Rollins College
Rollins Scholarship Online

Honors Program Theses

Spring 2020

Computational analysis of woodwind instruments using the lattice Boltzmann method

Jack D. Gabriel
Rollins College, jgabriel@rollins.edu

Follow this and additional works at: https://scholarship.rollins.edu/honors

 Part of the Fluid Dynamics Commons

Recommended Citation

Gabriel, Jack D., "Computational analysis of woodwind instruments using the lattice Boltzmann method" (2020). *Honors Program Theses*. 128.
https://scholarship.rollins.edu/honors/128

This Open Access is brought to you for free and open access by Rollins Scholarship Online. It has been accepted for inclusion in Honors Program Theses by an authorized administrator of Rollins Scholarship Online. For more information, please contact rwalton@rollins.edu.

ROLLINS COLLEGE

HONORS THESIS

Computational analysis of woodwind instruments using the lattice Boltzmann method

---

Author:

Jack GABRIEL

Supervisor:

Dr. Whitney COYLE

A thesis submitted in fulfillment of the requirements

for the degree of Artium Baccalaureus Honoris

in the

Physics Department

May 8, 2020

Declaration of Authorship

I, Jack GABRIEL, declare that this thesis titled, "Computational analysis of woodwind instruments using the lattice Boltzmann method" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

_____

Date:

_____

Ozymandias

I met a traveller from an antique land
Who said: “Two vast and trunkless legs of stone
Stand in the desert . . . Near them, on the sand,
Half sunk, a shattered visage lies, whose frown,
And wrinkled lip, and sneer of cold command,
Tell that its sculptor well those passions read
Which yet survive, stamped on these lifeless things,
The hand that mocked them, and the heart that fed:
And on the pedestal these words appear:
‘My name is Ozymandius, king of kings:
Look on my works, ye Mighty, and despair!’
Nothing beside remains. Round the decay
Of that colossal wreck, boundless and bare
The lone and level sands stretch far away."

Percy Bysshe Shelley

ROLLINS COLLEGE

Abstract

Physics Department

Artium Baccalaureus Honoris

Computational analysis of woodwind instruments using the lattice Boltzmann method

by Jack GABRIEL

Through the use of the lattice Boltzmann method, a series of computational models were created to simulate air flow in woodwind instruments. Starting as a two-dimensional code in Matlab running on the CPU, the model went through a series of iterations before becoming a three-dimension code in Fortran that was accelerated through the use of GPU parallel computing. The accuracy and stability of the model are shown by comparison to various published benchmark tests. Thus far, the air flow in organ pipes for a two dimensional model was simulated showing oscillating flow by the labium as expected. This thesis offers the mathematical and computational background as well as a description of the implementation of the basic LBM for simulating flow in musical instruments. The method described here is meant as a first step to a code that is highly flexible and can be used to study many aspects of acoustics in musical instruments. Future applicability of the model includes observing flow at the exit of both square and round organ pipes in addition to modeling the reed-mouthpiece system of the clarinet.

Acknowledgements

To start, I would like to thank my adviser, Dr. Whitney Coyle for her guidance, advice and trust. Whether it was as a freshman walking into her office knowing nothing about the clarinet other than that Squidward plays it on "SpongeBob SquarePants" or for this thesis in proposing to create this computational model that has been the basis of PhD dissertations, she trusted that I would be able to be successful in at least part of what I set out to do. For that reason I am extremely indebted to her.

Additionally I would like the thank Dr. Thomas Moore for our interesting conversations on end corrections and flow out the end of pipes, and, along with Dr. Rochelle Elva, for serving on my thesis committee.

I am grateful for my many friends at Rollins who helped make Central Florida my home over these past four years. I would like to give a special thanks to my honors friends for providing a forum to vent while we all completed our respective theses and to my fellow physics majors who I see as an extension of my family and without whom, I would have struggled to find a reason to stay at Rollins.

Finally I would like to thank my parents for allowing me to attend Rollins and, along with my brother and sister, for humoring me as I tried to explain to them my research throughout the years.

Contents

List of Figures

List of Tables

Listings

List of Algorithms

Dedicated to my wife and kids that I apparently have…

Chapter 1

Introduction

Woodwind instruments including, among others, flutes and clarinets, have been around in close to their present form for the past couple of centuries. Most woodwind instruments were not built with the physics in mind, but rather with the intention of making the instrument sound 'better' to the user and audience. Over the last century, it has been the goal of acousticians to better understand the physics of the sound produced by these instruments, perhaps to optimize the manufacturing, or improve consumer experiences, or in many cases to merely better understand the complexities underlying the instruments.

Musical acousticians sought understanding first through experimental and theoretical approaches, however, the advent and rapid development of computers has opened up a new way to study the acoustics of woodwind instruments. Numerical simulations are particularly advantageous in cases where theoretical modeling or experimentation is either difficult or invasive as can occur, for example, when analyzing end corrections or fluid structure interaction in woodwind instruments. The goal of this thesis is to analyze air flow computationally, within a particular set of woodwind instruments. As the governing equations for flow in woodwind instruments are quite complex, exact analytical solutions are restricted in scope [1], [2]. This means that for this work, a numerical solution is required. The following sections will introduce the approach and method as well as offer context for the problem

at hand.

This research began nearly three years ago as an experimental investigation of air flow inside the clarinet mouthpiece. The work was difficult to validate with the current methods and the research group sought to add computational studies in order to have confidence in the work. The clarinet mouthpiece also offered an extra layer of difficulty in that there were moving parts, the reed on the mouthpiece (see Section 2.2.1). In order to build a solid computational foundation, the work will begin by introducing a simpler system, the flute, then offer suggestions for future work that could eventually lead to development of simulations of the clarinet mouthpiece – the end goal of this line of research in Dr. Coyle's lab. The geometry and specifics of the flute will be introduced in Section 2.3.

As an introduction to the computational techniques used in this thesis, first we must discuss the two main approaches when simulating the physical transport equations (mass and momentum for this work): continuum and discrete [3]. For the continuum approach, the domain is first discretized into a mesh which consists of volumes, grids or elements depending on the method used. This allows for the partial differential equations that are being solved, the Navier-Stokes equations, to be converted into a system of algebraic equations. These algebraic equations are then solved iteratively until convergence below some target threshold is reached. This process is performed at each time step until the simulation is complete. There are many examples of this approach being used in musical acoustics. Kühnelt used a finite element model as part of his work in simulating vortex shedding in the flue pipe [4]. Additionally, Giordano used a finite difference scheme to simulate the effects of different geometries of the flue exit in the flue pipe in both 2D and 3D [5], [6]. The main benefit to the contiuum approach is that the solution comes from directly solving the Navier-Stokes equations as opposed to an approximation. However, due to complexity involved in solving these

equations, an extraordinary amount of computation power is required. This means that either the time steps of the simulation are large, the simulation takes a long time to run or a supercomputer is used.

On the opposite extreme, instead of solving the governing equations over the entire domain, the interaction between individual particles can be considered. On the microscopic scale, the governing equation is Newton's second law, which is a simple ordinary differential equation. For the discrete approach, the velocity and position of each particle is identified and an inter-particle force function needs to be applied at each time step. This time step needs to be able to resolve the fastest motion in the domain, meaning that the time step needs to be on the order of femtoseconds, $\mathcal{O}(10^{-15})s$ [3]. With the computational power available today with modern computers, this means that the total simulation time needs to be less than $\mathcal{O}(10^{-9})s$, which makes this approach not currently feasible for musical acoustics where the simulation time is $\mathcal{O}(10^{-3})s$ or larger. In addition to the time step being small, the domain needs to be subsequently small, which additionally makes this approach not feasible for this work. While not feasible, there are benefits of this approach that need to be discussed. The main benefit of this approach is its simplicity. This comes from Newton's second law being computationally easy to solve, reducing the computational power required for each time step. The issue is then that there are too many time steps needed for even the fastest computers to make this approach feasible for this work.

This leads to the lattice Boltzmann method (LBM), which operates between the continuum and discrete approaches. Instead of considering the behavior of each individual particle, the idea of the LBM is to consider the behavior of a collection of particles using statistical mechanics [3]. The rationale for this comes from the fact that for fluid dynamics, the movement of each individual particle is not important; rather, the overall flow over the

entire domain is. The LBM is relatively new and has gained a lot of attraction over the last 30 years, especially in the realm of fluid dynamics [7]. The main advantage of the LBM is its simplicity while at the same time it is capable of resolving the Navier-Stokes equations, which will be discussed in Section 4.6. Additionally, due to the local nature of the LBM, it is inherently parallelizable as discussed in Section 4.4, which allows for simulations that require either long time scales or high spatial resolutions to be performed.

Originating from the lattice gas automata model, which acted as a simplified molecular dynamics model, the LBM has been used in multiple areas, but especially for simulating complex fluid systems and systems with complex boundaries [3], [7]. First performed in studying musical acoustics by Skordos in 1995 with his simulations of recorders and pipes, the LBM has been shown to work well in simulating the flow inside woodwind instruments [8]. More recently, there have been numerous simulations on more complicated instruments and larger scales [4], [9], [10]. Kühnelt used a high performance computing cluster to perform 3D simulations of the flow about the labium in the organ pipe and flute [4]. Moving from static to moving boundary simulations, da Silva performed simulations on the mouthpiece of a clarinet in order to study the fluid structure interaction between the reed and air flow in the mouthpiece [9]. Shi improved upon da Silva's model, parallelizing it to run on the graphics processing unit (GPU) and adding an acoustic resonator on the end of the mouthpiece [10].

