3 & 6 & 3 & 0 & -9 \\ 4 & 2 & 1 & -6 & -3 & -6 & -3 & 0 & 9 \\ 4 & 2 & 1 & 6 & 3 & -6 & -3 & 0 & -9 \end{pmatrix}. \qquad (2.27)$$

where $a = 1/36$, and the diagonal matrix $\mathbf{S}$ reads

$$\mathbf{S} = \begin{pmatrix} s_0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & s_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & s_2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & s_3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & s_4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & s_5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & s_6 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & s_7 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & s_8 \end{pmatrix}. \qquad (2.28)$$

In the compact notation, the matrix $\mathbf{S}$ is :

$\mathbf{S} = diag\,(1.0, 1.4, 1.4, s_3, 1.2, s_5, 1.2, s_7, s_8)$, where $s_7 = s_8 = 1/\tau$ and s_3; s_5 are arbitrary and can be set initially to 1.0.

2.5.2 MRT-D2Q9 model for scalar variables

As mentioned in Sec. 2.2, the model D2Q5 is commonly applied to solve the momentum field, while for the temperature and concentration fields some authors prefer to use the D2Q5 because of its quick convergence. This is so because the model has less degrees of freedom than D2Q9 (i.e., only 4 discrete velocities). Along with a low computational cost, it also possesses good accuracy and implementation simplicity. In the literature, there exists a lot of information related to these methods [24, 35, 36]. In this Ph.D. thesis, the D2Q9 MRT model is applied to the velocity as well as the temperature and concentration fields, because the focus is on the precision of the results rather than the saving of computational costs. The relaxation equation for the distribution function of scalar variables is of the same form as Eq. (2.23),

$$\phi_i(x + \mathbf{c}_i \Delta t, t + \Delta t) - \phi_i(x, t) = -\mathbf{M}^{-1} \cdot \mathbf{S}_\phi \cdot [\mathbf{m}(x, t) - \mathbf{m}^{eq}(x, t)] \quad (2.29)$$

where the nine components of the equilibrium moment vector $\mathbf{m}^{eq}$ are analogous to those in Eq (2.24), with minor differences. That means for example, that $m_7^{eq} = 0$ and $m_8^{eq} = 0$ the value is zero, whereas the other equilibrium moments are

$$\begin{aligned} m_0^{eq} &= \phi \\ m_1^{eq} &= -2\phi \\ m_2^{eq} &= 2\phi \\ m_3^{eq} &= u_x\phi \\ m_4^{eq} &= -u_x\phi \\ m_5^{eq} &= u_y\phi \\ m_6^{eq} &= -u_y\phi \\ m_7^{eq} &= 0 \\ m_8^{eq} &= 0 \end{aligned} \quad (2.30)$$

The diagonal matrix $\mathbf{S}_\phi$ reads

$$\mathbf{S}_\phi = \begin{pmatrix} s_0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & s_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & s_2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & s_3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & s_4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & s_5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & s_6 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & s_7 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & s_8 \end{pmatrix}. \tag{2.31}$$

and its compact notation is $\mathbf{S}_\phi = diag\,(1.0, 1.1, 1.1, s_3, s_4, s_5, s_6, 1.2, 1.2)$, where $s_3 = s_4 = s_5 = s_6 = 1/\tau_\phi$, and s_7; s_8 are arbitrary and are set initially to 1.2 in this thesis.

2.6 Boundary conditions

A crucial part of any computational fluid dynamics method is the implementation of initial and boundary conditions (BC). Depending on the boundary conditions, a unique solution is obtained. For the case of the LBM, the application of boundary conditions is not different. There are different types and approaches to use BC, and the accuracy of the method depends on the right implementation of the boundary conditions. This implies that different types of BC may be used depending of the case. A peculiar aspect occurs for BC applied in LBM, where the distribution function equation (i.e., Eq. 2.2) contains all the information about the fluid, which is integrated in the domain and needs to be determined at the boundaries.

Currently, different approaches are being used and suggested by several authors. The most relevant are explained in the following sub-sections.

2.6.1 Bounce Back

The bounce-back scheme was originally taken from Lattice Gas method [38]. This model was used for solid walls at rest, moving boundaries and flow over obstacles. The bounce-back scheme for the LBM may be considered ideal to simulate fluid flow in complex geometries, such as porous medium. Also called no-slip boundary condition, the bounce-back scheme takes a particle going from a fluid node towards a solid boundary (e.g., a wall) and bounces it back to the last node of the flow domain (e.g., fluid flow).

Figure 2.7 shows a simple bounce back where the particles are exactly located at the wall node; before the streaming step, the distribution functions pertaining to opposite directions are exchanged. The bounce back scheme is a simple approach to fix the unknown distribution functions at the wall nodes. This scheme is first order accurate [10]. Thus the distribution functions, f_5, f_2 and f_6, are simply, $f_5 = f_7$, $f_2 = f_4$ and $f_6 = f_8$, where f_7, f_4 and f_8 are known from the streaming step, and can be easily calculated directly at the solid node.

Still another type of bounce-back is used in LBM (see Fig. 2.8). This scheme locates the wall at half the distance from the lattice sites, i.e., between a fluid and a solid node. The distribution functions, f_4, f_7 and f_8 are known from the streaming step. Its is assumed that, when these known distribution functions hit the wall, bounce back to the solution domain. Therefore, $f_5 = f_7$, $f_2 = f_4$ and $f_6 = f_8$, where f_4 at node (x, y) is equal to f_2 at node $(x, y - 1)$. This scheme is called half-way bounce-back and is second order accurate, so it has has better accuracy than the full-way bounce-back. These bounce-back schemes also fulfill conservation of mass and momentum at the boundaries [10, 24, 27].

This Ph.D. considers different schemes of boundary conditions implemented in the different study cases of the next chapters.

2.6.2 Periodic boundary condition

Periodic boundary condition become necessary to apply to isolate a repeating flow conditions. This boundary condition are used commonly in some

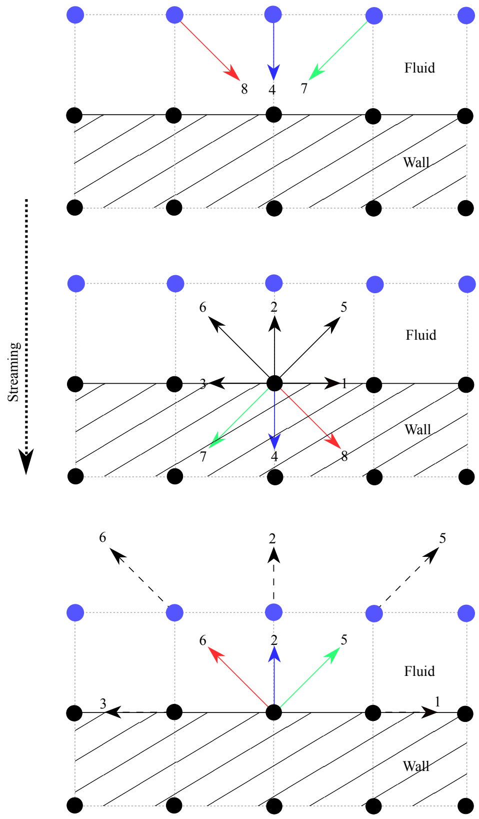


FIGURE 2.7: Full-way bounce back scheme

benchmarks to validate the system. For instance, this scheme may be used in the case of a flow in a channel, where bounce-back boundaries may be applied at the slit wall and periodic boundaries may be applied at the inlet and the outlet of the channel. The distribution functions that enter the inlet are the same as the distribution functions leaving the domain at the position $x = L$ (i.e., outlet). Consequently, the distribution functions, f_1, f_5, and f_8 are unknown at the inlet $x = 0$ and f_3, f_7, and f_6 are unknown at the outlet $x = L$. Hence, the application of these boundary conditions is as follows,

$$\begin{aligned} &f_1^{x=0} = f_1^{x=L}, \quad f_5^{x=0} = f_5^{x=L}, \quad f_8^{x=0} = f_8^{x=L} \\ &f_3^{x=L} = f_3^{x=0}, \quad f_7^{x=L} = f_7^{x=0}, \quad f_6^{x=L} = f_6^{x=0} \end{aligned} \tag{2.32}$$

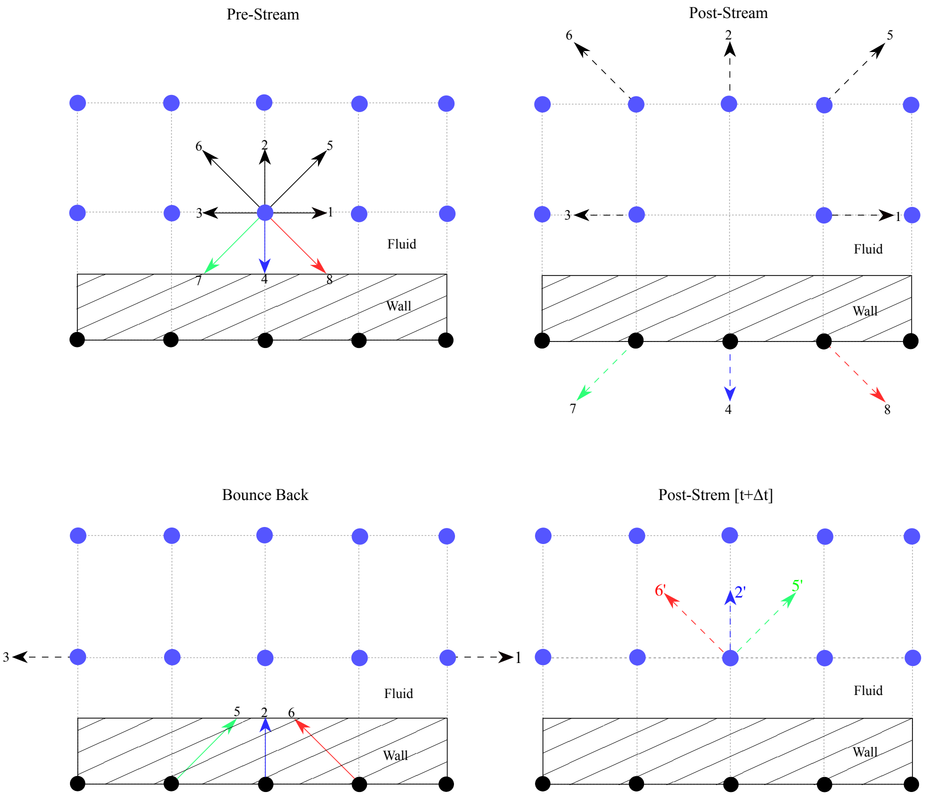


FIGURE 2.8: Half-way bounce back scheme

2.6.3 Zou-He boundary condition

In some physical cases, boundary conditions for the velocity or pressure are known. For example, in a channel flow, where the velocity at the inlet is specified, the three unknown distribution functions may be calculated as seen in Fig. 2.9. Let us suppose that the velocity components for west are known. The Zou-He boundary conditions may be used to calculate the unknown distribution functions [31]. Eq. (2.8) and Eq. (2.9) are imposed,

$$\rho = f_0 + f_1 + f_2 + f_3 + f_4 + f_5 + f_6 + f_7 + f_8, \tag{2.33}$$

$$\rho \mathbf{u}_x = f_1 + f_5 + f_8 - f_6 - f_3 - f_7 \tag{2.34}$$

$$\rho \mathbf{u}_y = f_5 + f_2 + f_6 - f_7 - f_4 - f_8, \tag{2.35}$$

after the streaming, f_0, f_2, f_4, f_6, f_3, f_7 are known and the color arrows (see Fig. 2.9) f_5, f_1 and f_8 on the west need to be calculated with some equations presented hereafter.

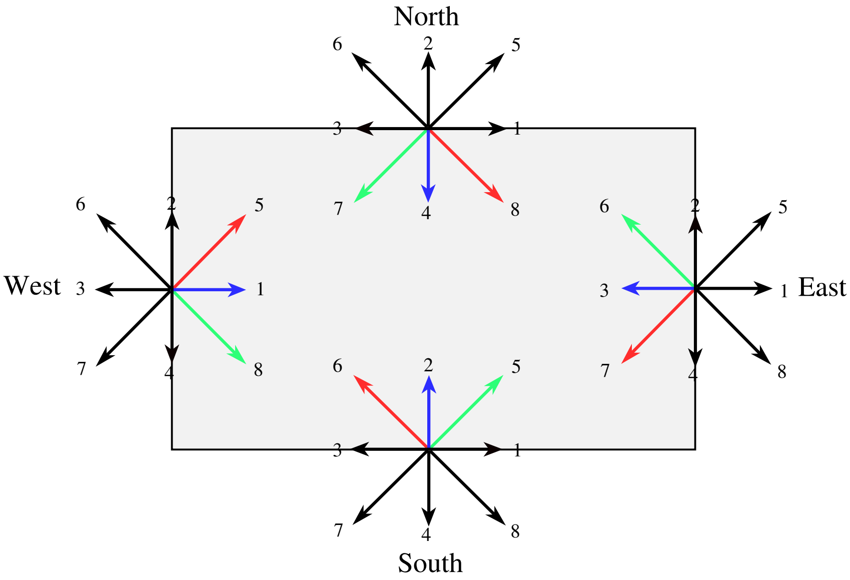


FIGURE 2.9: Boundaries and direction of streaming velocities.

Zou-He describes a method to calculate these unknown distributions assuming that the bounce-back rule still holds for the non-equilibrium part of the particle distribution normal to the boundary. In the case of Fig. 2.9, the east-west exchange of the non-equilibrium distributions at the inlet reads

$$f_1 - f_1^{eq} = f_3 - f_3^{eq}, \tag{2.36}$$

where the equilibrium distributions f_1^{eq} and f_3^{eq} can be calculated using Eq. (2.3). After substituting one obtains

$$f_1 - f_1^{eq} = f_3 + \frac{2}{3}\rho\mathbf{u}_x. \tag{2.37}$$

For the other two unknown distribution functions f_5 and f_8, the procedure is also applied,

$$\rho = \frac{1}{1-\mathbf{u}_x}[f_0 + f_2 + f_4 + 2(f_3 + f_6 + f_7)] \tag{2.38}$$

$$f_5 = f_7 - \frac{1}{2}(f_2 - f_4) + \frac{1}{6}\rho\mathbf{u}_x + \frac{1}{2}\rho\mathbf{u}_y \tag{2.39}$$

$$f_8 = f_6 + \frac{1}{2}(f_2 - f_4) + \frac{1}{6}\rho\mathbf{u}_x - \frac{1}{2}\rho\mathbf{u}_y. \tag{2.40}$$

Thus the distribution functions are now known in the west part of the domain. The same procedure can be used for the rest of the boundaries if needed (e.g., the velocity components are known at the east, north or south boundaries). For instance, f_3, f_7 and f_6 are calculated as

$$\rho = \frac{1}{1 - \mathbf{u}_x}[f_0 + f_2 + f_4 + 2(f_1 + f_5 + f_8)] \tag{2.41}$$

$$f_3 = f_1 - \frac{2}{3}\rho\mathbf{u}_x \tag{2.42}$$

$$f_7 = f_5 + \frac{1}{2}(f_2 - f_4) - \frac{1}{6}\rho\mathbf{u}_x - \frac{1}{2}\rho\mathbf{u}_y \tag{2.43}$$

$$f_6 = f_8 - \frac{1}{2}(f_2 - f_4) - \frac{1}{6}\rho\mathbf{u}_x + \frac{1}{2}\rho\mathbf{u}_y, \tag{2.44}$$

for cases in which the velocity component is known in the east boundary. If the velocity is known in the north boundary, the unknown distribution functions f_4, f_7 and f_8 are calculated in the following way

$$\rho = \frac{1}{1 - \mathbf{u}_y}[f_0 + f_1 + f_3 + 2(f_2 + f_6 + f_5)] \tag{2.45}$$

$$f_4 = f_2 - \frac{2}{3}\rho\mathbf{u}_y \tag{2.46}$$

$$f_7 = f_5 + \frac{1}{2}(f_1 - f_3) - \frac{1}{6}\rho\mathbf{u}_y - \frac{1}{2}\rho\mathbf{u}_x \tag{2.47}$$

$$f_8 = f_6 + \frac{1}{2}(f_3 - f_1) + \frac{1}{2}\rho\mathbf{u}_x - \frac{1}{6}\rho\mathbf{u}_y. \tag{2.48}$$

Finally, for the south boundary the unknown distribution functions are f_2, f_5 and f_6, computed as

$$\rho = \frac{1}{1 - \mathbf{u}_y}[f_0 + f_1 + f_3 + 2(f_4 + f_7 + f_8)] \tag{2.49}$$

$$f_2 = f_4 + \frac{2}{3}\rho\mathbf{u}_y \tag{2.50}$$

$$f_5 = f_7 - \frac{1}{2}(f_1 - f_3) + \frac{1}{6}\rho\mathbf{u}_y + \frac{1}{2}\rho\mathbf{u}_x \tag{2.51}$$

$$f_6 = f_8 + \frac{1}{2}(f_1 - f_3) + \frac{1}{6}\rho\mathbf{u}_y - \frac{1}{6}\rho\mathbf{u}_x. \tag{2.52}$$

2.6.4 Dirichlet boundary condition

For instance, let us suppose that in Fig. 2.9, the heated wall on the west satisfies a boundary condition with a specified temperature (e.g., $T_w = 1$, in dimensionless units). A Dirichlet boundary condition can be used to calculate the flux through a conservation principle [24, 22]. The unknown distribution functions, g_1, g_5 and g_8 are evaluated as follows. For g_1 one has

$$g_1 - g_1^{eq} + g_3 - g_3^{eq} = 0. \tag{2.53}$$

Since g_1^{eq} at the heated wall is equal to $\omega_1 T_w$, the above equation can be rewritten as

$$g_1 = T_w(\omega_1 + \omega_3) - g_3, \tag{2.54}$$

where w_1 and w_3 are the weights of the applied lattice model explained in Section. 2.2. Similarly, the rest of the unknown distribution functions can be calculated as

$$g_5 = T_w(\omega_5 + \omega_7) - g_7 \tag{2.55}$$

$$g_8 = T_w(\omega_8 + \omega_6) - g_6. \tag{2.56}$$

In cases when the wall is at a colder temperature (i.e., $T_c = 0$),

$$g_1 = \cancelto{0}{T_c}(\omega_1 + \omega_3) - g_3, \tag{2.57}$$

the equations are expressed in the following form,

$$g_1 = -g_3, \quad g_5 = -g_7, \quad g_8 = -g_6. \tag{2.58}$$

2.6.5 Open Boundaries

As an opposite case to the very common problem that the velocity at the inlet (e.g., a flow in a channel) is known, as for instance when the velocity is specified at the inlet as a uniform flow field along the channel direction, there is also the case when the outlet velocity is unknown. A solution to solve this problem is calculating the unknown distribution functions using a double extrapolation at the outlet boundary. Figure 2.10 represents the outlet and the nodes used to calculate the unknown distribution functions. The equations are as follows,

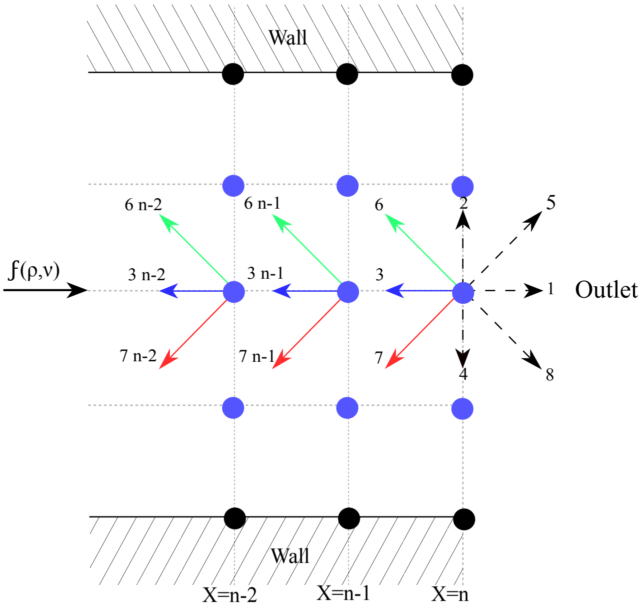


FIGURE 2.10: Open boundary condition scheme, outlet nodes of a channel.

$$f_{3,n} = 2 \cdot f_{3,n-1} - f_{3,n-2} \tag{2.59}$$

$$f_{6,n} = 2 \cdot f_{6,n-1} - f_{6,n-2} \tag{2.60}$$

$$f_{7,n} = 2 \cdot f_{7,n-1} - f_{7,n-2} \tag{2.61}$$

where n is the last node of the domain (i.g., channel), $n-1$ and $n-2$ are the nodes two position backwards in the lattice domain. The outlet boundary conditions for the temperature and the concentration are written similarly, where f is replaced respectively by g and A^{σ}.

2.7 Lattice units

It is well known that in the field of computational fluid dynamics (CFD), the main goal is to represent in a simulation a physical problem with real units, and obtain a numerical solution. During the implementation of the LBM, difficulties are encountered sometimes when mapping the variables of such real problem. One of the difficulties with LBM is the macroscopic variable conversion, as LBM works in a mesoscopic scale with Lattice variables (e.g., density and viscosity of the lattice, discrete velocities, etc). These variables are not scaled like physical variables, and for this reason it is necessary to convert the physical variables into the lattice scale to be equivalent and then compared.

The correct way to convert the physical variables into LBM variables and vice versa is the following. The first step consists in converting the physical variables into dimensionless variables, then these dimensionless variables can be transformed into lattice variables. Sometimes the physical parameters are unknown or irrelevant and the process of the unit conversion can be calculated the other way around, that is, arbitrary lattice variables can be used and then converted into physical variables if needed. Furthermore the solution to the incompressible Navier stokes equations depends on the values of certain dimensionless parameters (e.g., Reynolds number, Richardson number, among others). Therefore, the Reynolds number for example, has to be the same in both physical and lattice units. For this purpose one can set the appropriate values of the discretization parameters Δx and Δt. These parameters are important to determine the correct space and time scales for the characteristic discrete velocity, in order to ensure the stability and accuracy of the method. For a better understanding, a simple example case is shown in the following paragraph.

Lest us start with a simple physical case in two dimensions. Figure 2.11 shows a rectangular channel with an inlet on the left and an outlet on the right side. The top and bottom consist of fixed walls, $u_x = u_y = 0$.

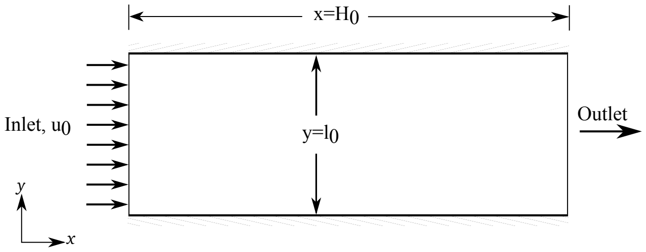


FIGURE 2.11: Physic problem scheme, Flow in channel with uniform inlet velocity.

For the inlet-flow, a uniform velocity is fixed as $u_0 = 0.1004 cm/s$. The kinematic viscosity of the fluid is $\nu = 0.01004 cm^2/s$, the reference length $l_0 = 10 cm$. Knowing these parameters, the Reynolds number for the system can be calculated,

$$Re = \frac{u_0 \cdot l_0}{\nu}, \tag{2.62}$$

hence,

$$Re = \frac{0.1004 cm/s \cdot 10 cm}{0.01004 cm^2/s} = 100. \tag{2.63}$$

To ensure the correctness of the simulation, the Reynolds number to apply in the Lattice Boltzmann method must be exactly the same obtained from Eq. (2.62), (i.e., Re_{lbm} = Re). Knowing this condition, one can proceed to the next step, which consists in chosing the discretization parameters, for example the grid size; 100×300. That means that we set 100 nodes for the width l_0 in the x-coordinate, and 300 nodes in the y-coordinate. Thus, the discrete space interval Δx (or Δy) is defined with the number of nodes (N) in the characteristic length of the computational domain. Therefore,

$$\Delta x = \frac{1}{N}. \tag{2.64}$$

Thus, $\Delta x = 1/100 = 0.01$.

It is not intuitive to select a value for Δt. In other schemes Δt is often linked to Δx to provide stability in the calculation. To ensure the stability and accuracy of the LBM, most common relation used is $\Delta t \sim \Delta x^2$ [24]. In LBM Δt and Δx are linked by a different constraint, involving u_{lbm} which is calculated through $\Delta t/\Delta x$. The value of u_{lbm} cannot be larger than the speed of sound c_s, moreover the Mach number $\mathrm{Ma} = u/c_s$ should be kept small for an accurate incompressible-flow solution. Therefore, the value of u_{lbm} should be in the range $0.1 \backsim 0.2$, in order to ensure the range of $\mathrm{Ma} \leqslant 0.3$ for incompressible flow (LBM does not support supersonic flows). As a matter of fact, LBM is in the class of quasi-incompressible fluid solvers. Understanding this, a good choice for Δt to start is

$$\Delta t \sim \Delta x^2. \tag{2.65}$$

Consequently, $\Delta t = 0.0001$. But this choice is not restrictive, if accuracy and stability of the method is a concern the value of Δt could be increase this is achieved in particular increase a bit the value of Δx. Several author used the following approximation (e.g., $\Delta t \sim \Delta x^2 \cdot 2$). Knowing the values of Δt and Δx, the other variables such as velocity and viscosity may be converted from physical to dimensionless variables, and afterwards to lattice variables as follows. For instance for the dimensionless velocity,

$$u_d = \frac{t_0}{l_0} u_0. \tag{2.66}$$

where t_0 is the characteristic time of the system and could be calculated as $t_0 = l_0/u_0$, that means the time needed by a passive scalar in the fluid to travel a distance l_0. To help the intuition, one may consider that in the dimensionless system $u_d = 1$, and the dimensionless viscosity as $\nu_d = 1/Re$.

Thus,

$$u_d = \frac{99.601594\ s}{10\ cm} \cdot 0.1004\ cm/s = 1. \tag{2.67}$$

We can calculate the lattice characteristic velocity u_{lbm} and the lattice viscosity ν_{lbm} from the following equations,

$$u_{lbm} = \frac{\Delta t}{\Delta x} u_d, \tag{2.68}$$

and

$$\nu_{lbm} = \frac{\Delta t}{\Delta x^2} \frac{1}{Re}, \tag{2.69}$$

Therefore, we have

$$u_{lbm} = \frac{0.0001}{0.01} 1 = 0.01 \qquad \nu_{lbm} = \frac{0.0001}{(0.01)^2} \frac{1}{100} = 0.01 \tag{2.70}$$

and

$$Re_{lbm} = \frac{u_{lbm} N}{\nu_{lbm}} = \frac{0.01 \cdot 100}{0.01} = 100, \tag{2.71}$$

It is not necessary to use Eq. (2.69) if we first know the value of Re, that means that we can calculate the value of ν_{lbm} from Eq. (2.71); $\nu_{lbm} = u_{lbm} \cdot N/\mathrm{Re}_{lbm}$. If we are using Single Relaxation Time SRT, it has to be ensured that the value of τ for the hydrodynamic lattice is not in the range $\tau < 0.53$ (because we may face accuracy and stability problems) when calculating τ through Eq. (2.19).

In conclusion, these are required parameters in the set up of the lattice variables to reproduce the physical problem.

Table 2.5: Parameters of the example (see text), physical and lattice variables for the LBM simulation

Parameters

Physical units	Lattice Units (ℓu)
$u_0 = 0.1004\ cm/s$	$u_{lbm} = 0.01$
$\nu_0 = 0.01004\ cm^2/s$	$\nu_{lbm} = 0.01$
$l_0 = 10\ cm$	$N = 100$
$t_0 = 99.6016\ s$	$t_{lbm} = 10000$
$Re = 100$	$Re_{lbm} = 100$
	$\tau = 0.53$

It can be seen in in Table 2.5, that the value of $\tau = 0.53$ is relatively low, and depending on the case we may face stability and accuracy problems. To solve this difficulty, some parameter linked with τ should be modified, to increase its value. Instead of SRT, Multiple relaxation time MRT is used for this low-range value of τ, but this configuration implies a higher computational cost than SRT. There are simple solutions or alternatives within the SRT scheme that can be used to increase the value of τ:

- Increase the values of ν_{lbm}, so that in order to modify this value we have to use the equation for the Reynolds number, Eq. (2.71).
- Increase the number of nodes in the characteristic length N
- Increase the value of the lattice velocity u_{lbm}, according to certain conditions in which the LBM works under the assumption of incompressible regime, the value of u_{lbm} need to be $\leqslant 0.1$

The parameters above can be used to modify the range of τ through Eq. (2.71), changing one value and keeping the rest of the parameters fixed in the equation. In order to increase the value of τ, we then change the number of nodes in the characteristic length N from the last example, obtaining the new value of the viscosity ν_{lbm},

$$\nu_{lbm} = \frac{u_{lbm}N}{\mathrm{Re}_{lbm}} = \frac{0.01 \cdot 200}{100} = 0.02 \qquad \tau = 3 \cdot 0.02 + 1/2 = 0.56. \quad (2.72)$$

If the increase of the number of nodes is not feasible, we can choose any of the rest of parameters that can be modified (i.e., u_{lbm}). Thus,

$$\nu_{lbm} = \frac{0.05 \cdot 100}{100} = 0.05 \qquad \tau = 3 \cdot 0.05 + 1/2 = 0.65 \quad (2.73)$$

Equation (2.73) we have freely chosen the lattice velocity u_{lbm}. For this reason the value of Δt needs to be recalculated, as u_{lbm} was computed before through Eq (2.65) using the discretized lattice space and time $(\Delta x, \Delta t)$. Consequently, the new value Δt can be obtained as follows,

$$\Delta t = u_{lbm} \cdot \Delta x \ = 0.05 \cdot 0.01 = 0.0005. \quad (2.74)$$

When dealing with thermal flows, where cold and heated walls are considered, the reference temperature, T_c can be chosen arbitrarily and it does not affect the evolution of the flow, as the buoyancy term only depends on the difference $T_h - T_c$. For example, for the Rayleigh Bernard convection the temperature in two apposite boundaries is defined as

$$T_{0,hot} = T_c + \frac{1}{2}\Delta T_p \qquad T_{0,cold} = T_c - \frac{1}{2}\Delta T_p \tag{2.75}$$

where ΔT is the temperature difference between the two boundaries. As mentioned above, a constant non zero T_c is not eligible as a reference, as its value have no influence on the flow evolution.

In LBM, fixing ΔT is a less crucial step than fixing the value of Δx and Δt. ΔT does not intervene in the calculation of the lattice variables except for determining the value of T_{lbm} (e.g., initial condition). Even so, the temperature scale imposed through ΔT is not expected to affect the numerics in a significant way, as the advection-diffusion equation for the temperature is linear. For simplicity, $T_{c,lbm} = 0$ and $T_{h,lbm} = 1$ are chosen, and

$$T = \frac{T_{lbm} - T_{c,lbm}}{T_{h,lbm} - T_{c,lbm}} \tag{2.76}$$

where T is the dimensionless temperature, T_{lbm} is the local temperature of lattice. For example, in a closed square cavity with two different temperatures on two parallel walls (see Fig. 2.12), the conditions are chosen such that the heated wall temperature is $T_{0,hot} = 313.150$ K and the temperature of the cold wall is $T_{0,cold} = 293.150$ K), being the top and bottom adiabatic boundaries (i.e., thermal insulation).

To represent these parameters in a LBM simulation, it is enough to define the hot and cold temperature as $T_{h,lbm} = 1$ and $T_{c,lbm} = 0$, respectively (see Fig. 2.12)

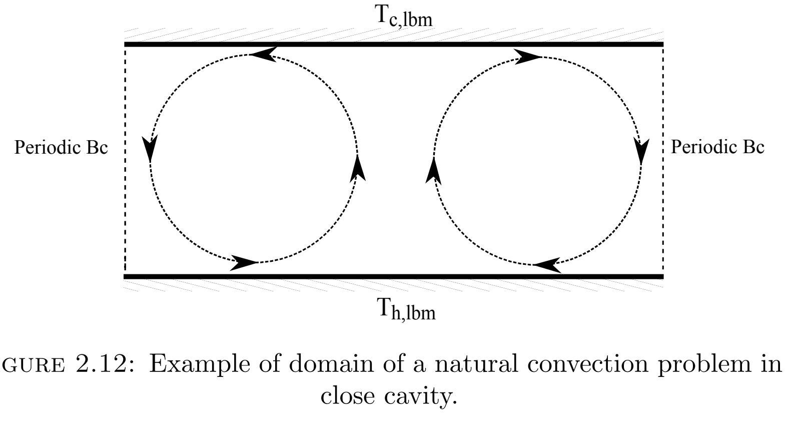


FIGURE 2.12: Example of domain of a natural convection problem in a close cavity.

The results of the temperature field in the domain obtained from the simulation are in a interval of range $T = [0, 1]$. With a simple process we can get the physic values of the temperature, starting from the LBM variables, by means of the following conversion,

$$T_0 = (T_{0,hot} - T_{0,cold} \cdot T_{lbm}) + T_{0,cold}. \tag{2.77}$$

For example, if $T_{lbm} = 0.56$ is obtained in the simulation, the value in the physical system is

$$T_0 = (313.150K - 293.150K \cdot 0.56) + 293.150K = 305.47K, \tag{2.78}$$

To conclude, in the inverse situation, we use the following form to convert the physical temperature into the LBM temperature,

$$T_{lbm} = \frac{T_0 - T_{0,cold}}{T_{0,hot} - T_{0,cold}}. \tag{2.79}$$

Other parameters can as well be converted, for example time,

$$T_{ime,lbm} = \frac{T_{step,lbm} \cdot u_{lbm}}{\Delta t}, \tag{2.80}$$

$$T_{ime,d} = \frac{T_{ime,lbm}}{(N/u_{lbm})}, \tag{2.81}$$

where $T_{step,lbm}$ is the time step in the LBM simulation, $T_{ime,lbm}$ is the time in lattice units and $T_{ime,d}$ is the dimensionless time in the physical system.

2.8 Algorithm for the Lattice Boltzmann Method

Algorithm of LBM, a step by step description of the different phases of the algorithm is presented.

1. The first step consists in rescaling the physical system to a dimensionless system suitable to be applied the LBM (e.g., obtaining u_0, l_0). Conversion units are explained in the next section.

2. The second step is discretizing the problem from the dimensionless physical system to the lattice system, building the uniform grid in cartesian coordinates generated by the spacial step Δx and time step Δt according to the geometry, and ensure the restriction on the Mach number $Ma < 0.3$ to maintain the incompressible regime in a subsonic flow simulation.

3. In the initialization step, the distribution function is constructed and the initial macroscopic variables are computed (e.g., $\rho(x, t)$, $\mathbf{u}(x, t)$), The density of the LBM is initialized as $\rho(x, t) = 1.0$.

4. Impose boundary conditions at the macroscopic level. Construct the unknown distribution functions streaming from the solid to the bulk on the kinetic level, based on the imposed boundary conditions. Different boundary dynamics are chosen according to the requirement of stability, accuracy and efficiency.

5. In the Collision step, according with the collision rule explained in section (2.1) and the boundary conditions imposed, the distribution function is relaxed towards the local equilibrium in all nodes. In the streaming step, the distribution function propagates to neighbor nodes at the discrete set of velocities $\mathbf{c}_i$ of the chosen model, and according to the boundary condition imposed.

6. The last step is the post processing, if certain stopping condition is satisfied. Otherwise go back to step (Boundary conditions) and recalculate again the imposed boundary condition.

2.8.1 Code

Code, Fortran 90

The code used was acquired from the open-source Palabos organization. This code was modified and implemented the different existing models in the literature related on the LBM, such as Temperature and concentration field, Multiple Relaxation Time (MRT-model) for momentum, temperature and concentration, and the different schemes of the boundary condition for LBM. In addition, a new subroutine was created in order to export the data result to a visualization software and (Gnuplot, Paraview). This code was implemented in Fortran 90 programming language, and can be found in the appendix of this thesis. (http://www.palabos.org/)

2.8.2 Visualization tools and data analysis

Paraview

The Paraview software is an open-source language used for data analysis and visualization. It has been used by several authors to depict frame rate for large datasets. To this end, the Paraview visualization tool-kit (VTK) libraries offer the advantage of a quick and easy implementation to create contour of different scalars, iso-volumes, streamlines, 2D-3D glyphs vectors, animations. It also contains a calculator tool which can be used to compute new variables using the existing calculated variables. (http://www.paraview.org/).

Gnuplot

Is an open-source command line program used to generate accurate plots, Mathematica functions and interactive data in two and three dimension. It also allows to directly generate output graphs or images in different formats (eps, jpeg, svg, png, etc.). It contains a built-in module for inserting large or complex equations.

Chapter 3

LBM for incompressible fluid flow

3.1 Introduction

In this section, numerical simulations are presented which were carried out using the Lattice Boltzmann method (LBM) in two-dimensions, in the incompressible regime and laminar flow. In order to verify the accuracy of the LBM, these results are compared with either analytical or numerical solutions using other methods. The numerical stability is analyzed for single relaxation time (SRT) and multiple relaxation time (MRT), as well as the performance of the boundary condition used. A uniform grid was implemented for all the studied cases in this thesis. For the stationary solid wall of the simulations, full-way bounce-back boundary condition and Zou-He boundary condition was used for these cases of study, in the form explained in Sec. 2.6.

A total of three cases of study were selected for this section, with the main goal of validating the codes:

- 2D - Poiseuille flow
- Lid - Driven cavity flow
- 3D - Poiseuille flow

These cases will be analyzed in the following paragraphs.

3.1.1 2D - Poiseuille flow

A basic example is the Hagen-Poiseuille flow, a steady state incompressible flow which develops between two stationary parallel plates at rest, and an inlet flow boundary condition in the form of a pressure gradient or an specified inlet velocity. In this study, a velocity boundary condition will be used at the inlet of the domain. The x-component of the Navier-Stokes equations reduces in this simple case to:

$$\frac{\partial p}{\partial x} = \mu \frac{\partial^2 u}{\partial y^2} \tag{3.1}$$

and the y component

$$-\frac{1}{\rho}\frac{\partial p}{\partial y} = 0 \tag{3.2}$$

Assuming the higher pressure region is on the left side and the lower pressure region on the right, the pressure gradient $\partial p/\partial x$ would be negative as the fluid velocity is directed to the right side, meaning that the fluid is moving from the higher pressure region to the lower pressure one. Such as the plane Couette flow, the velocity lies in the x direction and is a function of y alone, but the pressure gradient is not zero, but a constant instead $-C$. Hence, $\mu\partial^2 u/\partial y^2 = -C$. According to the equations defined previously, this expression can be solved to obtain the maximum velocity as follows [39]

$$u_{max} = \frac{C \cdot D^2}{8\mu} = \frac{3}{2}u_0. \tag{3.3}$$

Thus the velocity solution of the incompressible Navier-Stokes equations corresponding to the Poiseuille flow inside the channel is

$$u(y) = u_{max}\left[1 - \frac{\| R - y \|^2}{R^2}\right], \tag{3.4}$$

where u_{max} is the maximum value of the velocity, $R = D/2$ is the radius of the channel, (D being the characteristic length of the system). In order to ensure a fully developed velocity profile, the channel should be long enough. In the laminar flow regime this can be approximated as [39]

$$\frac{L}{D} = 0.06 \cdot \text{Re}; \quad L = 6D, \tag{3.5}$$

for Re in the order of 100. Figure 3.1 shows the domain for this case. The fluid in this simulation is water with a kinematic viscosity of $\nu =$ 1e-6 m^2/s and a characteristic length $D = 0.05\ m$. The uniform velocity is fixed at the inlet and is determined from the Reynolds number, Eq. (2.62),

$$u_0 = \frac{\text{Re} \cdot \nu}{D}; \quad = \frac{50 \cdot \text{1e}{-6}\ m^2/s}{0.05\ m}; \quad = 0.001\ m/s \tag{3.6}$$

for a Reynolds number of Re =50.

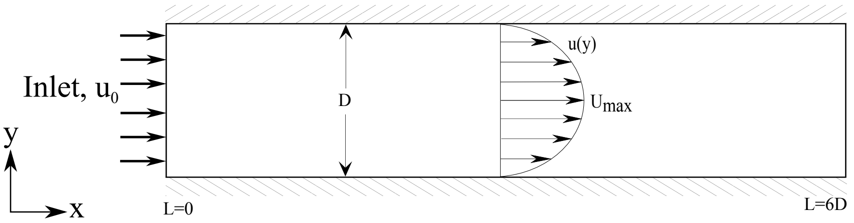


FIGURE 3.1: Study domain of the channel flow

3.1.1.1 Simulation setup

The uniform grid is buildt using $D = 52$ and $L = 312$ nodes. To determine the correct space and time scales for the discrete velocity, the uniform velocity of the entrance of the channel is fixed, and it is calculated using the discrete variable Δx and Δt. According to the approximation explained in Section. 2.2 of the Eq. (2.65), Δt is calculate as $\Delta t \sim \Delta x^2$. Hence, $\Delta x = 1/N = 1/52 = 0.01923$ and $\Delta t = 3.846 \times 10^{-4}$. The discrete velocity is then $u_{lbm} = \Delta t/\Delta x = 0.02$.

Figure 3.2 shows the scheme of the computational domain and the unknown distribution function which will be calculated (arrows in color).

For the solid wall, bounce-back boundary schemes are applied for the stationary solid walls (see Section. 2.6.1) for f_4, f_7, f_8 for the upper wall and f_2, f_5, f_6 for the bottom wall. At the entrance of the channel, the density is fixed $\rho = 1.0$. The unknown distribution functions for the inlet f_1, f_5, and f_8 are calculated through the Zou-He boundary condition (see Section. 2.6.3), with Eqs. (2.37, 2.39, 2.39) in the following form.

Inlet boundary, Zou and He scheme:

$$\begin{aligned}
\rho =& \frac{f_0 + f_2 + f_4 + 2(f_3 + f_6 + f_7)}{1 - \mathbf{u}_x} \\
f_1 =& f_3 + \frac{2}{3}\rho\mathbf{u}_x \\
f_5 =& f_7 - \frac{1}{2}(f_2 - f_4) + \frac{1}{6}\rho\mathbf{u}_x \\
f_8 =& f_6 + \frac{1}{2}(f_2 - f_4) + \frac{1}{6}\rho\mathbf{u}_x,
\end{aligned}$$

and for the outlet open boundary condition is implemented as follows:

$$\begin{aligned}
f_3 =& 2\cdot f_{3,L=6D-1} - f_{3,L=6D-2} \\
f_6 =& 2\cdot f_{6,L=6D-1} - f_{6,L=6D-2} \\
f_7 =& 2\cdot f_{7,L=6D-1} - f_{7,L=6D-2}
\end{aligned}$$

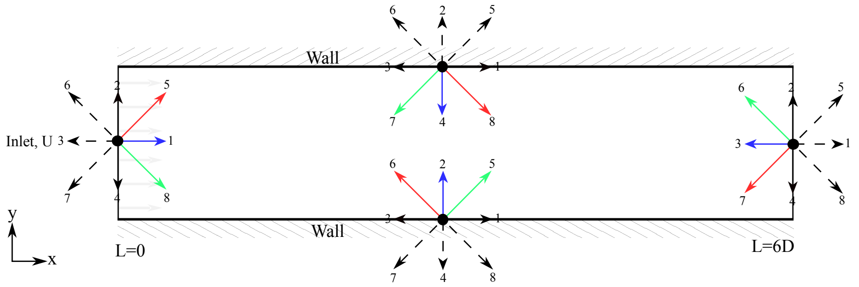


FIGURE 3.2: Domain for the 2D Poiseuille flow, LBM simulation with the D2Q9 model, Re=50, u_{lbm}=0.1.

The (lattice) kinematic viscosity can be calculated with Eq. (2.72), using the previous parameters, thus

$$\nu_{lbm} = \frac{u_{lbm} \cdot N}{\text{Re}} = \frac{0.02 \cdot 52}{50} = 0.0208, \tag{3.7}$$

Then the kinematic viscosity is used to calculated the relaxation time τ using Eq. (2.19). Then, $\tau = 3{\cdot}\nu{+}1/2 = 3{\cdot}0.0208{+}0.5 = 0.5624$. According to the stability of the method explained in Sec. 2.4, the value of τ has to be of about $\geqslant 0.53$ to ensure a correct stability and accuracy of the method. In the present case, $\tau = 0.5624$ ensures the stability of the method. The norm error $\boldsymbol{L}^2$ is defined with the following form:

$$\boldsymbol{L}^2 = \sqrt{\frac{\sum_{i=0}^{i=N_y} \parallel u_{(y)} - u_{(y),lbm} \parallel^2}{Ny}} \tag{3.8}$$

3.1.1.2 Results and validation

The analytical solution of the Poiseuille flow is presented and compared with the LBM result. The Poiseuille profile will not be developed immediately past the entrance of the channel. For this reason, some distance from the inlet is necessary before the profile is fully developed, Eq. (3.5).

Figure (3.3, 3.4) shows the velocity profiles as obtained from the LBM simulation, with the effect due to the development of the flow in the channel past the entrance, for $Re = 50$. The location along the channel for the selected velocity profiles is $X = 0.0016$, 0.0160, 0.1602, 0.3205, 0.5608, measured in a dimensionless distance from the inlet.

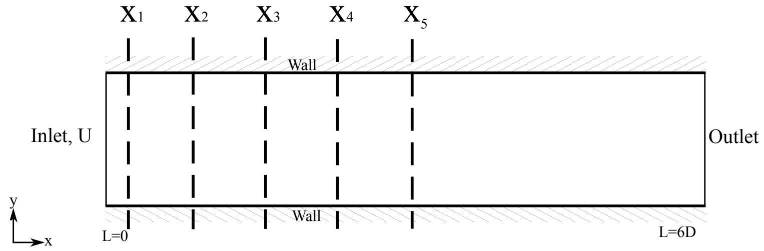


FIGURE 3.3: Distances along the channel for the plot of the velocity profiles, SRT-LBM simulation, Re=50.

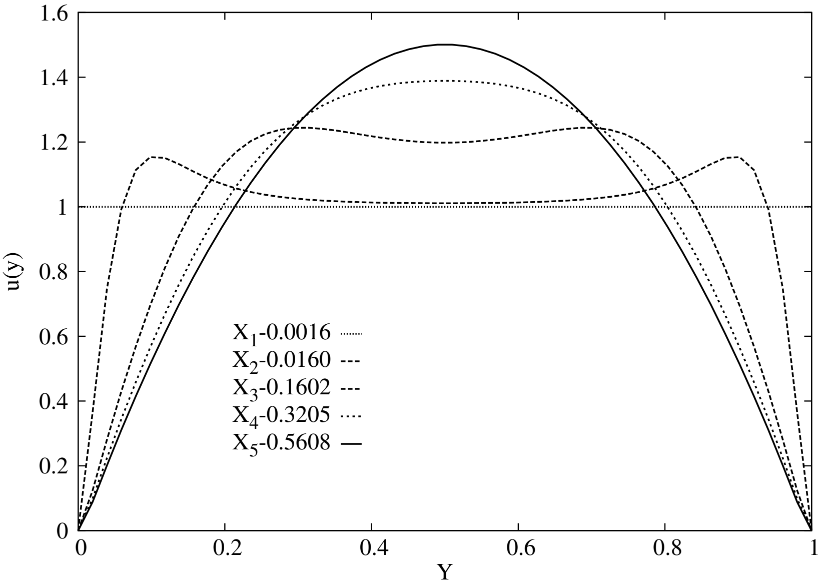


FIGURE 3.4: Velocity profiles at different distances from the inlet, SRT-LBM simulation, Re=50.

Three different SRT-LBM simulations were carried out for a grid-independence test to observe the convergence of the solution. The results are shown in Fig. 3.5, in green for $N = 52$, blue for $N = 104$, and red for $N = 204$, compared with the analytical solution. An appropriate quantity to characterize the error is the deviation with respect to the exact solution, $\boldsymbol{L}^2$. The norm $\boldsymbol{L}^2$ is tabulated in Table 3.1 for the three different grid sizes. As expected, the value of the error decreases as the number of lattice nodes increases.

TABLE 3.1: Norm error $\boldsymbol{L}^2$ for the grid convergence test performed with the laminar-Poiseuille-flow validation example.

N	Norm $\boldsymbol{L}^2$
52	6.40e−4
104	1.46e−4
204	5.68e−5

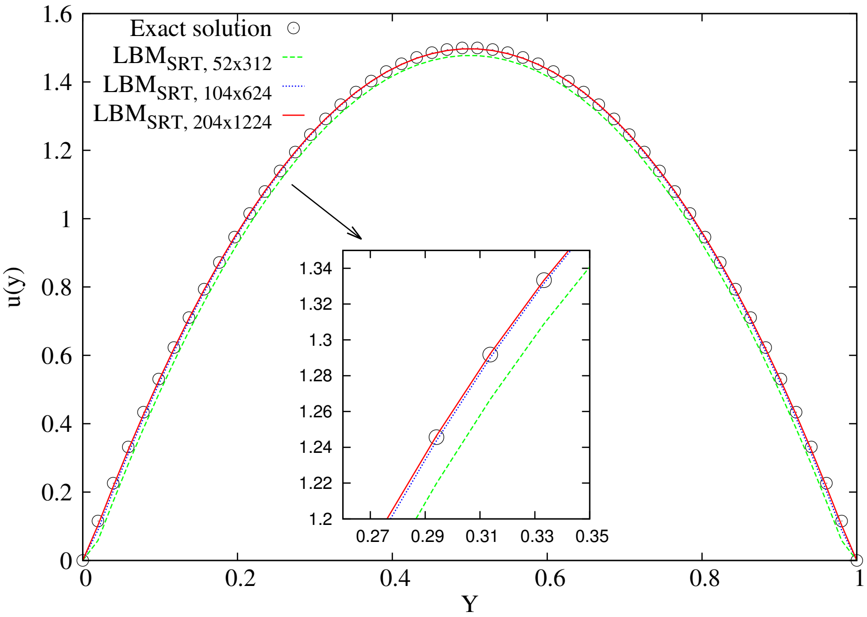


FIGURE 3.5: Dimensionless velocity in the vertical center line of the channel. Analytical and SRT-LBM solutions.

Figure 3.6 displays the contour plot of the dimensionless magnitude of the velocity for $N = 52$. On the left side (inlet) a small distortion can be seen due to an incipient numerical instability, which can be better observed in the zoom of the region in Fig. 3.7. This occurs because of the value of the relaxation time parameter τ, which for the case of $N = 52$ approaches the limit for the SRT model, $\tau \sim 0.53$. The cases $N = 104$ and $N = 204$ do not present these problems since τ increases with the number of nodes above the critical value.

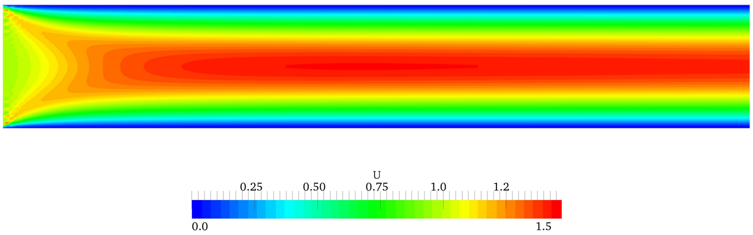


FIGURE 3.6: Dimensionless velocity contour plot for the SRT model: Re=50, u_{lbm}=0.02, $N = 52 \times 312$.

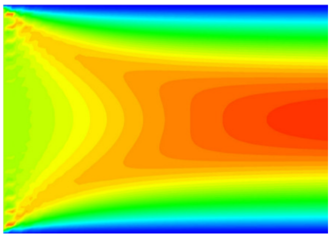


FIGURE 3.7: Zoom of the inlet region showing the dimensionless velocity contour plot.

Two more simulations were carried out in order to compare the SRT and the MRT models. It was observed that the MRT model does not present the instability for the case $N = 52$ (see Fig. 3.8). Likewise, the error $\boldsymbol{L}^2 = 3.52\text{e}{-4}$, is smaller than that found in the SRT simulation. In Fig. 3.9, the results from both models in the three cases are depicted along with the analytical solution[1].

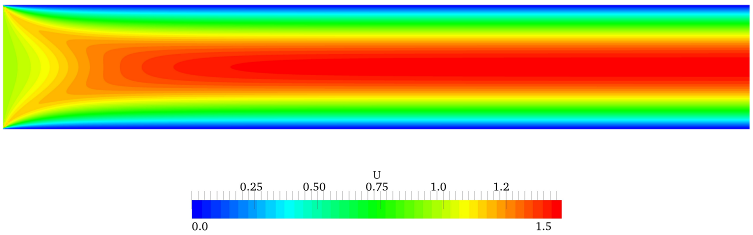


FIGURE 3.8: Dimensionless velocity contour plot for the MRT model: Re=50, u_{lbm}=0.02, $N = 52 \times 312$.

[1] The LBM codes for the Poiseuille flow in a channel were written in **Fortran 90** programming language, and can be found in the **Appendix**C

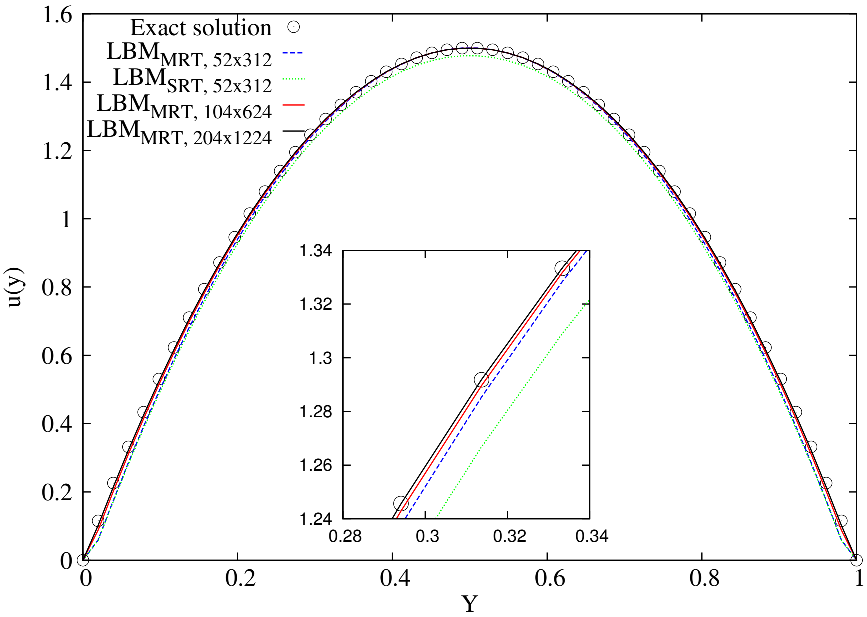


FIGURE 3.9: Dimensionless velocity in the vertical center line of the channel: analytical solution and LBM solution, showing also the MRT *vs.* SRT profiles for the case $N = 52$.

Table 3.2 shows the errors obtained in simulations for both models. In the first place, it has to be emphasized that for the same grid size, the MRT-model renders smaller errors, however this is more pronounced in the case $N = 52$. In conclusion, the grid size $N = 104$ can be used for either model to obtain similar results, whereas $N = 204$ presents a slightly smaller error at the cost of a significant increase of the computational trade-off.

TABLE 3.2: Norm error $\boldsymbol{L}^2$ for the SRT and MRT models obtained in the test cases.

N	$\boldsymbol{L}^2$ SRT-model	$\boldsymbol{L}^2$ MRT-model
52	6.40e−4	3.52e−4
104	1.46e−4	1.43e−4
204	5.68e−5	4.81e−5

3.1.2 2D Lid-Driven cavity flow

This case, commonly known as the Lid-Driven cavity (LDC) problem, is one of the most common benchmark test problems for incompressible fluid flow, and has been used to evaluate the accuracy and efficiency of numerical methods used in the Computational Fluid Dynamic field (CFD), such as multi-grid method, finite element, volume method, etc. The LDC is a type of flow developed within a square box in two-dimensions. The flow is driven by moving the top wall of the cavity with a constant velocity. A schematic example is provided in Fig. 3.10. For the rest of the walls, the velocity is set to zero, which in LBM schemes is accomplished by using bounce-back boundary conditions (see Sec. 2.6.1). The main goal of LDC is to study the effect of the Reynolds number on the structure of the steady vortices in the cavity, several author analyze the location of the different vortex generated, a primal vortex appear near to the center of the cavity, of the corners two small counter-rotation vortex are developed (secondary vortices), when the Re is increases, the secondary vortices grow up and the primal vortex is moves to the center of the cavity.

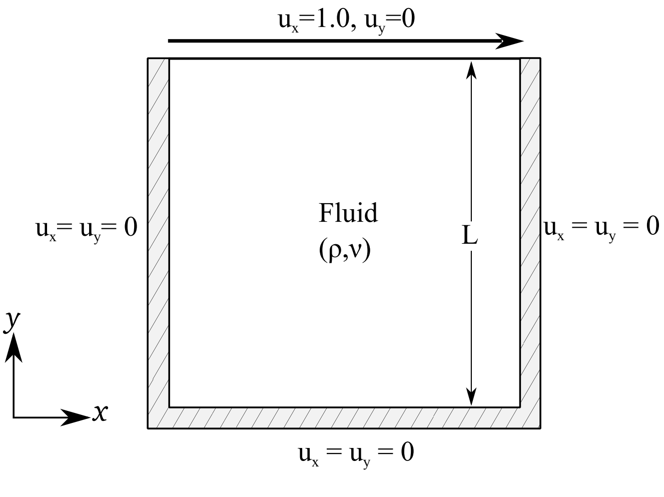


FIGURE 3.10: Lid-Driven cavity flow scheme.

On the other hand, for LBM, the moving top wall needs a special treatment. Zou and He [31] proposed to use the boundary condition explained in Sec. 2.6.3 to calculate the distribution function stream of the moving wall. The boundary conditions for the LDC are used as follows: Top(north) boundary condition; the Zou-He scheme is used, the velocity of the moving wall is fixed, $\mathbf{u}_x = u_x$= constant, the distribution function is calculated with this velocity in the x-component; north boundary condition for moving top wall, $\mathbf{u}_x = u_x$, $\mathbf{u}_y = 0$:

$$\begin{aligned}\rho =& f_0 + f_1 + f_3 + 2(f_2 + f_6 + f_5)\\ f_4 =& f_2\\ f_7 =& f_5 + \frac{1}{2}(f_1 - f_3) - \frac{1}{2}\rho\mathbf{u}_x\\ f_8 =& f_6 + \frac{1}{2}(f_3 - f_1) + \frac{1}{2}\rho\mathbf{u}_x.\end{aligned}$$

Three numerical studies were carried out for validation at different Reynolds numbers, Re=100, 1000 and 7500, with the Reynolds number of the cavity flow being defined by Re $= u_0 \cdot l_0/\nu$, where u is the velocity of the moving wall, l_0 is the characteristic length of the cavity (L Dimensionless length) and ν is the kinematic viscosity. The LBM results were validated by means of the most relevant study of the cavity flow by Ghia et al [1], who in the 1980's published a thorough numerical study based on a multi-grid method, analyzing the position of the vortex centers appearing in the left bottom corner, the right bottom corner and left top corner of the cavity.

3.1.2.1 Simulation setup for the grid independence test

For the grid-independence test, different grid sizes were selected depending on the Reynolds number. For Re = 100, the grids were $N = 128 \times 128$, $N = 160 \times 160$, $N = 260 \times 260$ and $N = 400 \times 400$. For Re = 1000 and Re = 7500, only the grids $N = 400 \times 400$ and $N = 600 \times 600$ were used, as smaller grids cannot ensure, at these values of the Reynolds number, the stability and accuracy of the method, even if MRT was used in all cases[2]. The discretization units and the rest of the relevant parameters for the LBM simulations are calculated as follows.

[2] The **Fortran 90** code for the Lid Driven cavity flow can be found in **Appendix**D

For Re = 100:

- The characteristic velocity of the moving wall is fixed at $u_{lbm} = 0.1$; the discrete density $\rho = 1.0$ is also imposed initially.
- The discrete length, $\Delta x = 1/N$ gives 0.0078125, 0.00625, 0.003847 and 0.0025 for the grid sizes specified above. The value of the kinematic viscosity in lattice units is calculated through the Reynolds equation 2.71, resulting in a range of $0.4 \leqslant \nu_{lbm} \geqslant 0.128$.
- The value of the relaxation time parameter is calculated using Eq. (2.19). The obtained values are in the range $1.7 \leqslant \tau \geqslant 0.884$ for the grid sizes used.

For Re = 1000:

- The two different grid sizes used in this case result in a spatial discretization of $\Delta x = 1/N = 0.0025$ for $N = 400 \times 400$ and 0.001667 for $N = 600 \times 600$. The values of the kinematic viscosity for the two grid sizes are $\nu_{lbm} = 0.040$ and 0.06, whereas the relaxation times are $\tau = 0.62$ and 0.68, respectively.

For Re = 7500:

- As the grids are the same as in the previous case, $\Delta x = 1/N$ are also the same, whereas the values of the kinematic viscosity for the two grids give $\nu_{lbm} = 0.00533, 0.0080$, and the relaxation times $\tau = 0.516$, 0.524.

3.1.2.2 Results and validation

The results presented in this section were obtained from the Lattice Boltzmann method and were compared with benchmark by Ghia et al [1] (1982), since it includes several results for various Reynolds numbers.

Figure 3.11 shows the magnitude of the dimensionless velocity of the y-component U_y for Re = 100. Different grid sizes were used, $N = 128 \times 128$, 160×160, 260×260 and 400×400. For $N = 128$ (yellow dashed line), the mean relative error with respect to the reference values is of $\bar{\epsilon} = 5.2$ %, with a maximum relative error of $\epsilon = 12$ %. In the case of $N = 160$ (green dashed line), the mean relative error decreased to $\bar{\epsilon} = 3.7$ % with a maximum value of $\epsilon = 8.4$ %. For $N = 260$ (red solid line), a mean relative error of $\bar{\epsilon} = 1.9$ % and a maximum of $\epsilon = 3.4$ % were obtained. For the most refined grid $N = 400$ (blue dashed line), the mean relative error decreased considerably to $\bar{\epsilon} = 1.0$ %, with a maximum relative error of $\epsilon = 1.7$ %.

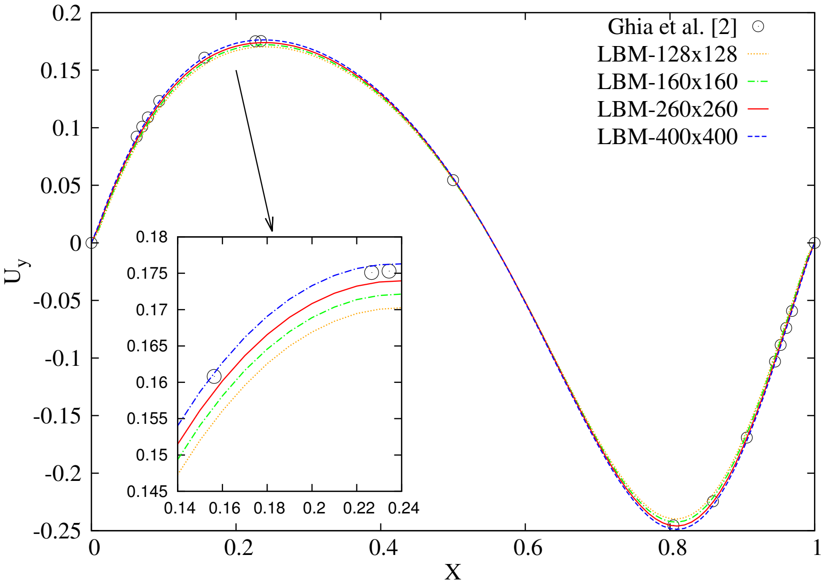


FIGURE 3.11: Grid-independence test for Re =100: y-component of the velocity along the centreline of the square cavity, U_y.

Figure 3.12 shows the location of the different vortices appearing in the cavity at Re=100. The streamlines of the dimensionless velocity were compared with the results by Ghia et al. [1].

At Re = 100, a primary vortex is located near the center of the cavity. At the bottom corners, two counter-rotating secondary vortices appear on the left and right side of the walls. Table. 3.3 at the end of the sub-section, shows the x-location and y-location of the vortex centers, as compared to the reference values. The agreement is good in general; the relative error for the primary vortex is $\epsilon = 0.49$ and 0.12 % respectively. For the left-corner vortex, a greater discrepancy was observed, with a relative error of $\epsilon = 13$ %, in the x-location. Seemingly, N is not large enough to adequately resolve the bottom left vortex. Therefore a smaller grid space (i.e., Δx) near the walls could be a solution, resulting in a local refinement at the corners. For the left vortex, bigger than the left one, the errors are much smaller: $\epsilon = 0.67$ and 0.00 % for the x- and y-location of the vortex center.

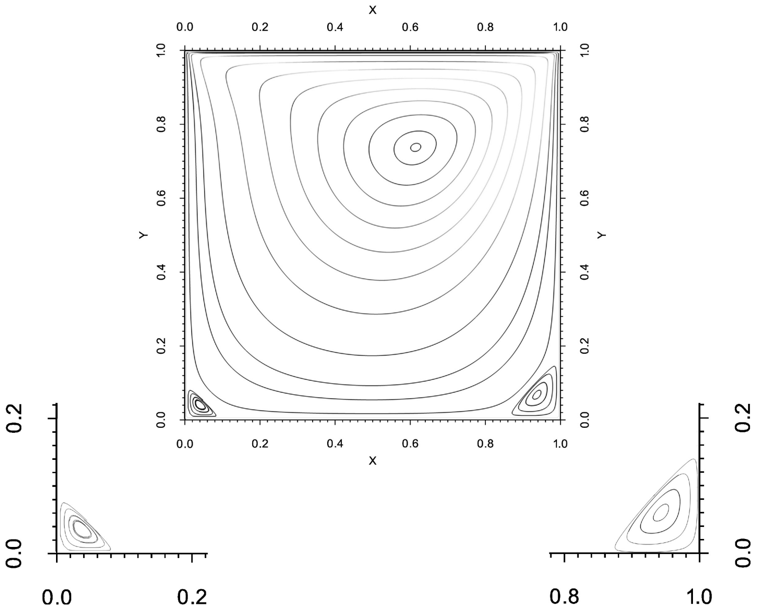


FIGURE 3.12: Streamlines for Re=100, LBM simulation.

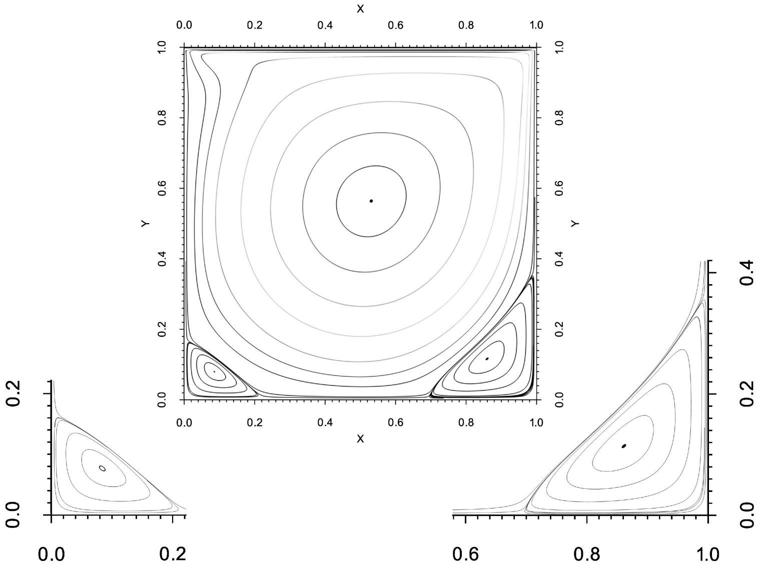


FIGURE 3.13: Streamlines for Re=1000, LBM simulation.

At Re = 1000 (see Fig. 3.13), the primary vortex has moved to the center of the cavity. Also, the left and right counter-rotating vortices in the bottom corners grow in size as compared with Re = 100. In the upper left corner, a small recirculation appears in the transient stages, but it is not further developed and is not visible in the steady state. In the study of Ghia et al. [1], this vortex appears in the steady flow solution when the value of the Reynolds' number approaches Re$\geqslant$2000. In this study, it was not necessary to test all Reynolds' number ranges, given that the main goal is to validate the LBM code by comparing the results obtained with those from other studies.

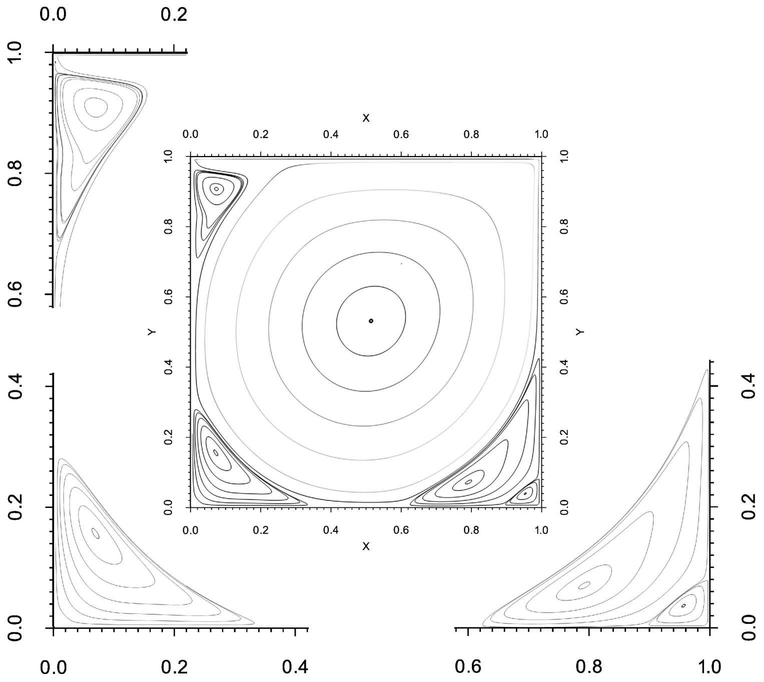


FIGURE 3.14: Streamlines for Re=7500, LBM simulation.

At Re=7500 (see the Fig. 3.14), the primary vortex is situated in the center of the cavity, with the two counter-rotating vortices of the left and right bottom corners presenting a substantial growth. A secondary vortex appears in the right top corner of the cavity, now fully developed in the steady state.

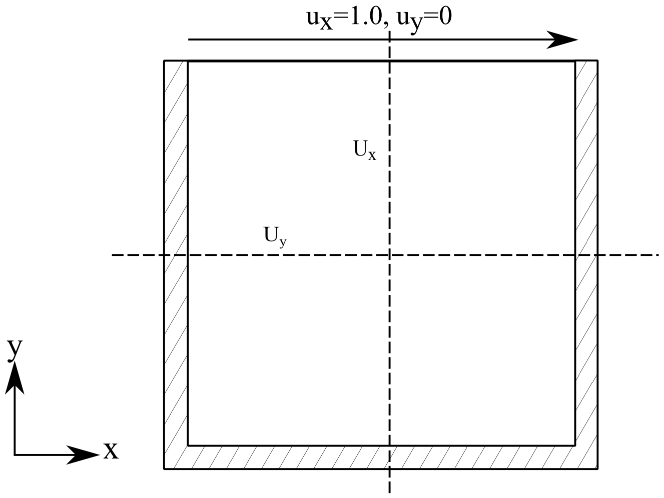


FIGURE 3.15: Scheme for the vertical and horizontal mid plane of x, y-component of the velocity.

In addition, the x-velocity (U_x) and y-velocity (U_y) profiles were analyzed in the center lines of the cavity (see Fig. 3.15 for a scheme). A good agreement with respect to the reference values is obtained in this case, as observed from Figs. 3.16 and 3.17. For Re = 100, using a grid of $N = 400$, the mean relative error obtained for U_x was $\bar{\epsilon} = 1.0$ %, and $\bar{\epsilon} = 1.7$ % for U_y. For Re = 1000, using a grid of $N = 600$, the mean relative error obtained for U_x was $\bar{\epsilon} = 1.9$ % and an $\bar{\epsilon} = 1.5$ % for U_y. For Re = 7500, the mean relative errors were $\bar{\epsilon} = 2.7$ % for U_x and $\bar{\epsilon} = 2.2$ % for U_y using a grid of $N = 600$. The velocity profiles at high Re becomes linear in most of the cavity: linear profiles in the central core confirm the approximately uniform vorticity region generated in the center of the cavity at high Re numbers.

Figure 3.18 shows the dimensionless velocity contour. The left side presents the contour of the dimensionless velocity field, while the right side shows the contour of the dimensionless vorticity field for each Re. For Re = 100, a higher region of vortex near the moving wall can be observed as the Re is increases. Several regions of high vorticity gradients are observed in the cavity when the Re increases, and the effect of the rotation generated a uniform vorticity in the center of the cavity with a contrary effect near the wall and to the top corner close to the moving wall. Moreover, some instability is produced in the corner of the cavity when Re$\geqslant$7500. In

Fig. 3.19, the dimensionless pressure contour is presented for the different cases. Notice that for Re = 100, the pressure is constant through the whole cavity. As the Re increases, the pressure gradient decreases in the center of the cavity (see Fig. 3.19(C))

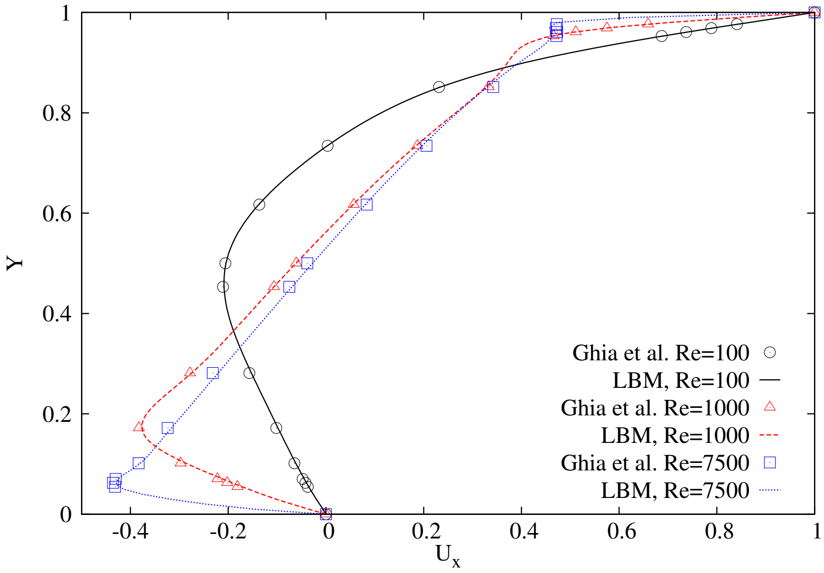


FIGURE 3.16: Dimensionless x-component of the velocity U_x along the vertical centerline of the cavity, Re = 100, 1000 and 7500.

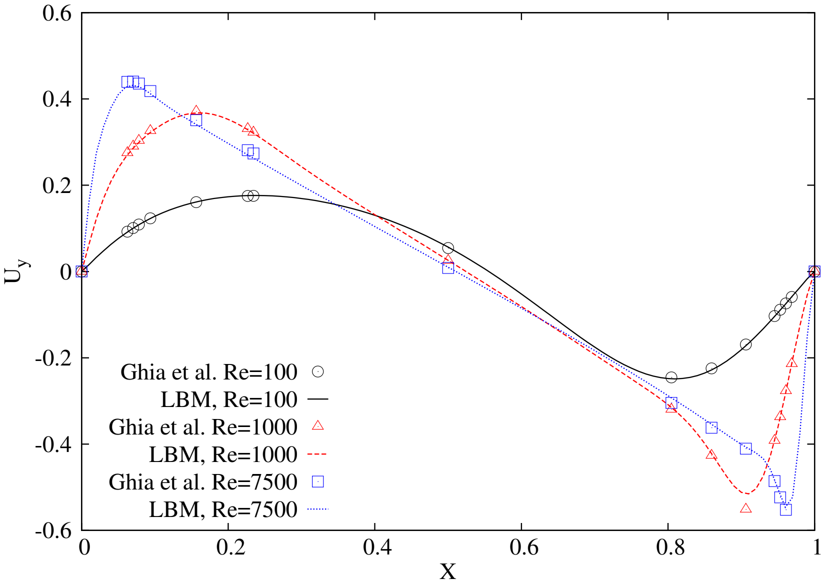


FIGURE 3.17: Dimensionless y-component of the velocity U_y along the horizontal centerline of the cavity, Re = 100, 1000 and 7500.

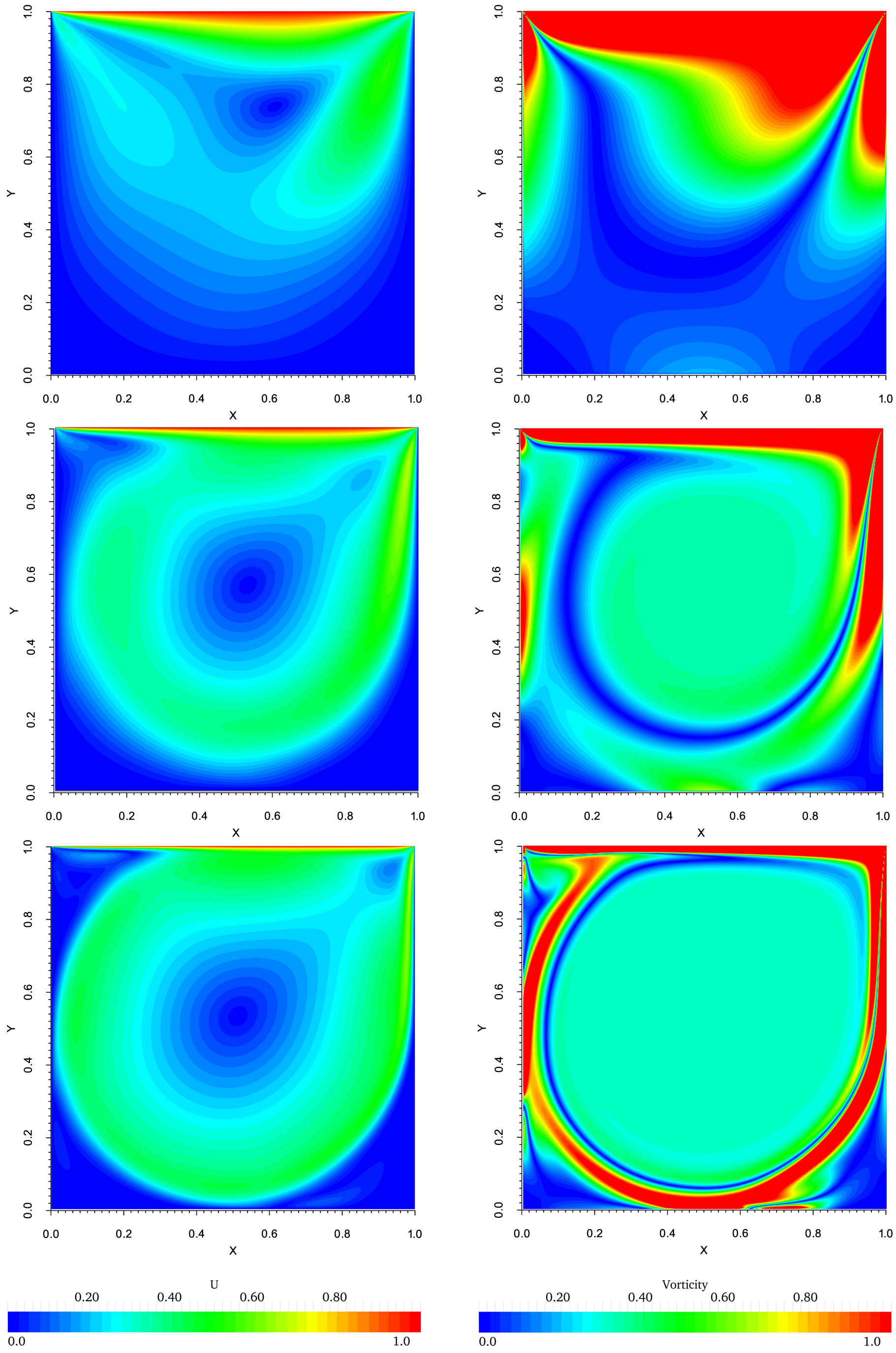
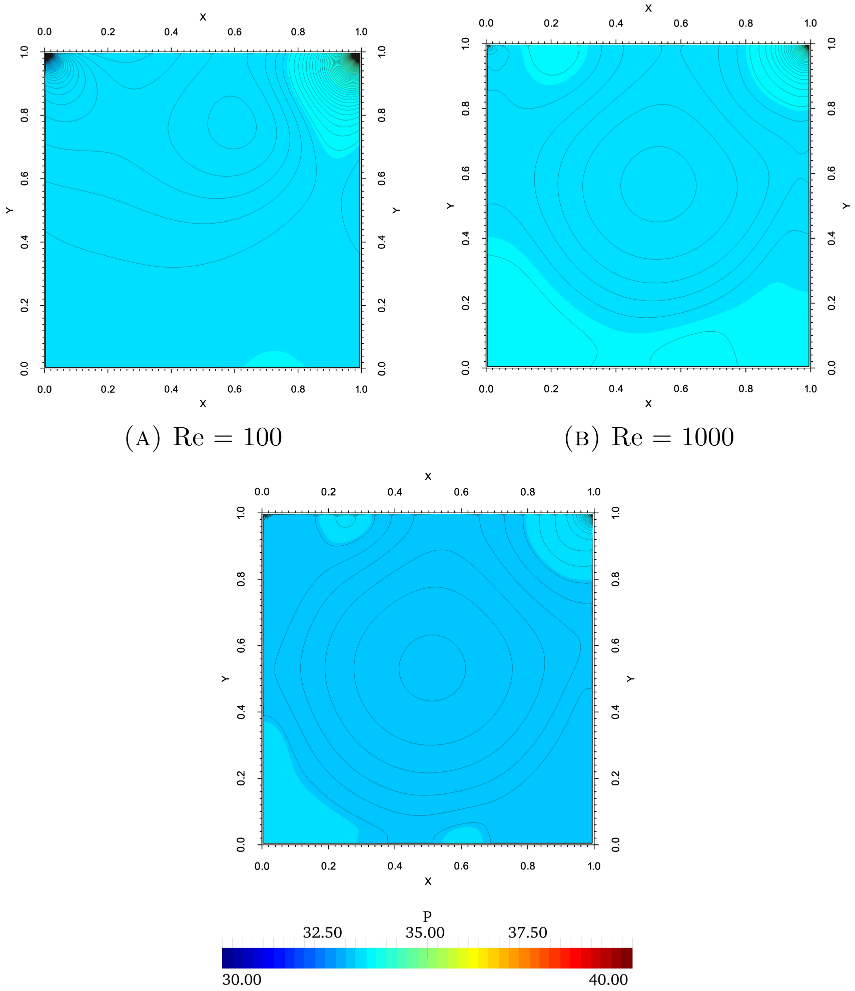


FIGURE 3.18: Contours of the dimensionless magnitude of the velocity (left) and vorticity (rigth); Re = 100 (top), Re = 1000 (middle) and Re = 7500 (bottom).



(A) Re = 100

(B) Re = 1000

(C) Re = 7500

FIGURE 3.19: A) Pressure contour for Re = 100, u_{lbm}=0.02 lattice units(ln), 256×256 nodes was used for this case. B) Re = 1000, u_{lbm}=0.02,400x400 nodes C) Re = 7500, u_{lbm}=0.02, 400×400 nodes.

TABLE 3.3: Location of the vortices found in LDC flow at Re=100, 1000 and 7500. LBM simulations and bibliographic data.

$Re = 100$

Primary	LBM	Ghia et al.[1]	ϵ %
Location, x,y	0.6142, 0.7353	0.6172, 0.7344	0.470, 0.12
First left vortex			
Location, x,y	0.0353, 0.0369	0.0313, 0.0391	12.78, 5.63
First right vortex			
Location, x,y	0.9390, 0.0625	0.9453, 0.0625	0.67, 0.00

$Re = 1000$

Primary	LBM	Ghia et al.[1]	ϵ (%)
Location, x,y	0.5316, 0.5639	0.5313, 0.5625	0.056, 0.25
First left vortex			
Location, x,y	0.0824, 0.0760	0.0859, 0.0781	4.07, 2.67
First right vortex			
Location, x,y	0.8633, 0.1125	0.8594, 0.1094	0.42, 2.84

$Re = 7500$

Primary	LBM	Ghia et al.[1]	ϵ %
Location, x,y	0.5126, 0.5321	0.5117, 0.5322	0.18, 0.019
Top left vortex			
Location, x,y	0.0688, 0.9116	0.0664, 0.9118	3.62, 0.022
First left vortex			
Location, x,y	0.0665, 0.1504	0.0645, 0.1504	3.10, 0.00
First right vortex			
Location, x,y	0.7903, 0.0662	0.7813, 0.0625	1.15, 0.06
Second right vortex			
Location, x,y	0.9526, 0.03829	0.9492, 0.0430	0.36, 10.96

3.1.3 3D - Poiseuille flow

The Poiseuille flow is a recurrently used problem to validate different methods. Due to its simplicity, it has been selected as a benchmark case to demonstrate the accuracy and stability of our 3D model. A simulation of the 3D-Poiseuille flow was carried out in order to validate our implementation of the D3Q19 model for three dimensional problems. A full-way bounce-back scheme was imposed on all solid walls as follows,

$$f_{i,n_{wall}} = f_{i,n_{wall},\ \mathit{opposite}}$$

where $i = 6, 13, 14, 16$ and 18 for the upper wall $i = 5, 11, 12, 15$ and 17 for the bottom wall and. At the outlet, an open boundary condition was imposed. The unknown distribution functions at the outlet were calculated with the second order extrapolation scheme explained in Sec. 2.6.

The double extrapolation is applied at the outlet node in the form:

$$f_{i,n} = 2 \cdot f_{i,n-1} - f_{i,n-2}$$

where here i are the discrete directions for the unknown distribution functions at the outlet, f_2, f_9, f_{10}, f_{12} and f_{14}.

Figure 3.20 displays the unknown distribution functions equation that need to be determinated, Sec. 2.2.

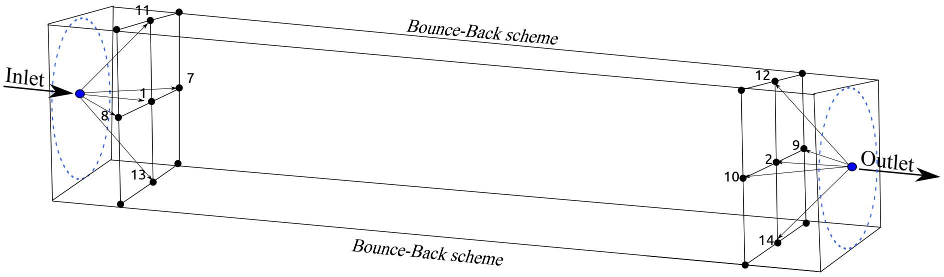


FIGURE 3.20: Scheme for Inlet/Outlet of the unknown distribution functions equations $f(x, y)$. 3D-Poiseuille flow.

For the inlet boundary condition, the density was fixed to $\rho = 1.0$ and the velocity $u(x, y)$, being the velocity at the pipe entrance. In addition, the unknown distribution functions f_1, f_7, f_8, f_{11} and f_{13} were calculated with the equilibrium distribution equation

3.1.3.1 Simulations setup:

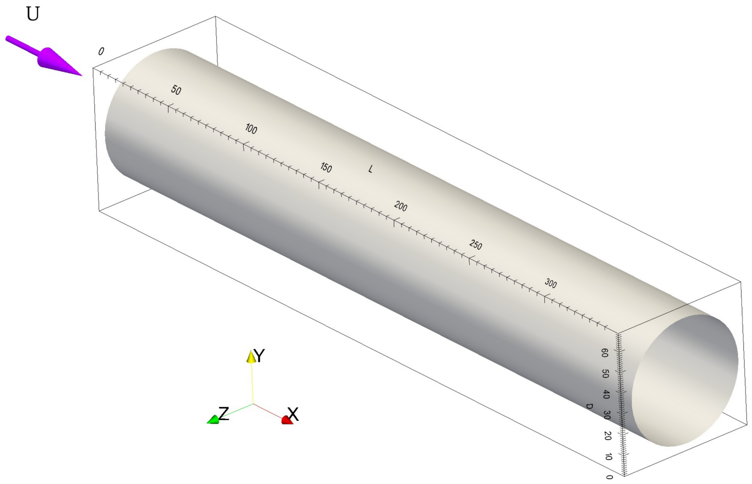


FIGURE 3.21: Domain for 3D-Poiseuille flow, D3Q19-LBM simulation.

Figure 3.21 shows the computational domain. The length of the channel needs to be long enough to ensure that the velocity profile is fully developed. In this case, the characteristic length of the system is the diameter of the pipe, which is equal to $D = 60$ nodes. Consequently, the length of the pipe should be $L \approx 6D$ nodes. Notice that the length of our computational domain is equal to $L = 350$ nodes. Considering the chosen grid size (i.e., 60 nodes in the y and z directions, and 350 in the x-direction), the values of the LBM velocity u_{lbm} are calculated through Δx and Δt. $\Delta x = 1/D = 1/60 = 0.01667$ and in this case, the velocity was chosen as $u_{lbm} = 0.02$, which complies with the restriction $u_{lbm} \leqslant 0.1$ to ensure the incompressible limit.

Since $u_{lbm} = \Delta t/\Delta x$, thus Δt can be calculate as $\Delta t = u_{lbm} \cdot \Delta x = 0.02 \cdot 0.01666 = 0.0003333$ (resulting in $\Delta t \approx \Delta x^2$).

The Reynolds number was fixed to Re=20, and the kinematic viscosity of lattice is calculated through Eq. 2.16.

$$\nu_{lbm} = \frac{0.02 \cdot 60}{20} = 0.06,$$

Finally, the relaxation time τ is obtained using Eq. (2.19):

$$\tau = 3 \cdot 0.06 + 0.5 = 0.68$$

3.1.3.2 Results and validation

The simulation results were converted into laboratory dimensionless variables and then compared with the analytical solution of the Poiseuille flow in a pipe. The analytical velocity profile in a pipe is given by

$$u(y) = u_{max}\left[1 - \frac{\| R - y \|^2}{R^2}\right], \tag{3.9}$$

where u_{max} is the maximum value of the velocity, calculated as $u_{max} = 2u_0$, u_0 being the characteristic velocity of the flow. R is the radius of the channel, and $D = 2R$ is the characteristic length of the channel.

Figure 3.22 displays the comparison of the LBM velocity profiles with respect to the exact solution. Three simulations were carried out; the first one (blue dash line) used $N = 40$ nodes in the characteristic length, giving an error relative to the exact solution of $\bar{\epsilon} = 3.9$ %. The second simulation (red dashed line) is the result for $N = 60$, showing a closer agreement with a mean relative error equal to $\bar{\epsilon} = 1.4$ %. The results for $N = 100$ (solid black line) are the most accurate, with a mean relative error of $\bar{\epsilon} = 0.7965$ %.

Figure 3.23 shows the results obtained from the D3Q19-LBM simulation.

Figure 3.23a displays the development of the velocity profiles along the pipe at different positions, as well as the 3D view in Fig. 3.23b.

In Fig. 3.23c a longitudinal section was made to show the velocity along the x-coordinate, and in Fig. 3.23d a section of the pipe near the outlet was made in order to observe a correct velocity field distribution, ensured by an appropriate choice of outlet boundary conditions (open boundary conditions in this case).

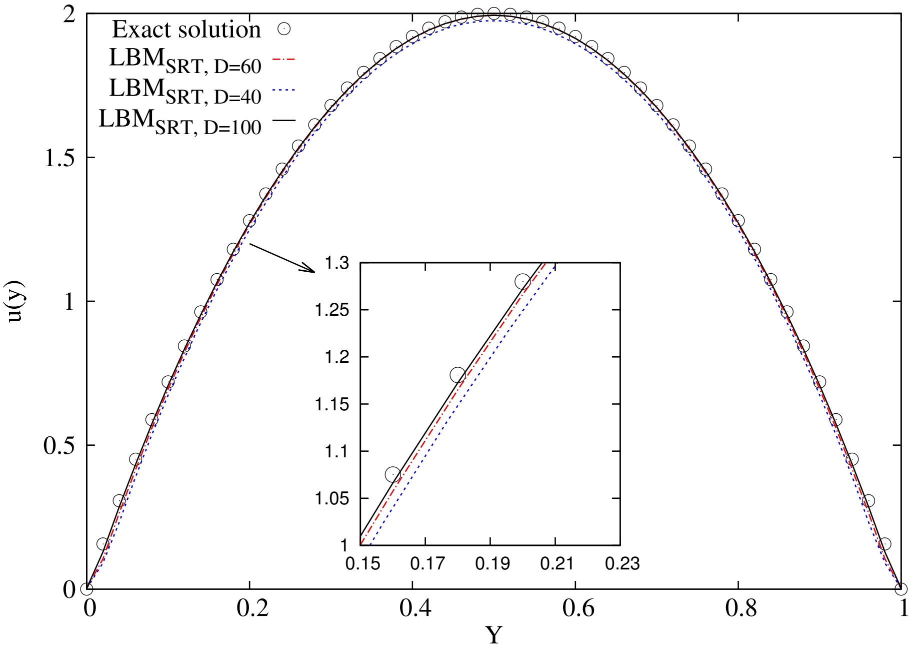


FIGURE 3.22: Velocity profiles of LBM simulation and analytical solution for 3D-Poiseuille flow.

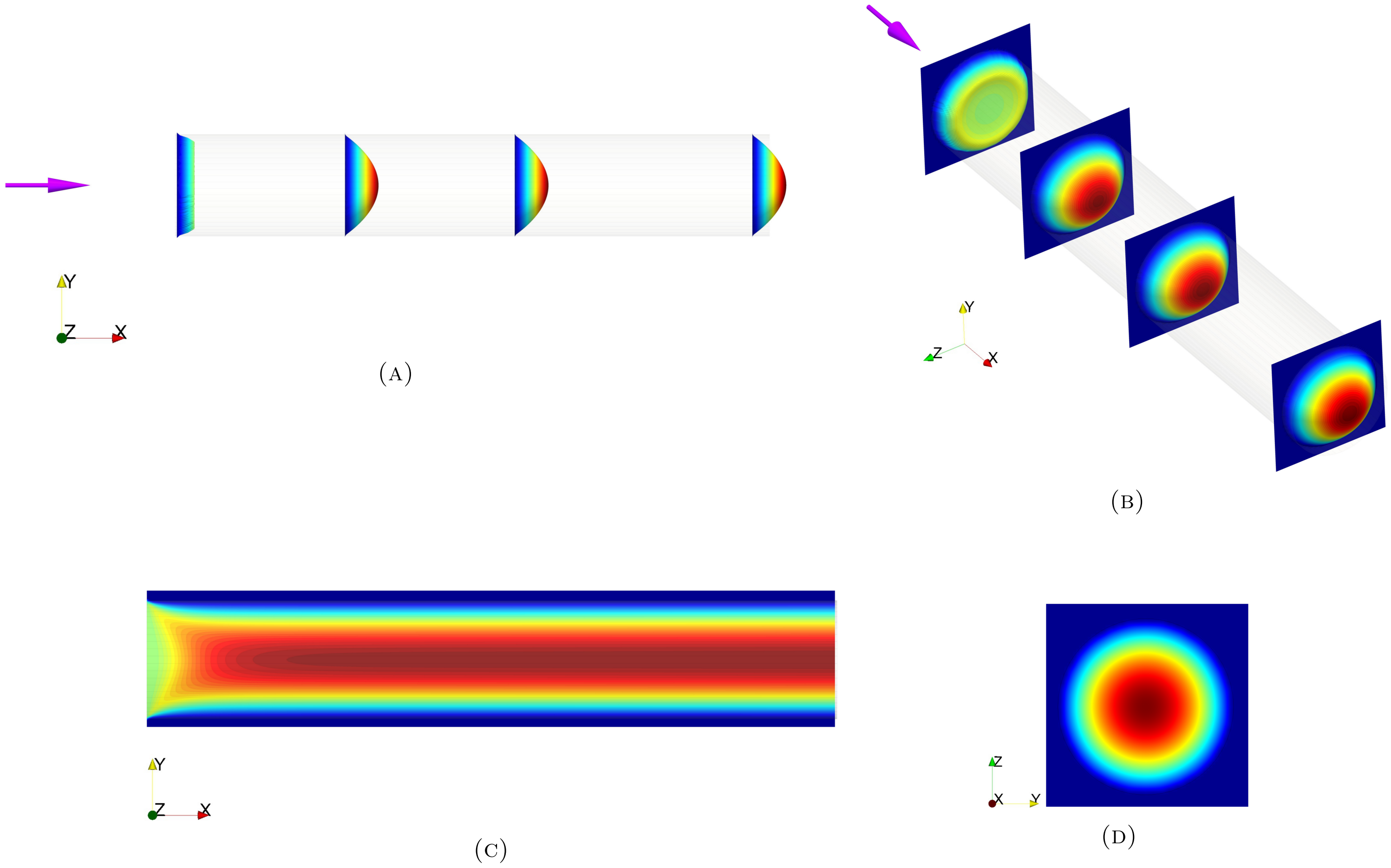


FIGURE 3.23: LBM simulation of the 3D-Poiseuille flow. A: shaded velocity profiles in the mid xy-plane; B: 3D-view of the velocity profiles along the pipe; C: a cut along the central xy-plane of the pipe; D: frontal view at the end of the pipe.

Chapter 4

LBM for Heat and Mass transfer phenomena

4.1 Advection-Diffusion equation

In this chapter, heat and mass transfer processes using LBM as the solution method are considered for advection-diffusion equation. In the process of advection-diffusion, the advection and diffusion occur simultaneously. Advection process, the governing advection equation for one-dimension for the transport of heat, mass and momentum can be expressed as

$$\frac{\partial \phi}{\partial t} + u\frac{\partial \phi}{\partial x} = 0 \tag{4.1}$$

where ϕ is the scalar quantity (e.g., energy, mass, momentum etc.) and u is the advection velocity, fluid flow velocity. for two-dimensional advection-diffusion equation is gives as

$$\frac{\partial \phi}{\partial t} + u\frac{\partial \phi}{\partial x} + v\frac{\partial \phi}{\partial yx} = \eta\left(\frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2}\right) \tag{4.2}$$

where η is the diffusion coefficient (i.e., mass diffusion or thermal diffusion for heat transport)

In order to validate the Lattice Boltzmann method in two-dimensions for some phenomena of heat and mas transfer processes, three simulations were carried out in the state-steady regimen for an incompressible fluid flow.

- Advection - Diffusion case
- Free or Natural Convection
- Mass Flux Case

In this section, part of the obtained results will be compared with their respective analytical solution, while the rest will be compared with results obtained by other authors, using different numerical methods as well as commercial software. For all studied cases, a uniform grid is used.

4.1.1 Advection - Diffusion equation for Lattice Boltzmann method

The advection-diffusion problem with or without source or reaction term is very common in engineering applications. In the Lattice Boltzmann method, this problem can be solved through Eq. (2.15), explained in Sec. 2.1,

$$A_i(x + \mathbf{c}_i \Delta t, t + \Delta t) = A_i(x,t) + \frac{1}{\tau_A}\left[A_i^{eq}(x,t) - A_i(x,t)\right], \tag{4.3}$$

where A_i is the discretized distribution function, τ_A and ω_i are the relaxation time parameter and weight factors for the model (see Section. 2.2) and $i = 0, 1, ..8$ are the discrete velocities for the model used.