For this work, an LBM code was written that allows for future improvements to be made. While Shi and da Silva used a 2D axisymmetric model and Kühnelt used CPU parallel computing, the current work presented here will offer the first 3D GPU LBM model with the purpose of studying woodwind instruments. In building up to this point, to the author's knowledge, this work also presents the first 2D GPU LBM simulations of a closed flue pipe. Far from an end product, the code presented in this work is meant

to be the first step in a code that can be rapidly manipulated and improved upon to study numerous aspects of the acoustics of woodwind instruments. The outline of the thesis is as follows:

Chapter 2 gives an introduction to the acoustics underlying woodwind instruments. This overview presents a framework for how the woodwind instruments are related, which helps in understanding why certain simulations were performed. The starting point for this work was studying the acoustics of the clarinet (a more complicated system). This chapter presents the series of approximations which are generally used to simplify the system.

Chapter 3 disucsses the underlying theory of the LBM. Starting from introductory kinematic theory, the all important Boltzmann equation is derived in terms of the one particle distribution function. Additionally, the two most common approximations for the collision operator are presented, and the discretized lattice Boltzmann equation is derived, which is the equation upon which the LBM is based.

Chapter 4 introduces the LBM itself as well as its implementation. This includes a discussion of different models that are typically used, from 1D to 3D models along with different boundary conditions. Finally, the procedure is included along with software implementation with regard to GPU computing.

Chapter 5 discusses the evolution of the code from a Matlab 2D code running on the CPU to a Fortran 3D code that utilizes GPU computing. Additionally, the results of the simulations are given for both the Fortran 2D and Fortran 3D code. These simulations show the validity of the LBM codes created for this thesis.

To end, Chapter 6 provides the conclusion along with many suggestions for future research and how the code can be further improved.

Chapter 2

Acoustics of Musical Instruments

2.1 Introduction

Preliminary work performed before the start of this thesis as a part of Rollins College's Student-Faculty Collaborative Scholarship Program involved studying the fluid-structure interaction of the clarinet as discussed in Section 2.2.1. While this work was ultimately unsuccessful, it was the starting point for which the idea of this thesis was created. As a member Dr. Coyle's research lab, the ultimate goal is the study the clarinet, therefore while simulations of the flow within the clarinet will not be presented in this thesis, it is for this reason that the acoustics of the clarinet is where this thesis begins.

2.2 Clarinet Acoustics

The clarinet is a member of the woodwind family of instruments of which other instruments include the flute and saxophone. These instruments are characterized by the way in which they produce sound. As opposed to brass instruments where the air stream passes through a player's vibrating lips into the resonator, woodwind instruments produce sound by having the air stream interact with a sharp edge, as is the case for the labium in a flute, or a reed, as is the case for the clarinet [11].

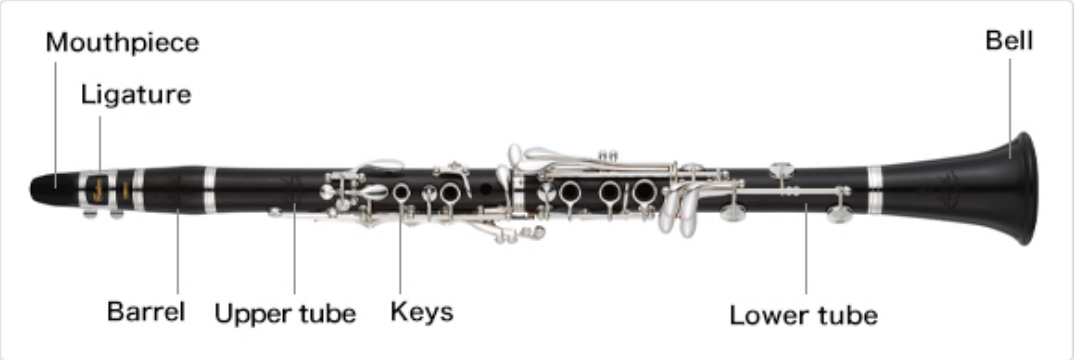


FIGURE 2.1: The clarinet with all of its components labeled. This can be modeled as a cylinder with one open end and the other end closed [12].

The clarinet is composed of five parts as seen in Fig. 2.1. From left to right, the ligature holds the reed onto the mouthpiece through which the musician blows. The barrel is responsible for connecting the mouthpiece to the upper tube which is where the musician would place their left hand. Along the length of the clarinet are keys, which are holes in the instrument that the musician opens or closes with their fingers to change the note. At the end of the instrument, connected to the lower tube is the bell, which helps radiate the sound of the clarinet.

Due to the complexity of the physical qualities of the clarinet, it is necessary to break the instrument into simpler parts, for both theoretical and computational purposes. Typically, the clarinet is divided into two pieces: the generator and the resonator. The reed-mouthpiece system, where air is input into the instrument, acts as the generator, and the remainder of the instrument then acts as the resonator.

2.2.1 Reed Mouthpiece System

The reed mouthpiece system, as seen in Fig. 2.2, is comprised of the reed and mouthpiece, which are held together by the ligature. By clamping the reed to the mouthpiece at only one end, the tapered end of the reed is free to

vibrate. The rest position of the reed sits a small distance above the edge of the mouthpiece. This is known as the lay.

In order to play the clarinet, the musician places their mouth around the mouthpiece and blows into the instrument. This input air pressure forces the reed to close against the mouthpiece, but, due to the reed material, the reed bends back to its initial position. This process repeats in an oscillatory manner similar to that of a spring, allowing for the reed to be modeled as a mass-spring system [13].

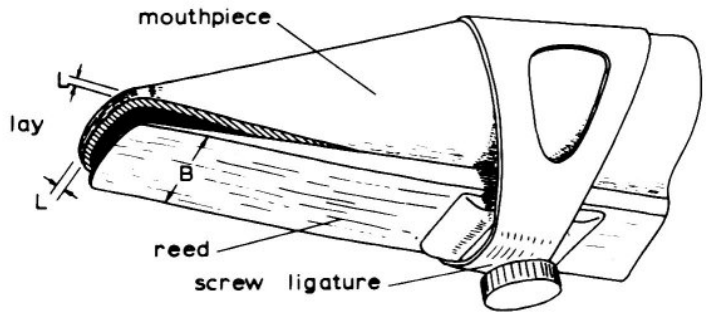


FIGURE 2.2: The clarinet reed mouthpiece system where the ligature holds the reed to the mouthpiece [14].

In addition to this initial source of flow from the musician, there is a second component of flow from the oscillations of the reed. This second component of flow is known as the reed induced flow. When the reed oscillates, it displaces air and this displaced air also interacts with the reed, altering its motion. This fluid-structure interaction plays a role in the reed induced flow, which in turn has a large effect on the playing frequency of the clarinet [9], [15]. These two components of flow can be seen in Fig. 2.3.

The reed induced flow, u_r, is described by,

$$u_r = -S_r \frac{dy}{dt}, \tag{2.1}$$

where $\frac{dy}{dt}$ is the velocity of the reed and S_r is the relevant surface area of the reed. Values for S_r are not very well understood, with values in literature

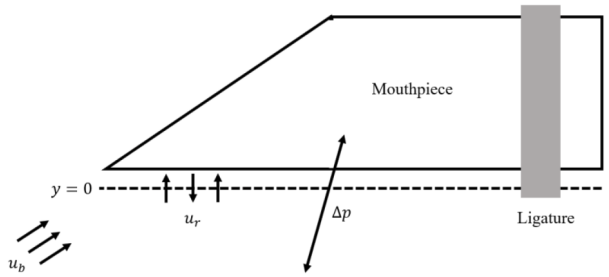


FIGURE 2.3: The clarinet reed mouthpiece system where the ligature holds the reed, dotted line, to the mouthpiece. The reed induced flow, u_r, the input flow from the muisician, u_b and the pressure difference between the inside of the mouth and mouthpiece, Δp, can be seen [16].

ranging from 60 - 200 mm^2 [17]. Due to the reed being inside the mouth of the musician while the clarinet is being played, it is difficult to measure this surface area experimentally.

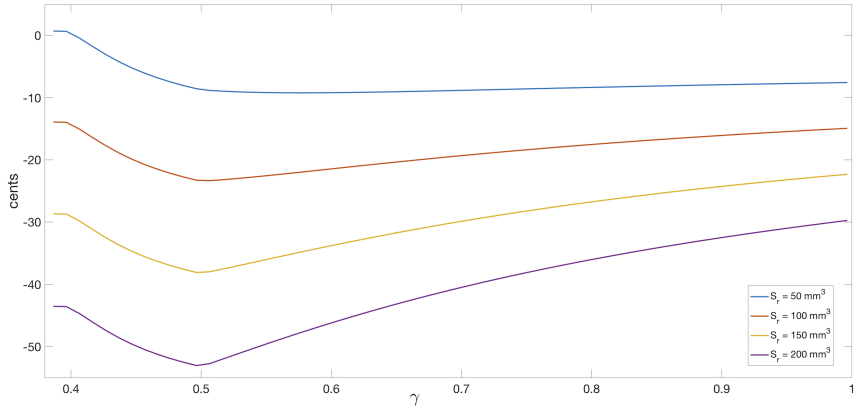


FIGURE 2.4: The results of the sensitivity analysis for the relevant surface area of the reed, S_r, where γ is a dimensionless blowing parameter and cents is a measure for describing ratios of frequencies. A difference of around 10 cents can be heard by the average musician, so these differing values of S_r would be audible [15].

Previous work at Rollins included performing a sensitivity analysis of S_r using our analytical model as seen in Fig. 2.4. The model works by taking in multiple parameters in addition to the resonance frequency of the clarinet in order to calculate the playing frequency of the clarinet. For example in Fig. 2.4, the 0 cents mark corresponds to the resonance frequency and each line

corresponds to the playing frequency, the frequency that is actually heard, for a particular S_r and blowing pressure. More information on the analytical model can be found in [15]. The average musician can detect a difference of around 10 cents, so the 50 cents difference between the smallest and largest values of S_r would be audible [15].

2.2.2 Resonator

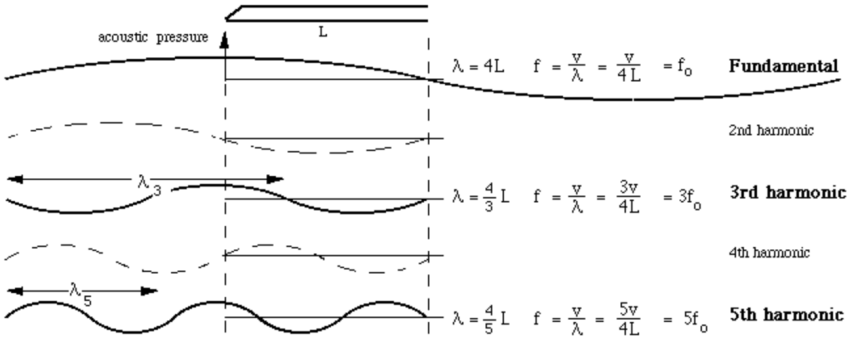


FIGURE 2.5: A represenationation of the acoustical pressure in a closed open tube for the first five resonance frequencies where λ is the wavelength, L is the length of the tube, f_0 is the fundamental and v is the velocity of the wave. Notice that only the odd multiples of the fundamental are possible [18].

After the reed-mouthpiece system, the rest of the clarinet is acoustically very similar to that of a closed open tube. The bell corresponds to the open end of the tube, as that is from where the sound is radiated and the mouthpiece corresponds to the closed end. While not physically closed, it is acoustically closed in that no air comes out, allowing for reflection at the boundary and the creation of standing waves. This restriction imposes boundary conditions allowing for the frequency of the tube to be determined. Specifically, two of these conditions are that the pressure at the closed end of the tube is an antinode and that the pressure at the open end of the tube is 0, a node. This can be seen in Fig. 2.5. These boundary conditions allow for only certain frequencies described by,

$$f_n = \frac{nc}{4L}, \tag{2.2}$$

for odd multiples of n where c is the speed of sound and L is the length of the tube. These frequencies are known as resonance frequencies. The keys along the length of the clarinet allow for the effective length of the clarinet to be changed by closing and opening the holes.

2.3 Flutes

While the end goal of the current work is to study the clarinet, the instrument does not lend itself to being an easy place to start. Being a reed woodwind instrument, the instrument has moving parts, the reed, that causes the air column in the instrument to vibrate and produce its sound. The fluid structure interaction as described in Section 2.2.1 is difficult to model, so in an attempt to build up to the clarinet, this study will start with the other type of woodwind instruments, flutes.

The flute family can be further divided into two subfamilies: non-fipple (also known as non-duct or open) and fipple (also known as duct or closed) flutes [11], [19]. The difference between these two subfamilies is in how the air flow is directed to the sharp edge. In non-fipple flutes, the musician directs the air flow against the edge whereas in fipple flutes, the airstream is directed to the sharp edge by a duct, which is known as a fipple.

An example of a non-fipple flute would be the transverse flute. To play the transverse flute, a musician blows across the top of the embouchure hole which acts as a sharp edge to split the airflow. This split airstream then acts upon the air in the hollow flute causing it to vibrate and produce sound. While still a woodwind instrument, the hollow flute is typically open at both ends so it can be modeled as an open open tube. This means that the non-fipple flute is not a perfect parallel to the clarinet.

One example of a fipple flute is a flue pipe, which can be seen in Fig. 2.6. To play the flue pipe, air is driven through the flue where it then interacts

with the labium which acts as a sharp edge to split the airflow. This causes for the air column in the pipe resonator to resonate hence producing sound. In effect, the labium acts as the reed does in the clarinet allowing for the flue pipe to behave as a simplified clarinet with no moving parts [20]. This comparison between the clarinet and flue pipe is not perfect because, similar to the transverse flute, the flue pipe is open by the labium and at the end of the pipe resonator. However, the end of the pipe resonator can be closed yielding a closed flue pipe which can be modeled as a closed open tube. It is for these reasons that the closed flue pipe is first analyzed in this study.

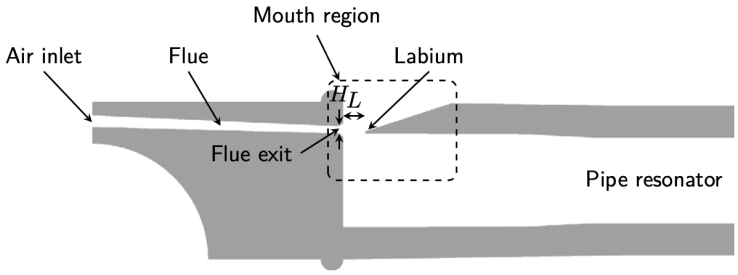


FIGURE 2.6: A diagram of the parts of a flue pipe where L is the distance from the flue exit to the labium and H is the height of the flue exit. The flue pipes studied for this thesis are closed flue pipes, so the end of the pipe resonator is closed (not pictured) [4].

2.4 Impedance

2.4.1 Acoustical Impedance

Before moving on to how these instruments will be modeled computationally, the concept of impedance will be discussed. The acoustical impedance, Z, is the complex ratio of acoustic pressure, P, to acoustic volume flow, U, so

$$Z = \frac{P}{U}. \tag{2.3}$$

In other words, spacial variations in pressure give rise to air flow and this relationship is the acoustical impedance [18]. For woodwind instruments, impedance varies with frequency. Instruments resonate at close to maximum impedance, so the resonance frequencies of an instrument closely correspond to peaks in its impedance spectra. This can be seen in Fig. 2.7 where the peaks correspond to the resonance frequencies described by Eq. 2.2.

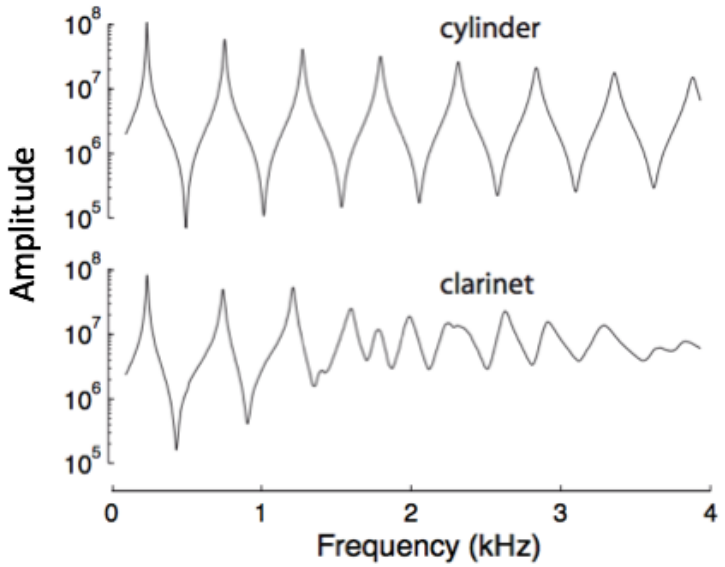


FIGURE 2.7: The impedance spectra of a closed-open cylinder and a clarinet. It can be seen that the frequency at which the peaks occur are similar for the first few peaks [18].

The impedance spectra in Fig. 2.7 shows why the approximation between the clarinet and cylinder can be made as the first few resonance frequencies of both the clarinet and cylinder are very similar. While it is a good approximation, elements of the clarinet such as the flared bell at the end are the cause for a difference in sound quality between the clarinet and a simple closed open tube. These differences in the spectra at higher frequencies are shown in Fig. 2.7.

2.4.2 Radiation Impedance

In Section 2.2.2, the resonance frequency was discussed using Eq. 2.2. While the equation is not incorrect, an alteration, which is discussed in Section 2.5, will need to be made due to the radiation impedance of air. The radiation impedance of a medium is a quantitative statement of the manner in which it reacts against the motion of a vibrating surface [21].

A useful example can be seen in sound production from a speaker. In a speaker, sound is produced by the vibrations of the diaphragm. In addition to the energy needed to vibrate the diaphragm, energy is radiated into the air by the diaphragm. Some of this radiated energy is useful and represents the sound output from the speaker whereas the remainder is reactive energy that is returned to the diaphragm. In relating this to acoustical impedance, in Eq. 2.3 acoustical impedance is defined as a complex ratio, so it has a real component, which determines the radiated power, and it has an imaginary component, which determines the reactive power [21].

2.5 End Corrections

In discussing the boundary conditions of the closed open tube set up in Section 2.2.2, it was assumed that the pressure at the open end was zero. This would be true if not for the radiation impedance of the open end, which acts like a small piston radiating into open air [21]. For a tube with a small diameter with respect to its length, this radiation impedance is small and has the same effect as to slightly increase the length of the tube. This part of the instrument, where the sound wave approaches the open end, is important in the instrument's acoustic behavior, such as the resonance frequency and its ability to radiate sound. In woodwind instruments, accurate representation of this area is vitally important because small changes in the resonant frequency are audibly noticeable.