In this example, a D2Q9-model is implemented, being C^A the concentration of the substance A that diffuses in the medium. The equilibrium distribution function A_i^{eq} is expressed as

$$A_i^{eq} = \omega_i C^A \left[1 + \frac{\mathbf{c}_i \cdot \mathbf{u}}{c^2}\right], \tag{4.4}$$

The relaxation time parameter τ_A for the concentration is calculated by the diffusion coefficient $\mathcal{D}^A$, corresponding to species A as:

$$\tau_A = 3\mathcal{D}^A + \frac{1}{2} \tag{4.5}$$

4.1.1.1 Advection - Diffusion problem: Gaussian pulse

The Gaussian pulse problem is the evolution of an initial concentration pulse inside of a rectangular domain with dimensionless length $x(0,2)$ and $y(0,2)$.

The initial Gaussian pulse is advected with constant velocity in a diagonal line across the domain, and is given by the following expression,

$$C(x,y) = a\text{exp}\left(-\frac{(x-x_0)^2}{\boldsymbol{D}_x} - \frac{(y-y_0)^2}{\boldsymbol{D}_y}\right) \tag{4.6}$$

where $C(x,y,0)$ is the Gaussian function of the concentration, a is the amplitude and x_0 and y_0 are the positions of the center of the pulse at $t=0$. The results obtained for the LBM simulation were compared with the exact solution. The analytical solutions for the concentration are given by

$$C(x,y,t) = \frac{1}{4t+1}\text{exp}\left(-\frac{(x-ut-x_0)^2}{\boldsymbol{D}_x(4t+1)} - \frac{(y-vt-y_0)^2}{\boldsymbol{D}_y(4t+1)}\right), \tag{4.7}$$

where $\boldsymbol{D}_x$ and $\boldsymbol{D}_y$ are the dimensionless diffusion coefficients in their respective x and y direction, t is the time, and u and v are the velocity components for the x and y directions.

4.1.1.2 Simulation setup

Two different uniform grid size were selected; $N = 80 \times 80$ and $N = 160 \times 160$. The LBM velocity for this case must be determined, however an initial guess to select a velocity can be calculated through Δx and Δt, to then obtain a characteristic velocity of the lattice.

Figure 4.1 shows the computational domain and the dimensionless parameters. The size of the domain is equal to x=y=2. The dimensionless diffusion coefficient is fixed to $\boldsymbol{D}_x = \boldsymbol{D}_y = 0.01$

The initial Gaussian pulse is generated using Eq. (4.6), positioned at coordinates $x = 0.5$ and $y = 0.5$ in the domain, with a maxima initial concentration $a = 1$. The diffusion coefficient is fixed to $\boldsymbol{D}_x = \boldsymbol{D}_y = 0.01$, and the characteristic velocity for the Gaussian pulse is selected as $U(x, y) = 1$. The concentration of the pulse was measured in the position $x = 1.5$ and $y = 1.5$ for the dimensionless time $t = 1$, and the characteristic length of the case is $L = 2$. Periodic boundary conditions were implemented in all boundaries (see Section. 2.6.2).

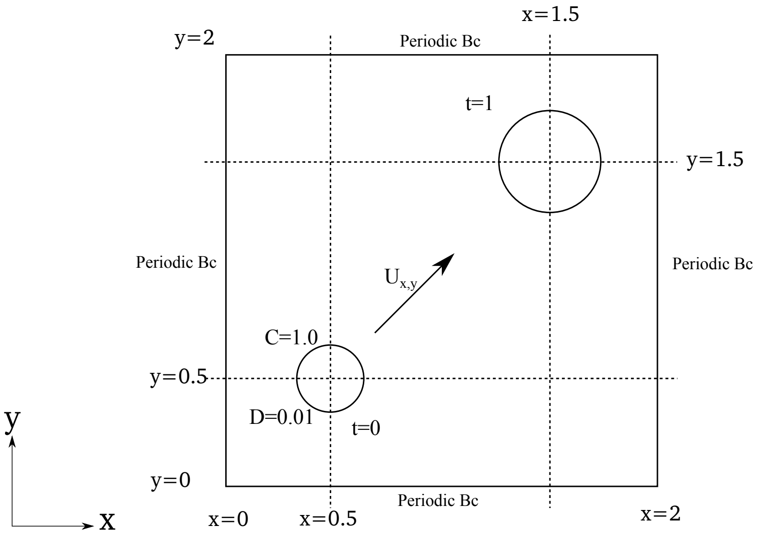


FIGURE 4.1: Domain scheme for the propagation of the Gaussian pulse (dimensionless parameters).

To proceed with the discretization of the lattice space and time scales, we calculate Δx and Δt using Eq. (2.64). Then, $\Delta x = 2/80 = 0.025$ for $N = 80 \times 80$ and $\Delta x = 2/160 = 0.0125$ for $N = 160 \times 160$, according to the procedure explained in Sec. 2.7 after Eq. (2.65), that is $\Delta t \sim \Delta x^2$. An approximation may also be useful in order to increase the accuracy of the method as $\Delta t \sim \Delta x^2 \cdot 2$, or $\Delta t \sim \Delta x^4 \cdot 2$, obtaining $\Delta t = 0.025^2 \cdot 4 = 0.0025$ for $N = 80$ and $\Delta t = 0.000625$ for $N = 160$. Therefore, the LBM velocity can be calculated through $u_{lbm} = \Delta t/\Delta x = 0.00125/0.025 = 0.05$ for the x- and y-directions.

The Reynolds number was fixed arbitrarily to Re=100 in order to calculate the respective viscosity of the lattice, using Eq. (2.71). Then, the viscosity is equal to $\nu_{lbm} = 0.05 \cdot 40/100 = 0.02$ for $N = 80 \times 80$ and $\nu_{lbm} = 0.05 \cdot 80/100 = 0.04$ for $N = 160 \times 160$, obtaining the relaxation time for both grid sizes as $\tau = 3 \cdot 0.02 + 0.5 = 0.56$ and $\tau = 3 \cdot 0.04 + 0.5 = 0.62$.

The LBM diffusion coefficient is calculated using the expression

$$\mathcal{D}^A = \frac{\Delta t}{\Delta x^2} \cdot \boldsymbol{D}, \tag{4.8}$$

where $\boldsymbol{D}$ represents the dimensionless diffusion coefficient mentioned previously, $\boldsymbol{D}_x = \boldsymbol{D}_y = 0.01$. Then,

$$\mathcal{D}^A = \frac{0.0025}{0.000625} \cdot 0.01 = 0.04, \qquad N = 80 \tag{4.9}$$

$$\tag{4.10}$$

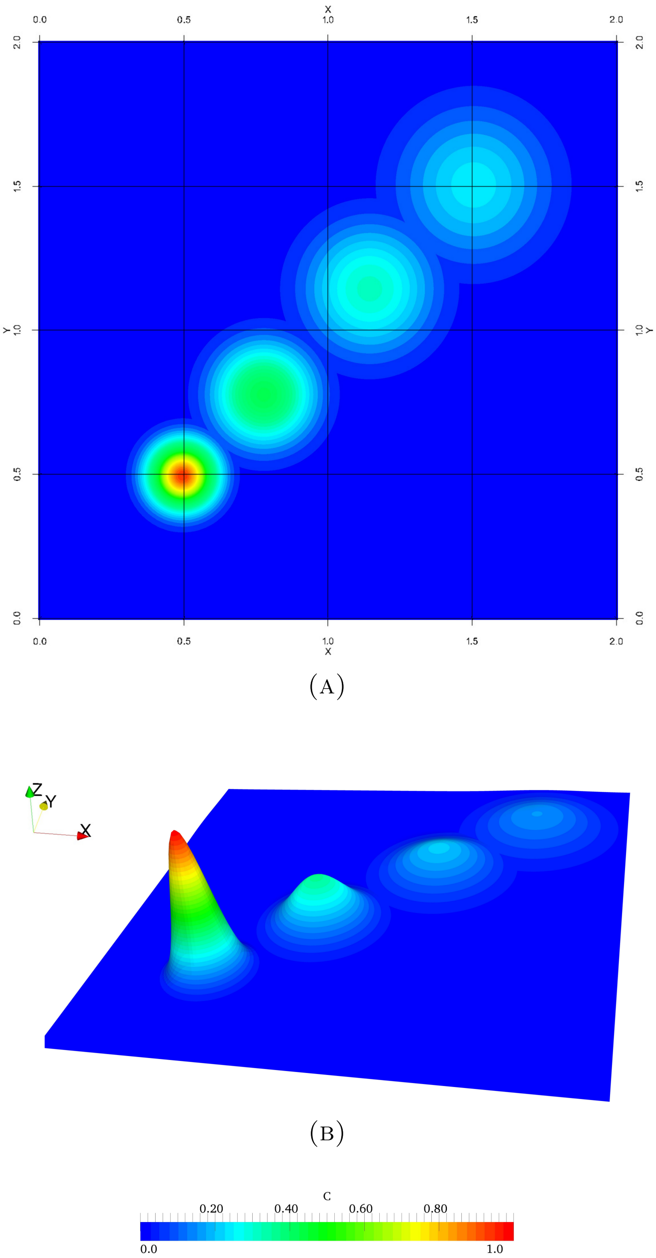
$$\mathcal{D}^A = \frac{0.000625}{0.00015625} \cdot 0.01 = 0.04, \qquad N = 160 \tag{4.11}$$

The relaxation time parameter for diffusion τ_A is calculated with the diffusion coefficient Eq. (2.18) previously obtained. Hence, $\tau_A = 3 \cdot \mathcal{D}^A + 0.5 = 3 \cdot 0.04 + 0.5 = 0.62$.

4.1.1.3 Results and validation

The analytical solution for the Gaussian pulse is compared with the LBM results for the two different grid sizes and for the MRT and the SRT models. Figure 4.2 displays the pulse located in the initial position x=y=0.5. The decrease in the initial concentration $C = 1$ at (x_0, y_0) due to diffusion is noticeable at $t = 1$, when the gaussian pulse has been advected to the position x=y=1.5, and where the maximum concentration is C=0.2. Figure 4.2b shows the two-dimensional domain, with a contour of the concentration with respect to time.

First, two simulations were carried out using the SRT model for $N = 80 \times 80$ and $N = 160 \times 160$. A deviation with respect to the analytic solution was observed, with the relative error for $N = 80$ being $\epsilon = 2.2\%$.



(A)

(B)

FIGURE 4.2: Contours of the dimensionless concentration field C: A) view of the pulse diffusing and moving from the initial to the final position. B) Embossed contours of the concentration.

For $N = 160$, an upgrade was achieved, obtaining a value of $\epsilon = 0.86\%$. Figure 4.3 shows the results for $N = 80$ (blue-dashed line) and $N = 160$ (red-dashed line).

Afterwards, other two simulations were ran using the MRT-model, obtaining a significant improvement with respect to SRT. Figure 4.3 shows the concentration profiles across a bisector for $N = 80$ (green-dashed line) and $N = 160$ (purple solid line) along with the exact result. The relative errors were $\epsilon = 1.7\%$ and $\epsilon = 0.10\%$ respectively, in the final peak concentration at t=1.

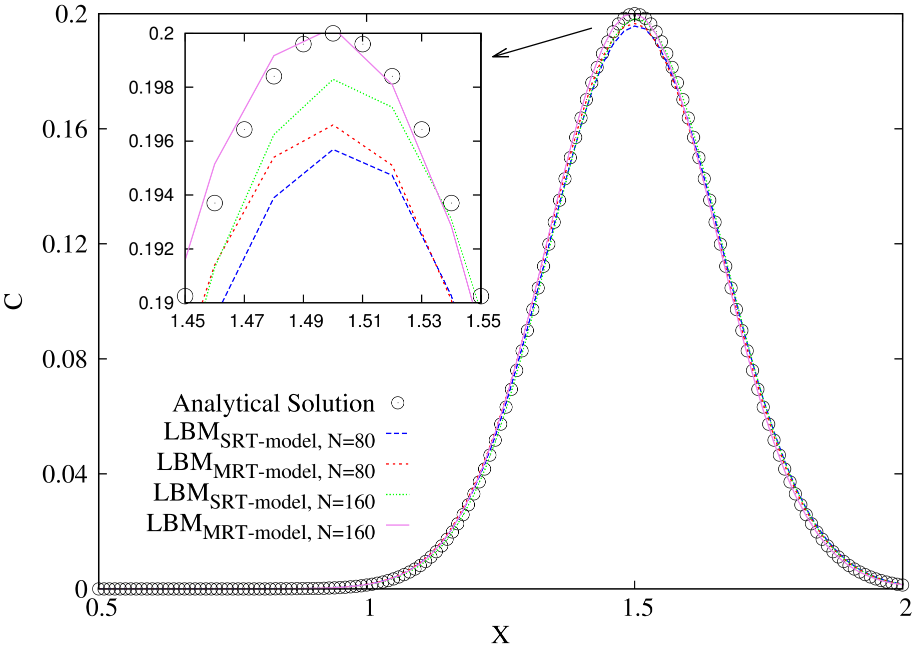


FIGURE 4.3: Results of the LBM simulations and their comparison with the analytical solution of the concentration pulse at the location $x = y$=1.5 at time t=1. Blue (SRT) and red (MRT) dashed lines are for $N = 80 \times 80$, and green dashed (SRT) and purple solid (MRT) lines are for $N = 160 \times 160$.

4.2 Convection

4.2.1 Incompressible Navier-Stokes Equations for Natural Convection

The incompressible form of the governing hydrodynamic equations may be derived from the general Navier-Stokes equation, which can be found in numerous bibliographic references [19, 20]. In this study, the steady-state natural convection will be considered. The buoyancy force term is simplified through the Boussinesq approximation, where the density in the buoyancy term is represented by a liner variation of the temperature. This means that all physical properties are constant, with the exception of the density, which is usually expressed in terms of the temperature,

$$\rho = \rho_0(1 - \beta\Delta T), \tag{4.12}$$

where ρ_0 is the bulk fluid density, ρ is the fluid density, $\Delta T = (T_0 - T_\infty)$ for T_0 is the local temperature of the fluid and T_∞ is the reference temperature and β is the coefficient of thermal expansion. For a single fluid incompressible flow, $\rho = \rho_0$, so that the thermal expansion can be calculated by $\beta = 1/T_\infty$ [20, 40].

The buoyancy term based on the approximation is included in the incompressible Navier-Stokes equation for two-dimensions in the momentum equations, and its formulation may be rewritten as follows,

The conservation of mass equation

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0, \tag{4.13}$$

the Boussinesq buoyancy model in the balance of x-momentum,

$$\rho\left(u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y}\right) = -\frac{\partial p}{\partial x} + \mu\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) + \rho g_x[1 - \beta(T_0 - T_\infty)] \tag{4.14}$$

the balance of y-momentum,

$$\rho\left(u\frac{\partial v}{\partial x}+v\frac{\partial v}{\partial y}\right)=-\frac{\partial p}{\partial y}+\mu\left(\frac{\partial^2 v}{\partial x^2}+\frac{\partial^2 v}{\partial y^2}\right)+\rho g_y[1-\beta(T_0-T_\infty)] \quad (4.15)$$

and the thermal energy transport equation in an incompressible gas,

$$u\frac{\partial T_0}{\partial x}+v\frac{\partial T_0}{\partial y}=\alpha\left(\frac{\partial^2 T_0}{\partial x^2}+\frac{\partial^2 T_0}{\partial y^2}\right) \quad (4.16)$$

The buoyancy force term per unit volume is $\beta g \rho_0 \Delta T$. It is sometimes preferred that the aforementioned equations are used in a dimensionless form, and thus these equations can be rewritten by selecting the characteristic quantities linked with the repective case of study (e.g., characteristic velocity u_0, characteristic length l_0, reference temperature T_∞, etc). Thus dimensionless parameters for the study of natural convection can be defined,

$$t=\frac{t_0}{l_0/\alpha}, \qquad X=\frac{x}{l_0}, \qquad U=\frac{u_0\cdot l_0}{\alpha} \quad (4.17)$$

$$T=\frac{T_0-T_\infty}{T_h-T_\infty}, \qquad P=\frac{p-l_0}{\rho_0\cdot\alpha^2}. \quad (4.18)$$

where T_h is the hot temperature here, the dimensionless governing equations with the body force can be rewritten as

$$\frac{\partial U}{\partial X}+\frac{\partial V}{\partial Y}=0, \quad (4.19)$$

$$U\frac{\partial V}{\partial X}+V\frac{\partial V}{\partial Y}-\frac{\partial P}{\partial Y}+Pr\left(\frac{\partial^2 V}{\partial X^2}+\frac{\partial^2 V}{\partial Y^2}\right)+RaPrT, \quad (4.20)$$

$$U\frac{\partial T}{\partial X}+V\frac{\partial T}{\partial Y}=\frac{\partial^2 T}{\partial X^2}+\frac{\partial^2 T}{\partial Y^2}. \quad (4.21)$$

In the study of natural convection, the buoyancy flow is characterized by the Rayleigh number Ra,

$$\mathrm{Ra}=\frac{g\beta(T_h-T_\infty)L^3}{\nu\alpha} \quad (4.22)$$

The Rayleigh number is related to the Prandtl and Grashof numbers by means of Ra= Gr·Pr, where

$$\mathrm{Gr} = \frac{g\beta\Delta T L^3}{\nu^2}, \qquad \mathrm{Pr} = \frac{\nu}{\alpha}. \tag{4.23}$$

Here β is the thermal expansion coefficient, $\Delta T = (T_h - T_\infty)$ is the difference between the surface temperature and reference temperatures, g is the gravity force, and ν and α are the kinematic viscosity and the thermal diffusion coefficient of the fluid. Therefore, the relative importance of the inertia and viscous forces depend on the Prandtl number; if Pr$\geqslant$1, the inertia term will be negligible and the viscosity term will balance the buoyancy term. Moreover if Pr$\leqslant$1, the inertia term is considerable and balances the buoyancy term in the steady state [19, 41].

4.2.2 Natural convection for the Lattice Boltzmann Method

To apply the Lattice Boltzmann Method to study natural convection phenomena, the distribution function equation Eq. (2.2) used previously for the momentum transfer needs to be modified.

Since a new term must be incorporated in the right hand side of the equation, the body force is added to the distribution function equation [4],

$$f_i(x + \mathbf{c}_i\Delta t, t + \Delta t) = f_i(x,t) + \frac{1}{\tau_v}\left[f_i^{eq}(x,t) - f_i(x,t)\right] + \Delta t F_i. \tag{4.24}$$

where the particular expression of F_i resorts to the Boussinesq approximation for the buoyancy force term, in which all fluid properties are constant except for the body force [42], where the density varies as $\rho = \bar{\rho}[1 - \beta(T_{lbm} - T_{lbm,\infty})]$. T_{lbm} is the local temperature, $T_{lbm,\infty}$ is the reference temperature, $\bar{\rho}$ is the corresponding density, and β the thermal expansion coefficient.

4.2.2.1 Force term

For natural convection driven flow, the force term is used as

$$\mathbf{F} = \rho \mathbf{g}_y \beta \Delta T \tag{4.25}$$

$\mathbf{g}_y$ being the gravitational vector and $\Delta T = (T_{lbm} - T_{lbm,\infty})$. The external force term is expressed in the LBM as follows [22],

$$F_i = \omega_i \mathbf{F} \cdot \mathbf{c}_i / c_s^2, \tag{4.26}$$

where $i = 0, ...8$ and $c_s = \mathbf{c}/\sqrt{3}$ are, as previously mentioned, the speed of sound in Lattice units. Therefore, Eq. (4.26) can be rewritten as

$$F_i = 3\omega_i \rho g_y \beta \Delta T. \tag{4.27}$$

There are different ways to add the external force to the collision operator in the LBM. Mohamad and Kuzmin [22] studied the results given by three different schemes, in particular, those suggested by L.S-Luo [43, 44], Shan and Chen [45] and Guo et al. [46]. After analyzing the velocities and the Nusselt number inside of a square cavity, the authors of these studies concluded that all schemes produced very similar results.

In this thesis, the scheme used by Mohamad et at. [22] has been implemented, using the expression for the external force as

$$F_i = 3\rho\omega_i g \beta (T_{lbm} - T_{lbm,\infty}). \tag{4.28}$$

Analogously, for the temperature field, a distribution function equation has to be implemented for the energy transfer as explained in Sec. 2.3, using Eq. (2.10).

4.2.2.2 Natural convection in a closed cavity

Natural convection in a close cavity can be found in many engineering situations, such as airflow in gaps, cavities of building walls, or air near heated surfaces. Several authors have studied this phenomenon, primarily with the aim of reducing heat transfer. Enhancing the accuracy of the different numerical methods used in simulations of heat transfer mechanisms has been another important goal in these studies. In the literature, several results and data about this case can be found, especially by G. De Vahl Davis. [2], who reported many interesting results of natural convection in closed cavities in two dimensions.

In order to validate our code, the results obtained with the Thermal Lattice Boltzmann method (TLBM) were compared with the method implemented by G. De Vahl Davis. [2], based on second-order, central difference approximations. The configuration of this case consisted in two differentially heated vertical side walls, with zero flux at top and bottom (i.e., adiabatic boundary condition), and zero velocity on the four lateral walls. Fig. 4.4 displays the computational domain, where H is the width and height of the cavity and g is acting along the direction of the negative y-coordinate. T_h and T_c are the dimensionless temperature for the hot and cold walls.

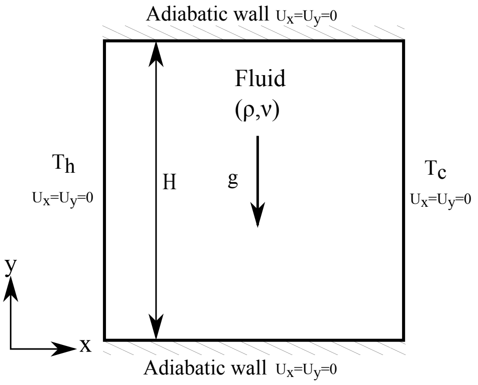


FIGURE 4.4: Scheme of the computational domain for the study of natural convection in a two-dimensional closed square cavity.

The most relevant dimensionless parameter when describing the convective heat transport processes is the Nusselt number Nu. This value is used to measure the heat transfer rate on a surface in a heated wall. The local Nusselt number is based on the cavity height or width, here $H = W$,

$$\mathrm{Nu} = -\frac{H}{\Delta T}\frac{\partial T}{\partial y}|_{wall}, \tag{4.29}$$

and the average Nusselt number is calculated by integrating Eq. (4.29) along the cavity height and dividing by the number of Lattice nodes,

$$\overline{\mathrm{Nu}} = \frac{1}{H}\int_0^H Nu\,dx. \tag{4.30}$$

4.2.2.3 Simulation setup

In accordance to the reference values [2], four values of the Rayleigh number were selected: Ra=10^3, 10^4, 10^5 and 10^6. In the study of natural convection in a closed cavity, the maximum horizontal velocity along the vertical mid-line of the cavity U_{max} and its location Y, and the maximum vertical velocity along the horizontal mid-line of the cavity V_{max} and its location X, are compared with the reference results. Also, the average Nusselt number $\overline{\mathrm{Nu}}$ over the heated wall is compared.

A grid-independence test was carried out for the purpose of observing the convergence of the variables as the grid is refined. Different grid sizes were selected, such as $N = 51 \times 51$, 61×61, 101×101, 151×151, 201×201, 301×301, and 401×401. No-slip boundary conditions were implemented on the four walls (in the form of the bounce-back scheme explained in Sec. 2.6.1). For the top and bottom walls, an adiabatic boundary condition was used. There are different schemes of the LBM for zero heat flux on a solid wall. For instance, the one used by A.A. Mohamed [24] which is implemented by writing the adiabatic boundary condition for the top wall as

$$g_{4,H} = g_{4,H-1}, \qquad g_{7,H} = g_{7,H-1}, \qquad g_{8,H} = g_{8,H-1},$$

whereas for the bottom wall, the unknown distribution functions can be calculated as

$$g_{2,1} = g_{4,2}, \qquad g_{6,1} = g_{6,2}, \qquad g_{5,1} = g_{5,2}.$$

Several authors use the Bounce-Back scheme (Section. 2.6.1) for the adiabatic boundary condition [5, 6, 47]. This scheme is applied according to the following equations. For the top adiabatic wall, it is calculated through

$$g_{7,H} = g_{5,H}, \qquad g_{8,H} = g_{6,H}, \qquad g_{4,H} = g_{2,H},$$

and for the bottom adiabatic wall, it reads

$$g_{2,1} = g_{4,1}, \qquad g_{6,1} = g_{8,1}, \qquad g_{5,1} = g_{7,1}.$$

In this thesis, we opted for the scheme suggested by A.A. Mohamed [24] fits better to the adiabatic boundary conditions. The results will be presented in the next paragraphs. The Prandtl number was fixed to Pr = 0.71(air) to ensure the stability of the method. Moreover, the viscosity of lattice was selected as $\nu_{lbm} = 0.01$, and the thermal coefficient on the lattice is calculated through Eq. (4.23), that is, $\mathrm{Pr} = \nu/\alpha$. Also, $\alpha = \nu/\mathrm{Pr} = 0.01/0.71 = 0.014085$ to guaranty the correct property of the fluid. The temperatures were established as $T_{hot,lbm} = 1$ for the heated wall and $T_{c,lbm} = 0$ for the cold wall[1].

4.2.2.4 Results and validation

Figure 4.5 displays the convergence for the dimensionless variables object of study and the number of nodes in the characteristic length of the cavity, for Ra $=10^3$. It was observed that for $N = 51$, the relative errors obtained for the maximum x- and y-components of the velocity, U_{max} and V_{max}, were $\epsilon = 3.6\%$ and $\epsilon = 4.0\%$, respectively. The average Nusselt number resulted in an error equal to $\epsilon = 0.37\%$. For $N = 401$, an upgrade was achieved, with relative errors of $\epsilon = 3.6\%$ for U_{max} and $\epsilon = 0.45\%$ for V_{max}, and an error in the average Nusselt number equal to $\epsilon = 0.044\%$.

The results obtained for other Rayleigh numbers, Ra $= 10^3$, 10^4, 10^5 and 10^6 are shown in Table. 4.1. In view of the results, the grid size $N = 401 \times 401$ was selected for all cases of study. As mentioned previously, the maximum velocities for the x- and y-component and the average Nusselt number were compared with the reference values. In addition, the location of the maximum velocities are presented in Table. 4.1 below.

A discrepancy was found in the location for the y-component of the velocity V_{max} for Ra $= 10^5$, with a relative error of $\epsilon = 1.51\%$. These discrepancies were also found by H.N. Dixit et al [47] and Y. Peng et al [42], who

[1] The LBM codes for the natural convection in a close cavity can be found in the **Appendix**E

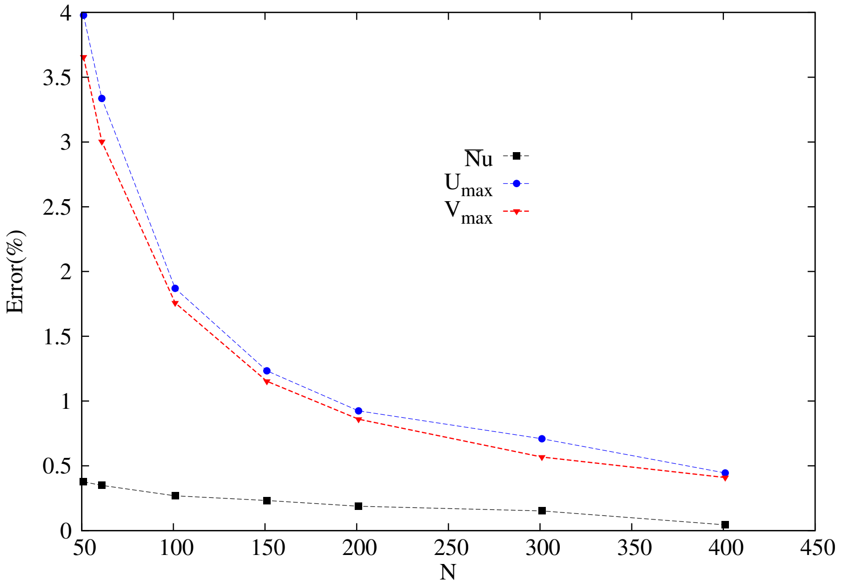


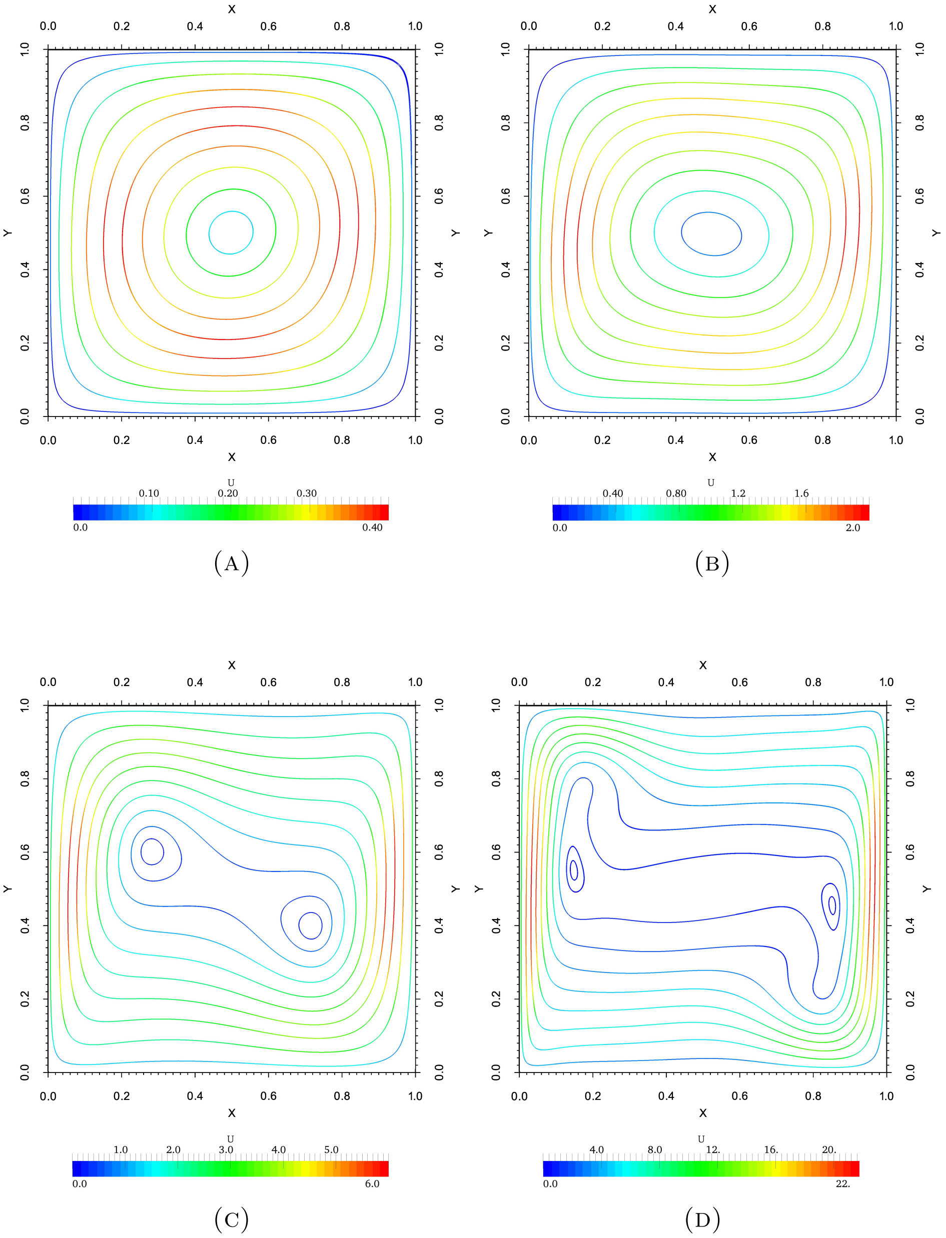
FIGURE 4.5: Grid-independence test for the problem of natural convection in a closed cavity, Ra = 10^3, with TLBM.

studied the natural convection in a closed cavity using both the LBM and simplified schemes of the TLBM. Figure 4.6 represents the streamlines of the velocity field in steady state, where the structures in the cavity for the range of the Ra $= 10^3, 10^4, 10^5, 10^6$ can be observed. In Figs. 4.6a,4.6b only one structure is observed, which is the one located in the center of the cavity. For Ra=10^5 (Fig. 4.6c) the structure in the center has split into two symmetrical structures; one located close to the top left corner and the other close to the bottom right corner. For Ra $= 10^6$ (Fig. 4.6d), these structures have moved closer to the hot and cold walls.

The dimensionless temperature contours are shown in Fig. 4.7. For Ra $= 10^3$, it can be deduced from the structure of the contour lines, that the heat transfer processes are dominated by conduction. As the Rayleigh number is increased, the contour lines are compressed close to the hot and cold walls respectively. A thermal stratification is generated, the flow in the cavity increases, as well as the velocity gradients, as the number of streamlines close to the wall indicate.

TABLE 4.1: Comparison of the LBM simulation with the benchmark solution of G.De Vahl Davis. [2], for Rayleigh number Ra = $10^3, 10^4, 10^5, 10^6$.

Ra		10^3	10^4	10^5	10^6
Grid used		(401x401)	(401x401)	(401x401)	(401x401)
U_{max}	LBM	3.6328	16.1211	34.8132	64.8046
	Vahl.D[2]	3.6490	16.1780	34.7300	64.6300
Error(%)		0.44	0.35	0.24	0.27
Y-Location	LBM	0.8134	0.8258	0.8564	0.8503
	Vahl.D[2]	0.8130	0.8230	0.8500	0.8500
Error(%)		0.05	0.35	0.17	0.01
V_{max}	LBM	3.6818	19.5354	68.2480	220.058
	Vahl.D[2]	3.6970	19.6170	68.5900	219.360
Error(%)		0.41	0.47	0.5	0.32
X-Location	LBM	0.1795	0.1243	0.0650	0.0374
	Vahl.D[2]	0.1800	0.1250	0.0660	0.0379
Error(%)		0.28	0.50	1.51	1.30
$\overline{Nu}$	LBM	1.1750	2.2471	4.5353	8.8310
	Vahl.D[2]	1.1700	2.2380	4.5090	8.8170
Error(%)		0.04	0.41	0.36	0.16



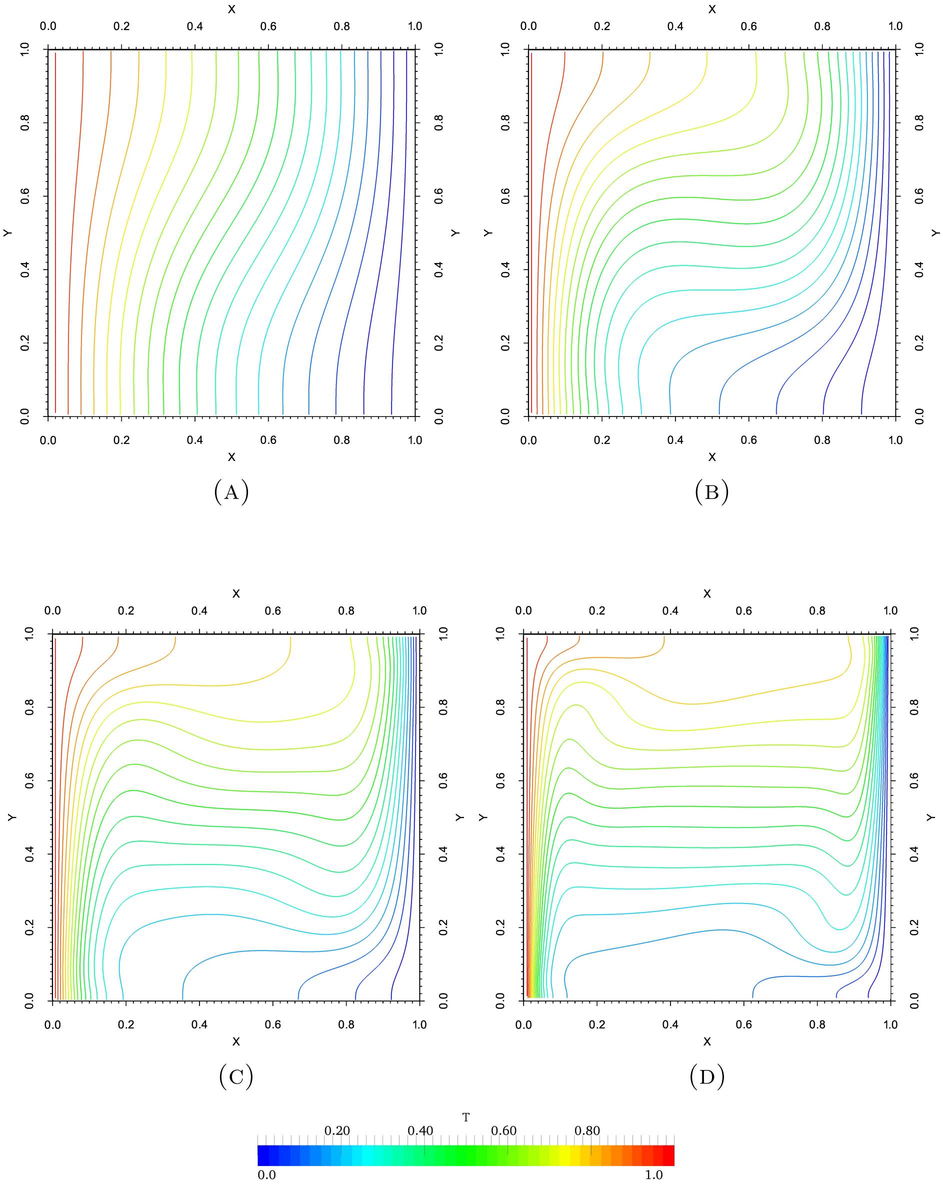
(A)

(B)

(C)

(D)

FIGURE 4.6: Streamlines of the dimensionless velocity. A) Ra $= 10^3$, B) Ra $= 10^4$, C) Ra $= 10^5$, D) Ra $= 10^6$, as obtained from LBM simulations using a grid $N = 401 \times 401$.



(A)

(B)

(C)

(D)

FIGURE 4.7: Contour of the dimensionless temperature field. A) Ra = 10^3, B) Ra = 10^4, C) Ra = 10^5, D) Ra = 10^6, as obtained from LBM simulations using a grid 401 × 401.

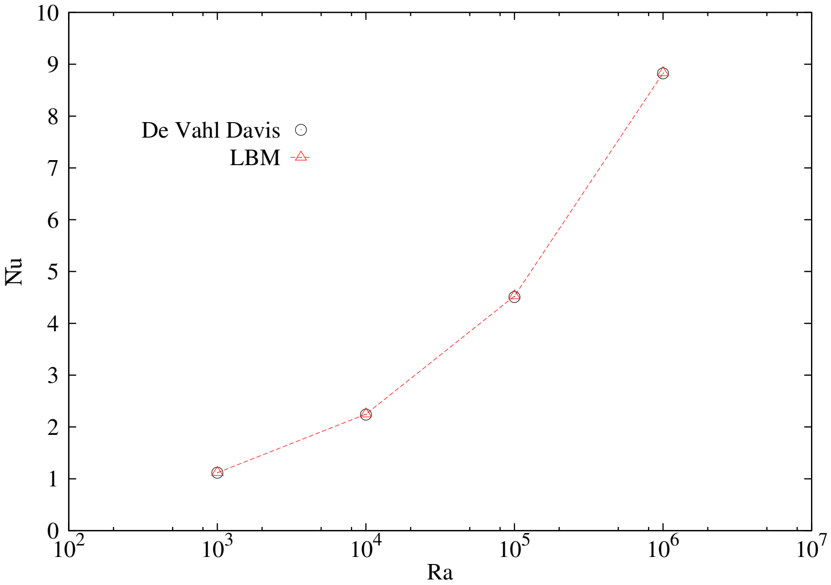


FIGURE 4.8: Averaged Nu as a function of Ra in the problem of natural convection in a closed cavity, LBM simulations on a 401 × 401 grid *vs.* reference values.

Figure 4.8 shows the comparison of the average Nusselt number $\overline{Nu}$ calculated over the heated wall (LBM on a 401×401 grid), taking as a reference the benchmark values obtained by De Vahl Davis [2]. As in Table. 4.1, the accuracy of the $\overline{Nu}$ results is demonstrated by obtaining a minimum discrepance of $\epsilon = 0.04\%$ (for Ra $= 10^3$) and a maximum of $\epsilon = 0.41\%$ (for Ra $= 10^4$).

4.3 Mass transfer

4.3.1 Lévêque problem for Mass flux

The Lévêque problem is a simple case of analysis of the mass transfer through a reactive surface (plate), where an engineering application is comparable to a chemical catalysis reaction. This problem consists in a two-dimensional Poiseuille flow fully developed in a laminar region, with a constant concentration at the entrance of the channel. The fluid flow passes through parallel plates separated by a fixed height. A fraction of the channel length contains a reaction region, where the concentration is determined by the mass flux through the boundary condition. In the Lévêque problem, in the region of the catalythic surface it is assumed that the reaction is infinitely fast and thus the dimensionless concentration is set to $C = 0$. The numerical solution is examined varying the value of the Péclet number Pe, which represents the relative strenght of diffusive and convective mass transport processes. Therefore, Pe is defined as the mass transfer rate occurring by advection to the mass transfer due to diffusion. For mass transfer, it is the product of Re and Sc. When Pe>1, the effects of convection exceed those of diffusion when determining the overall mass flux, and contrarily when Pe<1, the behavior is dominated by diffusion. The Péclet number is calculated as

$$\mathrm{Pe} = \frac{u \cdot L}{D_f} = \mathrm{Re} \cdot \mathrm{Sc} \tag{4.31}$$

where u is the velocity of the flow, L is the characteristic length, D_f is the diffusion coefficient of the substance and Sc represents the Schmidt number. Sc is the analogous to the Pr number for heat transfer processes, as a relation between the diffusion of momentum and the material diffusivity [20]. For a gas, both Sc and Pr∽ are equal to 0.7.

$$\mathrm{Sc} = \frac{\nu}{D_f} = \frac{\mu}{\rho D_f} \tag{4.32}$$

The interest in the Lévêque problem resides on measuring the total mass transfer in the reaction region on the whole length. As discussed above, the reaction on the catalysis surface is assumed to be infinity fast. Therefore, the mass transfer per unit length is characterized by the dimensionless parameter Sh (Sherwood number). The Sherwood number appears in many correlations, depending on the geometry in combination with other dimensionless parameters (e.g., Re or Sc). There are several correlations in the literature that can be used for a quick comparison of the results [20, 48]. Most notably over the whole length of a flat plate, Sh can be calculated according to the following correlation,

$$\mathrm{Sh} = \frac{kL}{D_f} = 0.664 Re_L^{1/2} \cdot Sc^{1/3} \tag{4.33}$$

where D_f is the diffusion coefficient of the component in the fluid, k is the convective mass transfer coefficient and L is the characteristic length of the system. The Reynolds number Re_L is based on the length of the flat plate (i.e., the catalysis region). Meanwhile, the Sherwood number Sh is analogous to the Nusselt number Nu for heat transfer processes, and can be expressed through the gradient of the dimensionless concentration on the flat plate. Thus,

$$\mathrm{Sh} = \frac{kL}{D_f} = \frac{\partial C}{\partial Y}|_{react-wall}, \tag{4.34}$$

and the average Sherwood number is calculated as $\overline{\mathrm{Sh}}$ as

$$\overline{\mathrm{Sh}} = \frac{1}{L}\int_0^L Sh\, dy. \tag{4.35}$$

4.3.1.1 Simulation setup

Figure 4.9 illustrates the computational domain of the Lévêque problem for mass flux through a plate. In order to reproduce the benchmark case, the velocity at the entrance of the channel is implemented as a fully developed profile.

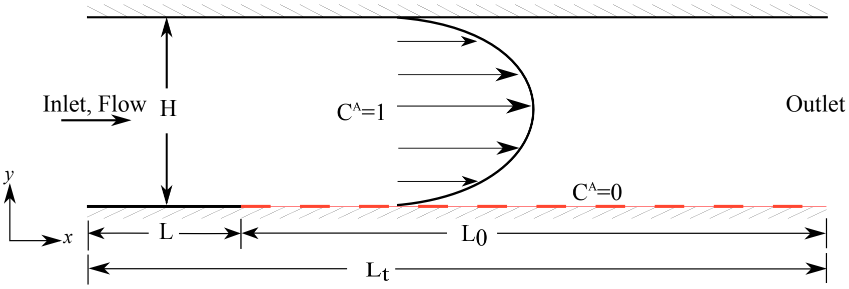


FIGURE 4.9: Scheme of the Léveque problem.

A constant concentration $C^A = 1$ is fixed at the entrance as a Dirichlet boundary condition (as explained in Sec. 2.6.4) which allows us to calculate the distribution functions $A_i(0, y)$ for the concentration,

$$\begin{aligned} A_1 &= C^A(\omega_1 + \omega_3) - A_3, \\ A_5 &= C^A(\omega_5 + \omega_7) - A_7, \\ A_8 &= C^A(\omega_8 + \omega_6) - A_6, \end{aligned}$$

where $w_1, w_3, w_5, w_7, w_8, w_6$ are the weights of the applied lattice model (D2Q9). At the outlet, an open boundary condition (see Section. 2.6.5) is implemented for the concentration field through the following settings,

$$\begin{aligned} A_{3,L_t} &= 2 \cdot A_{3,L_t-1} - A_{3,L_t-2}, \\ A_{6,L_t} &= 2 \cdot A_{6,L_t-1} - A_{6,L_t-2}, \\ A_{7,L_t} &= 2 \cdot A_{7,L_t-1} - A_{7,L_t-2}. \end{aligned}$$

Similarly, for the region of the plate where the mass transfer occurs, the value is fixed to $C^A = 0$ (infinitely fast reaction), applying the scheme for a Dirichlet boundary condition,

$$A_2 = -A_4, \qquad A_6 = -A_8, \qquad A_5 = -A_7.$$

the total length L_t of the domain is 10 times the height H of channel, the length of the catalyst layer in the flow direction is $L_0 = 9$, that is, $H = 100$ nodes for the height, and thus $L_t = 1000$ nodes in the x-direction. Also, the catalysis layer length is $L_0 = 100$. Then, it follows that $H = L$.

Different values of the Péclet number were set in order to compare with the values obtained by E. Holzbecher [7] on the Lévêque problem for a range of Pe between 0.01 and 10^8 using a package software (COMSOL). The range of Pe used in the LBM simulations is 0.01 to 2000. Once the values of Pe are chosen and using a range of characteristic lattice velocities $0.0001 \leqslant \mathrm{u}_{lbm} \leqslant 0.1$, they can be used in Eq. (4.31) to calculate the lattice diffusion coefficient $\mathcal{D}^A$,

$$\mathcal{D}^A = \frac{u_{lbm} \cdot L_t}{\mathrm{Pe}}. \tag{4.36}$$

Once the different values of $\mathcal{D}^A$ are obtained, the relaxation time parameter can be calculated for all cases using Eq. (2.18) for each of the diffusion values as ($\tau_A = 3 \cdot \mathcal{D}^A + 0.5$, SRT-model). As for the velocity field, the solution of the two-dimensional fully developed Poiseuille flow was directly implemented, as the study of the Lévêque problem is made in the steady state in a laminar flow with $Re = 10$.

Several grid sizes were used in order to obtain an appropriate relaxation time and to achieve the necessary stability and accuracy: $N = 52 \times 520$, 62×620, 72×720, 92×920, 102×1020 and finally 202×2020.

4.3.1.2 Results and validation

The results obtained from the LBM simulations are presented hereafter. Figura 4.10 shows the Sherwood number Sh as a function of the Péclet number Pe, along with the reference values by E. Holzbecher [7], where the package software (COMSOL) was used. The agreement is good up to Pe = 3000, where the value of the relaxation time parameter τ_A reached the minimum value permissible in the SRT model to ensure the accuracy and stability of the method ($\tau_A < 0.51$).

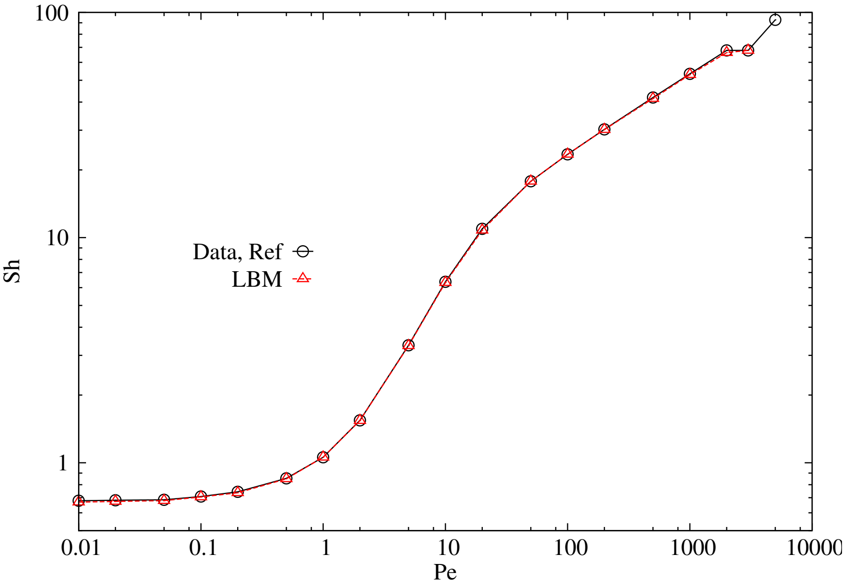
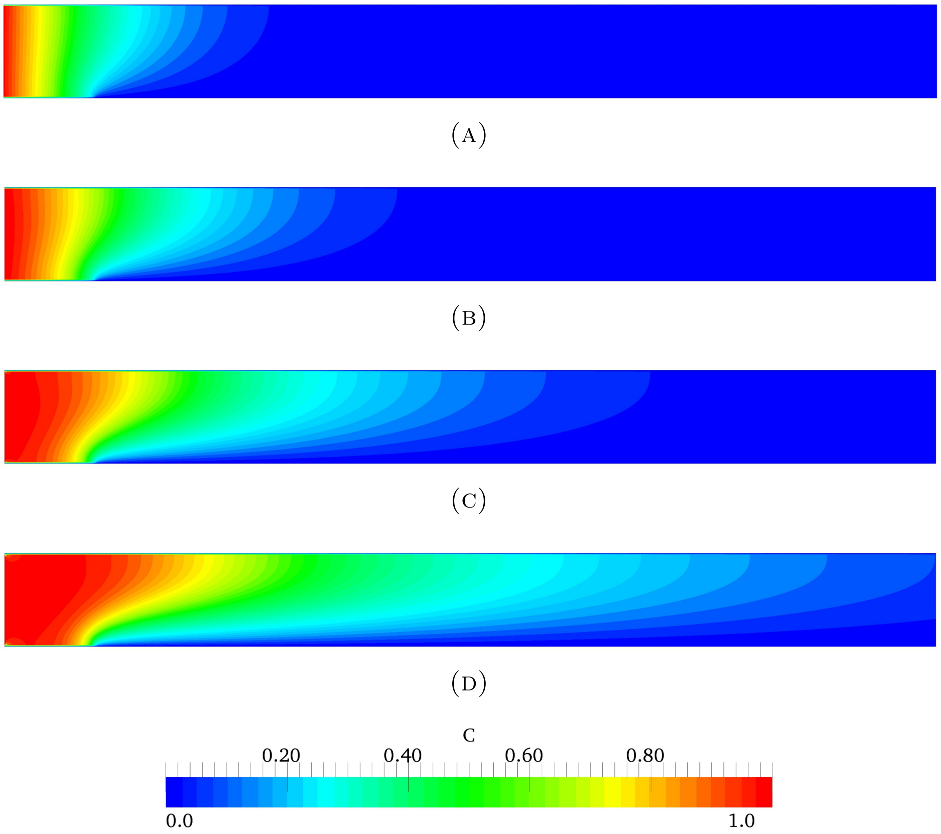


FIGURE 4.10: Results of the Sherwood number vs. Péclet number, as obtained from SRT-LBM simulations. The reference data are from E.Holzbecher [7], by using COMSOL software.

Figure 4.11 shows the contours of the dimensionless concentration in the channel for low Péclet numbers, Pe = 0.1, 2, 5, 10. For this range of Pe, it can be observed that the diffusion predominates over advection, the material boundary layer is large and thus the concentration is low across the channel.


(A)

(B)

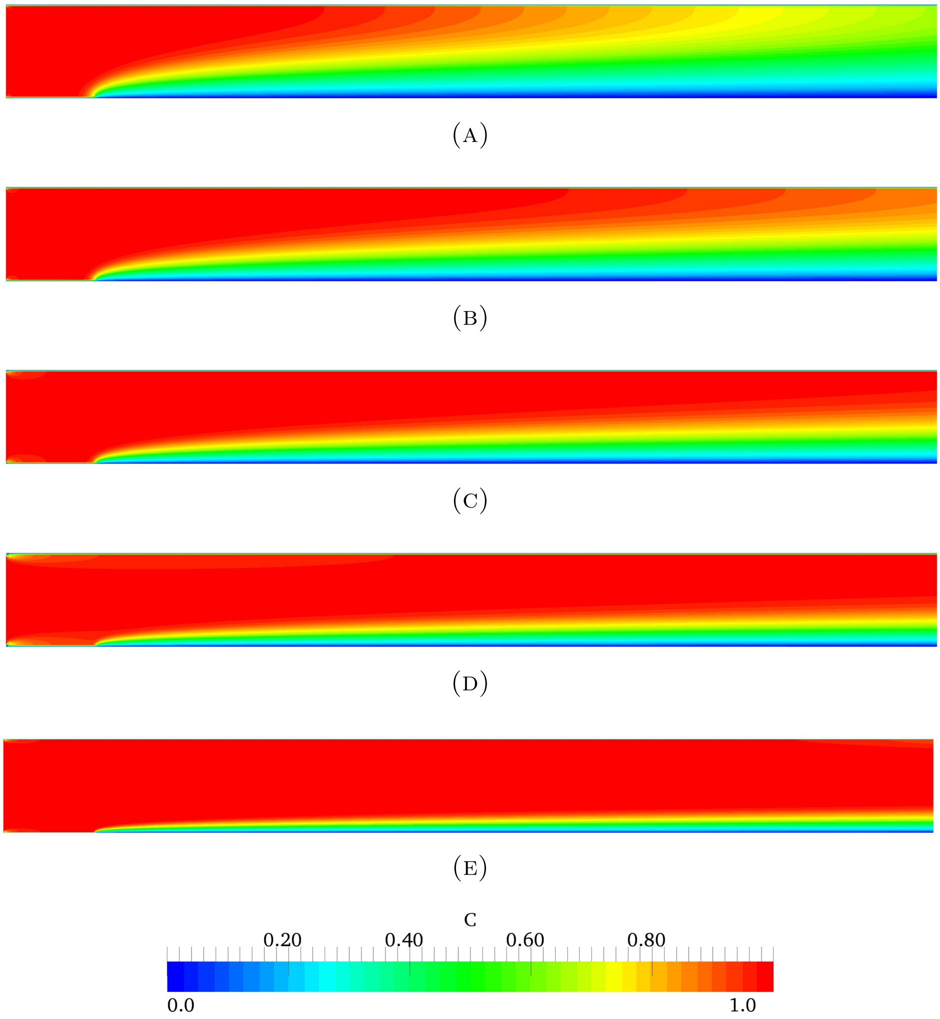
(C)

(D)

Figure 4.11: Contours of the dimensionless concentration C for low Péclet numbers: A) Pe = 0.1, B) Pe = 2, C) Pe = 5, D) Pe = 10, as obtained from LBM simulations.

Figure 4.12 displays the contours of the concentration for higher Péclet numbers, where the advection is dominant, the boundary layer is smaller and the concentration across the channel is larger, as Pe increases.

Table. 4.2 shows the results of the Sherwood number for the set of Péclet numbers studied, compared with the reference results through the relative error ϵ. Some discrepancies were found for Pe = 0.01, Pe = 20 and Pe = 500, with a maximum relative error of $\epsilon = 1.40\%$.



(A)

(B)

(C)

(D)

(E)

FIGURE 4.12: Contours of the dimensionless concentration C for higher Péclet numbers: A) Pe = 50, B) Pe = 100, C) Pe = 200, D) Pe = 500, E) Pe = 2000, as obtained from LBM simulations.

TABLE 4.2: Comparison of LB simulation with E.Holzbecher,[7] COMSOL Simulation for mass flux in a flat plate, Re=10

Pe	0.01	0.02	0.05	0.1	0.2	0.5	1
Grid used	(510x52)	(510x52)	(610x62)	(610x62)	(610x62)	(610x62)	(610x62)
u_{lbm}	0.0001	0.0001	0.0001	0.0005	0.0005	0.0020	0.0050
Sherwood number.(LBM)	0.6691	0.6740	0.6798	0.7049	0.7362	0.8495	1.0584
Sherwood number E.H,. [7]	0.6786	0.6819	0.6852	0.7087	0.7432	0.8534	1.0590
Error(%)	1.40	1.16	0.80	0.54	0.94	0.47	0.058

Pe	2	5	10	20	50	100	200
Grid used	(510x52)	(510x52)	(610x62)	(610x62)	(610x62)	(610x62)	(610x62)
u_{lbm}	0.0050	0.0050	0.0050	0.0050	0.0500	0.0500	0.0500
Sherwood number.(LBM)	1.5391	3.3095	6.3133	10.7909	17.8015	23.4450	30.1677
Sherwood number E.H,. [7]	1.5428	3.3260	6.3683	10.9430	17.8000	23.4490	30.2570
error(%)	0.24	0.50	0.86	1.39	0.08	0.01	0.3

Pe	500	1000	2000	5000	10000
Grid used	(72x720)	(92x920)	(102x1010)		
u_{lbm}	0.0900	0.1000	0.1000		
Sherwood number.(LBM)	41.4480	52.9284	67.8116		
Sherwood number E.H,. [7]	41.9140	53.376	67.7930	92.731	117.35
Error(%)	1.12	0.84	0.02		

Chapter 5

LBM for laminar mixed convection in an open cavity

5.1 Introduction

In the section, LBM is applied to study the heat transfer processes in laminar mixed convection. An open cavity with a heated bottom wall will be considered. This particular case can be found in various engineering applications such as solar devices, heat exchangers, nuclear reactors, electronic systems, etc. Experimental studies of buoyant flow in open cavities can be found in the literature [49, 50]. Mixed convection in cavities has received less attention, but several authors have previously analyzed this problem: in the early '90s, Papanicolaou et al. [51, 52, 53] studied the mixed convection in a rectangular enclosure, analyzing the flow for different inlet/outlet and heat source locations as it happens in a cooled electronic device. Manca et al. [54] studied the natural and mixed convection in rectangular cavities with a T-type of geometry and the effect of the position of the heated wall. Stiriba el at. [55, 56] carried out a numerical study of mixed convection for incompressible laminar flow past an open cubical cavity, showing that it exhibits a three-dimensional structure. These authors presented results for steady and unsteady laminar regimes in three dimensions, where the effect of the buoyancy force was analyzed for a range of Reynolds and Richardson numbers. The results obtained by previous authors clearly show that the geometry of the problem has a great influence in the flow and the instabilities observed.

In this section the Thermal Lattice Botlzmann method (TLBM) approach is used, combined with ANSYS-FLUENT simulations to validate our findings. Using the same T configuration as in the three-dimensional case[55, 56], we will show how in two dimensions and above a certain value of the Richardson number, i) the main vortex in the cavity is split into two cells and 2) an unsteady flow regime is found, exhibiting an intermittent pattern characterized by the periodic emission of hot plumes towards the channel outlet, originating from the upstream vertex of the cavity.

As usual in natural convection problems, the evolution of the distribution function includes the body force term as in Eq. (2.2) of Sec. 2.1. In the case of the temperature field, a distribution function equation accounts for the energy transfer mechanisms, as explained in Sec. 2.3. As well, the addition of an external force implies a modification of the equation for the macroscopic momentum, Eq. (2.9) of Sec. 2.3 [35]. The external force is computed as suggested by Mohamad et at. [22], as $F_i = 3\rho\omega_i g\beta(T_{lbm} - T_\infty)$ and added as follows

$$\rho\mathbf{u}(x,t) = \sum_i \mathbf{c}_i f_i(x,t) + \frac{F_i}{2}. \tag{5.1}$$

The dimensionless buoyancy force introduces the Richardson number Ri. The Richardson number, which determines the relative importance of forced convection vs. natural convection, is defined as

$$\mathrm{Ri} = \frac{g\beta(T - T_\infty)l_0}{u^2}, \tag{5.2}$$

where β is the thermal expansion coefficient of the fluid, u is the characteristic velocity of the flow, l_0 the characteristics length and g the gravity force. The Richardson number can also be expressed as a function of the Grashoff and Reynolds numbers as $\mathrm{Ri} = \mathrm{Gr}/\mathrm{Re}^2$.

The model system consists of an open square cavity with a dimensionless length L Fig. 5.1. The bottom of the cavity is heated at constant temperature T_h, while the other walls are adiabatic (zero flux temperature), at a

Prandtl number for the fluid equal to Pe = 0.7. The channel is of length $3L$, in order to minimize the effect of recirculation in some cases. For the inlet boundary and the top wall, we fix the temperature at T_∞, and an uniform inlet velocity u_0 is set. The choice of the velocity profile at the inlet mimicks the flow from a convergent nozzle, which is relatively easy to reproduce experimentally. Moreover, an inlet boundary condition where a fully developed velocity profile is set does not alter the features of the flow and the results obtained are similar.

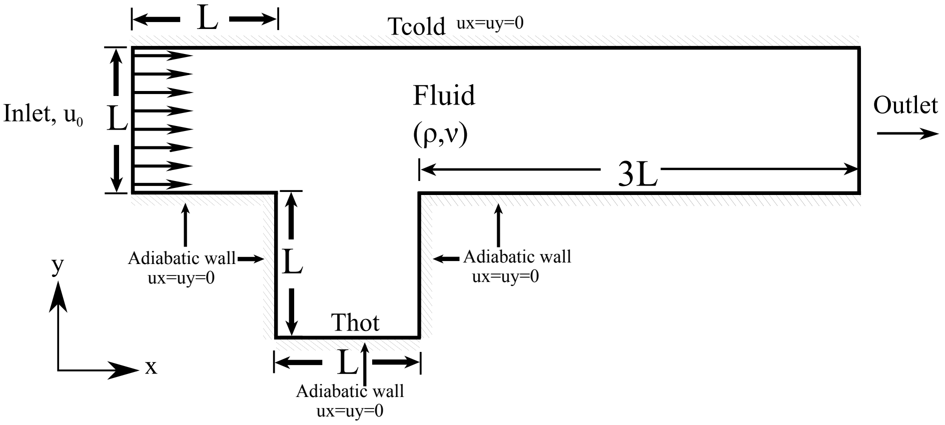


Figure 5.1: Illustration of the open cavity with boundary conditions.

5.1.1 Simulation setup

The values of $T_{0,hot}$ and u_0 were varied to give the different cases for the Reynolds numbers Re = 50, 100, 200, 400, 600, 1000 and Richardson numbers Ri = 0.01, 0.1, 1, 10 considered. Temperature T_∞ is chosen as 298.1450 K (room temperature) in real units, and $T_{\infty,lbm} = 0$ and $T_{h,lbm} = 1$ for the LB simulations.

A non-slip, bounce-back scheme (Sec. 2.6.1) is used and adiabatic boundary conditions are applied at all the other boundaries using the model implemented by A.A Mohamed [24]. The velocity is specified at the inlet as an uniform flow field along the channel direction. The Zou-He approach (see Sec. 2.6.3) is implemented to calculate the unknown distribution functions of the momentum field as

$$
\begin{aligned}
f_1 =& f_3 + \frac{2}{3}\rho \mathbf{u}_x \\
f_5 =& f_7 - \frac{1}{2}(f_2 - f_4) + \frac{1}{6}\rho \mathbf{u}_x \\
f_8 =& f_6 + \frac{1}{2}(f_2 - f_4) + \frac{1}{6}\rho \mathbf{u}_x,
\end{aligned}
$$

for the inlet, whereas an open boundary condition is used for the outlet to avoid reverse flow (see Sec. 2.6.5). Thus the unknowns at the outlet are extrapolated as

$$
\begin{aligned}
f_3 =& 2\cdot f_{3,L_{out}-1} - f_{3,L_{out}-2} \\
f_6 =& 2\cdot f_{6,L_{out}-1} - f_{6,L_{out}-2} \\
f_7 =& 2\cdot f_{7,L_{out}-1} - f_{7,L_{out}-2}
\end{aligned}
$$

The boundary conditions for the temperature are treated similarly as in the previous case studied when the value of the scalar is known: a Dirichlet boundary condition (Sec. 2.6.4) at the heated bottom wall which sets $T_{h,lbm} = 1$ in LBM units, is supplemented with the prescriptions for the unknowns g_6, g_2 and g_5. These are calculated in the heated nodes as follows

$$
\begin{aligned}
g_6 &= T_{h,lbm}(\omega_6 + \omega_8) - g_8 \\
g_2 &= T_{h,lbm}(\omega_2 + \omega_4) - g_4 \\
g_5 &= T_{h,lbm}(\omega_5 + \omega_7) - g_7.
\end{aligned}
\tag{5.3}
$$

For the outlet, an open boundary condition is used for the temperature, with a similar approach to that applied to the momentum distribution functions (i.e., f_3, f_6, f_7 above).

A grid independence test was carried out at the highest values of the Reynolds and Richardson numbers, Re= 1000 and Ri = 10. Figure 5.2 shows on the left hand side, the x-velocity component in the vertical mid line, as well as the y-velocity component in the horizontal mid line inside the cavity. On the right hand side, it shows the average dimensionless temperature along the vertical mid line of the cavity.

Results are obtained for three different grid refinements, where convergence is observed. The maximum discrepancy in the temperature is of 0.77%, between the intermediate (N = 151) and finest grid (N = 201). A similar convergence is obtained for the velocities.

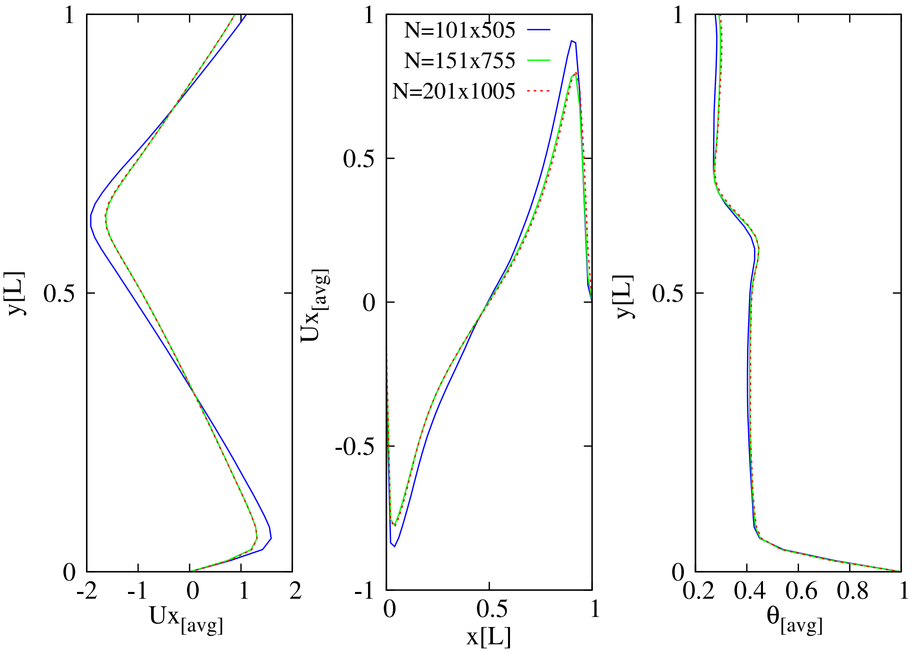


FIGURE 5.2: Grid independence test at Re=1000 and Ri=10 for LBM simulations in the cavity for the dimensionless velocities U_x, U_y and the dimensionless temperature θ.

The results obtained with LBM are compared with those from an independent method, ANSYS-FLUENT (ANS) for validation. Unlike LBM, non-uniform grids can be used here whenever required, to correctly model the boundary layers. For ANS software a grid refinement was applied close to the walls for an appropriate gradient resolution, which serves as a reference for our LBM simulations. Additionally, second order upwind is used for momentum and energy advection, and the second order implicit scheme is chosen for transient formulation. The convergence criterion is based on the absolute convergence of scaled residuals with the thresholds

10^{-8}, 10^{-3} and 10^{-7} for the continuity, momentum and energy equations, respectively.

5.1.2 Results and validation

Results are obtained for the different values of the Reynolds number studied: Re = 50, 100, 200, 400, 600 and 1000. For each of these values, the Richardson numbers Ri = 0.01, 0.1, 1, and 10 are chosen. These results will be analyzed with special attention to the stability of the flow and the structures generated. At sufficiently low Ri, where the external flow dominates, it is expected that the structure of the flow inside the cavity resembles that of the Lid-Driven-Cavity flow problem, with a main vortex in the middle (Chapter 3). First we analyze the heat transfer inseide the cavity. Table 5.1, shows the results of the comparison of the average Nusselt number $\overline{\text{Nu}}$ (computed as in Eq. (4.30) of Sec. 4.2). For all the Reynolds and Richardson numbers, $\overline{\text{Nu}}$ the results from LBM are compared with those obtained with ANSYS-FLUENT (ANS).

The results from both methods show good agreement, with small differences in general which do not exceed 2-4% at low values of the Richardson number. For Re = 600, 1000 and above Ri = 1, the discrepancy raises up to 7% in some cases. At these values of the Richardson number, the recirculating flow in the cavity is being reorganized into two main cells, and a slight drop can be observed in the plot of Nu as a function of Ri, Fig. 5.3, in such a way that the variation is non-monotonous. For Ri $\leqslant$1, the Nusselt number is approximately constant and the flow is steady, but above Ri = 1, it rapidly increases indicating an enhanced heat transport. This occurs specially at the highest values of the Reynolds number studied (Re = 400, 600 and 1000), therefore implying that the mechanism is coupled to the changes in the flow structure that are taking place in the cavity. These structures will be analyzed in the following subsection.

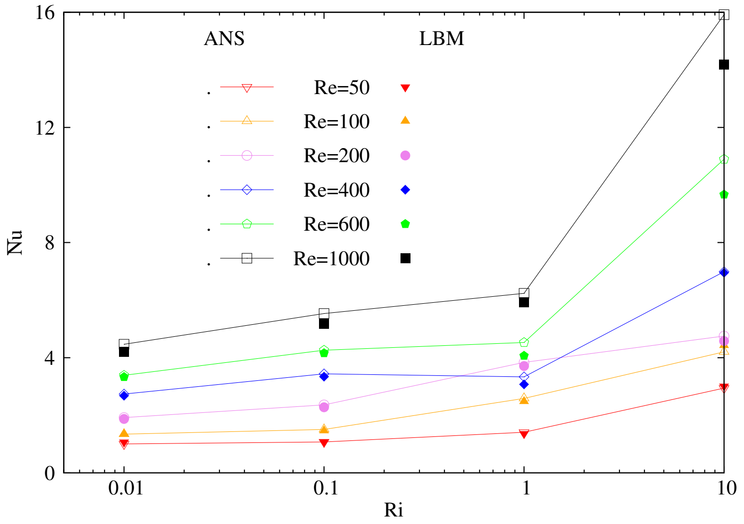


FIGURE 5.3: Average Nusselt number as a function of the Richardson number, for different Reynolds numbers.

5.1.2.1 Effect of the Richardson number

In this section we will be focused on analyzing the effect of the buoyancy force in the cavity and the behavior of the flow structures. The velocity profiles will be analyzed along the mid lines of the cavity, as shown in Fig. 5.4. The velocities have been made dimensionless by referring them to the velocity of the flow at the inlet, u_0. The dimensionless x-component of the velocity, U_x, is analyzed along the vertical line, and the y-component U_y along the horizontal line. In the cases where a steady state is developed, the analysis of the streamlines will be useful. The temperature field is tracked by means of the isotherms inside the cavity and the profiles obtained along the mid lines shown in Fig. 5.4. In the unsteady cases encountered, instantaneous temperature contours will be analyzed. All the results are obtained and compared for both the LBM and ANS simulations.

TABLE 5.1: Summary of the average Nusselt number obtained in ANS and LBM for the set of Richardson and Reynolds numbers. The values of the lattice velocity u_{lbm} and number of lattice nodes N along the cavity height L are specified

		Re = 50		Re = 100	
	Ri	$\overline{\text{Nu}}$		$\overline{\text{Nu}}$	
ANS	0.01	1.004		1.347	
LBM		1.070	u_{lbm}=0.01 N=101	1.344	u_{lbm}=0.02 N=101
ANS	0.1	1.075		1.507	
LBM		1.082	u_{lbm}=0.01 N=101	1.463	u_{lbm}=0.02 N=101
ANS	1	1.410		2.586	
LBM		1.364	u_{lbm}=0.01 N=101	2.480	u_{lbm}=0.02 N=101
ANS	10	2.945		4.202	
LBM		3.013	u_{lbm}=0.01 N=151	4.424	u_{lbm}=0.0135 N=151

		Re = 200		Re = 400	
	Ri	$\overline{\text{Nu}}$		$\overline{\text{Nu}}$	
ANS	0.01	1.919		2.738	
LBM		1.873	u_{lbm}=0.02 N=101	2.680	u_{lbm}=0.01 N=101
ANS	0.1	2.362		3.441	
LBM		2.278	u_{lbm}=0.02 N=101	3.341	u_{lbm}=0.02 N=101
ANS	1	3.837		3.334	
LBM		3.712	u_{lbm}=0.02 N=101	3.074	u_{lbm}=0.02 N=101
ANS	10	4.753		6.988	
LBM		4.580	u_{lbm}=0.01 N=151	6.952	u_{lbm}=0.02 N=151

		Re = 600		Re = 1000	
	Ri	$\overline{\text{Nu}}$		$\overline{\text{Nu}}$	
ANS	0.01	3.390		4.468	
LBM		3.331	u_{lbm}=0.01 N=151	4.201	u_{lbm}=0.025 N=101
ANS	0.1	4.262		5.535	
LBM		4.160	u_{lbm}=0.01 N=151	5.184	u_{lbm}=0.025 N=101
ANS	1	4.529		6.236	
LBM		4.068	u_{lbm}=0.01 N=151	5.921	u_{lbm}=0.025 N=151
ANS	10	10.90		15.92	
LBM		9.670	u_{lbm}=0.01 N=151	14.18	u_{lbm}=0.025 N=151

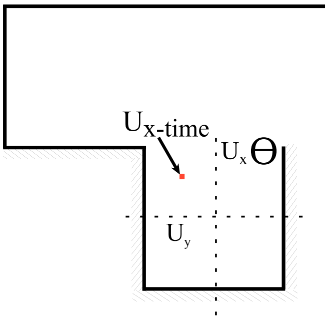


FIGURE 5.4: Scheme of the vertical and horizontal mid lines used to plot the velocity profiles, U_x and U_y as well dimensionless temperature, θ, and the location of the node where the instantaneous velocities have been recorded in the unsteady regime.

Several comparisons were carried out for LBM and ANS. Figure 5.5 displays the dimensionless contour of the temperature of the cavity for the range of 50 < Re < 400 and each one of the Richardson numbers considered: Fig. 5.5a shows the results of the LB simulation, and Fig. 5.5b the results of ANS. For all these cases, a steady state was established, characterized by the encapsulated flow in the cavity, which means that the external flow does not penetrate inside. For the lower Ri, the temperature contours are parallel to the hot wall, indicating that forced convection is dominant; when Ri increases, the temperature grows on the left wall of the cavity, indicative that the natural convection is dominant. In Fig. 5.6 streamlines are generated in order to observe the velocity field and the encapsulation of the structures in the cavity. Figure 5.6a displays the streamlines for the case Re = 50 and Ri = 10, where it can be observed that the external flow encapsulates the flow inside the enclosure, and a single main structure is generated. As both the Reynolds and the Richardson numbers increase, Fig. 5.6b, the main recirculation cell is pushed upwards, whereas a lower vortex develops from the bottom right corner of the cavity. The motion of the fluid on the bottom is reversed, and the right wall becomes warmer than the left wall. This has been observed in the range of Ri investigated in our simulations above Re = 200, where the second vortex is formed between Ri = 3.9 and Ri = 4: the higher the Reynolds number, the earlier the bifurcation is observed, in terms of the value of Ri.

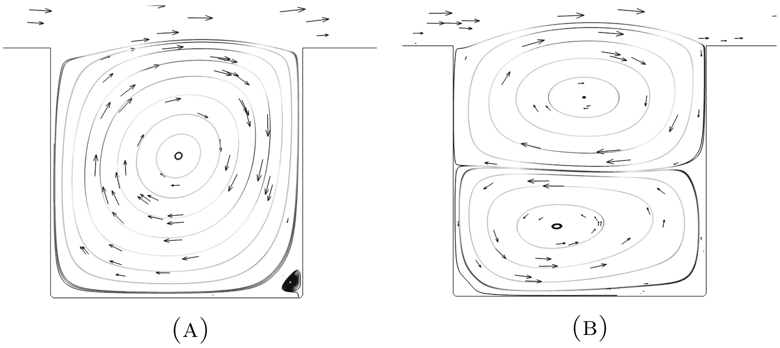


FIGURE 5.6: Streamlines and recirculation region in the cavity, LBM simulation. A) Re = 50, Ri = 10, B) Re = 200, Ri = 10.

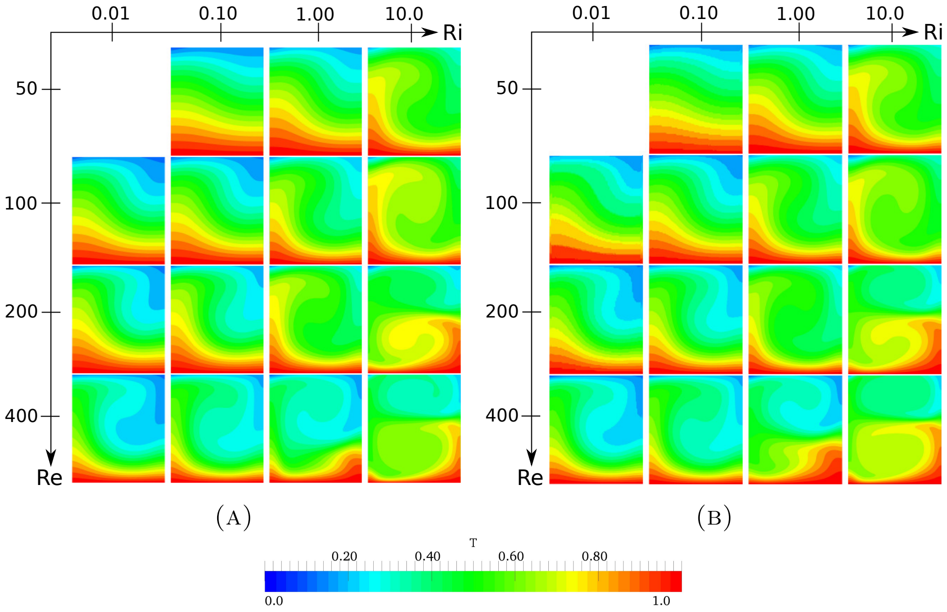


FIGURE 5.5: Contours of the dimensionless temperature inside the cavity, for A) LBM and B)ANSYS simulations.

The growth of the second recirculating cell is associated to the increase of the Richardson number. Let us momentarily consider the case where thermal effects are negligible, and the increase of the Reynolds number originates the development of secondary vortices at the cavity corners. At

Re = 50 they are barely visible, but as Re increases, these grow in size progressively, such that at Re = 1000 they involve a small but non-negligible portion of the flow in the cavity, attached to the bottom corners. Among the two secondary vortices, the downstream vortex is bigger. When the effects of buoyancy increase as Ri increases, the fluid at the bottom becomes lighter, and a pressure gradient is established in the vertical direction. At the same time, more fluid is incorporated into the lower vortex circulation. As a consequence, the growth of the lower vortex and the stratification of the flow is then observed: the upper vortex fills the region of higher pressure, whereas the lower region is occupied by the fluid circulating at a lower pressure. This is well illustrated by the contours of pressure and the streamlines of the case at Re = 600 which we will analyze in more detail in the next section, when the unsteady flow is discussed.

The profiles of the dimensionless temperature θ along the vertical mid line of the cavity, for Re = 50, 100, 200 and 400 can be observed in Figs. 5.7 for a quantitative comparison between LBM and ANS results. These profiles reflect the changes induced in the flow when varying the Reynolds and the Richardson numbers, and particularly the development of the second main recirculating cell. The agreement is very good in general, with small deviations. At higher Reynolds numbers, and as the flow is being reorganized into two main recirculating cells inside the cavity when Ri increases, the temperature is larger in the middle in ANS. That implies a slightly larger bottom vortex, where the fluid is warmer, than in our LBM simulations.

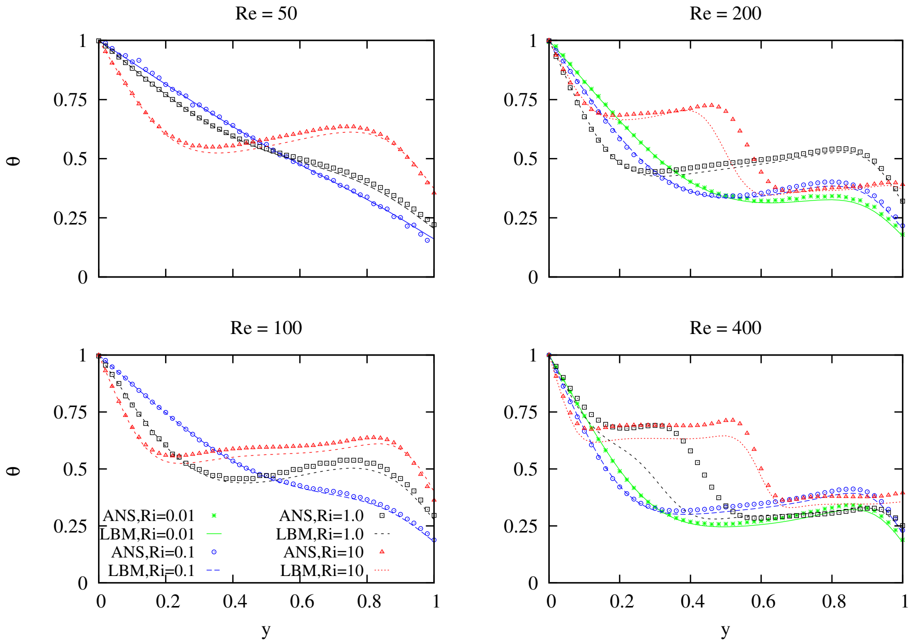
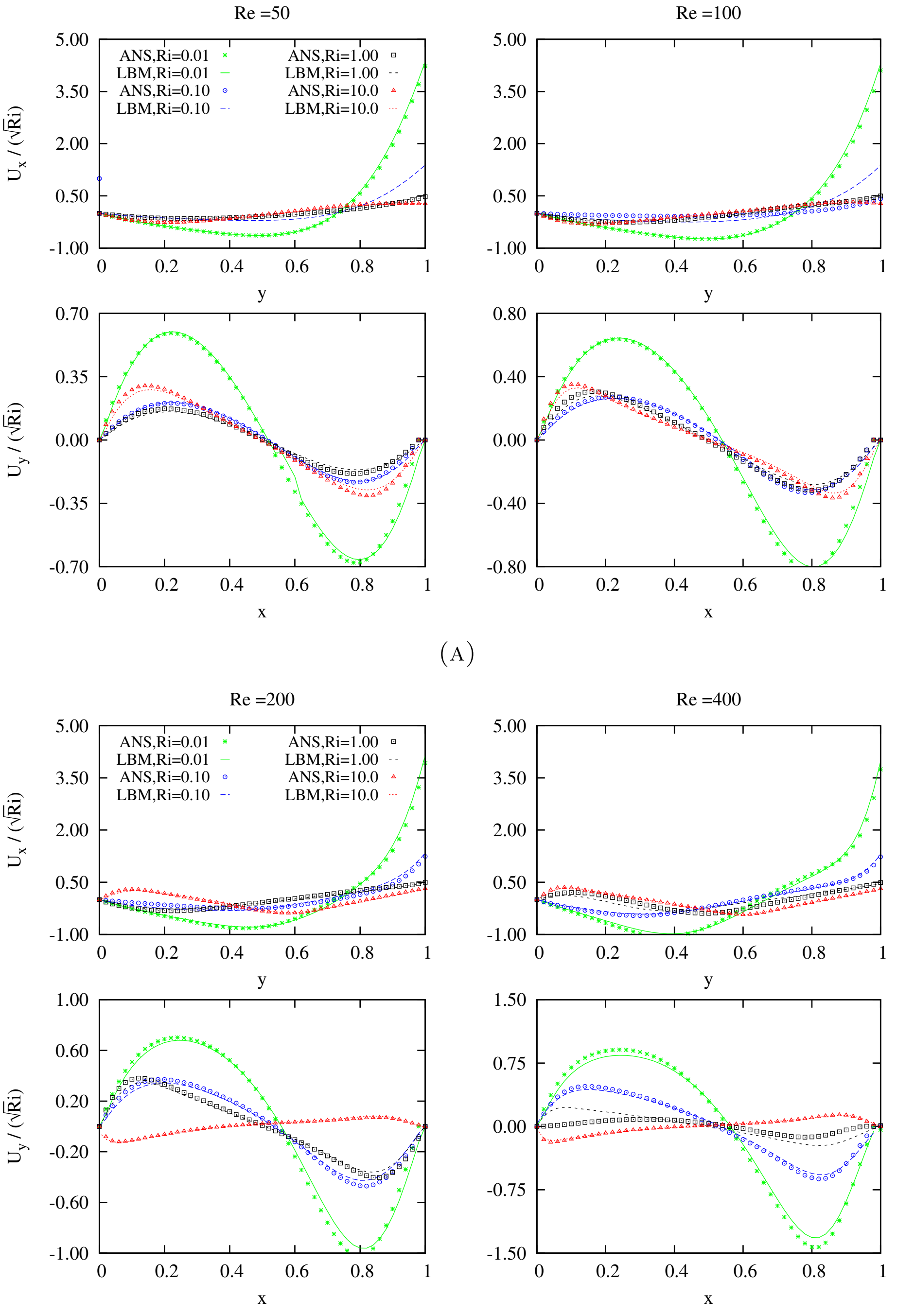


FIGURE 5.7: Dimensionless temperature θ in the vertical mid plane of the cavity, for Re = 50, 100, 200 and 400 and Ri = 0.01, 0.1, 1, 10. Ri = 0.01 has been omitted for Re = 50 and 100, because the temperature differences are too small and the changes are minimal.

The profiles of the velocity components U_x, U_y are shown in Figs. (5.8a, 5.8b). The x-component is plotted along the vertical mid line of the cavity, and the y-component along the horizontal mid line. The agreement between LBM and ANS is very good, and the differences do not exceed 8% in any case. The appearance of the second vortex and the changes in the circulation of the flow inside the cavity are manifested in the changes of sign of U_y, U_x, respectively at Re = 200 and 400, when Ri increases.



(A)

(B)

FIGURE 5.8: Dimensionless velocity profiles U_x and U_y along the vertical and horizontal mid lines of the open cavity, respectively, as Ri is varied from 0.01 to 10. A) Re = 50, 100, B) Re= 200, 400.

5.1.2.2 Unsteady flow

For Reynolds numbers above 400, in the range of Richardson numbers studied, one can observe the development of unsteady flow. At Re = 600, below a critical value of Ri, a steady flow regime is found up to Ri∼1, with a well developed double vortex at Ri = 1 and the typical temperature contours seen already at lower Reynolds numbers (Fig. 5.5). The qualitative and quantitative good agreement between both methods is apparent by comparing the velocity and the temperature profiles along the centerlines of the cavity. In Fig. 5.9 we plot the dimensionless velocity and temperature profiles, and in Fig. 5.10 the contours of temperature are shown.

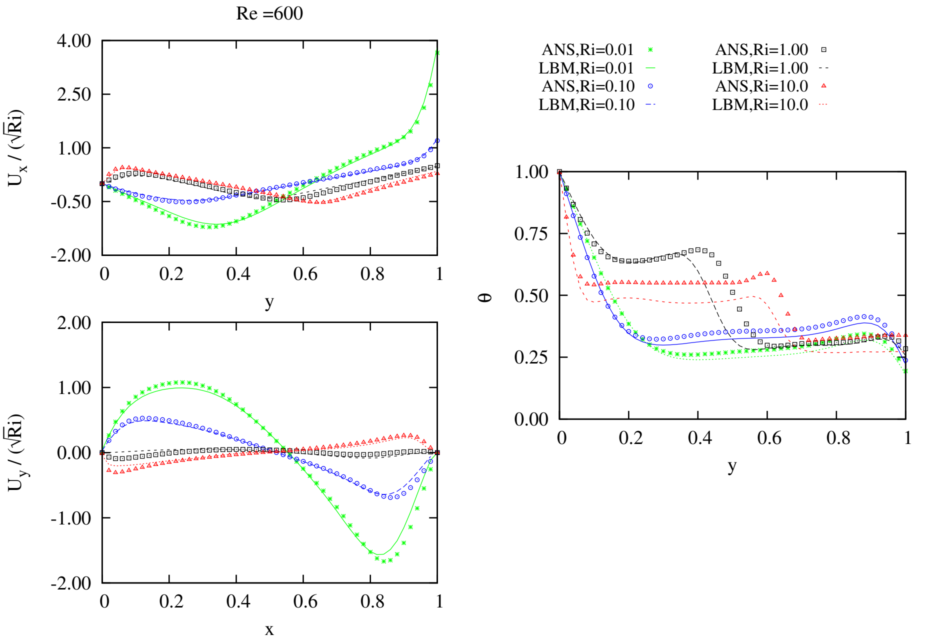


FIGURE 5.9: Dimensionless velocity profiles along the vertical and horizontal mid line of the cavity (left) and temperature (right), for Re = 600 and Ri = 0.01, 0.1, 1, 10. In the case of Ri = 10, the temperature and the velocity have been averaged over time once the unsteady regime is established. Observe the decrease of the temperature inside the cavity which accompanies the development of the unsteady flow.

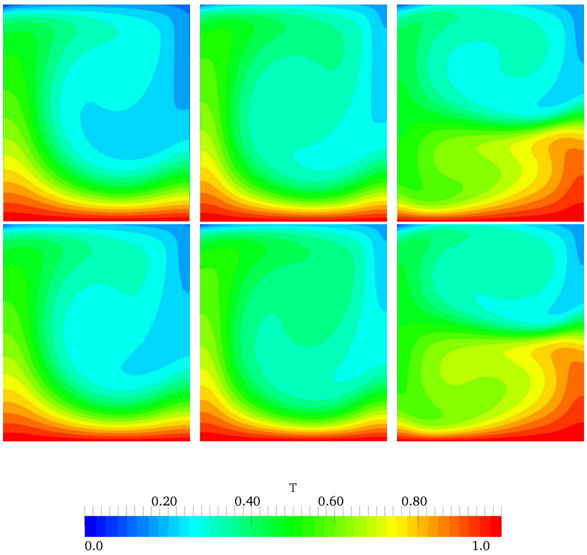


FIGURE 5.10: Comparison of the temperature contour plots for Re = 600 and Ri = 0.01 (left), Ri = 0.1 (center) and Ri = 1 (right): LBM (top) and ANS (bottom) simulations.

At Re = 600, Ri = 10 the unsteady flow has already reorganized into two main recirculating cells in the cavity (Fig. 5.11) visible not only in the averaged but also in the instantaneous contour plots, (Figs. 5.12, 5.13 respectively).

Beyond what the averaged results show, persistent oscillations of the flow variables have been recorded along time, manifested for example in the Nusselt number measured along the heated wall of the cavity. After a short transient, a pseudo-periodic regime sets up characterized by the intermittent emission of hot structures from the cavity (Fig. 5.13), which move downstream towards the outlet. The vortices are emitted periodically, in a pattern reminiscent of a Von Karman vortex street without alternation, although the primary nature of the instability is of thermal type. The instantaneous temperature contours show a complete sequence of the periodic emission of vertical structures from the downstream upper vertex of the cavity. The observation of the unsteady temperature fields of Fig. 5.13 reveals interesting information about the nature of the ejection mechanism. In the unsteady regime, for a sufficiently high value of the Richardson number, the mixing layer between the two vortices as well as the virtual interface separating the cold fluid in the channel from the warm

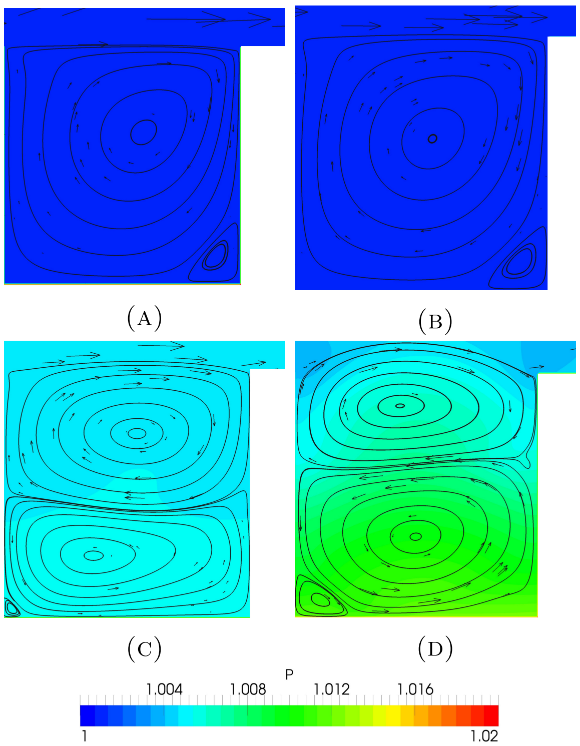
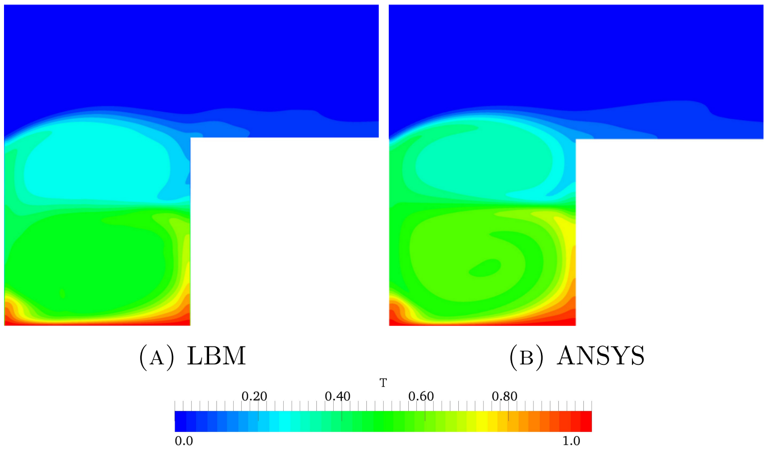
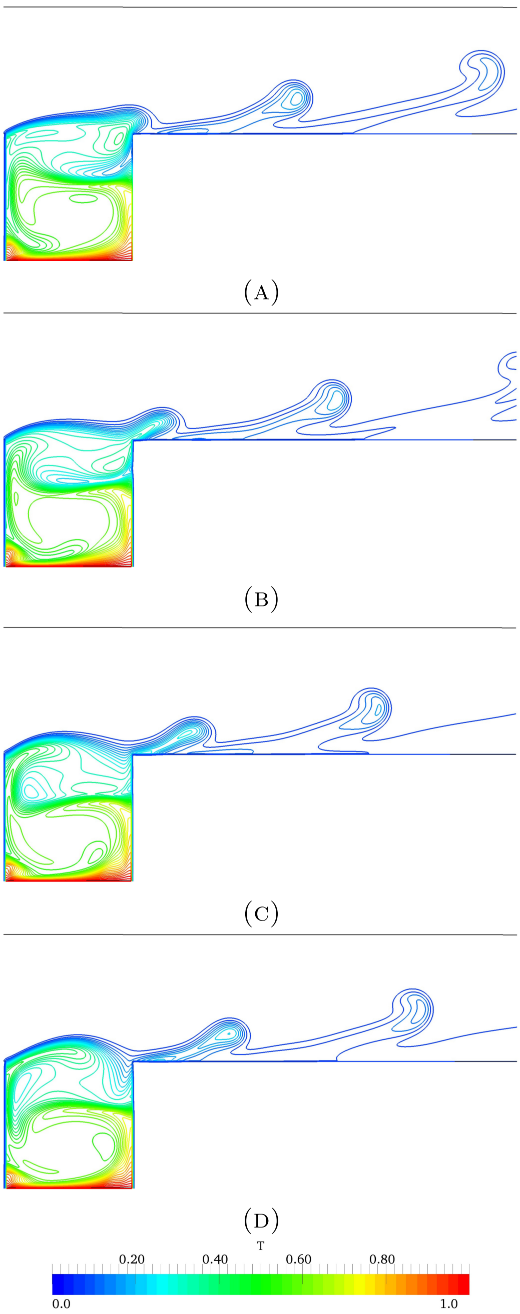


FIGURE 5.11: Pressure contours and streamlines (LBM) for Re = 600 and A) Ri = 0.01 B) Ri = 0.1 C) Ri = 1 D) Ri = 10. In (D) the flow is unsteady, and time-averaged results are shown. Observe the development of the secondary vortices at the bottom corners as Ri increases.


(A) LBM

(B) ANSYS

FIGURE 5.12: LBM and ANS averaged temperature contours in the unsteady regime, for Re = 600 and Ri = 10.


(A)

(B)

(C)

(D)

FIGURE 5.13: (A - D) Sequence of instantaneous temperature contours in the unsteady flow regime (LBM) showing the vortex production at the upstream corner of the enclosure and the changes in the mixing layer between the two vortices, for Re = 600 and Ri = 10.

fluid inside the cavity is unstable. As the fluid circulating in the lower vortex warms up and becomes more buoyant (A), destabilizes the mixing layer by climbing up the left vertical wall (B, C). The build-up is complete in (D) where the bulging shape of the upper vortex indicates that soon (A) the next puff of warm fluid is going to be ejected, while cold fluid from the channel is being incorporated into the enclosure at the downstream vertical wall. The entrance of cold fluid into the enclosure has the effect of lowering the temperature (compare the temperature profiles in Fig. 5.9), favoring mixing and heat transfer, and thus enhancing the rise of the Nusselt number (Fig. 5.3). The vorticity plot in Fig. 5.14, reveals the structure of the dual vortex emission, whereas the instantaneous streamlines show little disturbance in those regions, due to the more dominant contribution of the main flow. Fig. 5.15 shows the oscillations of the Nusselt number.

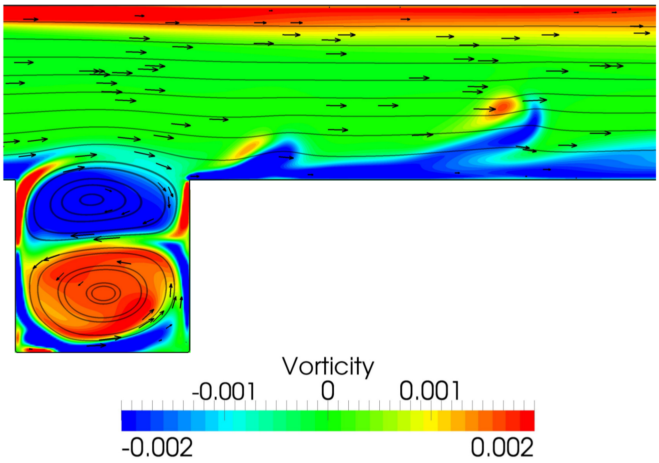


FIGURE 5.14: Vorticity field in the unsteady regime for Re = 600 and Ri = 10 (LBM), showing the dual character of the vortex production: regions with negative vorticity are shaded in blue, positive in red.