The reflection of the acoustic wave at the end of a tube is given by the ratio of the reflected wave pressure, p^-, to the incident wave pressure, p^+, so the reflection coefficient can be written as,

$$R = \frac{p^-}{p^+}. \tag{2.4}$$

The radiation impedance can then be given as,

$$Z_r = Z_c \frac{1+R}{1-R}, \tag{2.5}$$

where $Z_c = \rho c / S$ is the characteristic acoustic impedance, ρ is the density of air, c is the speed of sound and S is the cross-sectional area of the tube [10]. The reflection coefficient can also be written as product of its magnitude and a phase term,

$$R = -|R| e^{-2ikl}, \tag{2.6}$$

where k is the wavenumber and l is known as the length correction. One way to think about the end correction is to go back to the boundary conditions laid out in Section 2.2.2 with the pressure at the open end being zero. This implies that the reflected wave pressure is equal to the opposite of the incident wave pressure. Using Eq. 2.4, this means that the reflection coefficient is -1. Hence, the radiation impedance is zero from Eq. 2.5, and the phase of the reflection coefficient is π from Eq. 2.6. The end correction is then the length added to the tube necessary to make the phase $2kl = \pi$ [9]. A representation of this can be seen in Fig. 2.8. Whether there is another effect other than the phase shift is not known; however, as part of this thesis, a pipe with a quiescent flow is simulated, which could potentially be used to better understand the physicality of the end correction. This will be left for future work. The results of the 2D simulation are discussed in Section 5.4.

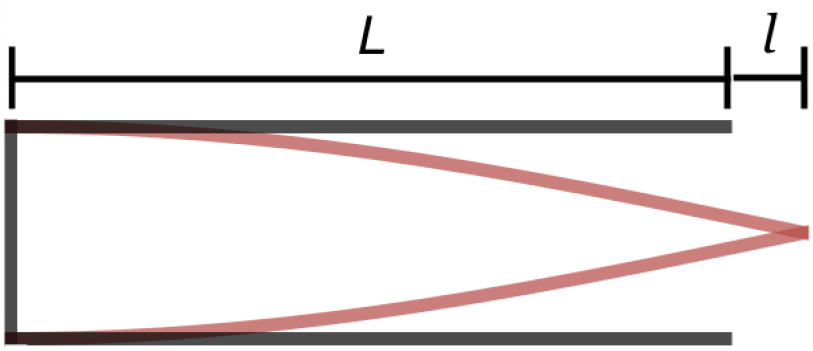


FIGURE 2.8: A representation of the pressure inside the closed open tube where the radiation impedance of the open end is nonzero with L being the length of the tube and l being the end correction.

Both the reflection coefficient and the length correction are strongly influenced by the geometry of the end of the tube. Due to how these characteristics influence the sound of the instrument, a lot of work has been done to calculate these parameters[22]–[25]. Analytical determinations of R are rather difficult and are typically derived based on many simplification such as replacing the end of a tube with a surrounding infinite flange [9]. While this simplifies the problem, it was demonstrated experimentally that this approximation yields an incorrect value for l [26]. Levine and Schwinger were the first to get an exact solution for an unflanged tube using the Wiener-Hopf technique [22]. In the low frequency limit, they found that $l = 0.6133a$ where a is the radius of the tube. Going back to Eq. 2.2, L is replaced by the corrected length, L', which yields for the resonance frequency of a closed open tube,

$$f_n = \frac{nc}{4L'} = \frac{nc}{4(L+l)}. \tag{2.7}$$

2.6 Conclusion

This chapter offered an introduction to musical acoustics, specifically the basics of the acoustics of the clarinet (the woodwind instrument studied most in

Dr. Coyle's research lab) and recorder (flute, the woodwind that will be modeled in this work). While there are many computational models that could be introduced in order to study flow in the instruments, this thesis focuses on one in particular. In Chapter 3, the mathematical background and derivations necessary to implement this method is introduced and will be followed by Chapter 4 where the implementation itself is detailed.

Chapter 3

Theory of Lattice Boltzmann Method

3.1 Introduction

In this chapter, the discretized lattice Boltzmann equation, the governing equation of the LBM, is derived. This derivation is not the author's own and is inspired by derivations from other sources [27], [28]. This derivation is in no way meant to be fully comprehensive, but rather serves as the minimum required to understand the motivation behind the LBM. Great care was taken in attempting to make this derivation as approachable to a wide range of readers while still maintaining a high mathematical level.

The derivation begins by defining the equations of motion for a single particle in one dimension in terms of the Lagrangian and then converting it in terms of the Hamiltonian. Due to the large number of particles present in a realistic system, it is necessary to focus on the one-particle distribution function as opposed to the motion of every particle. From this distribution function, the general movement of all of the particles can be determined through the use of the lattice Boltzmann equation.

3.2 Kinetic Theory

In general, when determining equations of motion, it can be useful to use the Lagrangian formulation where the function $\mathcal{L}(r_i, \dot{r}_i, t)$ is the Lagrangian and r_i (with $i = 1, \ldots, w$) are w generalized coordinates. This formulation is beneficial in this regard due to its ability to work in generalized coordinate systems and without explicit concern for constraints as is necessary with Newtonian formulation. The equations of motion for a single particle are,

$$\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{r}_i}\right) - \frac{\partial \mathcal{L}}{\partial r_i} = 0. \tag{3.1}$$

These w second order differential equation require $2w$ initial conditions to solve. For the sake of clarity and to minimize notational complications, the remainder of this discussion will be with a one-dimensional system. From here, the generalized momentum can be calculated as,

$$p = \frac{\partial \mathcal{L}}{\partial \dot{r}}. \tag{3.2}$$

This allows for Eq. 3.1 to be written as,

$$\dot{p} = \frac{\partial \mathcal{L}}{\partial r}. \tag{3.3}$$

Going forward, instead of having the equations of motion dependent on $(r, \dot{r})$, it is beneficial to have them dependent on (r, p). The rationale behind this is that the state of a system is defined by r and p, so this information allows for the state to be determined at all times in the future. Now, instead of a w dimensional configuration space, (r,p) defines a point in a $2w$-dimensional phase space [27]. In order to replace $\dot{r}$ with p, the Legendre transform can be used.

Through the use of a Legendre transformation, the Hamiltonian, $\mathcal{H}$, is defined as,

$$\mathcal{H}(r, p, t) = p\dot{r} - \mathcal{L}(r, \dot{r}, t), \tag{3.4}$$

where $\dot{r}$ has been eliminated in favor of p. Looking at the variation of the Hamiltonian,

$$\begin{aligned} d\mathcal{H} &= \dot{r}dp + pd\dot{r} - \frac{\partial \mathcal{L}}{\partial r}dr - \frac{\partial \mathcal{L}}{\partial \dot{r}}d\dot{r} - \frac{\partial \mathcal{L}}{\partial t}dt \\ &= \dot{r}dp - \frac{\partial \mathcal{L}}{\partial r}dr - \frac{\partial \mathcal{L}}{\partial t}dt. \end{aligned} \tag{3.5}$$

The Hamiltonian can also be written as a total differential yielding,

$$d\mathcal{H} = \frac{\partial \mathcal{H}}{\partial r}dr + \frac{\partial \mathcal{H}}{\partial p}dp + \frac{\partial \mathcal{H}}{\partial t}dt. \tag{3.6}$$

By equating the terms in Eqs. 3.5 and 3.6, Hamilton's equations can be written as,

$$\dot{p} = -\frac{\partial \mathcal{H}}{\partial r} \tag{3.7}$$

$$\dot{r} = \frac{\partial \mathcal{H}}{\partial p}. \tag{3.8}$$

One of the benefits of utilizing the Hamiltonian and this $2w$-dimensional phase space is that it allows for Liouville's Theorem to be employed, which states that as a region in phase space evolves over time, the shape will change, but the volume will remain constant. This means that the probability distribution of locating a particle on phase space acts like an incompressible fluid [27]. Since probability is locally conserved, it follows a continuity equation so the probability distribution obeys,

$$\frac{\partial f}{\partial t} + \frac{\partial}{\partial r}\left(\dot{r}f\right) + \frac{\partial}{\partial p}\left(\dot{p}f\right) = 0. \tag{3.9}$$

This is a general mathematical statement that becomes physical by applying

the Hamiltonian. Using Hamilton's equations in Eqs. 3.7 and 3.8, Eq. 3.9 becomes,

$$\frac{\partial f}{\partial t} + \frac{\partial f}{\partial r}\frac{\partial \mathcal{H}}{\partial p} - \frac{\partial f}{\partial p}\frac{\partial \mathcal{H}}{\partial r} = 0. \tag{3.10}$$

Eq. 3.10 is known as Liouville's equation, which says that probability of locating a particle in a given volume doesn't change as it moves along any trajectory in phase space [27]. In that regard it is very similar to Liouville's Theorem which can be written in its more common form as,

$$\frac{\partial f}{\partial t} = \{\mathcal{H}, f\}, \tag{3.11}$$

where $\{\cdot, \cdot\}$ is the Poisson bracket [29].