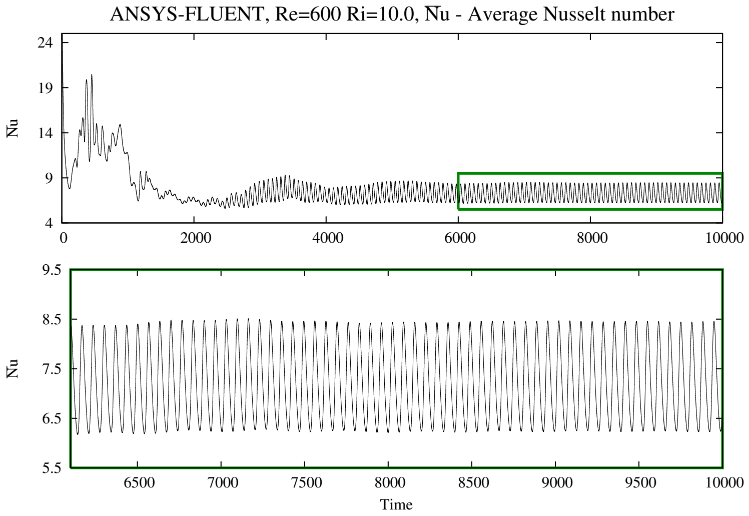
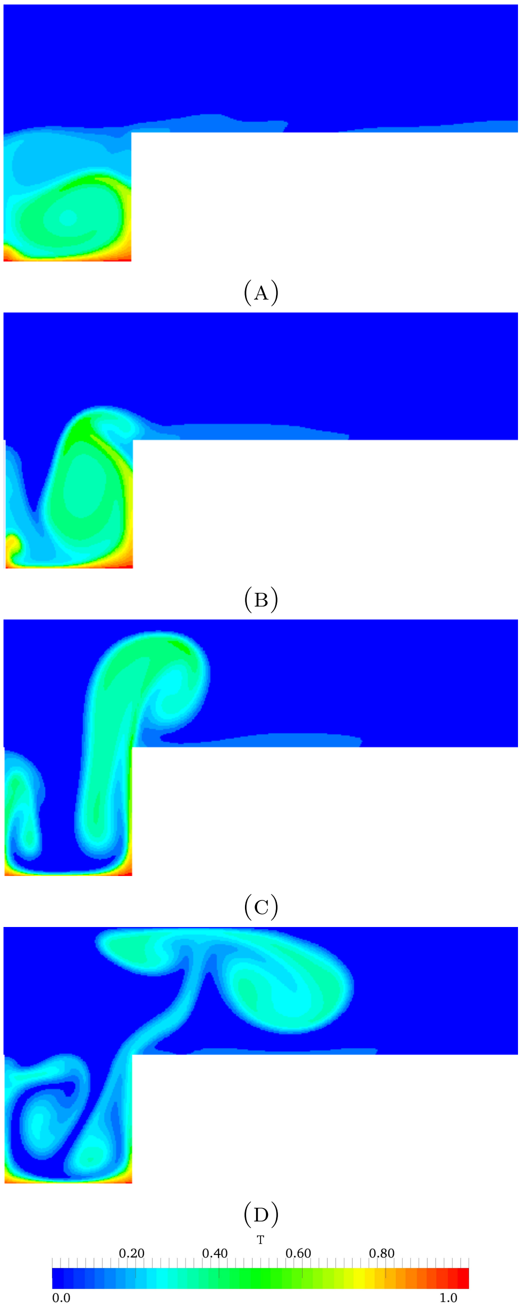


FIGURE 5.15: Nusselt number measured on the heated wall along time in the unsteady regime for Re = 600 and Ri = 10 (ANS).

The recirculation remains inside the cavity for all these cases, but at Ri = 10, the flow is no longer encapsulated. At Ri = 10, the buoyancy force is stronger than the external flow, and pushes the flow upstream, outside the cavity. At Ri = 10 and approaching Re = 500, the flow becomes unsteady as the double vortex becomes unstable, and a transient regime sets in where the cavity intermittently emits large plumes of warm fluid which slide over the downstream vertical wall of the cavity, while a current of cold fluid enters the enclosure from the upstream region (see Fig. 5.16. This pattern does not damp with time. The spatially averaged Nusselt number computed on the heated wall shows also very strong fluctuations and unpredictable temporal behavior, as in Fig. 5.17.



(A)

(B)

(C)

(D)

FIGURE 5.16: (A - D) Sequence of instantaneous temperature contours in the unsteady regime (LBM) showing the production of hot plumes at the upstream side of the enclosure, for Re = 500 and Ri = 10. The warm fluid on the bottom cell is periodically evacuated from the enclosure, sliding over the downstream vertical wall.

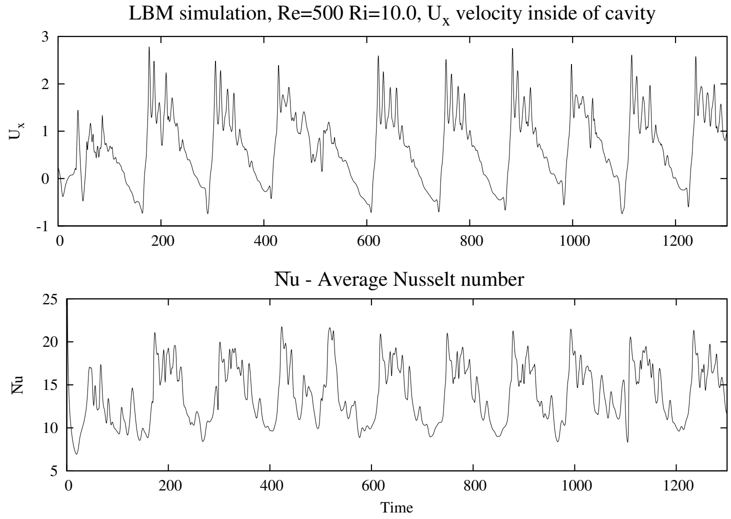


FIGURE 5.17: Velocity component U_x and spatially averaged Nusselt number recorded at a single lattice node (LBM) in the unsteady regime as a function of time, Re = 500, Ri = 10. The node is located in the middle of the upper-left quadrant of the square cavity, as shown in Fig. 5.4.

A further increase of the Reynolds number does not change the previously described characteristics of the flow. At Re = 1000, we find a similar behavior as for Re = 600, meaning that for Richardson numbers 0.01, 0.1, 1, the flow is steady amd the external flow encapsulates the flow inside the cavity. At Ri = 10, the flow is unsteady, as for Re = 600. Figure 5.18 shows again a good agreement between the steady temperature contours obtained by means of LBM and ANS, for Re = 1000.

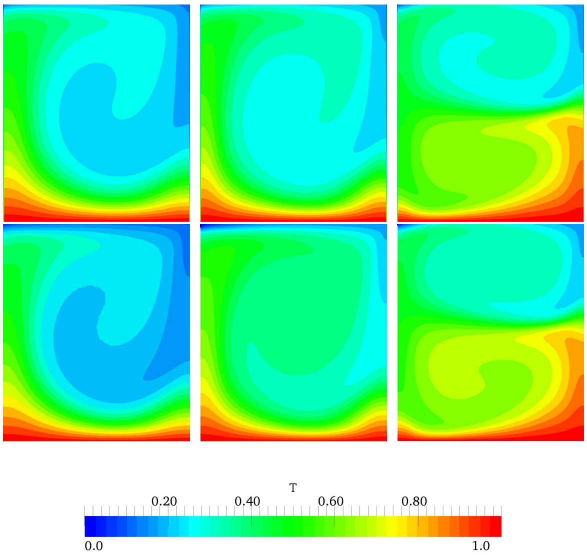
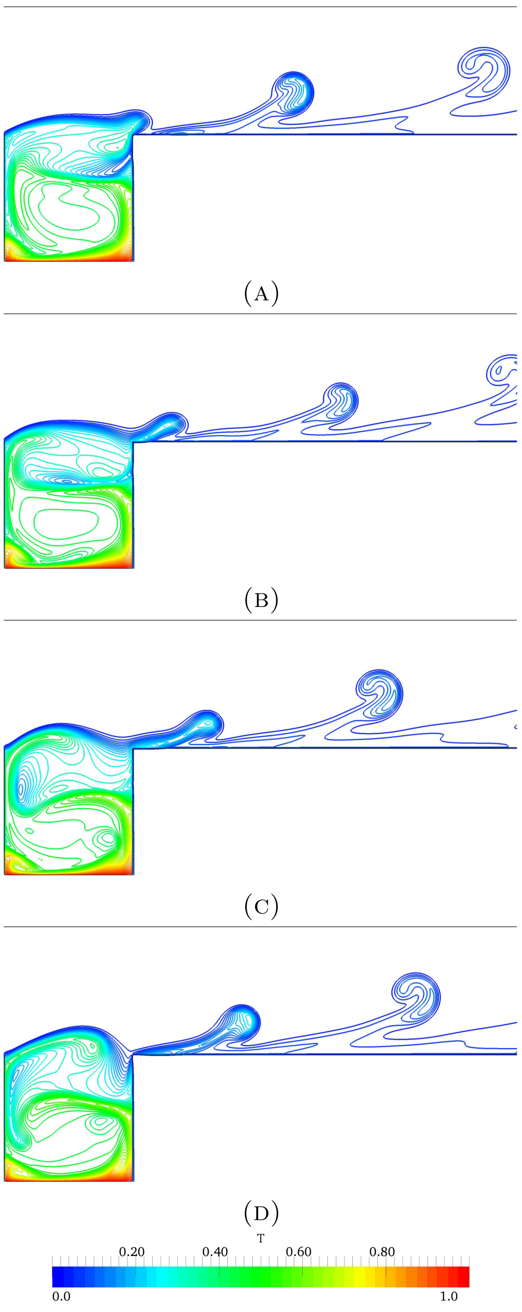


FIGURE 5.18: Comparison of the temperature contour plots for Re = 1000 and Ri = 0.01 (left), Ri = 0.1 (center) and Ri = 1 (right):LBM (top) and ANS (bottom) simulations.

At Re = 1000 and Ri = 10, the external flow cannot encapsulate the flow permanently and the buoyancy force pushes the flow outside the cavity. Figure 5.19 displays a sequence of instantaneous flow structures obtained from the LBM simulation. The comparison between the LBM and ANS results is shown in the average temperature contours of Fig. 5.20. Similarly to the case Re = 600, the buoyancy periodically generates structures that expand and propagate towards the channel output. The unsteady regime is accompanied by oscillations in the velocities and the temperatures inside the cavity. For example, observe the continued oscillations of the x-velocity component U_x along the vertical mid line of the cavity. The Nusselt number measured along the heated wall shows the same type of oscillations, as shown in Fig. 5.21. The comparison of the velocity and the temperature profiles along the centerlines of cavity can be observed in Fig. 5.22.

Papanicolaou and Jaluria [51] found similar, although more regular oscillations studying the mixed convection in a two dimensional cavity, but also found that after increasing the Reynolds number above 200, oscillations disappeared and a steady solution was observed. We do not observe such stabilizing effect in our simulations: first, increasing the Reynolds number

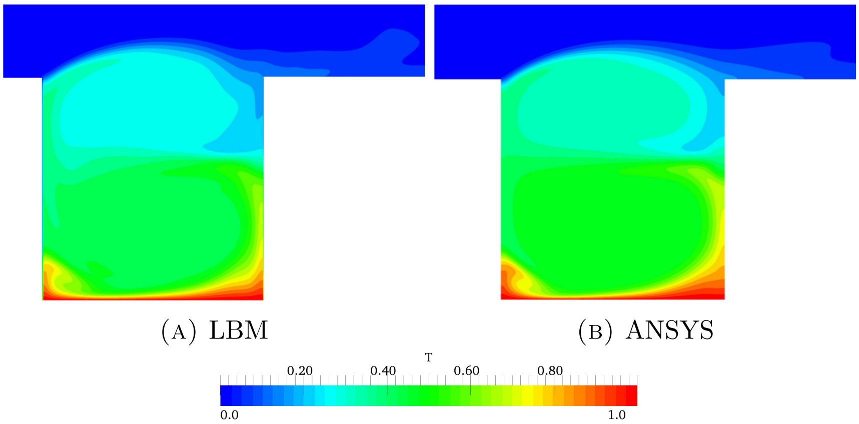

(A)

(B)

(C)

(D)

FIGURE 5.19: (A - D) Sequence of instantaneous temperature contours in the unsteady regime (LBM) showing the vortex production at the upstream corner of the enclosure, for Re = 1000 and Ri = 10.

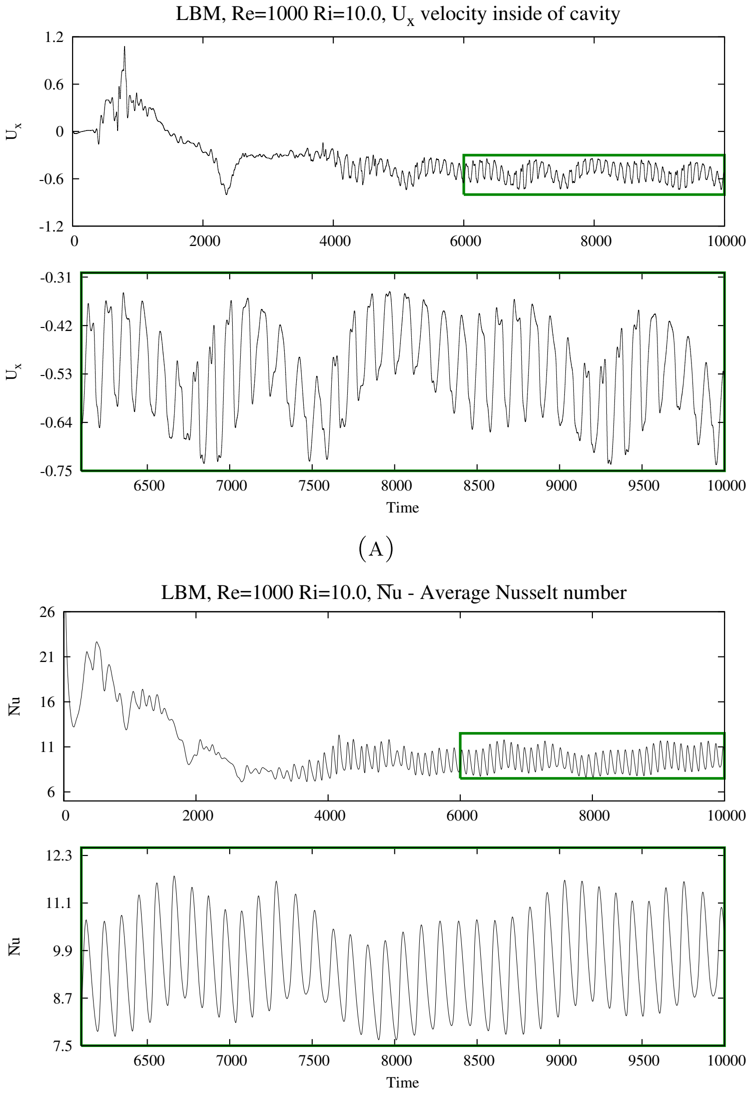


(A) LBM

(B) ANSYS

FIGURE 5.20: Average temperature contours for Re = 1000, Ri = 10.

has the effect of decreasing the critical Richardson number above which the main vortex is replaced by two steady main vortices (see Fig. 5.5). This pair of vortices inside the cavity was not observed in Ref. [51], but the geometry of the cavity, square with two opposite vertical openings, was different from the one used here, a true channel embedding a square cavity as an inclusion. Likewise, we do not observe oscillations at Re = 100 for the values of the Ri studied ($Ri \lesssim 10$), because a higher value of Ri should in principle be required (according to Ref. [51], Ri = 32 is the required value for oscillations at Re = 100).



(A)

(B)

FIGURE 5.21: A) Velocity component U_x recorded at a single lattice node (LBM) in the unsteady regime as a function of time, Re = 1000, Ri = 10. The node is located in the middle of the upper-left quadrant of the square cavity, as shown in Fig. 5.4. The bottom plot is the zoom of the rectangular region of the upper plot. B) The Nusselt number measured on the heated wall along time in the unsteady regime

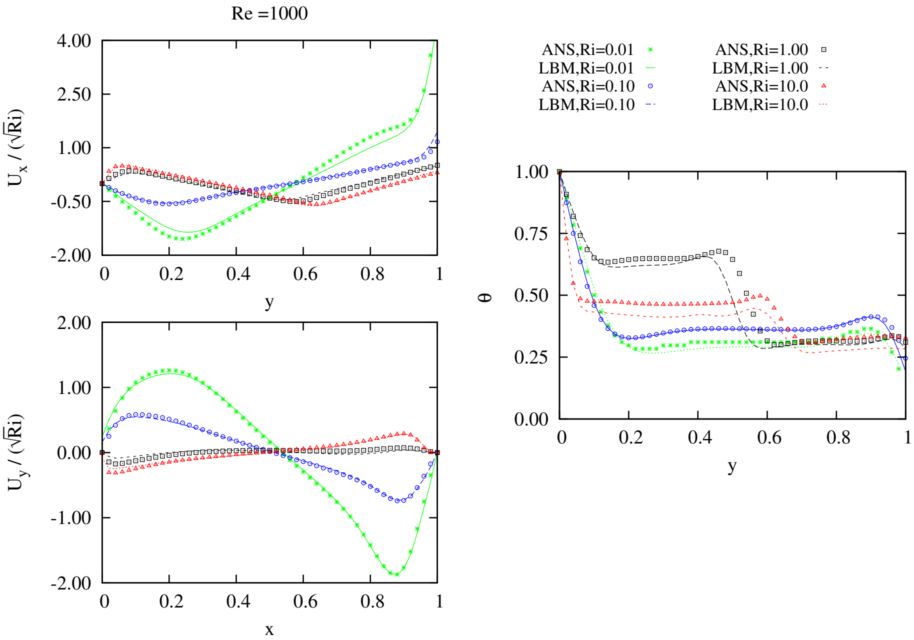
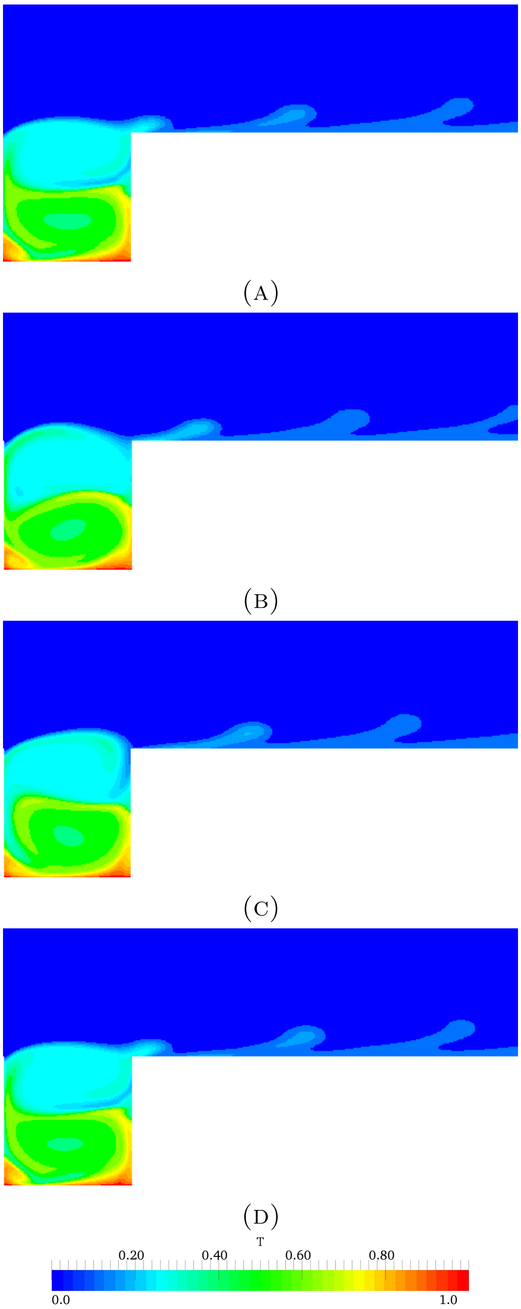


FIGURE 5.22: Dimensionless velocity profiles along the vertical and horizontal mid-lines of the cavity (left) and temperature (right), for Re = 1000 and Ri = 0.01, 0.1, 1, 10. In the case of Ri = 10, the temperature and the velocity have been averaged over time once the unsteady regime is established.

5.1.2.3 The oscillatory regime

The oscillatory regime is observed in a wide range of Reynolds and Richardson numbers, however the complexity of the oscillations increases as Re increases approaching the transitional regime in the channel, and also as Ri increases. Considering a fixed value of Ri = 10, oscillating variables like the spatially averaged Nu and the instantaneous velocity inside the cavity have been observed above Re $\simeq$ 500, and are still well visible at Re = 1500 (see Fig. 5.23). Oscillations start at Re = 500 with large amplitudes at a dominant low frequency mode (Fig. 5.17). These large variations of Nu imply also relatively large deviations when comparing the average values (Table 5.1) obtained by LBM and ANS, because the time series need to be very long. After the inception of the oscillatory regime, at Re = 600 oscillations have become regular (Fig. 5.15). As Re increases, the frequency of vortex emission increases and the Nusselt time series show a progressively larger spectral content (Figs. 5.21, 5.24).


(A)

(B)

(C)

(D)

FIGURE 5.23: Sequence of instantaneous temperature contours in the unsteady regime (ANS) for Re = 1500, Ri = 10.

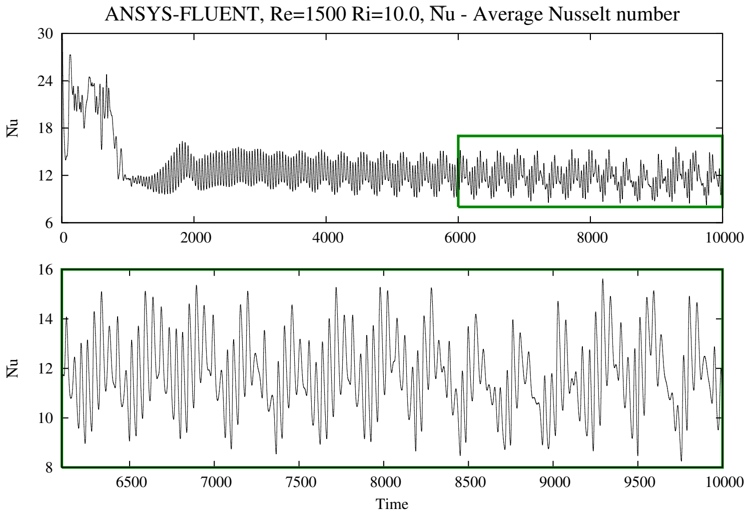


FIGURE 5.24: Spatially averaged Nusselt number as a function of the dimensionless time, measured on the heated wall in the unsteady regime for Re = 1500 and Ri = 10.

This is brought out by the FFT of the signal shown in Fig. 5.25, displaying a comparative panel of the power-spectrum of the velocity U_x measured in the middle of the upper-left quadrant of the square cavity, and the spatially averaged Nu for Re = 1000 and 1500, Ri = 10. The increase of the Reynolds number causes the main components of the power-spectrum to shift towards higher frequencies.

In order to investigate the effect of the Richardson number on the oscillations, a simulation at Re = 1000, Ri = 100 was conducted. As seen in Fig. 5.26, the vortices are emitted at a higher frequency and early detach from the channel wall in their way to the outlet as a consequence of the stronger buoyancy. The buoyancy force pushes the flow outside the cavity, generating pairs of vortices moving along and across the channel. This results in a more complex time series of the spatially averaged Nu and a richer power-spectrum in the frequency space, Fig. 5.27.

Two more numerical simulations were carried out in order to observe the effect of the position of the heated wall and the generation of flow structures in the cavity. The simulations reveal that the position of the heated wall is also crucial for the development of the corner vortices: if the square cavity is heated from a vertical wall the second vortex never develops. This

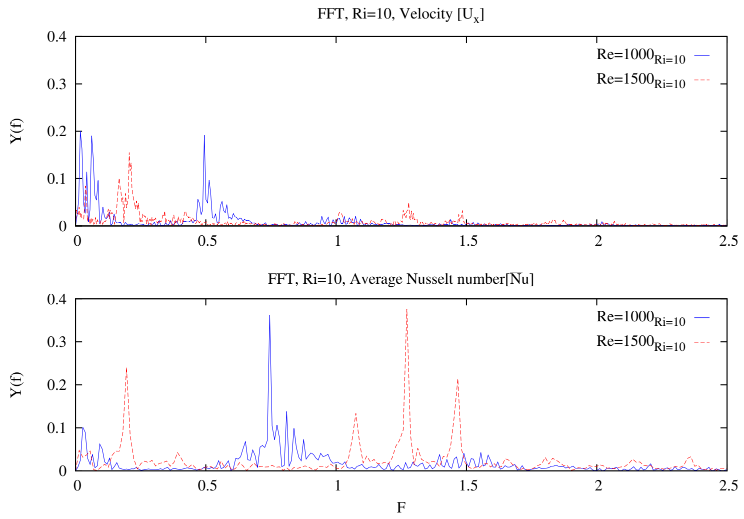
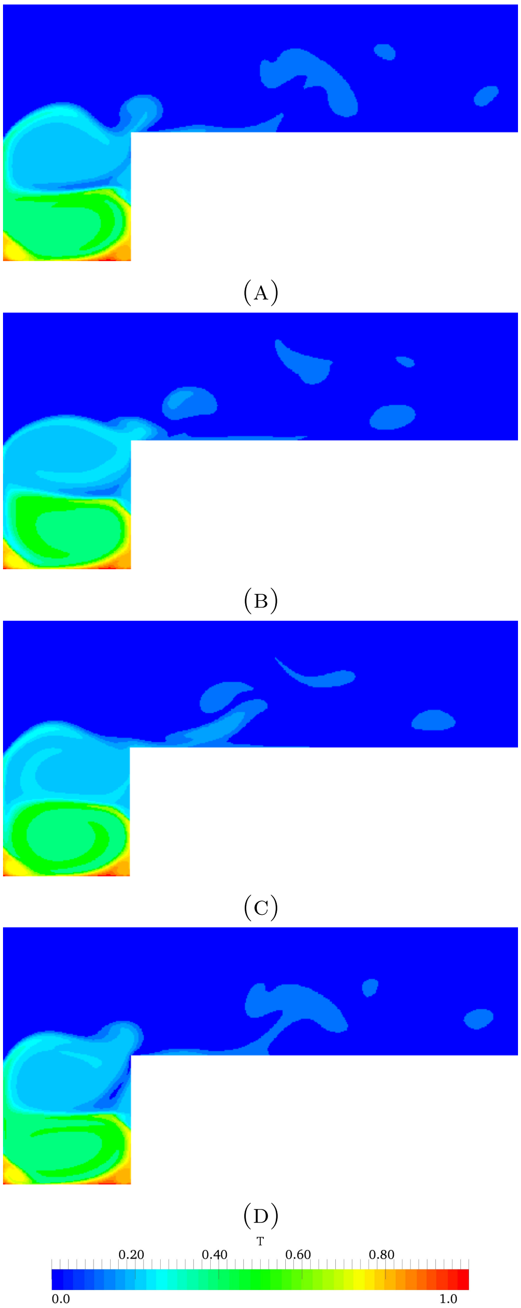


FIGURE 5.25: Power spectrum of the instantaneous velocity U_x and spatially averaged Nusselt number Re = 1000, Re = 1500 (ANS), as a function of the dimensionless frequency F.

and other results are included in Appendix A. On the other hand, the domain used for this mixed convection problem may serve to study the mass transfer processes in the open cavity configuration, where an external flow is imposed in the channel and a simple surface reaction is considered. The bottom wall of the cavity has been chosen as the catalytic surface, where a first order reaction is implemented in LBM without temperature dependence. It is assumed that the reaction does not affect the flow field. The surface reaction is regarded as irreversible and the mass transfer of reactant A and product B is simplified, where the species A entering the cavity from the inlet encounters a reactant at the bottom wall (e.g., catalytic flat plate) generating a substance B as the resulting product. These results, which will constitute the material of another publication, are briefly presented in Appendix B).



(A)

(B)

(C)

(D)

Figure 5.26: (A - D) Sequence of instantaneous temperature contours in the unsteady regime for Re = 1000 and Ri = 100 (ANS).

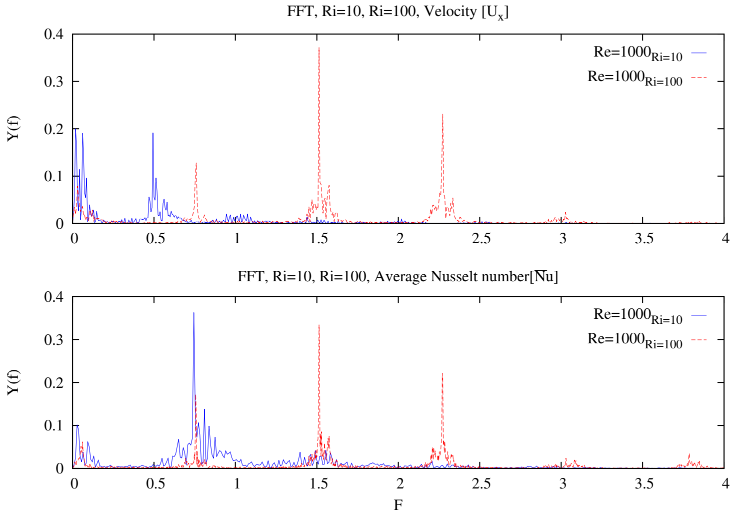


FIGURE 5.27: Power spectrum of the instantaneous velocity U_x and spatially averaged Nusselt number for Ri = 10 and 100, Re = 1000, as a function of the dimensionless frequency F.

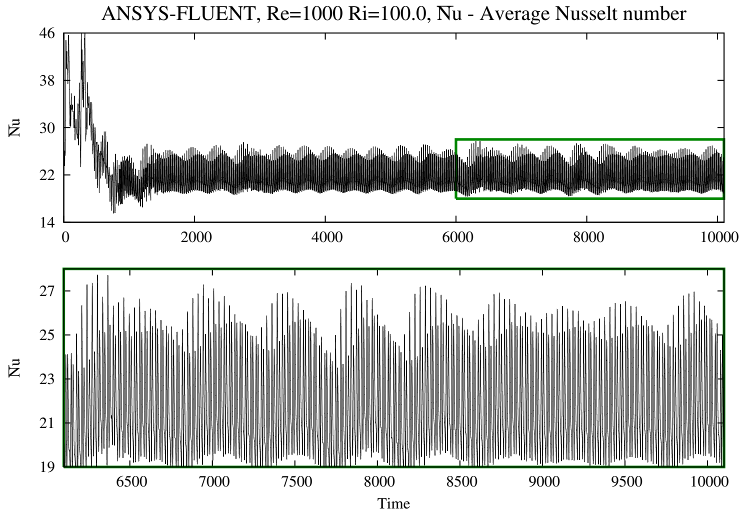


FIGURE 5.28: Spatially averaged Nusselt number as a function of the dimensionless time, measured on the heated wall in the unsteady regime for Re = 1000 and Ri = 100.

Chapter 6

Conclusions

Several numerical simulations were carried out using the LBM, both STR- and MRT-models (D2Q9) for incompressible fluid flow, to study and analyze momentum, heat and mass transfer phenomena. The implementation of the LBM implied the development of a code using Fortran90 programming language, modified from a basic existing code from Palabos (http://www.palabos.org/) for isothermal flows, and supplied with numerous novel subroutines in order to implement the required schemes: buoyancy force for natural and mixed convection, lattice meshes for mass and heat transfer, different boundary conditions, variety of geometries.

Chapter 3 analyzed different problems which were compared to exact solutions and/or to results obtained in previous research using alternative methods. In this study, different grid sizes were tried for the SRT-model, which in some cases, rendered the method unstable. Regarding the MRT-model, the D2Q9 model was appropriately implemented to maintain the stability and accuracy of the method. Boundary conditions and bounce-back schemes used by Zou and He, on one hand, and Dirichlet boundary conditions on the other hand, were used observing an excellent agreement for each one of the studied cases. The results obtained for the flow, temperature and concentration fields matched very closely bibliographic data, as expected.

Chapter 4 was devoted to natural convection in a closed cavity, to validate the Thermal-LBM solver (TLBM). In this case a better grid refinement (number of lattice nodes) would be necessary to obtain good results. This is due to some flow structures growing close to the walls, and the fact that the uniform grid of the LBM approach is not sufficient to properly capture the information contained in these small structures in the corners of the cavity. To this end, a local refinement may be the most appropriate solution to consider.

In Chapter 5, a study on mixed convection in an open square cavity heated from below was carried out. Steady and unsteady flows were analyzed in the incompressible limit as a function of the Reynolds and Richardson numbers. Temperature, velocity and Nusselt number were compared with ANSYS-FLUENT, obtaining very good agreement in general, which makes TLBM an accurate tool to study mixed convection in steady and unsteady regimes. The main differences between the solutions provided by both solvers are not found close to the walls, where the boundary layers are properly resolved, but in the temperature field around the mixing layer, in the middle of the cavity, for the velocity field the discrepancy is always small. For low Re =50, 100, the agreement was very good, at higher Re the development of the bottom vortex as Ri increases, is accompanied by the highest differences in the temperature field. The second vortex develops at Ri $<$ 4.0 starting from Re $=$ 200, and the greatest differences are found in the temperature field around the values of Ri where this happens. These differences are evident at Re $=$ 200, Ri $=$ 10, and even larger at Re $=$ 400, Ri $=$ 1. At higher Re, the second vortex appears at Ri $<$ 1. Accordingly, the greatest discrepancy of the temperatures in the middle of the cavity is observed for Re=600 and Re=1000 at Ri $=$ 1. In the unsteady regime, the LBM simulations give for Ri $=$ 10 consistently smaller values of the temperature inside the cavity which translate into slightly smaller Nusselt numbers. The behaviour of the spatially averaged Nu as a function of Ri with varying Re is similar as in the three dimensional case, revealing an enhanced heat transfer when Ri>1 in the range of Reynolds numbers studied 50$\leqslant$Re$\leqslant$1500. When Ri $\lesssim$ 1, the convective buoyancy force is negligible with respect to the external flow, and the flow remains steady and encapsulated. One and two recirculating cells are observed in the cavity: above a critical value of the Richardson number, lower as the Reynolds number increases, a double vortex develops inside the cavity as a steady solution, instead of the generally observed single vortex. At Ri $=$ 10, when Re$\simeq$500, the effect of the buoyancy force is important and the flow is no longer encapsulated. That initiates a fully unsteady regime, characterized by cyclic dual counter rotating vortex emission at a characteristic main frequency from the upper right corner of the enclosure, of growing complexity and spectral components as the Reynolds number increases. Oscillations of the flow variables and in particular of the spatially averaged Nu are observed and studied. As found Papanicolaou et al. [52], these oscillations are permanent and do not damp with time, however in our problem do not vanish as Re increases. As the buoyancy force is further increased (Ri = 100) the vortices emitted from the cavity detach from the channel wall and quickly mix with the main channel flow.

Appendix A

Effect of the heated wall position in the open cavity

The vertical heated wall on the inflow side of the cavity was set in order to observe the structure and the characteristics of the flow. Figure B.1 displays the computational domain and the boundary conditions for the LBM simulations. These conditions are transposed with those of the case where the heat comes from the bottom, since in this case, the upstream vertical wall is heated at constant temperature. As for the rest of the set-up, the flow conditions are identical. Three different Richardson numbers were considered for each Reynolds number in the range 50<Re<1000: Ri = 0.01, 0.1 and 1.0. However for Ri = 0.01, the effect of the buoyancy force was always negligible.

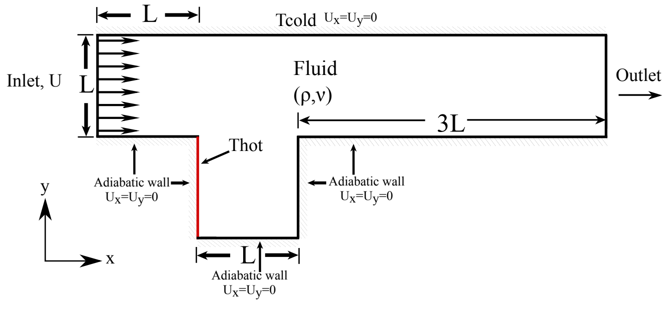


FIGURE A.1: Scheme of the open cavity with a vertical heated wall.

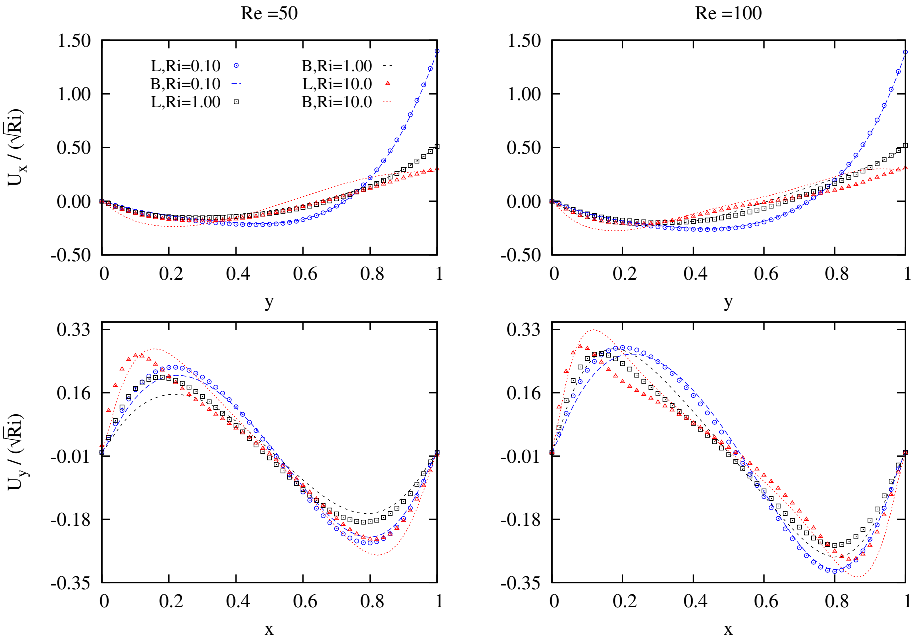
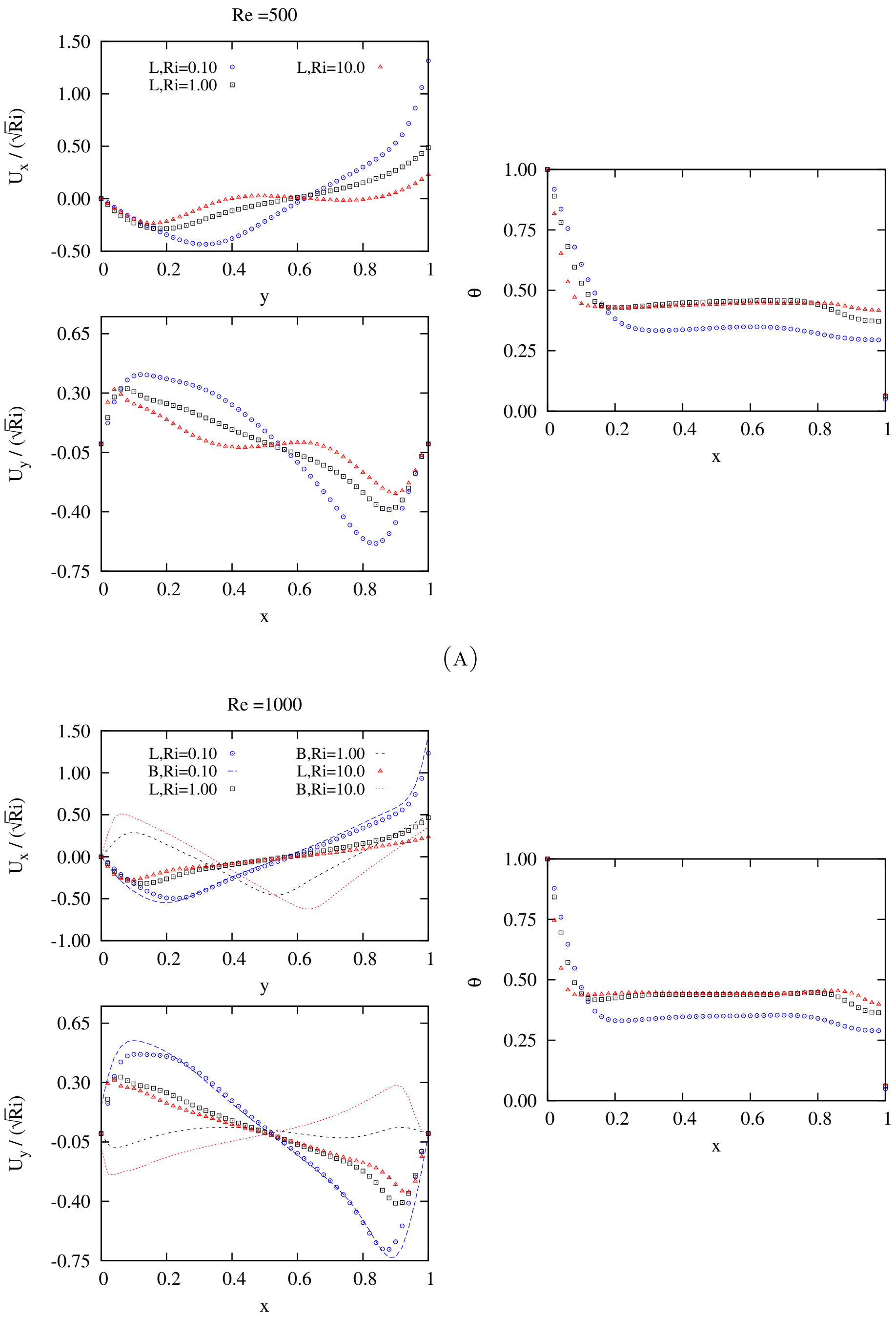


FIGURE A.2: Dimensionless velocity profile, vertical and horizontal mid line of the cavity, Re=50, Re=100. (L,Ri) represent the set of Richardson number for the position of the heated wall, left vertical heated wall and (B,Ri) bottom heat wall.

Figure A.2 shows the dimensionless velocity profile U_x,U_y for Re=50 and Re=100 on the left vertical heated wall case comparison with the velocities of the bottom heated wall case. Similarly to the previous case, the x-component of the velocity is plotted along the vertical mid-line of the cavity, and the y-component along the horizontal mid-line. Small variations of the profiles were found with respect to those of the previous case; the recirculation in the cavity presented a main single vortex as in the previous case for small Re values. Likewise, for Re =500 and Re = 1000 in the range of Ri = 0.1 to Ri = 10, a single main recirculating cell was also observed. Figure A.3a displays the velocity profile for Re = 500 and all the values of Ri. Figure A.3b shows the profiles for Re = 1000, where the y-velocity profiles differ with respect to the case when the heated wall is at the bottom, because with a vertical heated wall, a single main structure is generated, for all ranges of Re and Ri studied.

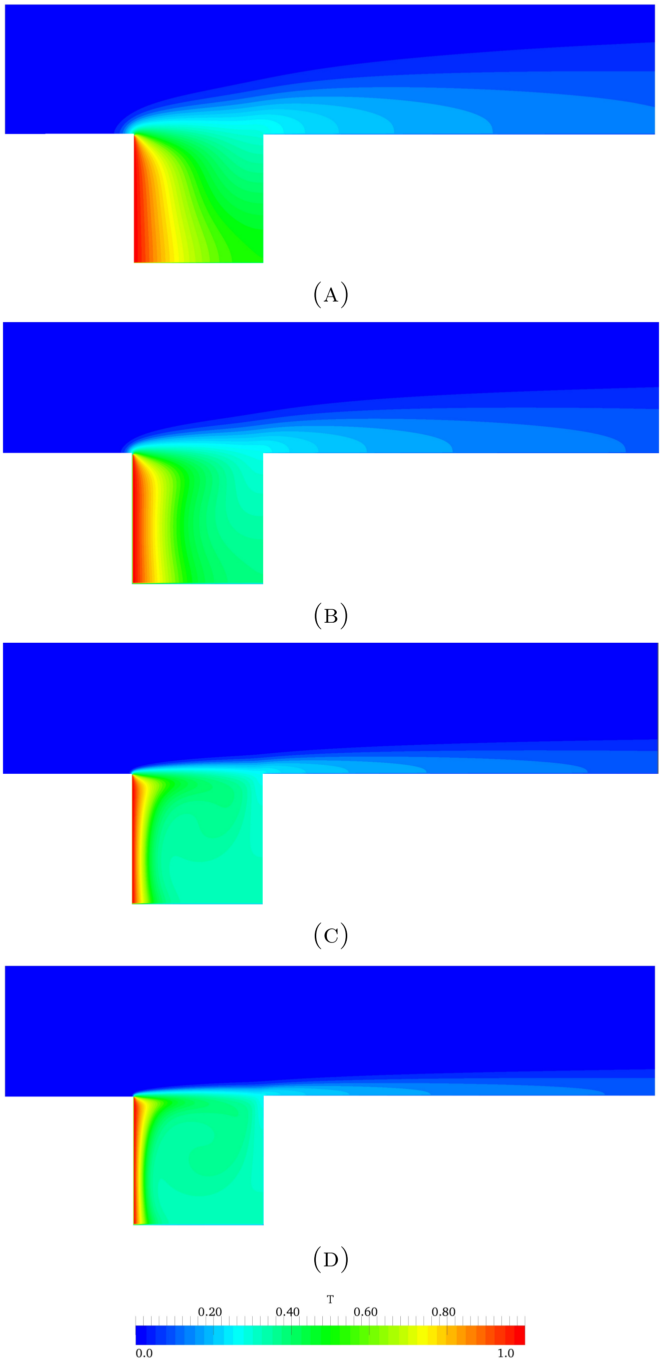
Figures. (A.4, A.5) show the contours of the dimensionless temperature for Ri =0 .1 and Ri = 1.0 for all the values of the Reynolds number Re = 50, 100, 500, 1000, in the steady state.



(A)

(B)

FIGURE A.3: Dimensionless velocity profiles along the vertical and horizontal mid lines of the cavity (left) and temperature (right), for A) Re=1000 and B) Re=1000.

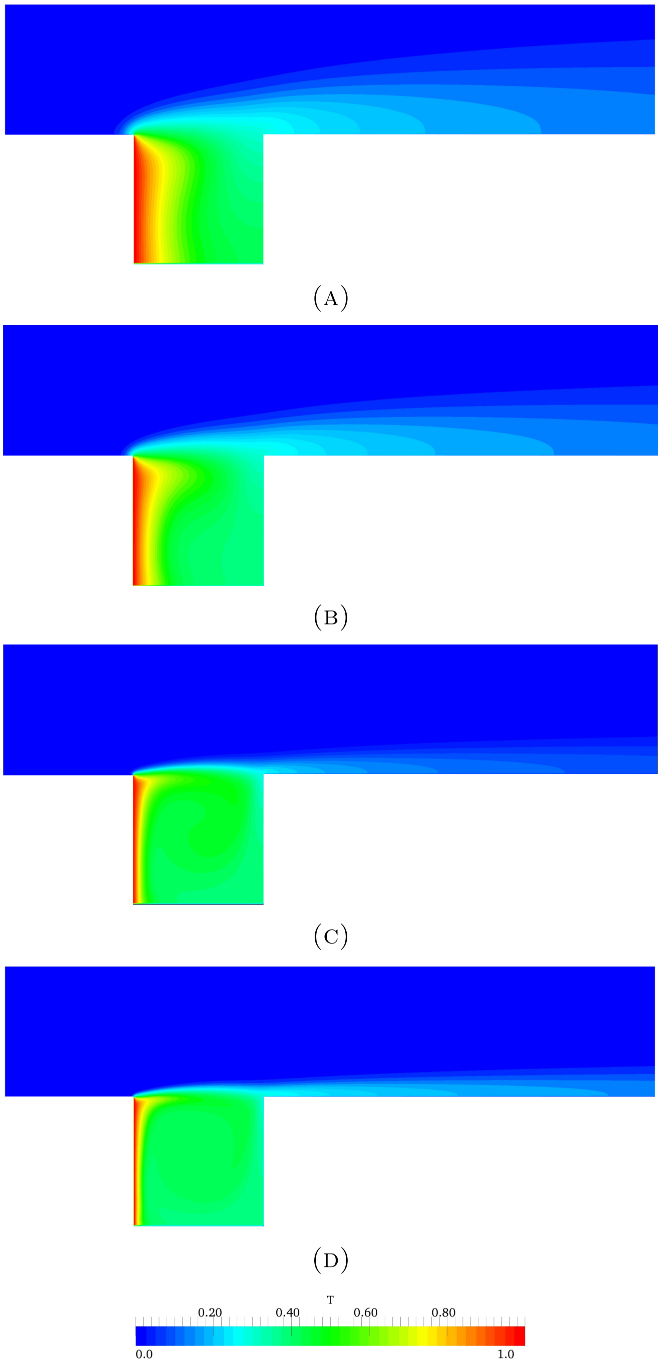

(A)

(B)

(C)

(D)

FIGURE A.4: Contours of the dimensionless temperature T for Ri=0.1: A) Re=50, B) Re=100, C) Re=500, D) Re=1000.

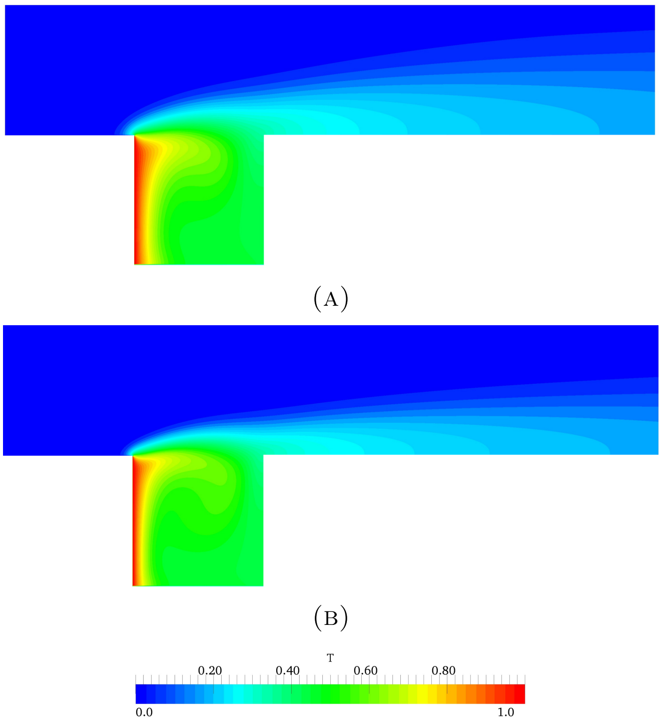

(A)

(B)

(C)

(D)

FIGURE A.5: Contours of the dimensionless temperature T for Ri = 1.0: A) Re = 50, B) Re = 100, C) Re = 500 , D) Re = 1000.


(A)

(B)

Figure A.6: Contours of the dimensionless temperature T for Ri = 10:
A) Re = 50, B) Re = 100.

Figure A.6 displays the contours of the dimensionless temperature for Re = 50 and Re = 100 (Ri = 10), where a single recirculation cell is found in the cavity and the flow continues encapsulated in the steady regime. Additionally, for Re=500 and Re=1000, the development of the unsteady flow is shown in Figs. (A.7, A.8), where the instantaneous contours of temperature can be observed.

In the same way as in the case of a bottom heated wall when Ri⩾10 and Re⩾500, a pseudo-periodic regime starts with an intermittent generation and emission of hot plumes. Then, the plume is dragged along the external channel. Figure A.11 shows the average contour of the dimensionless temperature of the cavity where a single main vertical structure can be observed.

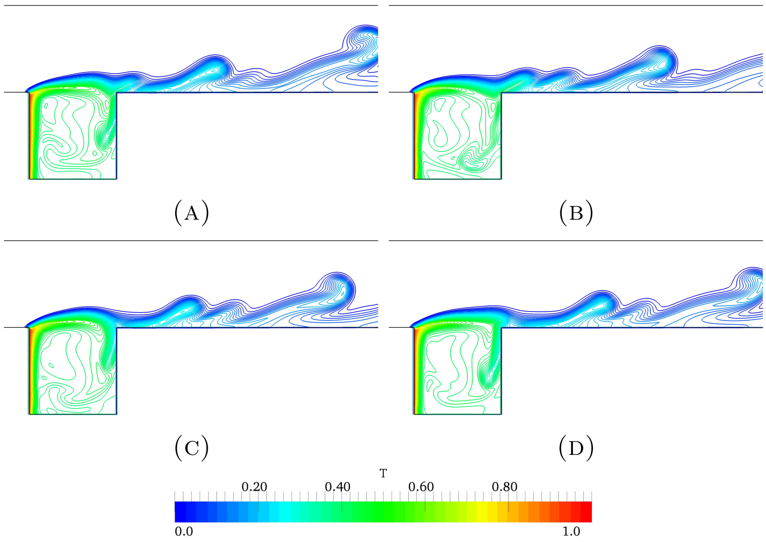


FIGURE A.7: (A - D) Sequence of instantaneous dimensionless temperature contours in the unsteady flow regime, showing the vortex production at the upstream corner of the enclosure with a single recirculation inside the cavity, for Re = 500 and Ri = 10.

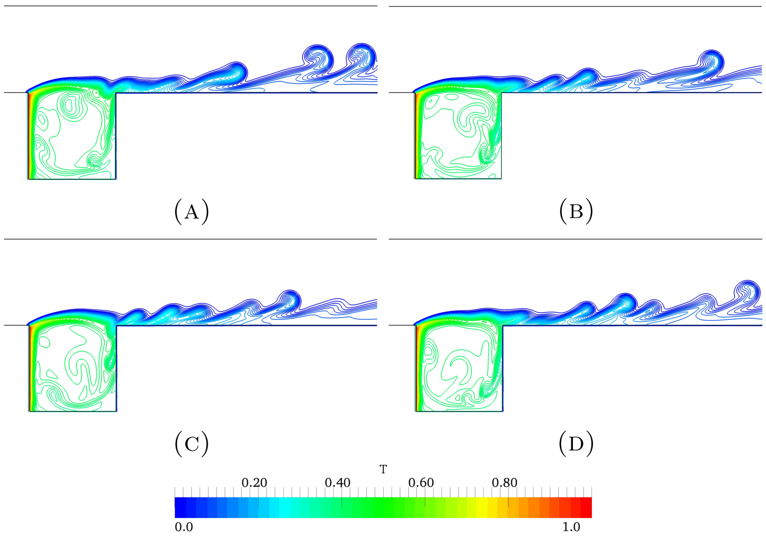
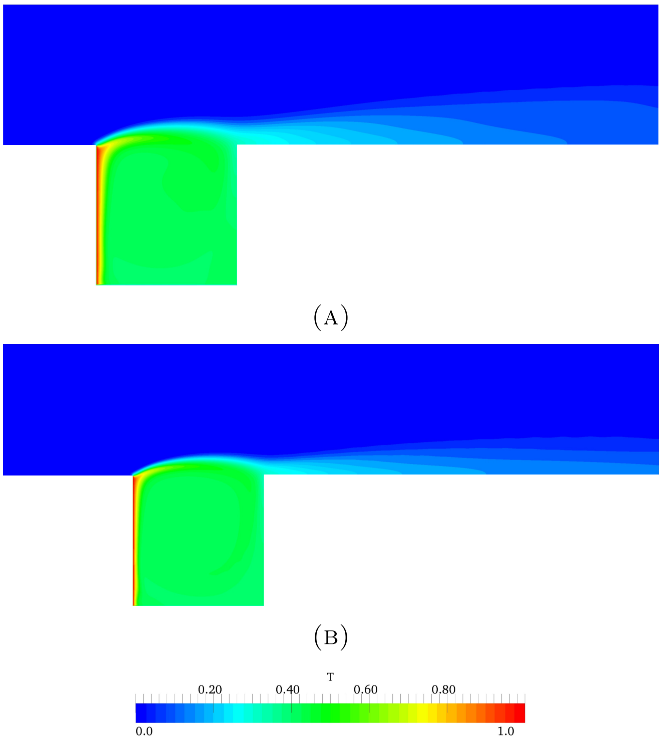


FIGURE A.8: (A - D) Sequence of instantaneous dimensionless temperature contours in the unsteady flow regime, showing the vortex production at the upstream corner of the enclosure with a single recirculation inside the cavity for Re = 1000 and Ri = 10.


(A)

(B)

FIGURE A.9: Contours of the average dimensionless temperature T for Ri = 10: A) Re = 500, B) Re = 1000.

For Re = 500 and Ri = 10, an interesting behavior was observed inside the cavity. Up to three small counter-rotating vortices were found inside the main recirculation region formed by the primal vortex of the cavity, rotating in opposed directions, with the main recirculation also moving these small structures in a particular direction. Some instantaneous of the streamlines of the velocity for Re = 500 and Ri = 10 are presented to better understand this distinctive effect (see Figs. A.10). The average streamlines of the velocity are shown in Fig. A.11a, where the small structures inside of the primal vortex are observed. For Re = 1000 and Ri = 10, these small structures do not appear, and only a pure main vortex is generated. However, a small structure is generated at the bottom left side of the cavity. These velocity streamlines are displayed in Fig. A.11b.

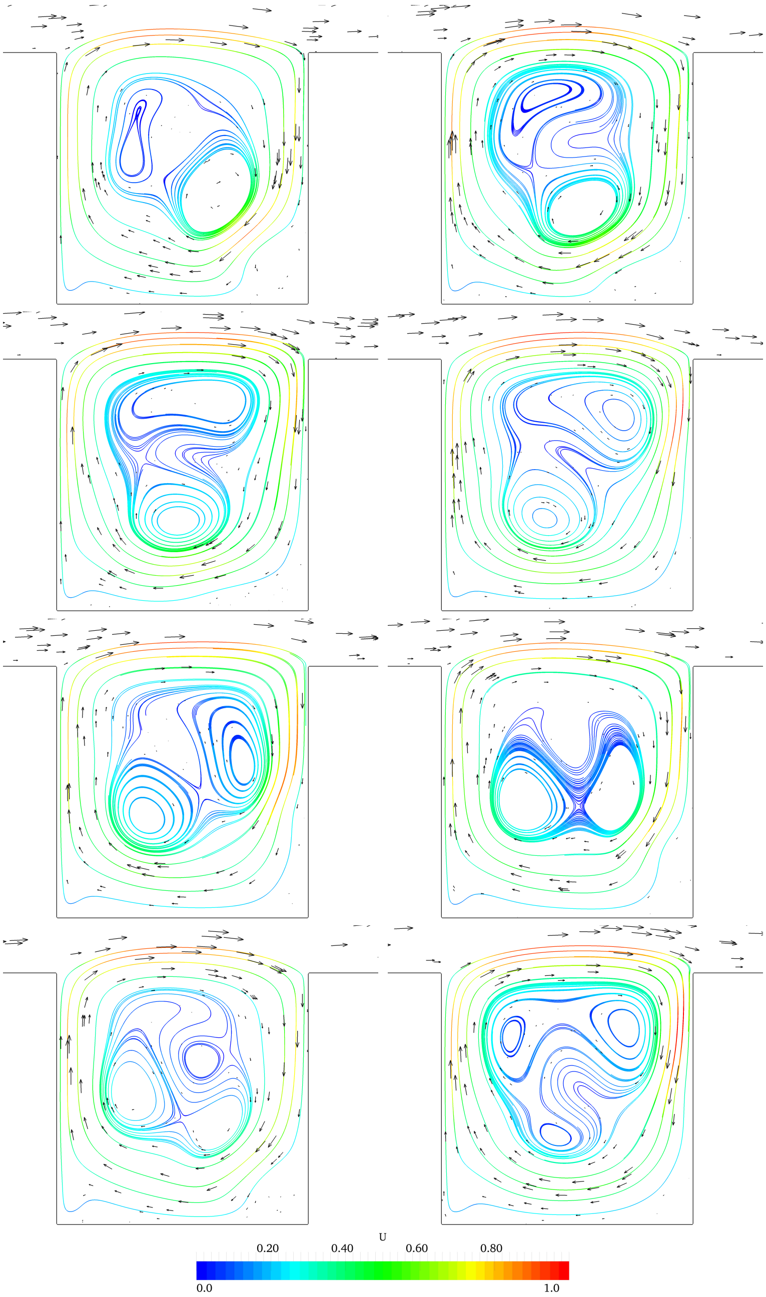
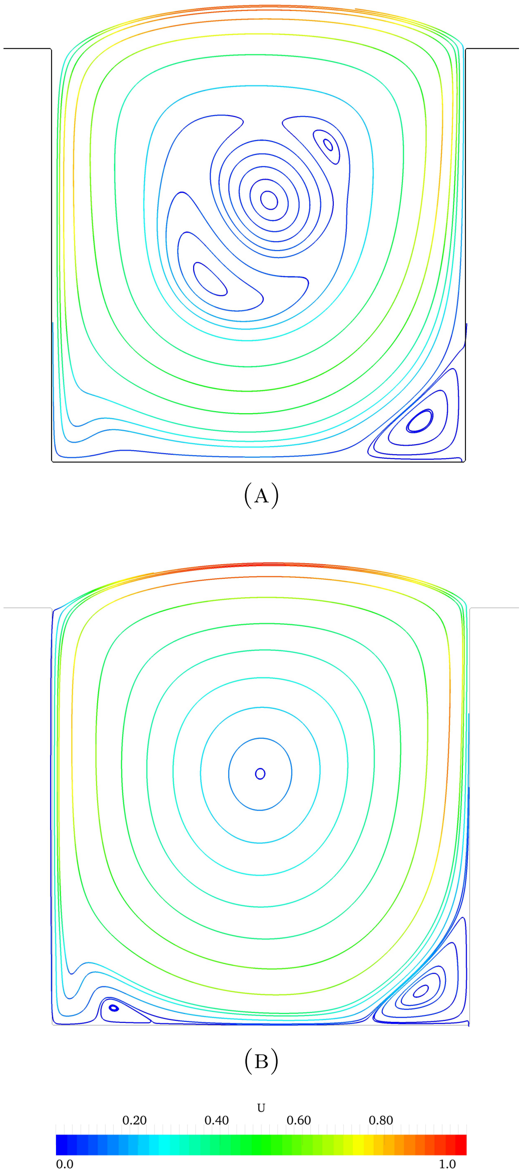


FIGURE A.10: Sequence of the instantaneous streamlines showing the flow structures present in the cavity, Re=500, Ri=10.



(A)

(B)

FIGURE A.11: Streamlines of the average dimensionless velocity in the cavity for Ri = 10: A) Re = 500, B) Re = 1000.

Appendix B

First order surface reaction in the open cavity

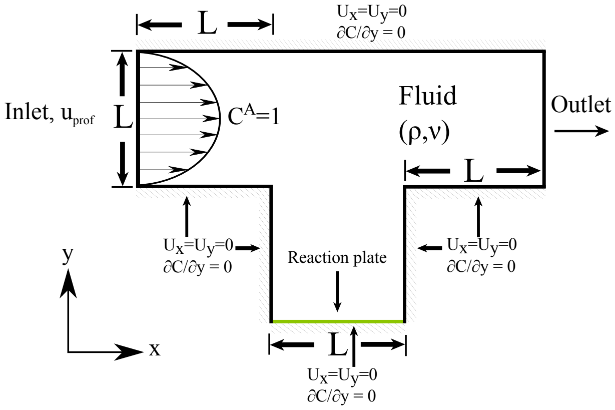


Figure B.1: Scheme of the open cavity with a reactive bottom flat plate.

$$A \longrightarrow B \tag{B.1}$$

During the course of the reaction, reactant A is consumed when coming into contact with the reaction flat plate, where product B is generated. We assume isothermal conditions and a reaction-diffusion dynamics, where both species are simply advected by an external flow. The geometry of the problem is the same as the system in Chapter. 5, with the reaction flat plate located at the bottom wall of the cavity. The reaction rate is determined by a measure of the concentration, either when the concentration of A decreases or the concentration of B increases. The consumption of reactant A is expressed in the following form

$$R = -\kappa C^A \tag{B.2}$$

where κ is the reaction constant and C^A, C^B are the concentrations of reactant A and product B, respectively. The flow field is modeled in LBM with the D2Q9 model, in a similar way as in Chapter. 5, where the same geometry has been used. For the concentration field of species A and B, two new lattice meshes for the mass transfer processes are required. The two new transport equations are written in the LBM formalism as relaxation equations for the distribution functions of both species. Additionally, a term for the chemical reaction S_y needs to be added as a source, which accounts for the rate of production of species B. It is calculated as

$$S_y = \omega_i \kappa C^A. \tag{B.3}$$

The LBM equations for species A and B then read

$$A_i(x + \mathbf{c}_i \Delta t, t + \Delta t) = A_i(x,t) + \frac{1}{\tau_A}\left[A_i^{eq}(x,t) - A_i(x,t)\right] - S_y, \tag{B.4}$$

$$B_i(x + \mathbf{c}_i \Delta t, t + \Delta t) = B_i(x,t) + \frac{1}{\tau_B}\left[B_i^{eq}(x,t) - B_i(x,t)\right] + S_y, \tag{B.5}$$

where A_i and B_i are the distribution function equations for the mass transfer of species A and B. The source term S_y in the distribution function of A_i is negative in order to express that the consumption of reactant A will correspond to the generation of product B.

The concentration of the species can be calculated through the following equations:

$$C^A(x,t) = \sum_{i=0}^{8} A_i(x,t), \qquad C^B(x,t) = \sum_{i=0}^{8} B_i(x,t), \tag{B.6}$$

The dimensionless parameter characteristic of the problem is the Damkohler number, defined as the ratio of the reaction rate over the diffusion rate,

$$\mathrm{Da} = \frac{\kappa\delta}{\mathcal{D}_{\mathcal{A}}} \tag{B.7}$$

where δ is the characteristic length of the reaction plate, and $\mathcal{D}_{\mathcal{A}}$ is the diffusion coefficient of species A.

For Da$>$ 1, the reaction rate is greater than the diffusion rate distribution. This is known as diffusion limited reaction (diffusion is slowest so its characteristics dominate and the reaction is assumed to be instantaneously in equilibrium). When Da$<$ 1, diffusion occurs faster than the reaction. Thus, diffusion reaches an equilibrium before the reaction reaches it. In this analysis, Da was fixed to 1, considering that the diffusion and reaction rate are bounded, which means that neither of them completely controls the system.

The Péclet number Pe can be determined with the relation of the Reynolds number and the Schmidt number as Pe=ReSc, where Sc is the Schmidt number and is defined as Sc=$\nu/\mathcal{D}$. The Schmidt number is the ratio of the momentum difussivity ν and the mass diffusivity $\mathcal{D}$ of the substance in the medium. For instance, the mass diffusivity of CO$_2$ in air is equal to $D_{co_2,air} = 19 \cdot 10^{-6}$, and the Schmidt number is Sc=1.14 [57]. In this scheme, the value of Sc was fixed to Sc$_A$=Sc$_B$ = 1. The reaction rate κ can be determined from Da to obtain the source term S_y in Eq. (B.3), needed in Eqs. (B.4,B.5) for A_i and B_i,

$$\kappa = \frac{Da \cdot \mathcal{D}_{\mathcal{A}}}{\delta}. \tag{B.8}$$

Subsequently, the diffusion coefficient is determined by

$$\mathcal{D}_{\mathcal{A}} = \frac{u\delta}{Pe} \tag{B.9}$$

where u is the characteristic flow velocity. The Péclet number Pe is calculated from Pe = ReSc for species A and B, varying Re = 1, 2, 5, 7, 10 for Da =1 .

We will show the results and a preliminary analysis. Figures. B.2, B.3 display the contours of the dimensionless concentration of A and B. The generation of B is controlled by Da via Eq. (B.8). The effect of the Reynolds number in the cavity is such that the material diffusion of B into the channel is reduced as Re increases, and the concentration of product B is then reduced.

Figure B.4 shows a plot with the concentrations of species A and B in the center of the cavity as a function of time. For Re = 1, the saturating concentration occurs as the product B reaches a value of 0.52, and the remaining of reactant A reaches 0.48. For Re = 10 instead, the maximum concentration of product B decreases to 0.37, whereas the concentration of A saturates at 0.63 as Fig. B.6 shows.

Figure (B.7) displays different simulations using different settings of the Damkohler number, Da = 0.1, 1.0, 10, 100. The Reynolds number was fixed at Re = 5 in order to observe the effect of the Damkohler number in the production of substance B. As one can observe, the increase of Da causes the concentration of B to increase (and the corresponding decrease of A) in both the cavity and the channel. Compare for example the results for Da = 0.1 (Figs. B.7a, B.7b) with those for Da = 100 (Figs. B.7g, B.7h). In the first case, the reaction speed is much lower, and the production of B reaches a maximum value of 0.06 in the center of the cavity. In the case of Da = 100, the reaction is much faster and the reactant A is absorbed almost entirely. Notice that the concentration of product B reaches a maximum at a value of 0.66 in the center of the cavity, as shown in Fig. B.10.

Finally, Fig. B.11 displays the value of the average Sherwood number Eq. (4.35) for product B and the set of Damkohler numbers used.

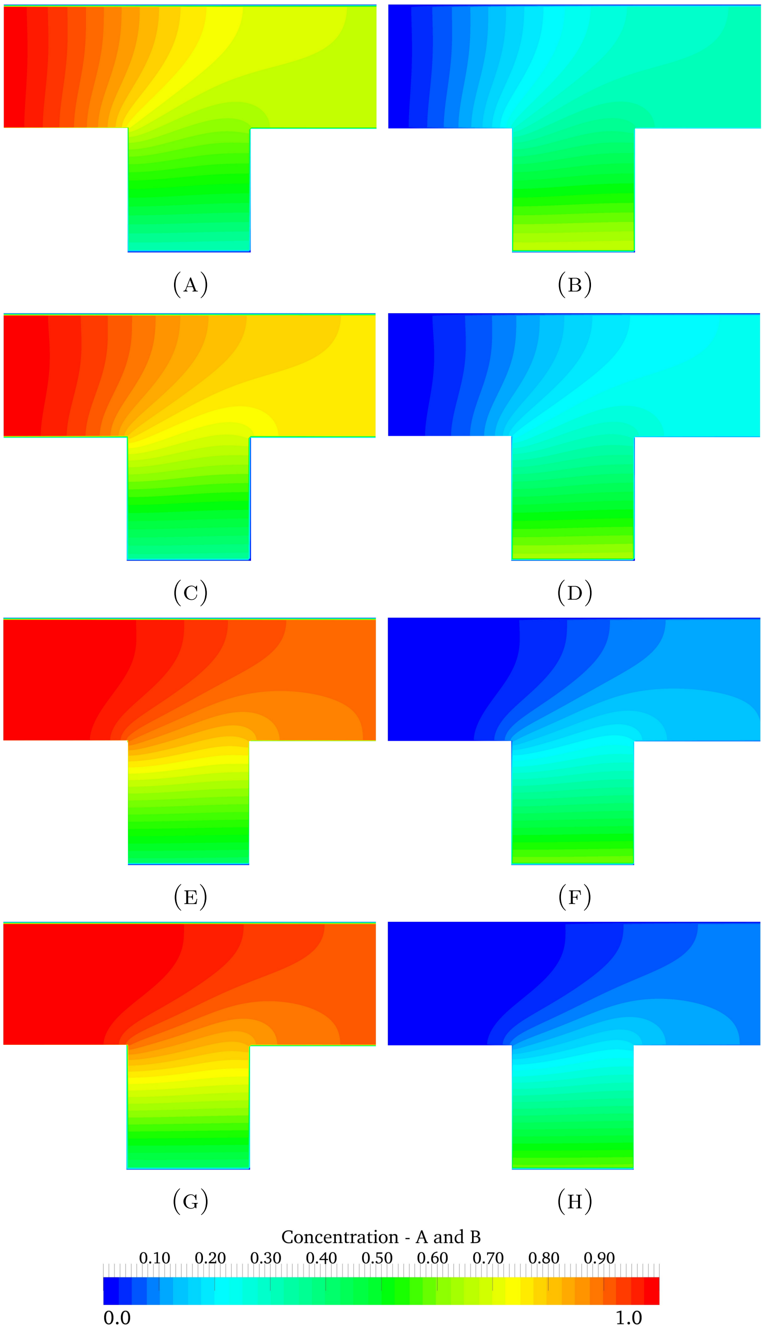


FIGURE B.2: Steady state dimensionless concentration for reactant A (left) and product B (right), for Da = 1 and varying Reynolds numbers: Re = 1 (A, B); Re = 2 (C, D); Re = 5 (E, F); Re = 7 (G, H).

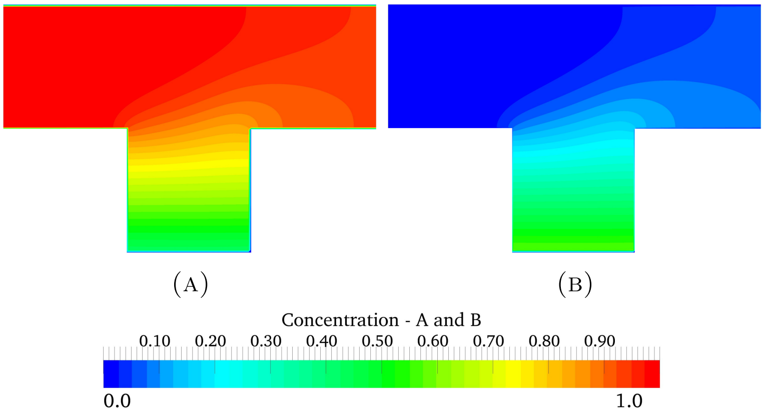


FIGURE B.3: Steady state dimensionless concentration for reactant A (left) and product B (right), for Da = 1 and Re =1 0.

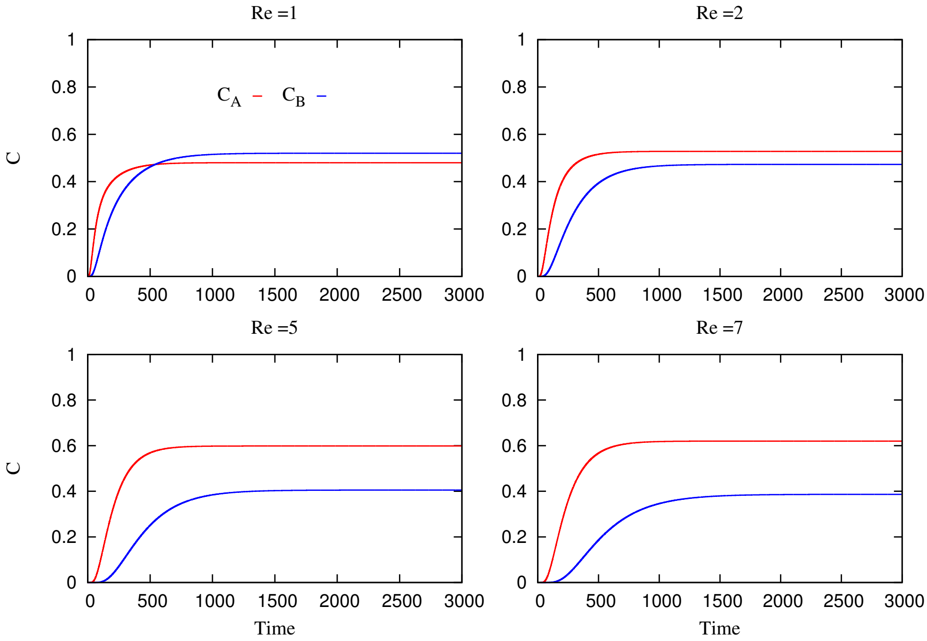


FIGURE B.4: Plot of the dimensionless concentration of A and B along time measured in the center of the cavity for Re = 1, 2, 5, 7 and Da = 1.

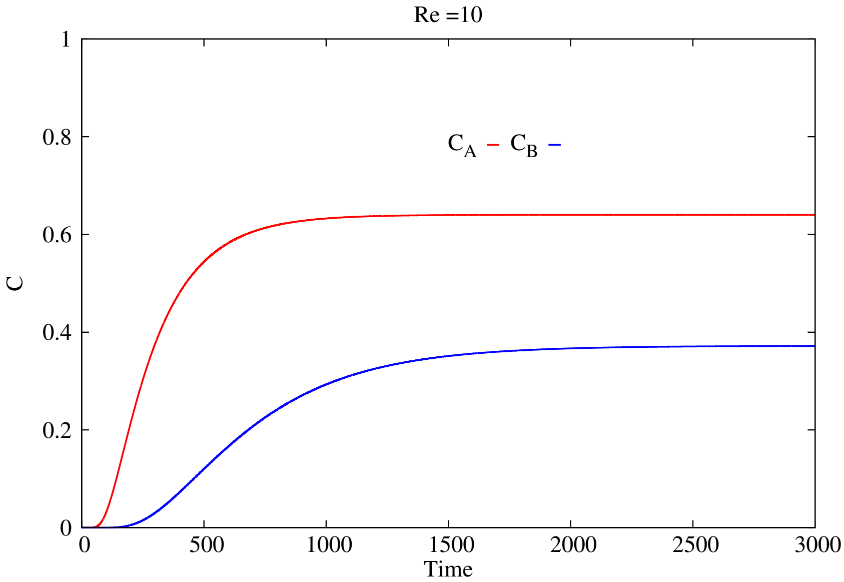


FIGURE B.5: Plot of the dimensionless concentration of A and B along time measured in the center of the cavity for Re = 10 and Da = 1.

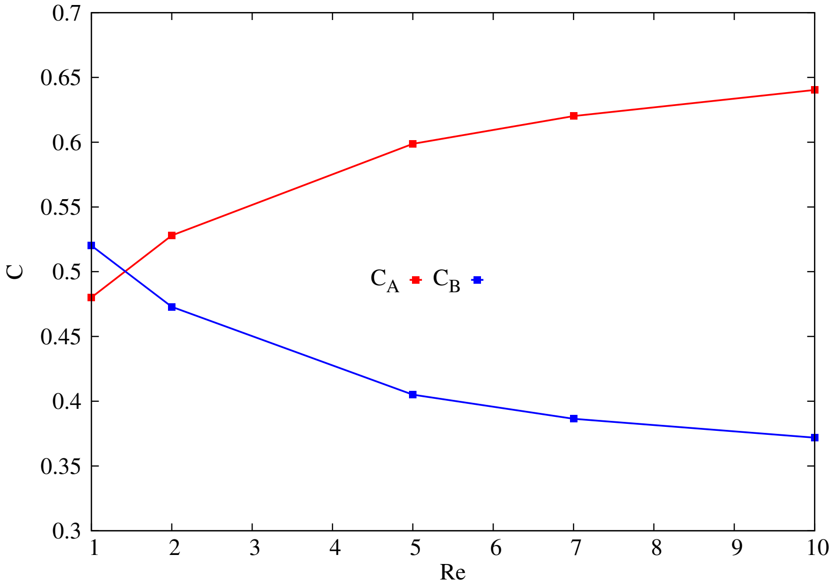


FIGURE B.6: Plot of the dimensionless concentration of A and B in the center of the cavity as a function of Re.

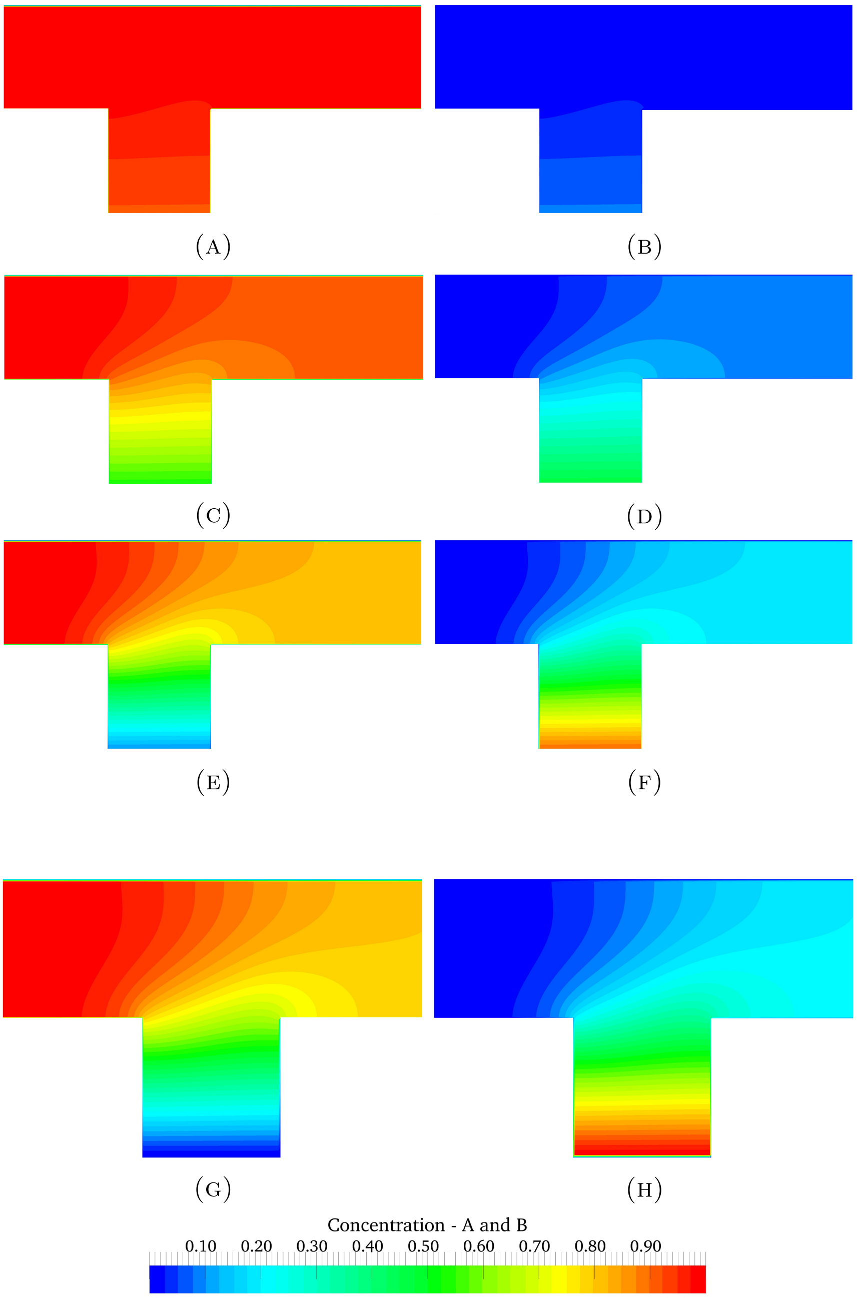


Figure B.7: Steady state dimensionless concentration for reactant A (left) and product B (right), for Re = 5 and varying Damkohler numbers: Da 0.1 (A, B); Da = 1 (C, D); Da = 10 (E, F); Da = 100 (G, H).

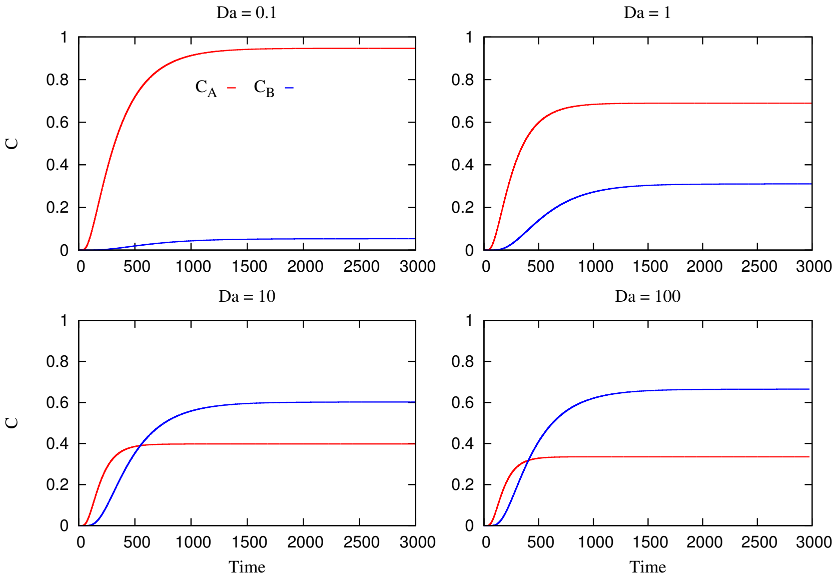


FIGURE B.8: Dimensionless concentration of A and B in the middle of the cavity, for Re = 5 and Da = 0.1, 1.0, 10 and 100.

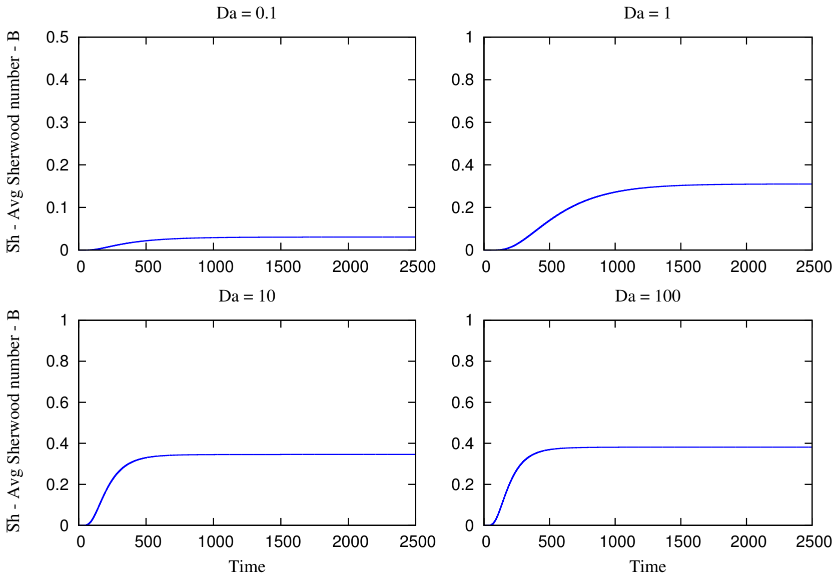


FIGURE B.9: Average Sherwood number for product B measured on the reaction plate at the bottom of the cavity, for Re = 5 and Da = 0.1, 1.0, 10 and 100.

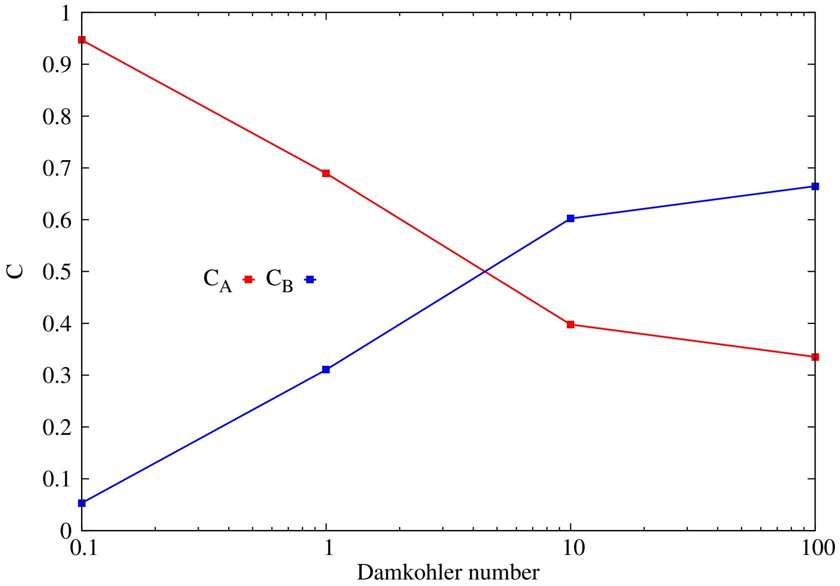


FIGURE B.10: Dimensionless concentration of A and B in the middle of the cavity as a function of Da, for Re = 5.

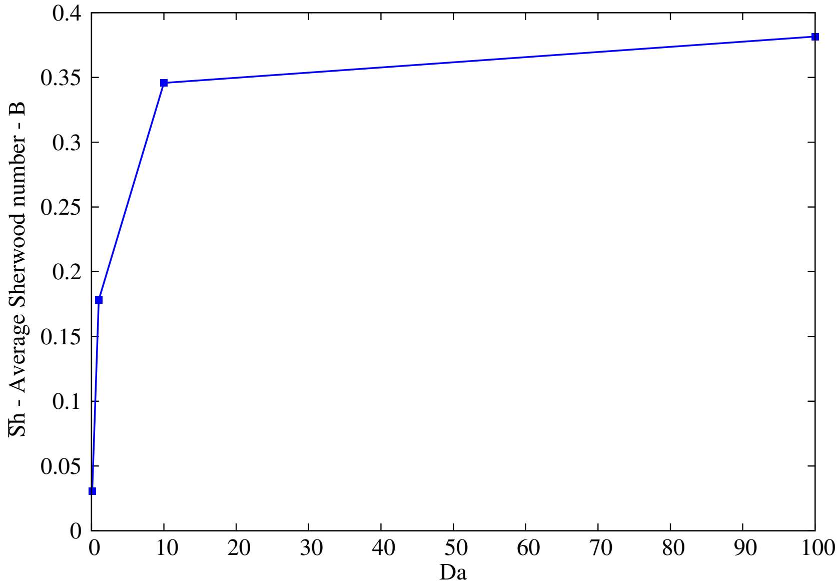


FIGURE B.11: Average Sherwood number for product B measured on the reaction plate at the bottom of the cavity as a function of Da, for Re = 5.