3.3 Lattice Boltzmann Equation

Up until this point, only one particle in a one dimensional space has been considered. It is at this point that it becomes necessary to revert to a more realistic system with N identical particles in 3 dimensional space. It is important to note that $N \sim \mathcal{O}(10^{23})$ [27]. The Hamiltonian of this system can be written in the form,

$$\mathcal{H} = \frac{1}{2m}\sum_{i=1}^{N} \vec{p_i}^2 + \sum_{i=1}^{N} V(\vec{r_i}) + \sum_{i<j} U(\vec{r_i} - \vec{r_j}), \tag{3.12}$$

where $\vec{F} = -\nabla V$ is an external force that acts equally on all particles, m is the mass of the particles and the two body interactions between particles is described by the potential energy U [27]. In looking at the probability distribution function from before, it now describes the probability distribution over this $6N$-dimensional phase space. In other words, it is now a function of $\sim 10^{23}$ variables, so instead the focus will be on the one-particle distribution function, f_1, that gives us the expected number of particles around a point

$(\vec{r}, \vec{p})$. This function is defined as,

$$f_1(\vec{r}, \vec{p}; t) = N \int \prod_{i=2}^{N} d^3 r_i d^3 p_i f(\vec{r}, \vec{r_2}, \ldots, \vec{r_N}, \vec{p}, \vec{p_2}, \ldots, \vec{p_N}; t), \tag{3.13}$$

where the N ensures that f_1 is normalized [27]. Since all particles are identical, the first particle can be chosen for this function without loss of generality. As it turns out, many thermodynamic properties of the system can be derived from f_1. For example, the average density of particles, average velocity of particles and energy flux are given respectively as,

$$\rho(\vec{r}; t) = \int d^3 p f_1(\vec{r}, \vec{p}; t), \tag{3.14}$$

$$\vec{u}(\vec{r}; t) = \int d^3 p \frac{\vec{p}}{m} f_1(\vec{r}, \vec{p}; t), \tag{3.15}$$

$$\vec{\xi}(\vec{r}; t) = \int d^3 p \frac{\vec{p}}{m} E(\vec{p}) f_1(\vec{r}, \vec{p}; t), \tag{3.16}$$

where $E = p^2/2m$ [27]. In order to obtain a governing equation for f_1, it is necessary to understand how it changes with time so,

$$\frac{\partial f_1}{\partial t} = N \int \prod_{i=2}^{N} d^3 r_i d^3 p_i \frac{\partial f}{\partial t}. \tag{3.17}$$

Using Liouville's Equation (3.11), this becomes,

$$\frac{\partial f_1}{\partial t} = N \int \prod_{i=2}^{N} d^3 r_i d^3 p_i \{\mathcal{H}, f\}. \tag{3.18}$$

Expanding this using the Hamiltonian defined in Eq. 3.12 yields,

$$\begin{aligned} \frac{\partial f_1}{\partial t} = N \int \prod_{i=2}^{N} d^3 r_i d^3 p_i \Bigg[& - \sum_{j=1}^{N} \frac{\vec{p}_j}{m} \cdot \frac{\partial f}{\partial \vec{r}_j} \\ & + \sum_{j=1}^{N} \frac{\partial V(\vec{r})}{\partial \vec{r}_j} \cdot \frac{\partial f}{\partial \vec{p}_j} + \sum_{j=1}^{N} \sum_{k<l} \frac{\partial U(\vec{r}_k - \vec{r}_l)}{\partial \vec{r}_j} \cdot \frac{\partial f}{\partial \vec{p}_j} \Bigg]. \end{aligned} \tag{3.19}$$

Notationally, the use of derivatives with respect to vectors is for the sake

of clarity and to be consistent with other sources in deriving the Boltzmann equation [27], [30]. In this sense, these derivatives are akin to the gradient. For a scalar function, $g(\vec{x})$ where $\vec{x} = [x_1, x_2]^T$, the gradient of $g(\vec{x})$ is,

$$\nabla g(\vec{x}) = \frac{\partial g}{\partial \vec{x}} = \left[\frac{\partial g}{\partial x_1}, \frac{\partial g}{\partial x_2}\right]^T = \frac{\partial g}{\partial x_1}\hat{x}_1 + \frac{\partial g}{\partial x_2}\hat{x}_2, \tag{3.20}$$

where $\hat{x}_1$ and $\hat{x}_2$ are unit vectors in the x_1 and x_2 direction respectively. Looking at the probability distribution function, since f is over phase space, the gradient includes both $\partial/\partial\vec{r}_i$ and $\partial/\partial\vec{p}_i$. The benefit of this notation is that it makes explicit reference to the argument of f to which the derivative is being taken.

While the integral in Eq. 3.19 appears to be rather unruly, it can be greatly simplified through the use of integration by parts,

$$\int u dv = uv - \int v du. \tag{3.21}$$

To get a better understanding of this, I will examine a section of the integral, the second summation term, for an arbitrary $j > 2$ yielding,

$$\begin{aligned} &\int \prod_{i=2}^{N} d^3r_i d^3p_i \frac{\partial V(\vec{r})}{\partial \vec{r}_j} \cdot \frac{\partial f}{\partial \vec{p}_j} \\ &= \int d^3r_2 d^3p_2 \cdots d^3r_j d^3p_j \cdots d^3r_N d^3p_N \frac{\partial V(\vec{r})}{\partial \vec{r}_j} \cdot \frac{\partial f}{\partial \vec{p}_j}. \end{aligned} \tag{3.22}$$

Identifying $dv = dp_j \frac{\partial f}{\partial p_j}$ and u as the remainder of the expression, the integral becomes,

$$\int \prod_{i=2}^{N} d^3r_i d^3p_i \frac{\partial V(\vec{r})}{\partial \vec{r}_j} \cdot \frac{\partial f}{\partial \vec{p}_j} = -\int \prod_{i=2}^{N} d^3r_i d^3p_i f \frac{\partial}{\partial p_j} \frac{\partial V(\vec{r})}{\partial \vec{r}_j}, \tag{3.23}$$

where the uv term goes to zero. Notice that V is only a function of $\vec{r}$ thus Eq. 3.23 is equal to zero. In fact, this same process of shifting the derivative away

from f and on to the other terms that are not functions of that variable to which the derivative is being taken can be repeated whenever $j = 2, \ldots, N$. This leaves only the $j = 1$ terms, so Eq. 3.19 is simplified to,

$$\begin{aligned}\frac{\partial f_1}{\partial t} = N \int \prod_{i=2}^{N} d^3 r_i d^3 p_i \Bigg[- \frac{\vec{p}}{m} \cdot \frac{\partial f}{\partial \vec{r}} \\ + \frac{\partial V(\vec{r})}{\partial \vec{r}} \cdot \frac{\partial f}{\partial \vec{p}} + \sum_{k=2}^{N} \frac{\partial U(\vec{r} - \vec{r_k})}{\partial \vec{r}} \cdot \frac{\partial f}{\partial \vec{p}} \Bigg],\end{aligned} \tag{3.24}$$

where $\vec{r_1} \equiv \vec{r}$ and $\vec{p_1} \equiv \vec{p}$ in accordance with previous notation. Defining the one-particle Hamiltonian as,

$$\mathcal{H}_1 = \frac{p^2}{2m} + V(\vec{r}), \tag{3.25}$$

allows for Eq. 3.24 to be written as a quasi-Liouville equation as,

$$\frac{\partial f_1}{\partial t} = \{\mathcal{H}_1, f_1\} + \sum_{k=2}^{N} \frac{\partial U(\vec{r} - \vec{r_k})}{\partial \vec{r}} \cdot \frac{\partial f}{\partial \vec{p}}. \tag{3.26}$$

Renaming the second term yields the Boltzmann equation,

$$\frac{\partial f_1}{\partial t} = \{\mathcal{H}_1, f_1\} + \left(\frac{\partial f_1}{\partial t} \right)_{coll}. \tag{3.27}$$

The first term is known as the streaming operator where it contains the information pertaining to how particles move when not colliding, and the second term is known as the collision operator where it contains the information about particle collisions.

It should be noted that referring to Eq. 3.27 as the Boltzmann equation is a bit disingenuous. While it is indeed equivalent to the Boltzmann equation, calling the collision operator in Eq. 3.27 equal to the second term in Eq. 3.26 is not a proper stopping point. In its current form, the collision term is not in terms of the one particle distribution function. The derivation of this collision

term in terms of f_1 is outside of the purview of this thesis, and would be superfluous as will be seen later because this term will be replaced entirely. A complete derivation of the collision term can be found in Ref. [27].

With the Boltzmann equation derived, it is only a matter of manipulation and discretization to arrive at the lattice Boltzmann equation. Expanding the streaming operator and applying Hamilton's equations (3.7 and 3.8), Eq. 3.27 becomes,

$$\frac{\partial f_1}{\partial t} + \frac{d\vec{r}}{dt}\frac{\partial f_1}{\partial \vec{r}} + \frac{d\vec{p}}{dt}\frac{\partial f_1}{\partial \vec{p}} = \left(\frac{\partial f_1}{\partial t}\right)_{coll}. \tag{3.28}$$

Realizing $\frac{d\vec{r}}{dt} = \frac{\vec{p}}{m}$ and $\frac{d\vec{p}}{dt} = \vec{F}$ yields,

$$\frac{\partial f_1}{\partial t} + \frac{\vec{p}}{m}\cdot\nabla f_1 + \vec{F}\cdot\frac{\partial f_1}{\partial \vec{p}} = \left(\frac{\partial f_1}{\partial t}\right)_{coll}. \tag{3.29}$$

In this case, there are no outside forces making $\vec{F} = 0$ and it is customary to now switch from momentum to velocity so $\frac{\vec{p}}{m} = \vec{v}$, making Eq. 3.29,

$$\frac{\partial f_1}{\partial t} + \vec{v}\cdot\nabla f_1 = \left(\frac{\partial f_1}{\partial t}\right)_{coll}. \tag{3.30}$$

3.3.1 Single Relaxation Time

The most common way to replace this collision term is to use the Bhatnagar, Gross, Krook (BGK) approximation where the collision term is replaced with a collision operator that depends on a single relaxation parameter Ω [31]. This yields the Boltzmann equation with the BGK approximation,

$$\frac{\partial f}{\partial t} + \vec{v}\cdot\nabla f = -\frac{1}{\tau}(f - f^{eq}), \tag{3.31}$$

where f_1 has been redefined as, $f_1 \equiv f = f(\vec{r}, \vec{v}, t)$, and $\tau = 1/\Omega$ is the relaxation time of f to the equilibrium state f^{eq}. The collision operator ensures that $f - f^{eq}$ decays exponentially as $e^{-t/\tau}$ where f^{eq} is a Maxwellian distribution function [31].

To obtain the discrete Boltzmann equation, the velocity space is discretized into a finite set of velocities, $\vec{v}_q$, with their respective distribution functions, $f_q(\vec{r}, t)$. The discrete Boltzmann equation can then be written as,

$$\frac{\partial f_q}{\partial t} + \vec{v}_q \cdot \nabla f_i = -\frac{1}{\tau}(f_q - f_q^{eq}). \tag{3.32}$$

How the space is discretized is discussed in Section 4.2.

From here, the variables are converted from whatever units they may have to lattice units which are dimensionless. This is done using the characteristic length scale, L, the reference speed, U, the reference density, n_r, and the time between particle collisions, t_c yielding,

$$\frac{\partial f_q'}{\partial t'} + \vec{c}_q \cdot \nabla' f_q' = -\frac{1}{\tau'\epsilon}(f_q' - f_q^{eq'}), \tag{3.33}$$

where $\vec{c}_q = \vec{v}_q/U, \nabla' = L\nabla, t' = t \cdot U/L, \tau' = \tau/t_c, f_q' = f_q/n_r$ and $\epsilon = t_c \cdot U/L$ is the Knudsen number [28]. Discretizing Eq. 3.33 using a first order finite difference scheme gives,

$$\begin{aligned}\frac{f_q'(\vec{r'}, t' + \Delta t') - f_q'(\vec{r'}, t')}{\Delta t'} + \sum_{s'=x',y',z'} c_{is'} \frac{f_q'(\vec{r'} + \Delta s', t' + \Delta t') - f_q'(\vec{r'}, t' + \Delta t')}{\Delta s'} \\ = -\frac{1}{\tau'\epsilon}(f_q' - f_q^{eq'}),\end{aligned} \tag{3.34}$$

where x', y' and z' are the normalized position components. By having $\Delta\vec{r'}/\Delta t' = \vec{c}_q$, this can be rewritten as,

$$\frac{f_q'(\vec{r'} + \vec{c}_q\Delta t', t' + \Delta t') - f_q'(\vec{r'}, t')}{\Delta t'} = -\frac{1}{\tau'\epsilon}(f_q' - f_q^{eq'}). \tag{3.35}$$

Finally, choosing $\Delta t = t_c$ and dropping all the primes leads to the BGK lattice Boltzmann equation,

$$f_q(\vec{r} + \vec{c_q}\Delta t, t + \Delta t) - f_q(\vec{r}, t) = -\frac{1}{\tau}(f_q - f_q^{eq}). \tag{3.36}$$

The left hand side of Eq. 3.36 dictates the streaming of the distribution functions, and the right hand side is the collision operator which describes how the distribution functions change due to collisions. It should be noted that Eq. 3.36 is explicit, meaning that $f_q(t + \Delta t)$ is determined from $f_q(t)$, and simple in that the underlying arithmetic operations are addition and multiplication. Additionally, the collision step is local in that $f(\vec{r}, t + \Delta t)$ depends only on $f(\vec{r}, t)$ while the streaming step is not local because $f(\vec{r} + \vec{c_q}\Delta t, t + \Delta t)$ depends on $f(\vec{r}, t)$. This will be of importance when discussing the inherent parallel nature of the LBM in Section 4.4.

The equilibrium distribution function is the Maxwell-Boltzmann distribution. By using the small Mach number approximation and a second order polynomial expansion about $\vec{u} = 0$, the discrete form of the equilibrium distribution function is

$$f_q^{eq} = w_q \rho \left(1 + \frac{\vec{c_q} \cdot \vec{u}}{c_s^2} + \frac{(\vec{c_q} \cdot \vec{u})^2}{2c_s^4} - \frac{\vec{u}^2}{2c_s^2}\right), \tag{3.37}$$

where c_s is the speed of sound in lattice units, $\vec{u}$ is the velocity and w_q are velocity weights that are calculated to ensure the conservation of mass and momentum [32]. As seen in Eqs. 3.14 and 3.15, the fluid density and velocity can be determined from the distribution function. In the discrete form, this relation mimics the conservation of mass and momentum as,

$$\rho = \sum f_q, \tag{3.38}$$

$$\vec{u} = \frac{1}{\rho} \sum f_q \vec{c_q}. \tag{3.39}$$

3.3.2 Multiple Relaxation Times

The LBGK model from Eq. 3.36 is known as a Single Relaxation Time (SRT) LBM model due its single relaxation parameter, Ω. The LBGK-SRT model is commonly used because of its simple implementation. This characterization as simple comes from the use of a single constant relaxation term and easily calculated functions. While the LGBK-SRT model is generally reliable, its dependence on a single relaxation term can cause stability issues. This can be the case for simulations with fluids that have small kinematic viscosities, such as air, where numerical instabilities arise due to the strong anisotropy of the dispersion relations from the lattice symmetry [33].

In order to overcome these instabilities, d'Humieres proposed using a Multiple Relaxation Time (MRT) model where the single relaxation parameter is replaced by a relaxation matrix [34]. Additionally, instead of performing the collision in the velocity space where the distribution function is relaxed towards equilibrium, the collision is performed in the moment space allowing for the time scale of moment relaxing towards equilibrium to be controlled separately. By allowing the moments to relax towards equilibrium on different time scales, numerical stability is increased.

Similar to Eq. 3.36, the LBE can be rewritten using the MRT scheme by replacing Ω with the multiple relaxation time matrix $\mathbf{S}$ as,

$$\mathbf{f}(\vec{r} + \vec{c_q}\Delta t, t + \Delta t) - \mathbf{f}(\vec{r}, t) = -\mathbf{S}(\mathbf{f} - \mathbf{f}^{eq}), \tag{3.40}$$

where $\mathbf{f} = (f_0, f_1, \ldots, f_{n-1})^T$ is the vector of n discrete distribution functions in the $n-$dimensional velocity space. It can be seen from Eq. 3.40 that the LGBK-SRT is a special case in which $\mathbf{S} = \Omega\mathbf{I}$ where $\mathbf{I}$ is the identity matrix. Complimentary to the velocity space is the $n-$dimensional moment space that is comprised of n moments, $\mathbf{m} = (m_0, m_1, \ldots, m_{n-1})^T$. In the moment

space, the collision process is,

$$\mathbf{m}(\vec{r} + \vec{c_q}\Delta t, t + \Delta t) - \mathbf{m}(\vec{r}, t) = -\hat{\mathbf{S}}(\mathbf{m} - \mathbf{m}^{eq}), \tag{3.41}$$

where $\hat{\mathbf{S}}$ is a diagonal matrix. The mapping of $\mathbf{f}$ in the velocity space onto $\mathbf{m}$ in the moment space is through the use of the transformation matrix, $\mathbf{M}$, as,

$$\mathbf{m} = \mathbf{M}\mathbf{f}. \tag{3.42}$$

Similarly, $\mathbf{m}$ can be mapped to $\mathbf{f}$ as,

$$\mathbf{f} = \mathbf{M}^{-1}\mathbf{m}. \tag{3.43}$$

The diagonal matrix $\hat{\mathbf{S}}$ is given by,

$$\hat{\mathbf{S}} \equiv \text{diag}(s_0, s_1, \ldots, s_{n-1}) = \mathbf{M}\mathbf{S}\mathbf{M}^{-1}. \tag{3.44}$$

The values of $\hat{\mathbf{S}}$ and $\mathbf{M}$ are dependent on which moments are considered and how the space is discretized which is discussed in Section 4.2. For the MRT model, the collision step is now broken into three steps. First, the distribution function is mapped onto the moment space using Eq. 3.42. Next, the collision is performed using Eq. 3.41. Finally, the moments are mapped back onto the velocity space using Eq. 3.43 where the distribution functions can then be streamed. More concisely, these last two steps can be combined into one yielding,

$$\mathbf{f}(\vec{r} + \vec{c_q}\Delta t, t + \Delta t) - \mathbf{f}(\vec{r}, t) = -\mathbf{M}^{-1}\mathbf{S}(\mathbf{m} - \mathbf{m}^{eq}). \tag{3.45}$$

This process of streaming the distribution function to adjacent nodes will be described in the following chapter.

3.4 Conclusion

The lattice Boltzmann method has been widely used in woodwind acoustics over the past 25 years to study the flow in musical instruments. The programming language and specifics of implementation have varied by author. In Chapter 4 the necessary steps for implementation of a few different versions of this model in two different programming languages are offered.

Chapter 4

Implementation of Lattice Boltzmann Method

4.1 Introduction

Previous chapters provided motivation for the work, necessary background in musical acoustics, and a mathematical basis for the lattice Boltzmann method. In this chapter the steps to implementing the LBM in order to simulate flow in musical instruments will be discussed.