Appendix C

2D - Poiseuille flow in a channel

LBM code D2Q9 model - Flow in a channel, Fortran 90.
ParaView subroutine for post-processing data output (.VTK format file)

```
MODULE cellConst
integer, parameter:: fluid = 0, wall = 1, inlet = 2 , outlet = 3
END MODULE cellConst
MODULE D2Q9Const
!=================================================================================
! Weights for D2Q9-model, Discrete directions (v), Matrix M and S for MRT-model
!=================================================================================
double precision,parameter:: t(0:8) =
     (/4.0d0/9.0d0,1.0d0/9.0d0,1.0d0/9.0d0,1.0d0/9.0d0,1.0d0/9.0d0,1.0d0/36.0d0,&
&1.0d0/36.0d0,1.0d0/36.0d0,1.0d0/36.0d0/)
integer:: v(0:8,0:1)
integer:: opposite(0:8) = (/0,3,4,1,2,7,8,5,6/)
real:: Ma(0:8,0:8)
real:: Minvert(0:8,0:8)
real:: S(0:8,0:8)
real::SMinvert(0:8,0:8)
END MODULE D2Q9Const
! ========================================================
! Simulation Parameters:
! Reynolds number, length and height, Velocity of Lattice
! ========================================================
MODULE simParam
integer, parameter:: yDim = 52
integer, parameter:: yDim_d = yDim/2-1
integer, parameter:: xDim = yDim*6
integer, parameter:: iterper = 100
integer, parameter:: tMax = 3000000
double precision, parameter:: uMax = 0.02d0
double precision, parameter:: Re = 50
END MODULE simParam
! =========================================================
! Main program, principal loop
! =========================================================
PROGRAM unsteady
USE simParam, ONLY: xDim, yDim, tMax,iterper
implicit none
double precision:: omega, time1, time2, timeTot, Cs, nu
double precision, dimension(:,:,:), allocatable:: f, fEq, u, meq, m
double precision, dimension(:,:), allocatable:: rho, uSqr, mom
integer, dimension(:,:), allocatable:: image
integer:: tStep, iter, cont_4
! =========================================================
! Flow
! =========================================================
allocate(f(yDim,xDim,0:8))
allocate(fEq(yDim,xDim,0:8))
allocate(u(yDim,xDim,0:1))
allocate(uSqr(yDim,xDim))
allocate(rho(yDim,xDim))
allocate(image(yDim,xDim))
! =========================================================
!   MRT_model
! =========================================================
allocate(m(yDim,xDim,0:8))
allocate(meq(yDim,xDim,0:8))
CALL computeOmega(omega,nu,Cs)
CALL constructImages(image,omega)
CALL initMacro(rho,u,uSqr,image,iter,cont_4,tStep)
CALL computeFeq(fEq,rho,u,uSqr)
CALL computemeq(mEq,rho,u,uSqr)
f = fEq
timeTot = 0.0d0
do tStep = tStep, tMax , 1
CALL CPU_TIME(time1)
CALL inletOutlet(f,rho,u,image,uSqr)
CALL computeMacros(f,rho,u,uSqr,image)
CALL computeFeq(fEq,rho,u,uSqr)
CALL computemeq(mEq,rho,u,uSqr)
CALL collide(f,fEq,omega,image,u,rho,meq,uSqr,m)
CALL stream(f)
CALL boundaries(f,image)
CALL write_data(u,rho,tStep,f,iter,image,cont_4,omega,nu)
CALL CPU_TIME(time2)
timeTot = timeTot + (time2-time1)
end do
write(*,*) dble(tMax) * (dble(yDim * xDim)) / timeTot ,’cells per second’
write(*,*) ’total time:’, timeTot, ’ seconds’
deallocate(f)
deallocate(fEq)
deallocate(u)
deallocate(uSqr)
deallocate(rho)
deallocate(image)
deallocate(m)
deallocate(meq)
END PROGRAM unsteady
! ========================================================
!       Computer relaxation time
! ========================================================
SUBROUTINE computeOmega(omega,nu,Cs)
USE simParam, ONLY: xDim,yDim,Re,Umax
implicit none
double precision, INTENT(INOUT):: omega, nu
double precision :: Cs
Cs = (1.0d0/sqrt(3.0d0))
nu =  uMax * dble(yDim-1)/ Re
omega = 1.0d0 / (3.0d0*nu+0.5d0)
END SUBROUTINE computeOmega
! ========================================================
! Construct an array the defines the flow geometry
! ========================================================
SUBROUTINE constructImages(image,omega)
USE cellConst
USE simParam
USE D2Q9Const, ONLY: v, Ma, S, Minvert
implicit none
integer, INTENT(INOUT):: image(yDim,xDim)
double precision, INTENT(INOUT):: omega
double precision:: a
integer:: x,y,i,j
v(0:8,0) = (/0,1,0,-1,0,1,-1,-1,1/)
v(0:8,1) = (/0,0,1,0,-1,1,1,-1,-1/)
! ========================================================
! Domain - Cartesian coordinate system
! ========================================================
image=fluid
image(:,1)= inlet
image(:,xDim)= outlet
image(1,:)= wall
image(yDim,:)= wall
! ========================================================
! Matrix M for MRT-model
! ========================================================
Ma(0:8,0) = (/ 1, 1, 1, 1, 1,1, 1, 1, 1/)
Ma(0:8,1) = (/-4,-1,-1,-1,-1,2, 2, 2, 2/)
Ma(0:8,2) = (/ 4,-2,-2,-2,-2,1, 1, 1, 1/)
Ma(0:8,3) = (/ 0, 1, 0,-1, 0,1,-1,-1, 1/)
Ma(0:8,4) = (/ 0,-2, 0, 2, 0,1,-1,-1, 1/)
Ma(0:8,5) = (/ 0, 0, 1, 0,-1,1, 1,-1,-1/)
Ma(0:8,6) = (/ 0, 0,-2, 0, 2,1, 1,-1,-1/)
Ma(0:8,7) = (/ 0, 1,-1, 1,-1,0, 0, 0, 0/)
Ma(0:8,8) = (/ 0, 0, 0, 0, 0,1,-1, 1,-1/)
a = 1.0d0/36.0d0
! ========================================================
! Matrix M^-1 for MRT-model
! ========================================================
Minvert(0:8,0)=(/4.d0*a,-4.d0*a,4.d0*a,0.d0,0.d0,0.d0,0.d0,0.d0,0.d0/)
Minvert(0:8,1)=(/4.d0*a,-1.d0*a,-2.d0*a,6.d0*a,-6.d0*a,0.d0,0.d0,9.d0*a,0.d0/)
Minvert(0:8,2)=(/4.d0*a,-1.d0*a,-2.d0*a,0.d0,0.d0,6.d0*a,-6.d0*a,9.d0*a,0.d0/)
Minvert(0:8,3)=(/4.d0*a,-1.d0*a,-2.d0*a,-6.d0*a,6.d0*a,0.d0,0.d0,9.d0*a,0.d0/)
Minvert(0:8,4)=(/4.d0*a,-1.d0*a,-2.d0*a,0.d0*a,0.d0*a,-6.d0*a,6.d0*a,-9.d0*a,0.d0/)
Minvert(0:8,5)=(/4.d0*a,2.d0*a,1.d0*a,6.d0*a,3.d0*a,6.d0*a,3.d0*a,0.d0*a,9.d0*a/)
Minvert(0:8,6)=(/4.d0*a,2.d0*a,1.d0*a,-6.d0*a,-3.d0*a,6.d0*a,3.d0*a,0.d0*a,-9.d0*a/)
Minvert(0:8,7)=(/4.d0*a,2.d0*a,1.d0*a,-6.d0*a,-3.d0*a,6.d0*a,-3.d0*a,0.d0*a,9.d0*a/)
Minvert(0:8,8)=(/4.d0*a,2.d0*a,1.d0*a,6.d0*a,3.d0*a,-6.d0*a,-3.d0*a,0.d0*a,-9.d0*a/)
! =========================================================
! Matrix S for MRT-model
! =========================================================
!S(:) =    (/1.0d0,1.40d0,1.40d0,1.0d0,1.20d0,1.0d0,1.20d0,tau,tau/)
S(0:8,0) = (/1.d0, 0.d0 , 0.d0 , 0.d0 , 0.d0 , 0.d0, 0.d0 , 0.d0, 0.d0/)
S(0:8,1) = (/0.d0, 1.4d0, 0.d0 , 0.d0 , 0.d0 , 0.d0, 0.d0 , 0.d0, 0.d0/)
S(0:8,2) = (/0.d0, 0.d0 , 1.4d0, 0.d0 , 0.d0 , 0.d0, 0.d0 , 0.d0, 0.d0/)
S(0:8,3) = (/0.d0, 0.d0 , 0.d0 , 1.d0 , 0.d0 , 0.d0, 0.d0 , 0.d0, 0.d0/)
S(0:8,4) = (/0.d0, 0.d0 , 0.d0 , 0.d0 , 1.2d0, 0.d0, 0.d0 , 0.d0, 0.d0/)
S(0:8,5) = (/0.d0, 0.d0 , 0.d0 , 0.d0 , 0.d0 , 1.d0, 0.d0 , 0.d0, 0.d0/)
S(0:8,6) = (/0.d0, 0.d0 , 0.d0 , 0.d0 , 0.d0 , 0.d0, 1.2d0, 0.d0, 0.d0/)
S(0:8,7) = (/0.d0, 0.d0 , 0.d0 , 0.d0 , 0.d0 , 0.d0, 0.d0 , omega,0.d0/)
S(0:8,8) = (/0.d0, 0.d0 , 0.d0 , 0.d0 , 0.d0 , 0.d0, 0.d0 , 0.d0, omega /)
END SUBROUTINE constructImages
! =========================================================
! Initialization of Parameter: Umax, Density
! =========================================================
SUBROUTINE initMacro(rho,u,uSqr,image,iter,cont_4,tStep)
USE simParam, ONLY: xDim, yDim,uMax
USE cellConst
implicit none
double precision, INTENT(INOUT):: rho(yDim,xDim), u(yDim,xDim,0:1),
     uSqr(yDim,xDim)
integer, INTENT(INOUT):: iter,cont_4,tStep
integer, INTENT(IN):: image(yDim,xDim)
integer:: y,x,i
iter = 0 ; tStep = 1  ; cont_4 = 0
do x = 1, xDim
do y = 1, yDim
if (image(y,x) == wall) then
u(y,x,0) = 0.0d0
u(y,x,1) = 0.0d0
else
u(y,x,0) = umax
u(y,x,1) = 0.0d0
end if
end do
end do
rho  = 1.0d0
uSqr = u(:,:,0) * u(:,:,0) + u(:,:,1) * u(:,:,1)
END SUBROUTINE initMacro
! =========================================================
! Compute Equilibrium distribution function
! =========================================================
SUBROUTINE computeFeq(fEq,rho,u,uSqr)
```

```
USE D2Q9COnst, ONLY: t, v
USE simParam, ONLY: xDim, yDim
implicit none
double precision, INTENT(IN):: rho(yDim,xDim), uSqr(yDim,xDim), u(yDim,xDim,0:1)
double precision, INTENT(INOUT):: fEq(yDim,xDim,0:8)
integer:: i, x, y
double precision:: uxy
do x = 1, xDim
do y = 1, yDim
do i = 0, 8
uxy = u(y,x,0) * v(i,0) + u(y,x,1) * v(i,1)
fEq(y,x,i) = t(i) * rho(y,x) * (1.0d0 + 3.0d0 * uxy + 4.5d0 * uxy * uxy - 1.5d0 *
     uSqr(y,x))
end do
end do
end do
END SUBROUTINE computeFeq
! ==========================================================
! Compute Momentum equations for MRT-model D2Q9
! ==========================================================
SUBROUTINE computemeq(mEq,rho,u,uSqr)
USE D2Q9COnst, ONLY: t, v
USE simParam, ONLY: xDim, yDim
implicit none
double precision, INTENT(IN):: rho(yDim,xDim), uSqr(yDim,xDim), u(yDim,xDim,0:1)
double precision, INTENT(INOUT):: mEq(yDim,xDim,0:8)
integer:: i, x, y
double precision:: uxy
do x = 1, xDim
do y = 1, yDim
meq(y,x,0) =rho(y,x)
meq(y,x,1) =-2.0d0*rho(y,x) + 3.0d0*rho(y,x)*((u(y,x,0)*u(y,x,0) +
     u(y,x,1)*u(y,x,1)))
meq(y,x,2) =rho(y,x) -  3.0d0 *rho(y,x)*(u(y,x,0)*u(y,x,0) + u(y,x,1)*u(y,x,1))
meq(y,x,3) =u(y,x,0)*rho(y,x)
meq(y,x,4) =   -u(y,x,0)*rho(y,x)
meq(y,x,5) =u(y,x,1)*rho(y,x)
meq(y,x,6) =   -u(y,x,1)*rho(y,x)
meq(y,x,7) =   (u(y,x,0)*u(y,x,0)) - (u(y,x,1)*u(y,x,1))*rho(y,x)
meq(y,x,8) =   (u(y,x,0)*u(y,x,1))*rho(y,x)
end do
end do
END SUBROUTINE computemeq
! ==============================================================
! Compute Macro-scope variables, Density and x and Y velocities
! ==============================================================
SUBROUTINE computeMacros(f,rho,u,uSqr,image)
USE simParam, ONLY: xDIm, yDim
USE cellConst
USE D2Q9COnst, ONLY: t, v
implicit none
double precision, INTENT(IN):: f(yDim,xDim,0:8)
double precision, INTENT(INOUT):: u(yDim,xDim,0:1),rho(yDim, xDim),uSqr(yDim,
     xDim)
```

```
integer, INTENT(IN):: image(yDim,xDim)
integer:: x,y,i
do x = 1, xDim
do y = 1, yDim
if (image(y,x) == wall) then
u(y,x,0) = 0.0d0
u(y,x,1) = 0.0d0
else
rho(y,x)  = f(y,x,0) + f(y,x,1) + f(y,x,2) + f(y,x,3) + f(y,x,4) + f(y,x,5) +
     f(y,x,6) + f(y,x,7) + f(y,x,8)
u(y,x,0)  = ((f(y,x,1) - f(y,x,3) + f(y,x,5) - f(y,x,6) - f(y,x,7) + f(y,x,8)) )/
     rho(y,x)
u(y,x,1)  = ((f(y,x,2) - f(y,x,4) + f(y,x,5) + f(y,x,6) - f(y,x,7) - f(y,x,8))) /
     rho(y,x)
uSqr(y,x) = u(y,x,0) * u(y,x,0) + u(y,x,1) * u(y,x,1)
end if
end do
end do
END SUBROUTINE computeMacros
! ============================================================================
! Implement Boundary conditions, Bounce-Back scheme for solid wall, Full way
! ============================================================================
SUBROUTINE boundaries(f,image)
USE D2Q9Const
USE cellConst
USE simParam, ONLY: xDim, yDim
implicit none
integer, INTENT(IN):: image(yDim,xDim)
double precision, INTENT(INOUT):: f(yDim,xDim,0:8)
double precision:: fTmp(0:8)
integer:: i, x, y
do x = 1, xDim
do y = 1, yDim
if (image(y,x) == wall) then
do i = 0, 8
fTmp(i) = f(y,x,opposite(i))
end do
do i = 0, 8
f(y,x,i) = fTmp(i)
end do
end if
end do
end do
END SUBROUTINE boundaries
! ================================================================
! Inlet Outlet Boundary condition,
! ================================================================
SUBROUTINE inletOutlet(f,rho,u,image,uSqr)
USE cellConst, ONLY: inlet, outlet
USE simParam
USE D2Q9Const, ONLY: t, v
implicit none
double precision, INTENT(INOUT):: f(yDim,xDim,0:8), u(yDim,xDim,0:1),
     rho(yDim,xDim)
```

```
double precision, INTENT(IN):: uSqr(yDim,xDim)
integer, INTENT(IN):: image(yDim,xDim)
double precision:: uxy(0:8)
integer:: x, y, i
do x = 1, xDim
do y = 1, yDim
if (image(y,x) == inlet) then
u(y,x,0) = umax
u(y,x,1) = 0.0d0
CALL inletZou(f(y,x,:),u(y,x,:),rho(y,x)) ! Zou-he on inlet boundary condition
else if (image(y,x) == outlet) then
do i = 0, 8
f(y,x,i) = 2 *f(y,x-1,i) - f(y,x-2,i) !Open boundary condition on outlet
end do
end if
end do
end do
CONTAINS
! ================================================================
! Zou-he scheme, Rutine for outlet distribution functions
! ================================================================
SUBROUTINE inletZou(f,u,rho)
implicit none
double precision, INTENT(INOUT):: f(0:8),rho
double precision, INTENT(IN):: u(0:1)
double precision:: fInt, fInt2
fInt   = f(0) + f(2) + f(4)
fInt2  = f(3) + f(6) + f(7)
rho= (fInt + 2.0d0 * fInt2) / (1.0d0 - u(0))
CALL zouWestWall(f,rho,u)
END SUBROUTINE inletZou
SUBROUTINE zouWestWall(f,rho,u)
implicit none
double precision, INTENT(INOUT):: f(0:8)
double precision, INTENT(IN):: rho, u(0:1)
double precision:: fDiff, rhoUx, rhoUy
fDiff = 0.5d0 * (f(2) - f(4))
rhoUx = rho * u(0) / 6.0d0
rhoUy = 0.5d0 * rho * u(1)
f(1) = f(3) + 4.0d0 * rhoUx
f(5) = f(7) - fDiff + rhoUx + rhoUy
f(8) = f(6) + fDiff + rhoUx - rhoUy
END SUBROUTINE zouWestWall
END SUBROUTINE inletOutlet
! ================================================================
! Streaming step for all distribution functions, Full way
! ================================================================
SUBROUTINE stream(f)
USE simParam
USE cellConst
implicit none
double precision, INTENT(INOUT):: f(yDim,xDim,0:8)
f(:,2:xDim,1) = f(:,1:xDim-1,1) ! right direction
f(2:yDim,:,2) = f(1:yDim-1,:,2) ! up direction
```

```
f(:,1:xDim-1,3) = f(:,2:xDim,3) ! left direction
f(1:yDim-1,:,4) = f(2:yDim,:,4) ! down direction
f(2:yDim,2:xDim,5) = f(1:yDim-1,1:xDim-1,5) ! up-right direction
f(2:yDim,1:xDim-1,6) = f(1:yDim-1,2:xDim,6) ! up-left direction
f(1:yDim-1,1:xDim-1,7) = f(2:yDim,2:xDim,7) ! down-left direction
f(1:yDim-1,2:xDim,8) = f(2:yDim,1:xDim-1,8) ! down-right direction
END SUBROUTINE stream
! =======================================================
! LBGK collision step
! ========================================================
SUBROUTINE collide(f,fEq,omega,image,u,rho,meq,uSqr,m)
USE simParam, ONLY: xDim, yDim
USE cellConst
USE D2Q9Const, ONLY: Ma, Minvert,S,SMinvert,v,t
implicit none
integer, INTENT(IN):: image(yDim,xDim)
double precision, INTENT(IN):: fEq(yDim,xDim,0:8),
     omega,u(yDim,xDim,0:1),uSqr(yDim,xDim)
double precision, INTENT(IN):: rho(yDim,xDim),meq(yDim,xDim,0:8)
double precision, INTENT(INOUT):: f(yDim,xDim,0:8),m(yDim,xDim,0:8)
double precision:: mom , sume
integer:: x,y,i,j,k
do x = 1, xDim
do y = 1, yDim
do i = 0,8
mom = 0.0d0
do k = 0,8
mom = mom +  Ma(k,i) * f(y,x,k)
end do
m(y,x,i) = mom
end do
end do
end do

do i = 0,8
do j = 0,8
SMinvert(j,i) = 0.0d0
do k = 0,8
SMinvert(j,i) = SMinvert(j,i) + ( Minvert(k,i) * S(j,k))
end do
end do
end do
do x = 1, xDim
do y = 1, yDim
do i = 0, 8
sume = 0.0d0
do k = 0,8
sume = sume + SMinvert(k,i) * ( m(y,x,k) - meq(y,x,k))
end do
if (image(y,x) /= wall) then
! ================================================================
! SRT model, (remove exclamation mark for on) and comment MRT-model
! ================================================================
 !f(y,x,i) = (1.0d0 - omega) * f(y,x,i) + omega * feq(y,x,i)
```

```
! =================================================================
! MRT-model, compute the different matrices
! =================================================================
f(y,x,i) =  (f(y,x,i) - sume )
end if
end do
end do
end do
END SUBROUTINE collide
! =================================================================
! Post-processing, VTK file format .VKT ( PARAVIEW software)
! =================================================================
SUBROUTINE write_data(u,rho,tStep,f,iter,image,cont_4,omega,nu)
USE simParam
USE cellConst
USE D2Q9const,ONLY: v, opposite
implicit none
double precision, INTENT(IN)::
     u(yDim,xDim,0:1),f(yDim,xDim,0:8),rho(yDim,xDim),omega
integer, INTENT(INOUT):: iter,cont_4
integer, INTENT(IN):: image(yDim,xDim),tStep
double precision, INTENT(IN):: nu
double precision ::  d_loc_t,Cs
integer:: x,y,i
character (LEN=100):: file_name,filename, path1, pathTMP
character (LEN=100):: itercar,Ncar,Recar,Ucar,Macar, Racar, Ricar
Cs = (1.0d0/sqrt(3.0d0))
d_loc_t = 0.0d0
cont_4 = 0
do x = 1 , xDim
do y = 1 , yDim
if (image(y,x) /= wall) then
do i = 0 , 8
d_loc_t = d_loc_t + f(y,x,i)
end do
cont_4 = cont_4 +1
end if
end do
end do
if (d_loc_t >= yDim*xDim*2.0d0*rho(yDim,xDim)) then
write(*,*) 'Stop program, error'
STOP
end if
! =========================================================
! <<Generator for PARAVIEW ".vtk">>
! =========================================================
if (mod(tStep,1000)==0) then
iter = iter+1
write(file_name,'(a4,i5.4,a4)') 'data',iter,'.vtk'
write(path1,'(a47)') '/home/lattice/Documents/VTKs/2_MRT_D2Q9_canal/N'
write(itercar,'(i3)')iter
write(Ncar,'(i3.2)') xDim
write(Recar,'(i3.3)') int(Re)
file_name = trim(file_name)
```

```
OPEN(unit=15,file=trim(path1)//trim(Ncar)//’’//’/Re’//trim(Recar)
//’’//’/’//file_name,STATUS=’unknown’)

write(15,’(a26)’)’# vtk DataFile Version 3.0’
write(15,’(a5,i5,a7,i6.1)’)’iter ’,iter,’ total ’,tStep
write(15,’(a5)’)’ASCII’
write(15,’(a23)’)’DATASET STRUCTURED_GRID’
write(15,’(a10,2i6.2,a2)’)’DIMENSIONS’, xDim, yDim, ’ 1’
write(15,’(a7,i8,a6)’)’POINTS ’,xDim*yDim,’ float’
do y= 0 , (yDim-1)
do x= 0 , (xDim-1)
write(15,’(i6.1,i6.1,a2)’)x,y,’ 0’
end do
end do
write(15,*)
write(15,’(a11,i8)’)’POINT_DATA ’,xDim*yDim
! ================================================================
! Writing Domain
! ================================================================
write(15,’(a23)’)’SCALARS dominio FLOAT 1’
write(15,’(a20)’)’LOOKUP_TABLE default’
do y=1, yDim
do x=1, xDim
write(15,’(i2.1)’) image(y,x)
end do
end do
! ================================================================
! Writing Velocities Field
! ================================================================
write(15,’(a23)’)’VECTORS velocidad float’
do y= 1 , yDim
do x= 1 , xDim
write(15,’(F23.6,F23.6,a2)’)u(y,x,0),u(y,x,1),’ 0’
end do
end do
! ================================================================
! Writing Density of Lattice
! ================================================================
write(15,’(a24)’)’SCALARS densitat FLOAT 1’
write(15,’(a20)’)’LOOKUP_TABLE default’
do y = 1, yDim
do x = 1, xDim
 write(15,’(F31.16)’)rho(y,x)
end do
end do
close(15)

end if
!=======================================================
! <<U vs Tstep  >>
!=======================================================
if (mod(tStep,1)==0) then
write(file_name,’(a5)’) ’U.txt’
write(path1,'(a47)') '/home/lattice/Documents/VTKs/2_MRT_D2Q9_canal/N'
write(Ncar,'(i3.2)') xDim
write(Recar,'(i3.3)') int(Re)
file_name = trim(file_name)
OPEN(unit=20,file=trim(path1)//trim(Ncar)//''//'/Re'//trim(Recar)//''//
'/Parametros'//'/'//file_name,STATUS='unknown',position ='append')
 y = yDim/2
 x = xDim/2
write(20,'(i8,F18.7)')  tStep ,  u(y,x,0)
close(20)

if (mod(tStep,1)==0) then
write(file_name,'(a9)') 'U_max.txt'
write(path1,'(a47)') '/home/lattice/Documents/VTKs/2_MRT_D2Q9_canal/N'
write(Ncar,'(i3.2)') xDim
write(Recar,'(i3.3)') int(Re)
file_name = trim(file_name)
OPEN(unit=16,file=trim(path1)//trim(Ncar)//''//'/Re'//trim(Recar)//''
//'/Parametros'//'/'//file_name,STATUS='unknown')
do y = 1, yDim
x = xDim/2
write(16,'(i8,F12.7)') y,  u(y,x,0)
end do
close(16)

end if
end if
END SUBROUTINE write_data
! ========================================================
! Print out simulation parameters to screen
! ========================================================
SUBROUTINE writeInput(omega,nu)
USE simParam
implicit none
double precision, INTENT(INOUT):: omega, nu
write(*,*) 'xDim = ', xDim
write(*,*) 'yDim = ', yDim
write(*,*) 'tMax = ', tMax
write(*,*) 'nu   = ', nu
write(*,*) 'omega= ', omega
END SUBROUTINE writeInput
```

Appendix D

Lid Driven Cavity

LBM code - Flow in a channel, Fortran 90.
attach the routine for ParaView data.

Replaced the following MODULES and SUBROUTINE of the previous Code

```
!==========================================================
!Main parameter (input),
!yDim and xDim = dimension of the grid,
!uMax = 0.1(the maximum velocity that can be used in LBM)
!===========================================================
MODULE simParam
integer, parameter:: yDim = 600
integer, parameter:: xDim = 600
integer, parameter:: iterper = 100
integer, parameter:: tMax = 10000000
double precision, parameter:: uMax = 0.1
double precision, parameter:: Re = 7500
integer, parameter:: obstR = 8
real, parameter:: Ri = 10.00
END MODULE simParam
! ========================================================
!Compute viscotiy of lattice, the relaxation time tau, t > 0.51
! ========================================================
SUBROUTINE computeOmega(omega,nu,Cs)
USE simParam, ONLY: xDim,yDim,Re,Umax
implicit none
double precision, INTENT(INOUT):: omega, nu
double precision :: Cs
Cs = (1.0d0/sqrt(3.0d0))
nu =  (uMax * (dble(yDim-1))) / Re
omega = 1.0d0 / (3.0d0*nu+0.5d0)
END SUBROUTINE computeOmega
```

```
! ========================================================
! Construct an array the defines the flow geometry
! ========================================================
SUBROUTINE constructImages(image,omega)
USE cellConst
USE simParam
USE D2Q9Const
implicit none
integer, INTENT(INOUT):: image(yDim,xDim)
double precision, INTENT(INOUT):: omega
integer:: x,y,i,j
double precision:: a
v(0:8,0) = (/0,1,0,-1,0,1,-1,-1,1/)
v(0:8,1) = (/0,0,1,0,-1,1,1,-1,-1/)
image  = fluid
image(:,1)= wall
image(:,xDim)= wall
image(1,:)= wall
image(yDim,:)= lid_wall

\emph{Include the respective matrix for MRT model and the module of the momentum
     vector}

!========================================================================
!inizialization of the macroscopic variables, Bondary condition for moving wall,
 Zou-He scheme
!========================================================================
SUBROUTINE initMacro(rho,u,uSqr,image,iter,cont_4,tStep)
USE simParam, ONLY: xDim, yDim,uMax
USE cellConst
implicit none
double precision, INTENT(INOUT):: rho(yDim,xDim), u(yDim,xDim,0:1),
     uSqr(yDim,xDim)
integer, INTENT(INOUT):: iter,cont_4,tStep
integer, INTENT(IN):: image(yDim,xDim)
integer:: y,x,i
iter = 0 ; tStep = 1  ; cont_4 = 0
do x = 1, xDim
do y = 1, yDim
if (image(y,x) == wall)then
u(y,x,0) = 0.0d0
u(y,x,1) = 0.0d0
else
u(y,x,0) = uMax
u(y,x,0) = 0.0d0
end if
end do
end do
rho  = 1.0d0
uSqr = u(:,:,0) * u(:,:,0) + u(:,:,1) * u(:,:,1)
END SUBROUTINE initMacro
SUBROUTINE boundaries(f,image,u,uSqr,rho)
USE D2Q9Const
USE cellConst
USE simParam, ONLY: xDim, yDim,uMax
implicit none
integer, INTENT(IN):: image(yDim,xDim)
double precision, INTENT(INOUT)::
    f(yDim,xDim,0:8),u(yDim,xDim,0:1),uSqr(yDim,xDim)
double precision, INTENT(INOUT):: rho(yDim, xDim)
double precision:: fTmp(0:8), uxy(0:8)
integer:: i, x, y
do x = 1, xDim
do y = 1, yDim
if (image(y,x) == lid_wall) then
u(y,x,0) = uMax
u(y,x,1) = 0.0d0
CALL Norte(f(y,x,:),u(y,x,:),rho(y,x))
else if (image(y,x) == wall) then
do i = 0, 8
fTmp(i) = f(y,x,opposite(i))
end do
do i = 0, 8
f(y,x,i) = fTmp(i)
end do
end if
end do
end do
CONTAINS
! ========================================================
! Zou/He boundary on norte(up)
! ========================================================
SUBROUTINE Norte(f,u,rho)
implicit none
double precision, INTENT(INOUT):: f(0:8),rho
double precision, INTENT(IN):: u(0:1)
double precision:: fInt, fInt2
fInt= f(0) + f(1) + f(3)
fInt2  = f(2) + f(6) + f(5)
rho = (fInt + 2.0d0 * fInt2) / (1.0d0 + u(1))
CALL zouWestWall(f,rho,u)
END SUBROUTINE Norte
SUBROUTINE zouWestWall(f,rho,u)
implicit none
double precision, INTENT(INOUT):: f(0:8)
double precision, INTENT(IN):: rho, u(0:1)
double precision:: fDiff, rhoUx, rhoUy
fDiff = 0.5d0 * (f(1) - f(3))
rhoUy = rho * u(1) / 6.0d0
rhoUx = 0.5d0 * rho * u(0)
f(4) = f(2) - 4.0d0 * rhoUy
f(7) = f(5) + fDiff - rhoUy - rhoUx
f(8) = f(6) + 0.5d0 * (f(3) - f(1)) + rhoUx - rhoUy
END SUBROUTINE zouWestWall
END SUBROUTINE boundaries
```

Appendix E

Natural convection in a closed cavity

LBM code - Flow in a channel, Fortran 90.
ParaView software for post-processing data (.VTK format file)

```
! ========================================================
! Constants that identify different cell-types according
!to the dynamics they implement
! ========================================================
MODULE cellConst
integer, parameter:: fluid = 0,  wall_th = 3, wall_tc = 4, wall_top = 5, wall_bot
     = 6
integer, parameter:: corner_1 = 2, corner_2 = 7, corner_3 = 8, corner_4 = 9, wall
     = 1
integer, parameter:: wall_1 = 10 , wall_2 = 11, wall_3 = 12, wall_4= 13
integer, parameter:: corner_5 = 14, corner_6 = 15, corner_7 = 16, corner_8 = 17
END MODULE cellConst
! ========================================================
! Lattice constants for the D2Q9 lattice
! ========================================================
MODULE D2Q9Const
! D2Q9 Weights
double precision,parameter:: t(0:8) =
     (/4.0d0/9.0d0,1.0d0/9.0d0,1.0d0/9.0d0,1.0d0/9.0d0,1.0d0/9.0d0&
&,1.0d0/36.0d0,1.0d0/36.0d0,1.0d0/36.0d0,1.0d0/36.0d0/)
!D2Q9 Directions
integer:: v(0:8,0:1)
!= (/(/0,1,0,-1,0,1,-1,-1,1/),(/0,0,1,0,-1,1,1,-1,-1/)/)
integer, parameter:: opposite(0:8) = (/0,3,4,1,2,7,8,5,6/)

END MODULE D2Q9Const
! ========================================================
! Constants for simulation setup
! ========================================================
MODULE simParam
integer, parameter:: yDim = 202
integer, parameter:: xDim = yDim
integer, parameter:: iterper = 50
integer, parameter:: tMax = 10000000
double precision, parameter:: uMax = 0.0d0
integer, parameter:: tc = 0.0d0
integer, parameter:: th = 1.0d0
integer, parameter:: Ra = 10000
double precision, parameter:: Pr = 0.71d0
END MODULE simParam
! ========================================================
! The main program, implementing a Square_cavity with Temp
! ========================================================
PROGRAM Square_cavity
USE simParam, ONLY: xDim, yDim, tMax,iterper
implicit none
double precision:: omega, nu, time1, time2, timeTot, Cs,
     omega_t,alfa,gbeta,tv,tf,C,delta_x,delta_t
double precision, dimension(:,:,:), allocatable:: f, fEq, u, gEq,g
double precision, dimension(:,:), allocatable:: rho, uSqr,temp
integer, dimension(:,:), allocatable:: image,imaget
integer:: tStep, iter, cont, cont_2, cont_3, cont_4
allocate(f(yDim,xDim,0:8))
allocate(g(yDim,xDim,0:8))
allocate(fEq(yDim,xDim,0:8))
allocate(gEq(yDim,xDim,0:8))
allocate(u(yDim,xDim,0:1))
allocate(uSqr(yDim,xDim))
allocate(rho(yDim,xDim))
allocate(temp(yDim,xDim))
allocate(image(yDim,xDim))
allocate(imaget(yDim,xDim))
CALL constructImages(image)
CALL constructImagesTemp(imaget)
CALL computeOmega(omega,nu,Cs,omega_t,alfa,gbeta,tv,tf,C,delta_x,delta_t)
CALL initMacro(rho,u,uSqr,iter,cont,cont_2,cont_3,cont_4,tStep)
CALL initMacroTemp(temp,imaget)
CALL computeFeq(fEq,rho,u,uSqr)
CALL computeqEq(gEq,temp,u,uSqr)
f = fEq
g = gEq
timeTot = 0.0d0
do tStep = tStep, tMax , 1
CALL CPU_TIME(time1)
CALL boundaries(f,image)
CALL boundariesg(g,imaget)
CALL computeMacros(f,rho,u,uSqr,image)
CALL computeMacrosTemp(g,temp,imaget)
CALL computeFeq(fEq,rho,u,uSqr)
CALL computeqEq(gEq,temp,u,uSqr)
CALL write_data(u,rho,tStep,f,iter,image,nu,cont,cont_2,cont_3,
cont_4,omega,omega_t,gbeta,alfa,temp)
```

```
CALL collide(f,fEq,omega,image,u,rho,gbeta,temp)
CALL collideg(g,geq,omega_t,u,temp)
CALL stream(f)
CALL streamg(g)
CALL CPU_TIME(time2)
timeTot = timeTot + (time2-time1)
end do
write(*,*) dble(tMax) * (dble(yDim * xDim)) / timeTot ,'cells per second'
write(*,*) 'total time:', timeTot, ' seconds'
deallocate(f)
deallocate(g)
deallocate(fEq)
deallocate(gEq)
deallocate(u)
deallocate(uSqr)
deallocate(rho)
deallocate(temp)
deallocate(image)
deallocate(imaget)
END PROGRAM Square_cavity
! =========================================================
!  Compute the relaxation parameter, Rayleigh number, Prandtl number
! =========================================================
SUBROUTINE computeOmega(omega,nu,Cs,omega_t,alfa,gbeta,tv,tf,C,delta_x,delta_t)
USE simParam
implicit none
double precision, INTENT(INOUT):: omega, nu, omega_t, alfa,gbeta
double precision :: Cs,tv,tf,C,delta_x,delta_t
Cs = (1.0d0/sqrt(3.0d0))
nu = 0.01
omega = 1.0d0 / (3.0d0*nu+0.5d0)
!=======================================================================
!omega Termico
alfa = nu / Pr
omega_t = 1.0d0 / (3.0d0*alfa+0.5d0)
!=======================================================================
gbeta = Ra * nu * alfa / (xDim*xDim*xDim) !Rayleigh number
END SUBROUTINE computeOmega
! =========================================================
! Construct an array the defines the flow geometry
! =========================================================
SUBROUTINE constructImages(image)
USE cellConst
USE simParam, ONLY: xDim, yDim
USE D2Q9Const, ONLY: v
implicit none
integer, INTENT(INOUT):: image(yDim,xDim)
integer:: x,y,i
v(0:8,0) = (/0,1,0,-1,0,1,-1,-1,1/)
v(0:8,1) = (/0,0,1,0,-1,1,1,-1,-1/)
image= fluid
image(:,1)= wall_2
image(:,xDim)= wall_4
image(1,:)= wall_1
image(yDim,:)= wall_3
image(1,1)= corner_5
image(yDim,1)= corner_6
image(yDim,xDim)= corner_7
image(1,xDim)= corner_8
write(*,*) xDim, yDim
END SUBROUTINE constructImages
SUBROUTINE constructImagesTemp(imaget)
USE cellConst
USE simParam, ONLY: xDim, yDim
USE D2Q9Const, ONLY: v
implicit none
integer, INTENT(INOUT):: imaget(yDim,xDim)
integer:: x,y,i
v(0:8,0) = (/0,1,0,-1,0,1,-1,-1,1/)
v(0:8,1) = (/0,0,1,0,-1,1,1,-1,-1/)
imaget(:,1)= wall_th
imaget(:,xDim)= wall_tc
imaget(yDim,:)= wall_top
imaget(1,:)= wall_bot
imaget(1,1)= corner_1
imaget(yDim,1)= corner_2
imaget(yDim,xDim)= corner_3
imaget(1,xDim)= corner_4
END SUBROUTINE constructImagesTemp
! =========================================================
! Initialize
! =========================================================
SUBROUTINE initMacro(rho,u,uSqr,iter,cont,cont_2,cont_3,cont_4,tStep)
USE simParam
USE cellConst
implicit none
double precision, INTENT(INOUT):: rho(yDim,xDim), u(yDim,xDim,0:1),
     uSqr(yDim,xDim)
integer, INTENT(INOUT):: iter,cont,cont_2,cont_3,cont_4,tStep
double precision:: uProf
integer:: y,x,i
iter = 0 ; tStep = 1
cont = 0 ; cont_2 = 0 ; cont_3 = 0 ; cont_4 = 0
do y = 1, yDim
do x= 1, xDim
u(y,x,0) = 0.0d0
u(y,x,1) = 0.0d0
end do
end do
rho  = 1.0d0
uSqr = u(:,:,0) * u(:,:,0) + u(:,:,1) * u(:,:,1)
END SUBROUTINE initMacro
! =========================================================
! Initialize the simulation to Temperature
! =========================================================
SUBROUTINE initMacroTemp(temp,imaget)
USE simParam, ONLY: xDIm, yDim, tc, th
USE cellConst
```

```
implicit none
double precision, INTENT(INOUT):: temp(yDim,xDim)
integer, INTENT(IN):: imaget(yDim,xDim)
integer:: y,xe
do y = 1,yDim
do x = 1,xDim
if (imaget(y,x) == wall_th) then
temp(y,x) = th
else
temp(y,x) = tc
end if
end do
end do
END SUBROUTINE initMacroTemp
! =========================================================
! Compute equilibrium distribution
! =========================================================
SUBROUTINE computeFeq(fEq,rho,u,uSqr)
USE D2Q9COnst, ONLY: t, v
USE simParam, ONLY: xDim, yDim
implicit none
double precision, INTENT(IN):: rho(yDim,xDim), uSqr(yDim,xDim), u(yDim,xDim,0:1)
double precision, INTENT(INOUT):: fEq(yDim,xDim,0:8)
integer:: i, x, y
double precision:: uxy
do i = 0, 8
do x = 1, xDim
do y = 1, yDim
uxy = u(y,x,0) * v(i,0) + u(y,x,1) * v(i,1)
fEq(y,x,i) = t(i) * rho(y,x) * (1.0d0 + 3.0d0 * uxy + 4.5d0 * uxy * uxy - 1.5d0 *
     uSqr(y,x))
end do
end do
end do
END SUBROUTINE computeFeq
 =========================================================
! Compute equilibrium distribution, Temperature
! =========================================================
SUBROUTINE computeqEq(gEq,temp,u,uSqr)
USE D2Q9COnst, ONLY: t, v
USE simParam, ONLY: xDim, yDim
implicit none
double precision, INTENT(IN):: temp(yDim,xDim), uSqr(yDim,xDim), u(yDim,xDim,0:1)
double precision, INTENT(INOUT):: gEq(yDim,xDim,0:8)
integer:: i, x, y
double precision:: uxy
do i = 0, 8
do x = 1, xDim
do y = 1, yDim
uxy = u(y,x,0) * v(i,0) + u(y,x,1) * v(i,1)
gEq(y,x,i) = t(i) * temp(y,x) * (1.0d0 + 3.0d0 * uxy + 4.5d0 * uxy * uxy - 1.5d0
     * uSqr(y,x))

end do
end do
end do
END SUBROUTINE computeqEq
! ========================================================
! Compute density and velocity from distribution functions
! ========================================================
SUBROUTINE computeMacros(f,rho,u,uSqr,image)
USE simParam, ONLY: xDIm, yDim
USE cellConst
implicit none
double precision, INTENT(IN):: f(yDim,xDim,0:8)
double precision, INTENT(INOUT):: u(yDim,xDim,0:1),rho(yDim, xDim),uSqr(yDim,
     xDim)
integer, INTENT(IN):: image(yDim,xDim)
integer:: x,y
do x = 1, xDim
do y = 1, yDim
rho(y,x)  = f(y,x,0) + f(y,x,1) + f(y,x,2) + f(y,x,3) + f(y,x,4) + f(y,x,5) +
     f(y,x,6) + f(y,x,7) + f(y,x,8)
u(y,x,0)  = (f(y,x,1) - f(y,x,3) + f(y,x,5) - f(y,x,6) - f(y,x,7) + f(y,x,8)) /
     rho(y,x)
u(y,x,1)  = (f(y,x,2) - f(y,x,4) + f(y,x,5) + f(y,x,6) - f(y,x,7) - f(y,x,8)) /
     rho(y,x)
uSqr(y,x) = u(y,x,0) * u(y,x,0) + u(y,x,1) * u(y,x,1)
end do
end do
END SUBROUTINE computeMacros
! ========================================================
! Compute Temperature from distribution functions
! ========================================================
SUBROUTINE computeMacrosTemp(g,temp,imaget)
USE simParam, ONLY: xDIm, yDim, tc, th
USE cellConst
implicit none
double precision, INTENT(IN):: g(yDim,xDim,0:8)
double precision, INTENT(INOUT):: temp(yDim, xDim)
integer, INTENT(IN)::  imaget(yDim,xDim)
integer:: x,y
do x = 1,xDim
do y = 1,yDim
if (imaget(y,x) == wall_th) then
temp(y,x) = th
else if (imaget(y,x) == wall_tc) then
temp(y,x) = tc
else
temp(y,x) =  g(y,x,0) + g(y,x,1) + g(y,x,2) + g(y,x,3) + g(y,x,4) + g(y,x,5) +
     g(y,x,6) + g(y,x,7) + g(y,x,8)
end if
end do
end do
END SUBROUTINE computeMacrosTemp
!=============================================================================
! Implement Bounce-back on upper/lower/left/rigth boundaries, corner boundaries
!=============================================================================
SUBROUTINE boundaries(f,image)
USE D2Q9Const, ONLY: opposite
USE cellConst
USE simParam, ONLY: xDim, yDim
implicit none
integer, INTENT(IN):: image(yDim,xDim)
double precision, INTENT(INOUT):: f(yDim,xDim,0:8)
double precision:: fTmp(0:8)
integer:: i, x, y
do y= 1,yDim
do x= 1,xDim
if (image(y,x) == wall_1) then
f(y,x,6) = f(y,x,8)
f(y,x,5) = f(y,x,7)
f(y,x,2) = f(y,x,4)
f(y,x,3) = f(y,x,1)
else if (image(y,x) == wall_4) then
f(y,x,3) = f(y,x,1)
f(y,x,7) = f(y,x,5)
f(y,x,6) = f(y,x,8)
f(y,x,4) = f(y,x,2)
 else if (image(y,x) == wall_2) then
f(y,x,1) = f(y,x,3)
f(y,x,5) = f(y,x,7)
f(y,x,8) = f(y,x,6)
f(y,x,4) = f(y,x,2)
else if (image(y,x) == wall_3) then
f(y,x,4) = f(y,x,2)
f(y,x,7) = f(y,x,5)
f(y,x,8) = f(y,x,6)
f(y,x,0) = f(y,x,0)
f(y,x,3) = f(y,x,1)
else if (image(y,x) == corner_5) then
f(y,x,5) = f(y,x,7)
else if (image(y,x) == corner_6) then
f(y,x,8) = f(y,x,6)
else if (image(y,x) == corner_7) then
f(y,x,7) = f(y,x,5)
else if (image(y,x) == corner_8) then
f(y,x,6) = f(y,x,8)
end if
end do
end do
END SUBROUTINE boundaries
!==========================================================================
! Implement Bounce-back on upper/lower/left/rigth boundaries, Temperature
!==========================================================================
SUBROUTINE boundariesg(g,imaget)
USE D2Q9Const
USE cellConst
USE simParam, ONLY: xDim, yDim,th,tc
implicit none
integer, INTENT(IN):: imaget(yDim,xDim)
double precision, INTENT(INOUT):: g(yDim,xDim,0:8)
```

```
integer:: i, x, y
do y= 1,yDim
do x= 1,xDim
!==========================================
!Hot wall
if (imaget(y,x) == wall_th) then
g(y,x,1) = th * (t(1)+ t(3))-g(y,x,3)
g(y,x,5) = th * (t(5)+ t(7))-g(y,x,7)
g(y,x,8) = th * (t(8)+ t(6))-g(y,x,6)
!==========================================
!Cold wall
else if (imaget(y,x) == wall_tc) then
g(y,x,6) = -g(y,x,8)
g(y,x,3) = -g(y,x,1)
g(y,x,7) = -g(y,x,5)
!========================================================
!A.A Mohamed LBM
! Adiabatica wall
!==========================================
else if (imaget(y,x) == wall_top) then
g(yDim,x,8) = g(yDim-1,x,8)
g(yDim,x,7) = g(yDim-1,x,7)
g(yDim,x,6) = g(yDim-1,x,6)
g(yDim,x,5) = g(yDim-1,x,5)
g(yDim,x,4) = g(yDim-1,x,4)
g(yDim,x,3) = g(yDim-1,x,3)
g(yDim,x,2) = g(yDim-1,x,2)
g(yDim,x,1) = g(yDim-1,x,1)
g(yDim,x,0) = g(yDim-1,x,0)
!==========================================
! Adiabatic wall
!==========================================
else if (imaget(y,x) == wall_bot) then
g(1,x,8) = g(2,x,8)
g(1,x,7) = g(2,x,7)
g(1,x,6) = g(2,x,6)
g(1,x,5) = g(2,x,5)
g(1,x,4) = g(2,x,4)
g(1,x,3) = g(2,x,3)
g(1,x,2) = g(2,x,2)
g(1,x,1) = g(2,x,1)
g(1,x,0) = g(2,x,0)
!******************************************************************************
!Other configuration of Adiabatic boundary conditions g7=g5_n-1
!******************************************************************************
!else if (imaget(y,x) == wall_top) then
!g(yDim,x,8) = g(yDim-1,x,6)
!g(yDim,x,7) = g(yDim-1,x,5)
!g(yDim,x,4) = g(yDim-1,x,2)
!else if (imaget(y,x) == wall_bot) then
!g(1,x,6) = g(2,x,8)
!g(1,x,5) = g(2,x,7)
!g(1,x,2) = g(2,x,4)
end if
end do
end do
END SUBROUTINE boundariesg
! ========================================================
! Streaming step
! ========================================================
SUBROUTINE stream(f)
USE simParam
implicit none
double precision, INTENT(INOUT):: f(yDim,xDim,0:8)
! right direction
f(:,2:xDim,1) = f(:,1:xDim-1,1)
! up direction
f(2:yDim,:,2) = f(1:yDim-1,:,2)
! left direction
f(:,1:xDim-1,3) = f(:,2:xDim,3)
! down direction
f(1:yDim-1,:,4) = f(2:yDim,:,4)
! up-right direction
f(2:yDim,2:xDim,5) = f(1:yDim-1,1:xDim-1,5)
! up-left direction
f(2:yDim,1:xDim-1,6) = f(1:yDim-1,2:xDim,6)
! down-left direction
f(1:yDim-1,1:xDim-1,7) = f(2:yDim,2:xDim,7)
! down-right direction
f(1:yDim-1,2:xDim,8) = f(2:yDim,1:xDim-1,8)
END SUBROUTINE stream
! ========================================================
! Streaming step, Temperature
! ========================================================
SUBROUTINE streamg(g)
USE simParam
implicit none
double precision, INTENT(INOUT):: g(yDim,xDim,0:8)
! right direction
g(:,2:xDim,1) = g(:,1:xDim-1,1)
! up direction
g(2:yDim,:,2) = g(1:yDim-1,:,2)
! left direction
g(:,1:xDim-1,3) = g(:,2:xDim,3)
! down direction
g(1:yDim-1,:,4) = g(2:yDim,:,4)
! up-right direction
g(2:yDim,2:xDim,5) = g(1:yDim-1,1:xDim-1,5)
! up-left direction
g(2:yDim,1:xDim-1,6) = g(1:yDim-1,2:xDim,6)-
! down-left direction
g(1:yDim-1,1:xDim-1,7) = g(2:yDim,2:xDim,7)
! down-right direction
g(1:yDim-1,2:xDim,8) = g(2:yDim,1:xDim-1,8)
END SUBROUTINE streamg
! ========================================================
! LBGK collision step
! ========================================================
SUBROUTINE collide(f,fEq,omega,image,u,rho,gbeta,temp)
USE simParam, ONLY: xDim, yDim,tc
USE cellConst
USE D2Q9COnst, ONLY: t, v
implicit none
integer, INTENT(IN):: image(yDim,xDim)
double precision, INTENT(IN):: fEq(yDim,xDim,0:8), omega,u(yDim,xDim,0:1),gbeta
double precision, INTENT(IN):: rho(yDim,xDim),temp(yDim,xDim)
double precision, INTENT(INOUT):: f(yDim,xDim,0:8)
double precision:: force
integer:: x,y,i
do x = 1, xDim
do y = 1, yDim
do i = 0, 8
force = 3.0 * t(i) * gbeta * temp(y,x) * v(i,1) * rho(y,x)
if (( x .eq. 1) .or. (x .eq. xDim)) force = 0.0
if (( y .eq. 1) .or. (y .eq. yDim)) force = 0.0
if ((image(y,x) /= wall_1).or.(image(y,x) /= wall_2).or.(image(y,x) /=
     wall_3).or.&
(image(y,x) /= wall_4).or.(image(y,x) /= corner_5).or.(image(y,x) /=
     corner_6).or.&
&(image(y,x) /= corner_7).or.(image(y,x) /= corner_8)) then
f(y,x,i) = (1.0d0 - omega) * f(y,x,i) + omega * feq(y,x,i) +  force
end if
end do
end do
end do
END SUBROUTINE collide
! ========================================================
! LBGK collision step, Temperature
! ========================================================
SUBROUTINE collideg(g,geq,omega_t,u,temp)
USE simParam, ONLY: xDim, yDim,tc
USE cellConst
USE D2Q9COnst, ONLY: t, v
implicit none
double precision, INTENT(IN):: omega_t,gEq(yDim,xDim,0:8),u(yDim,xDim,0:1)
double precision, INTENT(IN):: temp(yDim,xDim)
double precision, INTENT(INOUT):: g(yDim,xDim,0:8)
integer:: x,y,i
do x = 1, xDim
do y = 1, yDim
do i = 0, 8
g(y,x,i) = (1.0d0 - omega_t) * g(y,x,i) + omega_t * geq(y,x,i)
end do
end do
end do
END SUBROUTINE collideg
! ========================================================
! <<POSTPROCESSING>>
! Write coordinates, velocities, domain in vtk format
!calculation of coeffs.
! ========================================================
SUBROUTINE write_data(u,rho,tStep,f,iter,image,nu,cont,cont_2,cont_3,cont_4,
omega,omega_t,gbeta,alfa,temp)
USE simParam
USE cellConst
USE D2Q9const,ONLY: v, opposite
implicit none
double precision, INTENT(IN)::u(yDim,xDim,0:1),f(yDim,xDim,0:8),nu,
rho(yDim,xDim),omega,omega_t,gbeta,alfa,temp(yDim,xDim)
integer, INTENT(INOUT):: iter,cont,cont_2,cont_3,cont_4
integer, INTENT(IN):: image(yDim,xDim),tStep
double precision ::d_loc_t,Cs,
     snul,snur,rnul,rnur,avnl,avnr,tempe,cont5,rnul_extra,&
&snul_extra, avnl_extra
integer:: x,y,i
character (LEN=100):: file_name,filename, path1, pathTMP
character (LEN=100):: itercar,Ncar,Recar,Ucar,Macar, Racar
Cs = (1.0d0/sqrt(3.0d0))
d_loc_t = 0.0d0
cont_4 = 0
do y = 1 , yDim
do x = 1 , xDim
if ((image(y,x) /= wall_1).or.(image(y,x) /= wall_2).or.(image(y,x) /=
     wall_3).or.(image(y,x) /= wall_4)) then
do i = 0 , 8
d_loc_t = d_loc_t + f(y,x,i)
end do
cont_4 = cont_4 +1
end if
end do
end do
if (d_loc_t >= yDim*xDim*2.0d0*rho(yDim,xDim)) then
write(*,*) 'stop'
STOP
end if
! ========================================================
! <<Data generation PARAVIEW ".vtk">>
! ========================================================
!if ((mod(tStep,iterper)==0) .and. (tStep >= tMax-40*iterper)) then
!if (mod(tStep,iterper)==0) then
if (mod(tStep,1000000)==0) then
iter = iter+1
write(file_name,'(a4,i5.4,a4)') 'data',iter,'.vtk'
write(path1,'(a43)') '/home/lattice/Documents/VTKs/data_termico/N'
write(itercar,'(i3)')iter
if (xDim >= 100) then
write(Ncar,'(i3.2)') xDim
else
write(Ncar,'(i2.2)') xDim
end if
write(Racar,'(i5.3)') int(Ra)
file_name = trim(file_name)
OPEN(unit=15,file=trim(path1)//trim(Ncar)//''//'/Ra'//trim(Racar)//
'/'//file_name,STATUS='unknown')
write(15,'(a26)')'# vtk DataFile Version 3.0'
write(15,'(a5,i5,a7,i6.1)')'iter ',iter,' total ',tStep
write(15,’(a5)’)’ASCII’
write(15,’(a23)’)’DATASET STRUCTURED_GRID’
write(15,’(a10,2i6.2,a2)’)’DIMENSIONS’,xDim,yDim, ’ 1’
write(15,’(a7,i5,a6)’)’POINTS ’,xDim*yDim,’ float’
do y= 0 , (yDim-1)
do x= 0 , (xDim-1)
write(15,’(i6.1,i6.1,a2)’)x,y,’ 0’
end do
end do
write(15,*)
write(15,’(a11,i8)’)’POINT_DATA ’,xDim*yDim
write(15,’(a24)’)’SCALARS density FLOAT 1’
write(15,’(a20)’)’LOOKUP_TABLE default’
do y = 1, yDim
do x = 1, xDim
write(15,’(F31.16)’)rho(y,x)
end do
end do
!Dominio
write(15,’(a23)’)’SCALARS domain FLOAT 1’
write(15,’(a20)’)’LOOKUP_TABLE default’
do y=1, yDim
do x=1, xDim
write(15,’(i2.1)’) image(y,x)
end do
end do
write(15,*)
!Velocidades
write(15,’(a23)’)’VECTORS velocity float’
do y= 1 , yDim
do x= 1 , xDim
write(15,’(F23.6,F23.6,a2)’)u(y,x,0),u(y,x,1),’ 0’
end do
end do
write(15,’(a27)’)’SCALARS Temperature FLOAT 1’
write(15,’(a20)’)’LOOKUP_TABLE default’
do y = 1, yDim
do x = 1, xDim
 write(15,’(F30.15)’)temp(y,x)
end do
end do
close(15)
! ========================================================
! <<Umax  >> Horizontal mid plane
! ========================================================
if (tStep == tMax) then
!if (mod(tStep,10000)==0) then
write(file_name,’(a5,i5.4,a4)’) ’U_max’,iter,’.txt’
write(path1,’(a43)’) ’/home/lattice/Documents/VTKs/data_termico/N’
if (xDim >= 100) then
write(Ncar,’(i3.2)’) xDim
else
write(Ncar,’(i2.2)’) xDim
end if
```

```
write(Racar,’(i5.3)’) int(Ra)
file_name = trim(file_name)
OPEN(unit=16,file=trim(path1)//trim(Ncar)//’’//’/Ra’//trim(Racar)//
’/Parametros’//’/’//file_name,STATUS=’unknown’)
write(16,’(a5,i5,a7,i6.1)’)’iter ’,iter,’ total ’,tStep
do y = 1, yDim
x = xDim/2
write(16,’(i8,F12.7,F12.7)’)  y,  u(y,x,0)/ (alfa/xDim)
end do
close(16)
! ========================================================
! <<Vmax  >> Vertical mid plane
! ========================================================
write(file_name,’(a5,i5.4,a4)’) ’V_max’,iter,’.txt’
write(path1,’(a43)’) ’/home/lattice/Documents/VTKs/data_termico/N’
if (xDim >= 100) then
write(Ncar,’(i3.2)’) xDim
else
write(Ncar,’(i2.2)’) xDim
end if
write(Racar,’(i5.3)’) int(Ra)
file_name = trim(file_name)
OPEN(unit=17,file=trim(path1)//trim(Ncar)//’’//’/Ra’//trim(Racar)//
’/Parametros’//’/’//file_name,STATUS=’unknown’)
write(17,’(a5,i5,a7,i6.1)’)’iter ’,iter,’ total ’,tStep
do x = 1 ,xDim
y = yDim/2
write(17,’(i8,F12.7,F12.7)’)  x, u(y,x,1) /(alfa/yDim)
end do
close(17)
write(file_name,’(a2,i5.4,a4)’) ’Nu’,iter,’.txt’
write(path1,’(a43)’) ’/home/lattice/Documents/VTKs/data_termico/N’
if (xDim >= 100) then
write(Ncar,’(i3.2)’) xDim
else
write(Ncar,’(i2.2)’) xDim
end if
write(Racar,’(i5.3)’) int(Ra)
file_name = trim(file_name)
OPEN(unit=18,file=trim(path1)//trim(Ncar)//’’//’/Ra’//trim(Racar)//
’/Parametros’//’/’//file_name,STATUS=’unknown’)
write(18,’(a5,i5,a7,i6.1)’)’iter ’,iter,’ total ’,tStep
snul= 0.0
snur= 0.0
!******local Nusselt*****************************************
do y = 1,yDim
rnul = (temp(y,1)-temp(y,2))*(xDim)
rnul_extra = (temp(y,2)-temp(y,3))*(xDim)
rnur = (temp(y,xDim-1)-temp(y,xDim))*(xDim)
snul= snul + rnul
snul_extra= snul + rnul_extra
snur= snur + rnur
end do
! ========================================================
! <<Average Nusselt>>
! ==========================================================
avnl = snul/(xDim)
avnl_extra = snul_extra/(xDim)
avnr = snur/(xDim)
write(18,’(a5,a8,a12,a10)’) ’Ra ’,’ Nu_hot’,’ Nu_cool’,’  Nu_prm’
write(18,’(i5,a2,F8.4,a2,F8.4,F8.4)’) ra,’  ’, snul,’  ’, snur,’  ’,
     avnl,  avnl_extra
close(18)
! ==========================================================
! <<Temperature mid plane  >>
! ==========================================================
write(file_name,’(a11,i5.4,a4)’) ’Temp_center’,iter,’.txt’
OPEN(unit=21,file=trim(path1)//trim(Ncar)//’’//’/Ra’//trim(Racar)//
’/Parametros’//’/’//file_name,STATUS=’unknown’)
write(21,’(a5,i5,a7,i6.1)’)’iter ’,iter,’ total ’,tStep
x = xDim/2
y = yDim/2
write(21,’(i8,F12.7)’)  x,  temp(y,x)
close(21)
end if

write(*,*)’=================================’
write(*,*)’Parameters’
write(*,*)’=================================’
!write(*,*) ’Richardson number (Ri)= ’,Ri
!write(*,*)’Reynolds numbers (Re) = ’,Re
write(*,*) ’delta temp (T_hot-T_ref)  = ’,th-tc
write(*,*) ’Rayleigh number (Ra)  = ’,Ra
write(*,*) ’Prandtl number (Pr)= ’,Pr
!write(*,*)  ’caract velocity (Umax) = ’,Umax
write(*,*)’================================= ’
write(*,*)’Lattice’
write(*,*)’================================= ’
write(*,*) ’omega = ’,omega
write(*,*) ’t = ’,(3.0d0*nu+0.5d0)
write(*,*) ’viscosity (nu)= ’,nu
write(*,*)’================================= ’
write(*,*)’  Lattice termico  ’
write(*,*)’================================= ’
write(*,*) ’omega termico = ’,omega_t
write(*,*) ’t_temp= ’,(3.0d0*alfa+0.5d0)
write(*,*) ’Thermal diffusivity (alfa)= ’,alfa
write(*,*) ’gravity*Thermal exp coef  = ’,gbeta
write(*,*)’================================= ’
write(*,*)’Densidad total= ’,d_loc_t
!write(*,*)’Mach number inicial (Ma)  = ’, Umax/Cs
!write(*,*)’nusselt local = ’,snul
write(*,*)’================================= ’

end if
! ==========================================================
! <<Temperature vs Tstep  >>
! ==========================================================
if (mod(tStep,iterper)==0) then
write(file_name,’(a17)’) ’Temp_vs_Tstep.txt’
write(path1,’(a43)’) ’/home/lattice/Documents/VTKs/data_termico/N’
if (xDim >= 100) then
write(Ncar,’(i3.2)’) xDim
else
write(Ncar,’(i2.2)’) xDim
end if
write(Racar,’(i5.3)’) int(Ra)
file_name = trim(file_name)
OPEN(unit=20,file=trim(path1)//trim(Ncar)//’’//’/Ra’//
trim(Racar)//’/Parametros’//’/’//file_name,&
&STATUS=’unknown’,  position = ’append’)
write(20,’(i8,F12.7)’)  tStep ,  temp(yDim/2,xDim/2)
close(20)
end if
! ========================================================
! <<U vs Tstep  >>
! ========================================================
if (mod(tStep,iterper)==0) then
write(file_name,’(a14)’) ’U_vs_Tstep.txt’
write(path1,’(a43)’) ’/home/lattice/Documents/VTKs/data_termico/N’
if (xDim >= 100) then
write(Ncar,’(i3.2)’) xDim
else
write(Ncar,’(i2.2)’) xDim
end if
write(Racar,’(i5.3)’) int(Ra)
file_name = trim(file_name)
OPEN(unit=21,file=trim(path1)//trim(Ncar)//’’//’/Ra’//
trim(Racar)//’/Parametros’//’/’//file_name,&
&STATUS=’unknown’,  position = ’append’)
x = xDim/2
y = yDim/2
write(21,’(i8,F12.7,F12.7)’)  tStep,  u(y,x,0)
close(21)
end if

END SUBROUTINE write_data
```