4.2 LBM Models

In discretizing the space, the domain is broken into a lattice that can be 1, 2 or 3 dimensional depending on the model. Each node in the lattice is known as a cell that is comprised of q propagation sites. Each of these sites then corresponds with the distribution function f_q that propagates to the next cell at every time step, Δt. The cells in the lattice are connected by the velocity vectors, $\vec{c_q}$. The standard nomenclature for the lattice geometry of the model is $DmQn$ where m corresponds with the number of dimensions of the model and n corresponds with the number of propagation sites q [35]. Much to the chagrin of programming languages that are not zero-based indexing, the naming standard for the propagation sites starts with the non-propagating

site in the middle of the cell as $q = 0$. Below, a few of the most common models are discussed. The lattice speed of sound, c_s, and the velocity weight, w_q, are determined by the model used.

4.2.1 One Dimension

Starting with one dimension, the lattice cell representation of the D1Q3 model is depicted in Fig. 4.1. The speed of sound here is $1/\sqrt{3}$ and the velocity weights, w_q, and vectors, $\vec{c_q}$ are,

$$w_q = \begin{cases} 4/6 & \text{for } q = 0 \\ 1/6 & \text{for } q = 1, 2 \end{cases} \tag{4.1}$$

$$\vec{c_q} = \begin{cases} (0,0) & \text{for } q = 0 \\ (\pm 1, 0)c & \text{for } q = 1, 2. \end{cases} \tag{4.2}$$

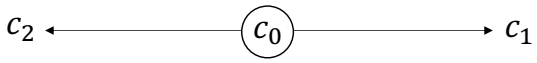


FIGURE 4.1: The lattice cell representation for the D1Q3 model where c_0 is the non-propagating site and c_1 and c_2 are the propagation sites. The velocity vectors correspond to the position of c_q multiplied by the speed.

4.2.2 Two Dimensions

Moving on to two dimensions, the lattice cell representation for the D2Q9 model is portrayed in Fig. 4.2. This is a common two dimensional model because it leads to second-order accurate solutions and can recover the Navier Stokes equations from Eq. 3.36 [32]. Additionally, the D2Q9 model allows for a conceptually convenient partition of the domain. The speed of sound is

$1/\sqrt{3}$ and the velocity weights, w_q and vectors, $\vec{c_q}$ are as follows,

$$w_q = \begin{cases} 4/9 & \text{for } q = 0 \\ 1/9 & \text{for } q = 1, \ldots, 4 \\ 1/36 & \text{for } q = 5, \ldots, 9 \end{cases} \tag{4.3}$$

$$\vec{c_i} = \begin{cases} (0,0) & \text{for } q = 0 \\ (\pm 1, 0)c, (0, \pm 1)c & \text{for } q = 1, \ldots, 4 \\ (\pm 1, \pm 1)c & \text{for } q = 5, \ldots, 9 \end{cases} \tag{4.4}$$

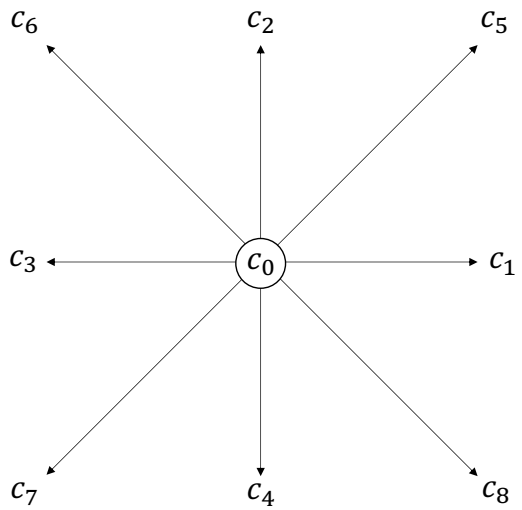


FIGURE 4.2: The lattice cell representation for the D2Q9 model where c_0 is the non-propagating site and $c_1, \ldots, c_8$ are the propagation sites. Notice that the naming convention starts with the edges of the square followed by its verticies in an anticlockwise manner.

The transformation matrix, $\mathbf{M}$, for the MRT model from section 3.3.2 is defined as,

$$\mathbf{M} \equiv \begin{pmatrix} \boldsymbol{\rho} \\ \mathbf{e} \\ \boldsymbol{\xi} \\ \mathbf{j_x} \\ \mathbf{q_x} \\ \mathbf{j_y} \\ \mathbf{q_y} \\ \mathbf{p_{xx}} \\ \mathbf{p_{xy}} \end{pmatrix} \equiv \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ -4 & -1 & -1 & -1 & -1 & 2 & 2 & 2 & 2 \\ 4 & -2 & -2 & -2 & -2 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & -1 & 0 & 1 & -1 & -1 & 1 \\ 0 & -2 & 0 & 2 & 0 & 1 & -1 & -1 & 1 \\ 0 & 0 & 1 & 0 & -1 & 1 & 1 & -1 & -1 \\ 0 & 0 & -2 & 0 & 2 & 1 & 1 & -1 & -1 \\ 0 & 1 & -1 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & -1 & 1 & -1 \end{pmatrix}, \tag{4.5}$$

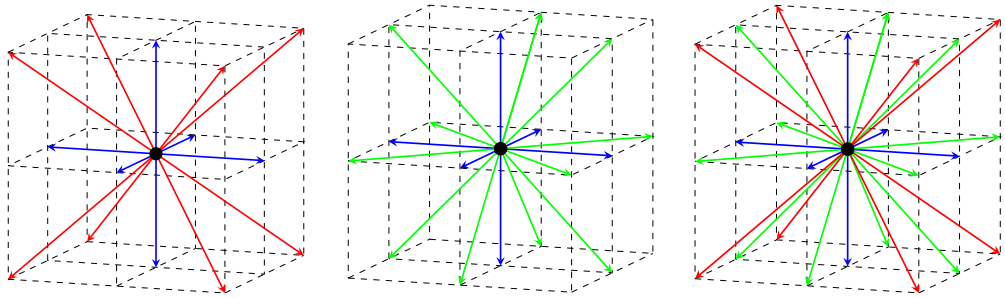
where $\boldsymbol{\rho}$ is the density mode, $\mathbf{e}$ is the energy mode, $\boldsymbol{\xi}$ is related to the energy square, $\mathbf{j_x}$ and $\mathbf{j_y}$ correspond to the x- and y-component of momentum, $\mathbf{q_x}$ and $\mathbf{q_y}$ correspond to the x- and y-component of energy flux and $\mathbf{p_{xx}}$ and $\mathbf{p_{xy}}$ correspond to the diagonal and off-diagonal component of the stress tensor. More details on this can be found in [33]. The diagonal matrix is defined as,

$$\hat{\mathbf{S}} \equiv \text{diag}(0, -s_2, -s_3, 0, -s_5, 0, -s_7, -s_8, -s_9) \tag{4.6}$$

where $s_2 = 2/(6\zeta + 1) = 1.63$ and $s_8 = s_9 = 2/(6\nu + 1) = 1/\tau$ with ζ and ν being the bulk and shear viscosity respectively, $s_3 = 1.14$ and $s_5 = s_7 = 1.92$ [33].

4.2.3 Three Dimensions

For three dimensions, there are three different models that are commonly used, the D3Q15, D3Q19 and D3Q27 models which are shown in Fig. 4.3. These models differ in which propagation sites they have on the cube. Each model connects the center to the six faces, but where they differ is in whether



(A) D3Q15 (B) D3Q19 (C) D3Q27

FIGURE 4.3: The lattice representations for three commonly used three dimensional lattice Boltzmann models. Notice that for all three models, the six faces of the cube are included (blue) however for 4.3a the verticies are included (red) unlike 4.3b where the edges are included (green). 4.3c includes all 27 propagation sites, the verticies, edges and faces [4].

they connect the center to the vertices (D3Q15), edges (D3Q19) or both (D3Q27). Practically, the differences between D3Q15 and D3Q19 are fairly small. The additional propagation sites lead to the D3Q19 model being more computationally intense; however, it also converges more quickly to a steady state. The D3Q27 model is different in that it is better suited for more complex problems because it is capable of representing an energy equation due to the number of degrees of freedom. The naming convention is that the non-propagation site is 0 as usual followed by the face sites, then the edge sites and finally the vertex sites as can be seen by the velocity vectors below. The velocity weights and vectors for the three models are:

For D3Q15,

$$w_q = \begin{cases} 2/9 & \text{for } q = 0 \\ 1/9 & \text{for } q = 1, \ldots, 6 \\ 1/72 & \text{for } q = 7, \ldots, 14 \end{cases} \tag{4.7}$$

$$\vec{c_q} = \begin{cases} (0,0,0) & \text{for } q = 0 \\ (\pm 1,0,0)c, (0,\pm 1,0)c, (0,0,\pm 1)c & \text{for } q = 1,\ldots,6 \\ (\pm 1,\pm 1,\pm 1)c & \text{for } q = 7,\ldots,14 \end{cases} \tag{4.8}$$

For D3Q19,

$$w_q = \begin{cases} 1/3 & \text{for } q = 0 \\ 1/18 & \text{for } q = 1,\ldots,6 \\ 1/36 & \text{for } q = 7,\ldots,18 \end{cases} \tag{4.9}$$

$$\vec{c_q} = \begin{cases} (0,0,0) & \text{for } q = 0 \\ (\pm 1,0,0)c, (0,\pm 1,0)c, (0,0,\pm 1)c & \text{for } q = 1,\ldots,6 \\ (\pm 1,\pm 1,0)c, (\pm 1,0,\pm 1)c, (0,\pm 1,\pm 1)c & \text{for } q = 7,\ldots,18 \end{cases} \tag{4.10}$$