Appendix F

Mixed convection in an open cavity

LBM code - Flow in a channel, Fortran 90.
ParaView software for post-processing data (.VTK format file)

```
MODULE cellConst
integer, parameter:: fluid = 0, wall = 1, inlet = 10 , outlet = 11
integer, parameter:: wall_a = 2,  wall_hot = 3,  inlet_temp = 4 , outlet_temp =
     5, wall_cold = 6
END MODULE cellConst
MODULE D2Q9Const
! D2Q9 Weights
double precision,parameter:: t(0:8) =
     (/4.0d0/9.0d0,1.0d0/9.0d0,1.0d0/9.0d0,1.0d0/9.0d0,1.0d0/9.0d0&
&,1.0d0/36.0d0,1.0d0/36.0d0,1.0d0/36.0d0,1.0d0/36.0d0/)
!D2Q9 Directions
integer:: v(0:8,0:1)
integer:: opposite(0:8) = (/0,3,4,1,2,7,8,5,6/)
real:: Ma(0:8,0:8)
real:: D(0:8,0:8)
real:: Minvert(0:8,0:8)
real:: S(0:8,0:8)
real::SMinvert(0:8,0:8)
END MODULE D2Q9Const
MODULE simParam
integer, parameter:: yDim = 202
integer, parameter:: yDim_d = yDim/2
integer, parameter:: xDim = yDim * 2.5
integer, parameter:: iterper = 100
integer, parameter:: tMax = 10000000
double precision, parameter:: uMax = 0.02d0
double precision, parameter:: Re = 500
real, parameter:: Ri = 1.00
```

```
double precision, parameter:: tc = 0.0d0
double precision, parameter:: th = 1.0d0
double precision, parameter:: Pr = 0.71d0
END MODULE simParam
!=================================================
!The main program
!=================================================
PROGRAM unsteady
USE omp_lib
USE simParam, ONLY: xDim, yDim, tMax,iterper
implicit none
double precision:: omega, time1, time2, timeTot, Cs,nu, alfa, omega_t, gbeta
double precision, dimension(:,:,:), allocatable:: f, fEq, u, meq, m, gEq, g
double precision, dimension(:,:), allocatable:: rho, uSqr, mom,temp
integer, dimension(:,:), allocatable:: image,imaget
integer:: tStep, iter, cont, cont_2, cont_3, cont_4
allocate(f(yDim,xDim,0:8))
allocate(fEq(yDim,xDim,0:8))
allocate(u(yDim,xDim,0:1))
allocate(uSqr(yDim,xDim))
allocate(rho(yDim,xDim))
allocate(image(yDim,xDim))
allocate(m(yDim,xDim,0:8))
allocate(meq(yDim,xDim,0:8))
allocate(g(yDim,xDim,0:8))
allocate(gEq(yDim,xDim,0:8))
allocate(imaget(yDim,xDim))
allocate(temp(yDim,xDim))
CALL computeOmega(omega,nu,Cs,alfa,gbeta,omega_t)
CALL constructImages(image,imaget,omega)
!'______________LOAD_ROUTINE______________________'
! CALL read_data(f,g,tStep,iter,u,rho,temp)
CALL initMacro(rho,u,uSqr,image,iter,cont,cont_2,cont_3,cont_4,tStep,temp,imaget)!
CALL computeFeq(fEq,rho,u,uSqr,gEq,temp)
CALL computemeq(mEq,rho,u,uSqr)
f = fEq
g = gEq
timeTot = 0.0d0
do tStep = tStep, tMax , 1
CALL CPU_TIME(time1)
CALL inletOutlet(f,rho,u,image,uSqr,g,temp,imaget)
CALL computeMacros(f,rho,u,uSqr,image,imaget,temp,g,gbeta)
CALL computeFeq(fEq,rho,u,uSqr,gEq,temp)
CALL computemeq(mEq,rho,u,uSqr)
CALL
     collide(f,fEq,omega,image,u,rho,meq,uSqr,m,nu,g,gEq,temp,imaget,omega_t,gbeta)
CALL stream(f,g)
CALL boundaries(f,image,g,imaget)
CALL write_data(u,rho,tStep,f,iter,image,omega,nu,omega_t,temp,alfa,gbeta,imaget)
!'______________SAVE_ROUTINE______________________'
CALL save_data(f,g,tStep,iter,u,rho,temp)
CALL CPU_TIME(time2)
timeTot = timeTot + (time2-time1)
end do
```

```
write(*,*) dble(tMax) * (dble(yDim * xDim)) / timeTot ,’cells per second’
write(*,*) ’total time:’, timeTot, ’ seconds’
deallocate(f)
deallocate(fEq)
deallocate(u)
deallocate(uSqr)
deallocate(rho)
deallocate(image)
deallocate(m)
deallocate(meq)
deallocate(g)
deallocate(gEq)
deallocate(temp)
deallocate(imaget)
END PROGRAM unsteady
SUBROUTINE computeOmega(omega,nu,Cs,alfa,gbeta,omega_t)
USE simParam, ONLY: xDim,yDim,Re,Umax,yDim_d,obstR,Ri,Pr,tc,th
implicit none
double precision, INTENT(INOUT):: omega, nu,omega_t, alfa,gbeta
double precision :: Cs,Gr
Cs = (1.0d0/sqrt(3.0d0))
nu=  (uMax * (dble(yDim_d-1))) / Re
omega = 1.0d0 / (3.0d0*nu+0.5d0)
alfa = nu / Pr
omega_t = 1.0d0 / (3.0d0*alfa+0.5d0)
gbeta = (uMax*uMax*Ri)/ (dble(yDim_d-1))
END SUBROUTINE computeOmega
!=========================================================
!Construct an array the defines the flow geometry
!=========================================================
SUBROUTINE constructImages(image,imaget,omega)
USE cellConst
USE simParam
USE D2Q9Const
implicit none
integer, INTENT(INOUT):: image(yDim,xDim),imaget(yDim,xDim)
double precision, INTENT(INOUT):: omega
integer:: x,y,i,j
double precision:: a
v(0:8,0) = (/0,1,0,-1,0,1,-1,-1,1/)
v(0:8,1) = (/0,0,1,0,-1,1,1,-1,-1/)
image = fluid
image(yDim/2:yDim,1)= inlet
image(yDim/2:yDim,xDim)= outlet
image(1:yDim/2,1:xDim/5)= wall
image(1:yDim/2,2*xDim/5:xDim)= wall
image(1,:)  = wall
image(yDim,:)= wall

imaget(yDim/2:yDim,1)= inlet_temp
imaget(yDim/2:yDim,xDim)= outlet_temp
imaget(yDim,:)= wall_cold
imaget(1:yDim/2,1:xDim/5)= wall_a
imaget(1:yDim/2,2*xDim/5:xDim)= wall_a
imaget(1,:)= wall_a
imaget(2:yDim/2-1,xDim/5)= wall_hot
!=========================================================================
!MATRIX M
!=========================================================================
Ma(0:8,0) = (/ 1, 1, 1, 1, 1,1, 1, 1, 1/)
Ma(0:8,1) = (/-4,-1,-1,-1,-1,2, 2, 2, 2/)
Ma(0:8,2) = (/ 4,-2,-2,-2,-2,1, 1, 1, 1/)
Ma(0:8,3) = (/ 0, 1, 0,-1, 0,1,-1,-1, 1/)
Ma(0:8,4) = (/ 0,-2, 0, 2, 0,1,-1,-1, 1/)
Ma(0:8,5) = (/ 0, 0, 1, 0,-1,1, 1,-1,-1/)
Ma(0:8,6) = (/ 0, 0,-2, 0, 2,1, 1,-1,-1/)
Ma(0:8,7) = (/ 0, 1,-1, 1,-1,0, 0, 0, 0/)
Ma(0:8,8) = (/ 0, 0, 0, 0, 0,1,-1, 1,-1/)
a = 1.0d0/36.0d0
!=========================================================================
!MATRIX M^-1
!=========================================================================
Minvert(0:8,0)=(/4.d0*a, -4.d0*a,  4.d0*a,  0.d0   ,  0.d0   ,  0.d0   ,  0.d0   ,
     0.d0   ,  0.d0/)
Minvert(0:8,1)=(/4.d0*a, -1.d0*a, -2.d0*a,  6.d0*a, -6.d0*a,  0.d0   ,  0.d0   ,
     9.d0*a,  0.d0/)
Minvert(0:8,2)=(/4.d0*a, -1.d0*a, -2.d0*a,  0.d0   ,  0.d0   ,  6.d0*a, -6.d0*a,
     -9.d0*a,  0.d0/)
Minvert(0:8,3)=(/4.d0*a, -1.d0*a, -2.d0*a, -6.d0*a,  6.d0*a,  0.d0   ,  0.d0   ,
     9.d0*a,  0.d0/)
Minvert(0:8,4)=(/4.d0*a, -1.d0*a, -2.d0*a,  0.d0*a,  0.d0*a, -6.d0*a,  6.d0*a,
     -9.d0*a,  0.d0/)
Minvert(0:8,5)=(/4.d0*a,  2.d0*a,  1.d0*a,  6.d0*a,  3.d0*a,  6.d0*a,  3.d0*a,
     0.d0*a,  9.d0*a/)
Minvert(0:8,6)=(/4.d0*a,  2.d0*a,  1.d0*a, -6.d0*a, -3.d0*a,  6.d0*a,  3.d0*a,
     0.d0*a, -9.d0*a/)
Minvert(0:8,7)=(/4.d0*a,  2.d0*a,  1.d0*a, -6.d0*a, -3.d0*a, -6.d0*a, -3.d0*a,
     0.d0*a,  9.d0*a/)
Minvert(0:8,8)=(/4.d0*a,  2.d0*a,  1.d0*a,  6.d0*a,  3.d0*a, -6.d0*a, -3.d0*a,
     0.d0*a, -9.d0*a/)
!=========================================================================
!MATRIX S
!=========================================================================
!S(:) =(/1.0d0,1.40d0,1.40d0,1.0d0,1.20d0,1.0d0,1.20d0,tau,tau/)
S(0:8,0) = (/1.d0, 0.d0 , 0.d0 , 0.d0 , 0.d0 , 0.d0, 0.d0 , 0.d0, 0.d0/)
S(0:8,1) = (/0.d0, 1.4d0, 0.d0 , 0.d0 , 0.d0 , 0.d0, 0.d0 , 0.d0, 0.d0/)
S(0:8,2) = (/0.d0, 0.d0 , 1.4d0, 0.d0 , 0.d0 , 0.d0, 0.d0 , 0.d0, 0.d0/)
S(0:8,3) = (/0.d0, 0.d0 , 0.d0 , 1.d0 , 0.d0 , 0.d0, 0.d0 , 0.d0, 0.d0/)
S(0:8,4) = (/0.d0, 0.d0 , 0.d0 , 0.d0 , 1.2d0, 0.d0, 0.d0 , 0.d0, 0.d0/)
S(0:8,5) = (/0.d0, 0.d0 , 0.d0 , 0.d0 , 0.d0 , 1.d0, 0.d0 , 0.d0, 0.d0/)
S(0:8,6) = (/0.d0, 0.d0 , 0.d0 , 0.d0 , 0.d0 , 0.d0, 1.2d0, 0.d0, 0.d0/)
S(0:8,7) = (/0.d0, 0.d0 , 0.d0 , 0.d0 , 0.d0 , 0.d0, 0.d0 , omega,0.d0/)
S(0:8,8) = (/0.d0, 0.d0 , 0.d0 , 0.d0 , 0.d0 , 0.d0, 0.d0 , 0.d0, omega /)
END SUBROUTINE constructImages
!=========================================================================
SUBROUTINE
     initMacro(rho,u,uSqr,image,iter,cont,cont_2,cont_3,cont_4,tStep,temp,imaget)
USE simParam, ONLY: xDim, yDim, th,tc,uMax
USE cellConst
implicit none
double precision, INTENT(INOUT):: rho(yDim,xDim), u(yDim,xDim,0:1),
     uSqr(yDim,xDim),temp(yDim,xDim)
integer, INTENT(INOUT):: iter,cont,cont_2,cont_3,cont_4,tStep
integer, INTENT(IN):: image(yDim,xDim),imaget(yDim,xDim)
double precision:: uProf
integer:: y,x,i
iter = 0 ; tStep = 1
cont = 0 ; cont_2 = 0 ; cont_3 = 0 ; cont_4 = 0
do x = 1, xDim
do y = 1, yDim
if ((image(y,x) == wall).or.(x > xDim/5).and.(x < 2*xDim/5).and.(y < yDim/2)) then
u(y,x,0) = 0.0d0
u(y,x,1) = 0.0d0
else
u(y,x,0) = uMax
u(y,x,1) = 0.0d0
end if
if (imaget(y,x) == wall_hot) then
temp(y,x) = th
else
temp(y,x) = tc
end if
end do
end do
rho  = 1.0d0
uSqr = u(:,:,0) * u(:,:,0) + u(:,:,1) * u(:,:,1)
END SUBROUTINE initMacro
!===================================================================
SUBROUTINE computeFeq(fEq,rho,u,uSqr,gEq,temp)
USE D2Q9COnst, ONLY: t, v
USE simParam, ONLY: xDim, yDim
implicit none
double precision, INTENT(IN):: rho(yDim,xDim), uSqr(yDim,xDim),
     u(yDim,xDim,0:1),temp(yDim,xDim)
double precision, INTENT(INOUT):: fEq(yDim,xDim,0:8),gEq(yDim,xDim,0:8)
integer:: i, x, y
double precision:: uxy
do x = 1, xDim
do y = 1, yDim
do i = 0, 8
uxy = u(y,x,0) * v(i,0) + u(y,x,1) * v(i,1)
fEq(y,x,i) = t(i) * rho(y,x) * (1.0d0 + 3.0d0 * uxy + 4.5d0 * uxy * uxy - 1.5d0 *
     uSqr(y,x))
gEq(y,x,i) = temp(y,x) * t(i) * (1.0d0 + 3.0d0 * uxy)
end do
end do
end do
END SUBROUTINE computeFeq
SUBROUTINE computemeq(mEq,rho,u,uSqr)
USE D2Q9COnst, ONLY: t, v
USE simParam, ONLY: xDim, yDim
USE omp_lib
```

```
implicit none
double precision, INTENT(IN):: rho(yDim,xDim), uSqr(yDim,xDim), u(yDim,xDim,0:1)
double precision, INTENT(INOUT):: mEq(yDim,xDim,0:8)
integer:: i, x, y
double precision:: uxy
do x = 1, xDim
do y = 1, yDim
meq(y,x,0) =rho(y,x)
meq(y,x,1) =-2.0d0*rho(y,x) + 3.0d0*rho(y,x)*((u(y,x,0)*u(y,x,0) +
     u(y,x,1)*u(y,x,1)))
meq(y,x,2) =rho(y,x) -  3.0d0 *rho(y,x)*(u(y,x,0)*u(y,x,0) + u(y,x,1)*u(y,x,1))
meq(y,x,3) =u(y,x,0)*rho(y,x)
meq(y,x,4) =   -u(y,x,0)*rho(y,x)
meq(y,x,5) =u(y,x,1)*rho(y,x)
meq(y,x,6) =   -u(y,x,1)*rho(y,x)
meq(y,x,7) =   (u(y,x,0)*u(y,x,0)) - (u(y,x,1)*u(y,x,1))*rho(y,x)
meq(y,x,8) =   (u(y,x,0)*u(y,x,1))*rho(y,x)
end do
end do
END SUBROUTINE computemeq
!=================================================================
SUBROUTINE computeMacros(f,rho,u,uSqr,image,imaget,temp,g,gbeta)
USE simParam, ONLY: xDIm, yDim, tc, th
USE cellConst
USE D2Q9COnst, ONLY: t, v
USE omp_lib
implicit none
double precision, INTENT(IN):: f(yDim,xDim,0:8),g(yDim,xDim,0:8),gbeta
double precision, INTENT(INOUT):: u(yDim,xDim,0:1),rho(yDim, xDim),uSqr(yDim,
     xDim),temp(yDim, xDim)
integer, INTENT(IN):: image(yDim,xDim),imaget(yDim,xDim)
integer:: x,y,i
double precision:: force
do x = 1, xDim
do y = 1, yDim
if (image(y,x) == wall) then
u(y,x,0) = 0.0d0
u(y,x,1) = 0.0d0
else
rho(y,x)  = f(y,x,0) + f(y,x,1) + f(y,x,2) + f(y,x,3) + f(y,x,4) + f(y,x,5) +
     f(y,x,6) + f(y,x,7) + f(y,x,8)
do i = 0, 8
force = 3.0d0 * t(i) * gbeta * (temp(y,x)) * v(i,1) * rho(y,x)
u(y,x,0)  = ((f(y,x,1) - f(y,x,3) + f(y,x,5) - f(y,x,6) - f(y,x,7) + f(y,x,8)) +
     (force/2.0d0))/ rho(y,x)
u(y,x,1)  = ((f(y,x,2) - f(y,x,4) + f(y,x,5) + f(y,x,6) - f(y,x,7) - f(y,x,8)) +
     (force/2.0d0)) / rho(y,x)
uSqr(y,x) = u(y,x,0) * u(y,x,0) + u(y,x,1) * u(y,x,1)
end do
end if
if (imaget(y,x) == wall_hot) then
temp(y,x) = th
else if (imaget(y,x) == wall_cold)then
temp(y,x) = tc
```

```
else
temp(y,x) =  g(y,x,0) + g(y,x,1) + g(y,x,2) + g(y,x,3) + g(y,x,4)+g(y,x,5) +
     g(y,x,6) + g(y,x,7) + g(y,x,8)
end if
end do
end do
END SUBROUTINE computeMacros
!===========================================================
SUBROUTINE boundaries(f,image,g,imaget)
USE D2Q9Const
USE cellConst
USE simParam, ONLY: xDim, yDim,th,tc
USE omp_lib
implicit none
integer, INTENT(IN):: image(yDim,xDim),imaget(yDim,xDim)
double precision, INTENT(INOUT):: f(yDim,xDim,0:8),g(yDim,xDim,0:8)
double precision:: fTmp(0:8),gTmp(0:8)
integer:: i, x, y
do x = 1, xDim
do y = 1, yDim
if (image(y,x) == wall) then
do i = 0, 8
fTmp(i) = f(y,x,opposite(i))
end do
do i = 0, 8
f(y,x,i) = fTmp(i)
end do
end if
if (imaget(y,x) == wall_hot) then
g(y,x,1) = th * (t(1) + t(3)) - g(y,x,3)
g(y,x,8) = th * (t(8) + t(6)) - g(y,x,6)
g(y,x,5) = th * (t(5) + t(7)) - g(y,x,7)
else if (imaget(y,x) == wall_cold) then
g(y,x,4) = -g(y,x,2)
g(y,x,8) = -g(y,x,6)
g(y,x,7) = -g(y,x,5)
else if (imaget(y,x) == wall_a) then
do i = 0, 8
gTmp(i) = g(y,x,opposite(i))
end do
do i = 0, 8
g(y,x,i) = gTmp(i)
end do
end if
end do
end do
END SUBROUTINE boundaries
!===============================================================
SUBROUTINE inletOutlet(f,rho,u,image,uSqr,g,temp,imaget)
USE cellConst, ONLY: inlet, outlet,inlet_temp, outlet_temp
USE simParam
USE D2Q9Const, ONLY: t, v
USE omp_lib
implicit none
```

```
double precision, INTENT(INOUT):: f(yDim,xDim,0:8), u(yDim,xDim,0:1),
     rho(yDim,xDim)
double precision, INTENT(INOUT):: g(yDim,xDim,0:8), temp(yDim,xDim)
double precision, INTENT(IN):: uSqr(yDim,xDim)
integer, INTENT(IN):: image(yDim,xDim),imaget(yDim,xDim)
double precision:: uProf, uxy(0:8)
integer:: x, y, i
do x = 1, xDim
do y = 1, yDim
if (image(y,x) == inlet) then !ZOU/He INTEL
u(y,x,0) = uMax
u(y,x,1) = 0.0d0
CALL inletZou(f(y,x,:),u(y,x,:),rho(y,x))
else if (image(y,x) == outlet) then  !OPEN BOUNDARY CONDITION
do i = 0, 8
!if ((i == 3) .or. (i == 7) .or. (i == 6)) then
uxy(i) = u(y,x-1,0) * v(i,0) + u(y,x-1,1) * v(i,1)
f(y,x,i) = t(i) * rho(y,x-1) * (1.0d0 + 3.0d0 * uxy(i) + 4.5d0 * uxy(i) * uxy(i)
     - 1.5d0 * uSqr(y,x-1))
!end if
end do
end if
if (imaget(y,x) == inlet_temp) then
temp(y,x) = 0.0d0
g(y,x,1) = - g(y,x,3)
g(y,x,5) = - g(y,x,7)
g(y,x,8) = - g(y,x,6)
else if (imaget(y,x) == outlet_temp) then
g(y,x,7) = 2 *  g(y,x-1,7) -  g(y,x-2,7)
g(y,x,6) = 2 *  g(y,x-1,6) -  g(y,x-2,6)
g(y,x,3) = 2 *  g(y,x-1,3) -  g(y,x-2,3)
end if
end do
end do
CONTAINS
!!!=========================================================
!!!!Zou/He boundary on inlet
!!!=========================================================
SUBROUTINE inletZou(f,u,rho)
implicit none
double precision, INTENT(INOUT):: f(0:8),rho
double precision, INTENT(IN):: u(0:1)
double precision:: fInt, fInt2
fInt   = f(0) + f(2) + f(4)
fInt2  = f(3) + f(6) + f(7)
rho= (fInt + 2.0d0 * fInt2) / (1.0d0 - u(0))
CALL zouWestWall(f,rho,u)
END SUBROUTINE inletZou
SUBROUTINE zouWestWall(f,rho,u)
implicit none
double precision, INTENT(INOUT):: f(0:8)
double precision, INTENT(IN):: rho, u(0:1)
double precision:: fDiff, rhoUx, rhoUy
fDiff = 0.5d0 * (f(2) - f(4))
```

```
rhoUx = rho * u(0) / 6.0d0
rhoUy = 0.5d0 * rho * u(1)
f(1) = f(3) + 4.0d0 * rhoUx
f(5) = f(7) - fDiff + rhoUx + rhoUy
f(8) = f(6) + fDiff + rhoUx - rhoUy
END SUBROUTINE zouWestWall
END SUBROUTINE inletOutlet
!===============================================================
SUBROUTINE stream(f,g)
USE simParam
USE cellConst
implicit none
double precision, INTENT(INOUT):: f(yDim,xDim,0:8),g(yDim,xDim,0:8)
f(:,2:xDim,1) = f(:,1:xDim-1,1)
f(2:yDim,:,2) = f(1:yDim-1,:,2)
f(:,1:xDim-1,3) = f(:,2:xDim,3)
f(1:yDim-1,:,4) = f(2:yDim,:,4)
f(2:yDim,2:xDim,5) = f(1:yDim-1,1:xDim-1,5)
f(2:yDim,1:xDim-1,6) = f(1:yDim-1,2:xDim,6)
f(1:yDim-1,1:xDim-1,7) = f(2:yDim,2:xDim,7)
f(1:yDim-1,2:xDim,8) = f(2:yDim,1:xDim-1,8)
g(:,2:xDim,1) = g(:,1:xDim-1,1)
g(2:yDim,:,2) = g(1:yDim-1,:,2)
g(:,1:xDim-1,3) = g(:,2:xDim,3)
g(1:yDim-1,:,4) = g(2:yDim,:,4)
g(2:yDim,2:xDim,5) = g(1:yDim-1,1:xDim-1,5)
g(2:yDim,1:xDim-1,6) = g(1:yDim-1,2:xDim,6)
g(1:yDim-1,1:xDim-1,7) = g(2:yDim,2:xDim,7)
g(1:yDim-1,2:xDim,8) = g(2:yDim,1:xDim-1,8)
END SUBROUTINE stream
!=======================================================
!LBGK collision step
!========================================================
SUBROUTINE
     collide(f,fEq,omega,image,u,rho,meq,uSqr,m,nu,g,gEq,temp,imaget,omega_t,gbeta)
USE simParam, ONLY: xDim, yDim
USE cellConst
USE D2Q9Const, ONLY: Ma, Minvert,S,SMinvert,D,v,t
USE omp_lib
implicit none
integer, INTENT(IN):: image(yDim,xDim),imaget(yDim,xDim)
double precision, INTENT(IN):: fEq(yDim,xDim,0:8),
     omega,u(yDim,xDim,0:1),uSqr(yDim,xDim),gEq(yDim,xDim,0:8)
double precision, INTENT(IN)::
     rho(yDim,xDim),meq(yDim,xDim,0:8),temp(yDim,xDim),omega_t,gbeta
double precision, INTENT(INOUT)::
     f(yDim,xDim,0:8),nu,m(yDim,xDim,0:8),g(yDim,xDim,0:8)
double precision:: mom ,sumb, a, sume, tau, s7,s8
integer:: x,y,i,j,k
double precision:: force
do x = 1, xDim
do y = 1, yDim
do i = 0,8
mom = 0.0d0
```

```
do k = 0,8
mom = mom +  Ma(k,i) * f(y,x,k)
end do
m(y,x,i) = mom
end do
end do
end do
do i = 0,8
do j = 0,8
SMinvert(j,i) = 0.0d0
do k = 0,8
SMinvert(j,i) = SMinvert(j,i) + ( Minvert(k,i) * S(j,k))
end do
end do
end do
do x = 1, xDim
do y = 1, yDim
do i = 0, 8
force = 3.0d0 * t(i) * gbeta * (temp(y,x)) * v(i,1) * rho(y,x)
sume = 0.0d0
do k = 0,8
sume = sume + SMinvert(k,i) * ( m(y,x,k) - meq(y,x,k))
end do
if (image(y,x) /= wall) then
f(y,x,i) =  (f(y,x,i) - sume ) + force
end if
end do
if (imaget(y,x) /= wall_a) then
do i = 0, 8
g(y,x,i) = (1.0d0 - omega_t) * g(y,x,i) + omega_t * geq(y,x,i)
end do
end if
end do
end do
END SUBROUTINE collide
!<<POSTPROCESSING>>
SUBROUTINE
     write_data(u,rho,tStep,f,iter,image,omega,nu,omega_t,temp,alfa,gbeta,imaget)
USE simParam
USE cellConst
USE D2Q9const,ONLY: v, opposite
USE omp_lib
implicit none
double precision, INTENT(IN)::
     u(yDim,xDim,0:1),f(yDim,xDim,0:8),rho(yDim,xDim),omega,omega_t,temp(yDim,xDim)
double precision, INTENT(IN):: gbeta,alfa
integer, INTENT(INOUT):: iter
integer, INTENT(IN):: image(yDim,xDim),tStep,imaget(yDim,xDim)
double precision, INTENT(IN):: nu
double precision :: d_loc_x,d_loc_y, d_loc_t,Cs,
     snul,snur,rnul,rnur,avnl,avnr,teta1,teta2,tempe,cont5,rnul_extra,&
&snul_extra, avnl_extra
integer:: x,y,i,cont_4
character (LEN=100):: file_name,filename, path1, pathTMP
character (LEN=100):: itercar,Ncar,Recar,Ucar,Macar, Racar, Ricar

!=========================================================
!<<PARAVIEW Generator".vtk">>
if (mod(tStep,1000)==0) then
iter = iter+1
write(file_name,’(a4,i5.4,a4)’) ’data’,iter,’.vtk’
write(path1,’(a42)’) ’/home/lattice/Documents/VTKs/T_2posicion/N’
write(itercar,’(i3)’)iter
write(Ncar,’(i3.2)’) xDim
write(Racar,’(i6.3)’) int(Ra)
write(Recar,’(i3.3)’) int(Re)
write(Ricar,’(F4.2)’) (Ri)
write(Ucar,’(F6.4)’) (uMax)
file_name = trim(file_name)OPEN(unit=15,file=trim(path1)//trim(Ncar)//’’//’/Re’//
trim(Recar)//’’//’/Ri’//trim(Ricar)//’/’//file_name,STATUS=’unknown’)
OPEN(unit=15,file=trim(path1)//trim(Ncar)//’’//’/Re’//trim(Recar)//&
&’’//’/Ri’//trim(Ricar)//’’//’/uMax’//trim(Ucar)//’/’//file_name,STATUS=’unknown’)
write(15,’(a26)’)’# vtk DataFile Version 3.0’
write(15,’(a5,i5,a7,i6.1)’)’iter ’,iter,’ total ’,tStep
write(15,’(a5)’)’ASCII’
write(15,’(a23)’)’DATASET STRUCTURED_GRID’
write(15,’(a10,2i6.2,a2)’)’DIMENSIONS’, xDim, yDim, ’ 1’
write(15,’(a7,i8,a6)’)’POINTS ’,xDim*yDim,’ float’
do y= 0 , (yDim-1)
do x= 0 , (xDim-1)
write(15,’(i6.1,i6.1,a2)’)x,y,’ 0’
end do
end do
write(15,*)
write(15,’(a11,i8)’)’POINT_DATA ’,xDim*yDim
!Dominio
write(15,’(a23)’)’SCALARS domain FLOAT 1’
write(15,’(a20)’)’LOOKUP_TABLE default’
do y=1, yDim
do x=1, xDim
write(15,’(i2.1)’) image(y,x)
end do
end do
!Velocidades
write(15,’(a23)’)’VECTORS velocity float’
do y= 1 , yDim
do x= 1 , xDim
write(15,’(F23.6,F23.6,a2)’)u(y,x,0),u(y,x,1),’ 0’
end do
end do
write(15,’(a27)’)’SCALARS Temperature FLOAT 1’
write(15,’(a20)’)’LOOKUP_TABLE default’
do y = 1, yDim
do x = 1, xDim
write(15,’(F30.15)’)temp(y,x)
end do
end do
write(15,’(a24)’)’SCALARS density FLOAT 1’
write(15,’(a20)’)’LOOKUP_TABLE default’
do y = 1, yDim
do x = 1, xDim
write(15,’(F30.15)’)rho(y,x)
end do
end do
close(15)
write(*,*)’=================================’
write(*,*)’Input parameters   ’
write(*,*)’=================================’
write(*,*) ’Richardson number (Ri)= ’,Ri
write(*,*)’Reynolds numbers (Re) = ’,Re
write(*,*) ’Rayleigh number (Ra)  = ’,Ra
write(*,*) ’Prandtl number (Pr)   = ’,Pr
write(*,*)  ’caract velocity (Umax)  = ’,Umax
write(*,*) ’omega = ’,omega
write(*,*) ’t = ’,(3.0d0*nu+0.5d0)
write(*,*) ’viscosity (nu)= ’,nu
write(*,*) ’omega termico = ’,omega_t
write(*,*) ’t_termico = ’,(3.0d0*alfa+0.5d0)
write(*,*) ’Thermal diffusivity (alfa)= ’,alfa
write(*,*) ’gravity*Thermal exp coef  = ’,gbeta
write(*,*)’================================= ’
write(*,*)’Densidad total= ’,d_loc_t
write(*,*)’================================= ’
end if
!======================================================
!<<U vs Tstep punto espec >>
!========================================================
if (mod(tStep,1)==0) then
write(file_name,’(a5)’) ’U.txt’
write(path1,’(a42)’) ’/home/lattice/Documents/VTKs/T_2posicion/N’
write(Ncar,’(i3.2)’) xDim
write(Recar,’(i3.3)’) int(Re)
write(Ricar,’(F4.2)’) (Ri)
write(Ucar,’(F6.4)’) (uMax)
file_name = trim(file_name)
OPEN(unit=20,file=trim(path1)//trim(Ncar)//’’//’/Re’//trim(Recar)//
’’//’/Ri’//trim(Ricar)//’’//&
&’/uMax’//trim(ucar)//’/Parametros’//’/’//file_name,STATUS=’unknown’,position
     =’append’)
y = 3*yDim/8
x = xDim/5+ 10
!WRITE(*,*) 3*yDim/8
write(20,’(i8,F18.7)’)  tStep ,  u(y,x,0)
close(20)
write(file_name,’(a6)’) ’Nu.txt’
write(path1,’(a42)’) ’/home/lattice/Documents/VTKs/T_2posicion/N’
write(itercar,’(i3)’)iter
write(Ncar,’(i3.2)’) xDim
write(Recar,’(i3.3)’) int(Re)
write(Ricar,’(F4.2)’) (Ri)
write(Ucar,’(F6.4)’) (uMax)
file_name = trim(file_name)
```

```
OPEN(unit=18,file=trim(path1)//trim(Ncar)//’’//’/Re’//trim(Recar)//
’’//’/Ri’//trim(Ricar)//’’//&
&’/uMax’//trim(ucar)//’/Parametros’//’/’//file_name,STATUS=’unknown’,position
     =’append’)
!write(18,’(a5,i5,a7,i6.1)’)’iter ’,iter,’ total ’,tStep
snul= 0.0
snur= 0.0
!!******local Nusselt*****************************************
do y = 2,yDim/2-1
rnul = (temp(y,xDim/5) - temp(y,xDim/5+1))*( float(yDim_d) )
snul= snul + rnul
end do
!!<<Nusselt Promedio  >>
avnl = snul/(ABS(2-yDim/2-1))
write(18,’(i8,F12.7)’) tStep, avnl
!write(18,’(F5.2,a3,F8.4,a3,F8.4)’) Ri,’   ’, snul,’   ’,avnl
close(18)
end if
END SUBROUTINE write_data
SUBROUTINE read_data(f,g,tStep,iter,u,rho,temp)
USE simParam
implicit none
double precision, INTENT(INOUT)::
     f(yDim,xDim,0:8),g(yDim,xDim,0:8),u(yDim,xDim,0:1)
double precision, INTENT(INOUT):: temp(yDim,xDim),rho(yDim,xDim)
integer, INTENT(INOUT):: tStep, iter
integer:: x,y,i
character(100):: cwd
call getcwd(cwd)
open(96,file=trim(cwd)//’/save/save_2.dat’)
write(*,*) ’reading file =save_2.dat’
!open(96,file=trim(cwd)//’/save/save_1.dat’)
!write(*,*) ’reading file = save_1.dat’
read(96,*) tStep, iter
read(96,*)
do y = 1,yDim
do x = 1,xDim
do i = 0,8
read(96,*) f(y,x,i), g(y,x,i),u(y,x,0),u(y,x,1),rho(y,x),temp(y,x)
end do
end do
end do
close(96)
END SUBROUTINE read_data
SUBROUTINE save_data(f,g,tStep,iter,u,rho,temp)
USE simParam
implicit none
double precision, INTENT(IN):: f(yDim,xDim,0:8),g(yDim,xDim,0:8),u(yDim,xDim,0:1)
double precision, INTENT(INOUT):: temp(yDim,xDim),rho(yDim,xDim)
integer, INTENT(IN):: tStep, iter
integer:: x,y,i
character(100):: cwd
call getcwd(cwd)
if (mod(tStep,10000)==0) then
```

```
open(69,file=trim(cwd)//’/save/save_2.dat’)
write(*,*) ’writing at tstep and file =’, tstep, ’save_2.dat’
else if  (mod(tStep,20000)==10000) then
open(69,file=trim(cwd)//’/save/save_1.dat’)
write(*,*) ’writing at tstep and file =’, tstep, ’save_1.dat’
end if
if (mod(tStep,10000)==0) then
write(69,*) tStep, iter
write(69,*)
do y = 1,yDim
do x = 1,xDim
do i = 0,8
write(69,*) f(y,x,i), g(y,x,i),u(y,x,0),u(y,x,1),rho(y,x),temp(y,x)
end do
end do
end do
close(69)
end if
END SUBROUTINE save_data
```

Bibliography

[1] U. Ghia, K. Ghia, and C. Shin, "High-Re solutions for incompressible flow using the Navier-Stokes equations and a multigrid method," *Journal of Computational Physics*, vol. 48, no. 3, pp. 387–411, 1982.

[2] G. De Vahl Davis, "Natural convection of air in a square cavity: A bench mark numerical solution," *International Journal for Numerical Methods in Fluids*, vol. 3, no. 3, pp. 249–264, 1983.

[3] G. Barrios, R. Rechtman, J. Rojas, and R. Tovar, "The lattice Boltzmann equation for natural convection in a two-dimensional cavity with a partially heated wall," 2005.

[4] A. Mohamad, M. El-Ganaoui, and R. Bennacer, "Lattice Boltzmann simulation of natural convection in an open ended cavity," *International Journal of Thermal Sciences*, vol. 48, pp. 1870–1875, Oct. 2009.

[5] G. Kefayati, S. Hosseinizadeh, M. Gorji, and H. Sajjadi, "Lattice Boltzmann simulation of natural convection in tall enclosures using water/SiO2 nanofluid," *International Communications in Heat and Mass Transfer*, vol. 38, pp. 798–805, July 2011.

[6] G. Kefayati, M. Gorji-Bandpy, H. Sajjadi, and D. Ganji, "Lattice Boltzmann simulation of MHD mixed convection in a lid-driven square cavity with linearly heated wall," *Scientia Iranica*, vol. 19, pp. 1053–1065, Aug. 2012.

[7] E. Holzbecher, "Numerical Solutions for the Lévêque Problem of Boundary Layer Mass or Heat Flux ," in *Excerpt from the Proceedings of the COMSOL Conference 2008 Hannover*, 2008.

[8] L. Li-shi, " the Lattice-Gas and Lattice Boltzmann Methods: Past, Present, and Future," *NASA Langley Research Center*, 2000.

[9] D. A. Wolf-Gladrow, *Lattice-gas cellular automata and lattice Boltzmann models: An Introduction.* No. 1725, Springer Science & Business Media, 2000.

[10] S. Chen and G. D. Doolen, "Lattice Botlzmann Method for fluid flows," *Annual Review of Fluid Mechanics*, vol. 30, pp. 329–364, Jan. 1998.

[11] B. Chopard, "Cellular automata modeling of physical systems," *Computational Complexity: Theory, Techniques, and Applications*, pp. 407–433, 2012.

[12] J. G. Zhou, *Lattice Boltzmann methods for shallow water flows*, vol. 4. Springer, 2004.

[13] U. Frisch, D. d'Humieres, B. Hasslacher, P. Lallemand, Y. Pomeau, J.-P. Rivet, *et al.*, "Lattice gas hydrodynamics in two and three dimensions," *Complex systems*, vol. 1, no. 4, pp. 649–707, 1987.

[14] G. R. McNamara and G. Zanetti, "Use of the Boltzmann equation to simulate lattice-gas automata," *Physical Review Letters*, vol. 61, no. 20, p. 2332, 1988.

[15] S. J. Almalowi and A. Oztekin, "Flow Simulations Using Two Dimensional Thermal Lattice Boltzmann Method," *Journal of Applied Mathematics*, vol. 2012, pp. 1–12, Jan. 2012.

[16] J. Wang, D. Wang, P. Lallemand, and L. S. Luo, "Lattice Boltzmann simulations of thermal convective flows in two dimensions," in *Computers and Mathematics with Applications*, vol. 65, pp. 262–286, 2013.

[17] M. Jafari, M. Farhadi, K. Sedighi, and E. Fattahi, "Numerical simulation of convection heat transfer in a lid-driven cavity with an open side," *World Academy of Science, Engineering and Technology*, vol. 59, pp. 435–439, 2011.

[18] K.-H. Lin, C.-C. Liao, S.-Y. Lien, and C.-A. Lin, "Thermal lattice Boltzmann simulations of natural convection with complex geometry," *Computers & Fluids*, vol. 69, pp. 35–44, Oct. 2012.

[19] T. Bergman, F. Incropera, D. DeWitt, and A. Lavine, *Fundamentals of Heat and Mass Transfer*. Wiley, 2011.

[20] J. Welty, C. Wicks, G. Rorrer, and R. Wilson, *Fundamentals of Momentum, Heat and Mass Transfer*. Wiley, 2007.

[21] J. Burgos, I. Cuesta, and C. Salueña, "Numerical study of laminar mixed convection in a square open cavity," *International Journal of Heat and Mass Transfer*, vol. 99, pp. 599–612, 2016.

[22] A. Mohamad and A. Kuzmin, "A critical evaluation of force term in lattice Boltzmann method, natural convection problem," *International Journal of Heat and Mass Transfer*, vol. 53, pp. 990–996, Feb. 2010.

[23] Q. Xiong, B. Li, J. Xu, X. Fang, X. Wang, L. Wang, X. He, and W. Ge, "Efficient parallel implementation of the lattice Boltzmann method on large clusters of graphic processing units," *Chinese Science Bulletin*, vol. 57, no. 7, pp. 707–715, 2012.

[24] A. A. Mohamad, *Lattice Boltzmann Method*. London: Springer London, 2011.

[25] G. R. McNamara and G. Zanetti, "Use of the Boltzmann equation to simulate lattice-gas automata," *Physical Review Letters*, vol. 61, no. 20, p. 2332, 1988.

[26] F. J. Higuera and J. Jiménez, "Boltzmann Approach to Lattice Gas Simulations," *EPL (Europhysics Letters)*, vol. 9, no. 7, p. 663, 1989.

[27] 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.*, vol. 94, pp. 511–525, May 1954.

[28] S. Succi, *The Lattice Boltzmann Equation: For Fluid Dynamics and Beyond*. Numerical Mathematics and Scientific Computation, Clarendon Press, 2001.

[29] Y. Qian, D. d'Humières, and P. Lallemand, "Lattice BGK models for Navier-Stokes equation," *EPL (Europhysics Letters)*, vol. 17, no. 6, p. 479, 1992.

[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*, vol. 56, pp. 6811–6817, Dec. 1997.

[31] Q. Zou and X. He, "On pressure and velocity boundary conditions for the lattice Boltzmann BGK model," *Physics of Fluids*, vol. 9, p. 1591, Nov. 1997.

[32] M. Hecht and J. Harting, "Implementation of on-site velocity boundary conditions for D3Q19 lattice Boltzmann simulations," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2010, no. 01, p. P01018, 2010.

[33] K. Suga, Y. Kuwata, K. Takashima, and R. Chikasue, "A D3Q27 multiple-relaxation-time lattice Boltzmann method for turbulent flows," *Computers & Mathematics with Applications*, vol. 69, no. 6, pp. 518–529, 2015.

[34] P. Lallemand and L.-S. Luo, "Theory of the lattice Boltzmann method: Dispersion, dissipation, isotropy, Galilean invariance, and stability," *Physical Review E*, vol. 61, no. 6, p. 6546, 2000.

[35] A. Mezrhab, M. A. Moussaoui, M. Jami, H. Naji, and M. Bouzidi, "Double MRT thermal lattice Boltzmann method for simulating convective flows," *Physics Letters A*, vol. 374, no. 34, pp. 3499–3507, 2010.

[36] R. Huang and H. Wu, "A modified multiple-relaxation-time lattice Boltzmann model for convection–diffusion equation," *Journal of Computational Physics*, vol. 274, pp. 50–63, 2014.

[37] D. d'Humières, "Multiple–relaxation–time lattice Boltzmann models in three dimensions," *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, vol. 360, no. 1792, pp. 437–451, 2002.

[38] S. Wolfram *et al.*, *Theory and applications of cellular automata*, vol. 1. World scientific Singapore, 1986.

[39] P. K. Kundu, I. M. Cohen, and D. R. Dowling, *Fluid mechanics.* Waltham, MA Academic Press, 5th ed ed., 2012.

[40] I. Currie, *Fundamental Mechanics of Fluids, Fourth Edition.* Civil and mechanical engineering, Taylor & Francis, 2012.

[41] F. Incropera, *Fundamentals of heat and mass transfer.* No. v. 1 in Fundamentals of Heat and Mass Transfer, John Wiley, 2007.

[42] Y. Peng, C. Shu, and Y. Chew, "Simplified thermal lattice Boltzmann model for incompressible thermal flows," *Physical Review E*, vol. 68, p. 026701, Aug. 2003.

[43] L.-S. Luo, "Lattice-Gas Automata and Lattice Boltzmann Equations for Two-Dimensional Hydrodynamics," *Thesis (PH.D.)-Georgia Institute of technology*, 1993.

[44] L.-S. Luo, "Unified Theory of Lattice Boltzmann Models for Nonideal Gases," *Physical Review Letters*, vol. 81, pp. 1618–1621, Aug. 1998.

[45] X. Shan and H. Chen, "Simulation of nonideal gases and liquid-gas phase transitions by the lattice Boltzmann equation," *Physical Review E*, vol. 49, pp. 2941–2948, Apr. 1994.

[46] Z. Guo, B. Shi, and C. Zheng, "A coupled lattice BGK model for the Boussinesq equations," *International Journal for Numerical Methods in Fluids*, vol. 39, no. 4, pp. 325–342, 2002.

[47] 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*, vol. 49, pp. 727–739, Feb. 2006.

[48] R. Bird, W. Stewart, and E. Lightfoot, *Transport Phenomena.* Wiley International edition, Wiley, 2007.

[49] Y. Chan and C. Tien, "Laminar natural convection in shallow open cavities," *Journal of Heat Transfer*, vol. 108, no. 2, pp. 305–309, 1986.

[50] F. Ampofo and T. G. Karayiannis, "Experimental benchmark data for turbulent natural convection in an air filled square cavity," *International Journal of Heat and Mass Transfer*, vol. 46, no. 19, pp. 3551–3572, 2003.

[51] E. Papanicolaou and Y. Jaluria, "Mixed convection from an isolated heat source in a rectangular enclosure," *Numer. Heat Transfer A*, vol. 4, no. 18, pp. 427–461, 1991.

[52] E. Papanicolaou and Y. Jaluria, "Transition to a periodic regime in mixed convection in a square cavity," *J. Fluid. Mech.*, no. 239, pp. 489–509, 1992.

[53] E. Papanicolaou and Y. Jaluria, “Mixed convection from a localized heat source in a cavity with conducting walls: A numerical study,” *Numer. Heat Transfer A*, vol. 4, no. 23, pp. 463–484, 1993.

[54] O. Manca, S. Nardini, K. Khanafer, and K. Vafai, “Effect of heated wall position on mixed convection in a channel with an open cavity,” *Numerical Heat Transfer, Part A: Applications*, vol. 43, no. 3, pp. 259–282, 2003.

[55] Y. Stiriba, “Analysis of the flow and heat transfer characteristics for assisting incompressible laminar flow past an open cavity,” *International Communications in Heat and Mass Transfer*, vol. 35, no. 8, pp. 901–907, 2008.

[56] Y. Stiriba, F. Grau, J. Ferré, and A. Vernet, “A numerical study of three-dimensional laminar mixed convection past an open cavity,” *International Journal of Heat and Mass Transfer*, vol. 53, pp. 4997–4808, Oct. 2010.

[57] Y. Çengel, *Transferencia de calor y masa: un enfoque práctico*. McGraw-Hill series in mechanical engineering, McGraw-Hill, 2007.