For D3Q27,

$$w_q = \begin{cases} 8/27 & \text{for } q = 0 \\ 2/27 & \text{for } q = 1,\ldots,6 \\ 1/54 & \text{for } q = 7,\ldots,18 \\ 1/216 & \text{for } q = 19,\ldots,26 \end{cases} \tag{4.11}$$

$$\vec{c_q} = \begin{cases} (0,0,0) & \text{for } q = 0 \\ (\pm 1,0,0)c, (0,\pm 1,0)c, (0,0,\pm 1)c & \text{for } q = 1,\ldots,6 \\ (\pm 1,\pm 1,0)c, (\pm 1,0,\pm 1)c, (0,\pm 1,\pm 1)c & \text{for } q = 7,\ldots,18 \\ (\pm 1,\pm 1,\pm 1)c & \text{for } q = 19,\ldots,26 \end{cases} \tag{4.12}$$

When considering which model would be best to use, both the simulation time and accuracy were taken into account. With fewer streaming sites, the

D3Q15 model will result in the fastest simulation time however it has been shown to be much less accurate and more susceptible to numerical instabilities than both the D3Q19 and D3Q27 models [36]. While the differences in terms of accuracy and stability are large between the D3Q15 and D3Q19 models, these differences between D3Q19 and D3Q27 are smaller. Additionally, since the flow simulations performed in this simulation are relatively simple, the additional streaming sites provided by the D3Q27 model were deemed unnecessary. It was thus decided that for the purposes of this thesis, a D3Q19 model would be sufficient. As the simulations become increasingly complex, with the addition of the fluid structure interaction between the reed and air flow for example, it could be necessary to upgrade the code to a D3Q27 model however that is left for future work.

In addition to the modes from the D2Q9 transformation matrix in Eq. 4.5, the D3Q19 tranformation matrix $\mathbf{M}$ includes the z component of $\mathbf{j}$ and $\mathbf{q}$ along with additional off dimensional elements of the stress tensor $\mathbf{p_{ij}}$. Additionally, two vectors of quadratic order that have the same symmetry as the diagonal part of the stress tensor $\mathbf{p_{ij}}$ and three vectors of cubic order that are parts of a third rank tensor that has symmetry of $\mathbf{j_k p_{nm}}$. The transformation matrix, $\mathbf{M}$, is written out in full in [36]. The diagonal matrix is defined as,

$$\hat{\mathbf{S}} \equiv \mathrm{diag}(0, s_1, s_2, 0, s_4, 0, s_4, 0, s_4, s_9, s_{10}, s_{13}, s_{13}, s_{13}, s_{16}, s_{16}, s_{16}) \tag{4.13}$$

where $s_9 = s_{13} = 2/(6\nu + 1)$,$s_1 = 1.19$, $s_2 = s_{10} = 1.4$, $s_4 = 1.2$ and $s_{16} = 1.98$ [33].

4.3 Boundary Conditions

Boundary conditions in LBM can be split into two categories: solid and open boundaries. A solid boundary is where the fluid contacts a solid object, and

an open boundary is where the fluid reaches the end of the simulation domain. While there are many different types of boundary conditions used in LBM models, only the boundaries used in this thesis will be discussed in this section. These conditions are the no-slip, free slip and absorbing boundary condition.

4.3.1 Solid Boundary

When it comes to solid boundaries, there are two types of conditions that are used: no-slip condition and free-slip condition. The no-slip boundary condition assumes that the solid has a high enough rugosity which makes it so the tangential velocity of the particles at the solid is zero. How this approach is applied can be seen in Fig. 4.4. In the first step, the three distribution functions at position, $\vec{x}$, and time, t, that will interact with the boundary can be seen. Next, the distribution functions are propagated to their neighboring sites through the boundary at an intermediary time between t and $t + \Delta t$. In the third step, the no slip condition is applied where the directions of propagation are inverted. Finally, the distribution functions are propagated along their new inverted directions at time $t + \Delta t$. It should be noted that steps two and three from Fig. 4.4 are not performed programatically, and are only included to give a conceptual understanding of the boundary condition. Programatically, this can be described as,

$$\begin{aligned} f_6(\vec{x}, t + \Delta t) &= f_8(\vec{x}, t), \\ f_2(\vec{x}, t + \Delta t) &= f_4(\vec{x}, t), \\ f_5(\vec{x}, t + \Delta t) &= f_7(\vec{x}, t). \end{aligned} \tag{4.14}$$

Note that these are only the conditions for this direction of collision as seen in Fig. 4.4. The other equations for collisions on a north, east or west wall can be determined in a similar manner.

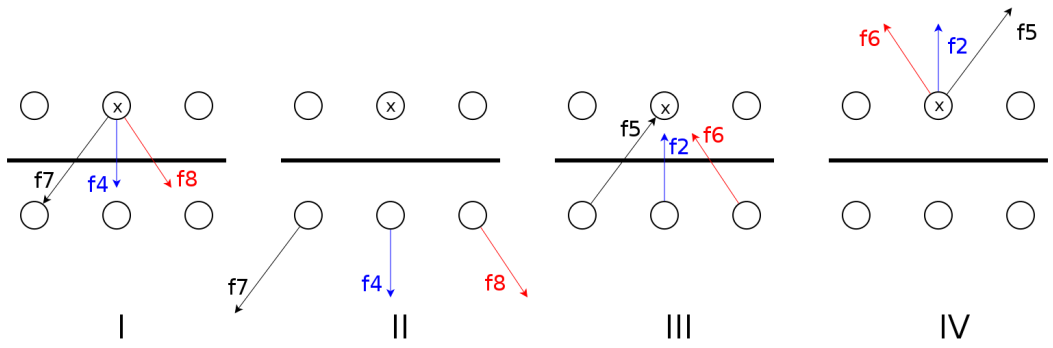


FIGURE 4.4: The procedure of the no-slip condition for the distribution functions. Notice that because the distribution function get inverted from steps two to three, the tangential velocity along the wall is zero. Steps two and three are not performed programatically, but are included as an aid to give a better understanding of the boundary condition [10].

As opposed to the no-slip boundary condition, the free-slip condition is used for smooth walls where the tangential velocity is unaffected. The implementation can be seen in Fig. 4.5. Starting at position, $\vec{x_1}$, and at time, t, the first two steps are identical to the no-slip procedure. In step three, instead of the directions of propagation being inverted, they are reflected leading to the final step where they are streamed in their new direction to the respective neighboring cell. Just like for the the no-slip condition, steps two and three are included solely as a conceptual aid. Programatically, the equations are,

$$\begin{aligned} f_6(\vec{x_0}, t + \Delta t) &= f_7(\vec{x_1}, t), \\ f_2(\vec{x_1}, t + \Delta t) &= f_4(\vec{x_1}, t), \\ f_5(\vec{x_2}, t + \Delta t) &= f_8(\vec{x_1}, t). \end{aligned} \tag{4.15}$$

Note that in comparing Figs. 4.4 and 4.5, for the no slip condition, the propagation is local in that it is not streamed to a neighboring cell whereas for the free slip condition, the distribution function is propagated to neighboring cells. This is what allows for a non zero tangential velocity at the wall for the free slip condition whereas there is a zero tangential velocity at the wall for the no slip condition.

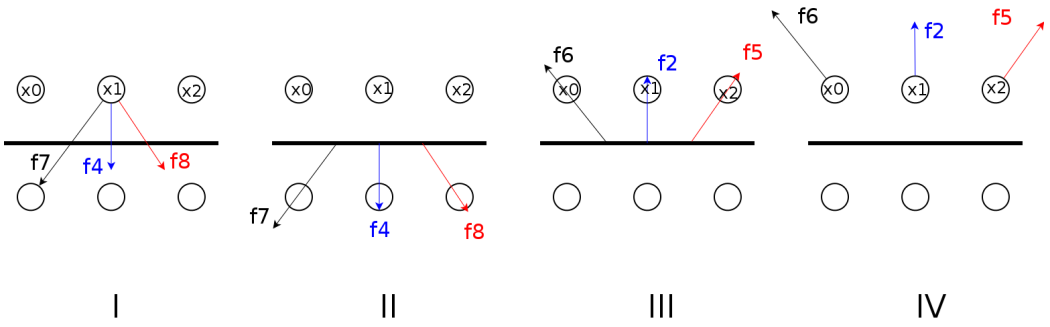


FIGURE 4.5: The procedure of the free-slip condition for the distribution functions. Notice that because the distribution functions get reflected between steps two and three, the tangential velocity along the wall is non-zero. Similar to the no-slip procedure, steps two and three are only included to give a better conceptual understanding [10].

4.3.2 Open Boundaries

Open boundary conditions are more difficult from a computational viewpoint than solid boundary conditions and must be carefully handled to ensure the stability of the code. As can be seen in Eqs. 3.38 and 3.39, it is simple to recover ρ and $\vec{u}$ from f_q; however, to derive f_q from ρ and $\vec{u}$ is not trivial. Due to the complexities involved, there are many different approaches to open boundaries that each have their benefits to particular problems. As stated earlier, these other schemes will not be addressed here, however, an overview of multiple open boundary conditions can be found in these sources [9], [10].

The open boundary condition used in this thesis is the absorbing boundary condition (ABC), which was first extended from direct numerical simulation to the LBM by Kam et al. [37]. The technique works by creating a buffer region between the fluid region and the open boundary that creates an asymptotic transition towards a pre-specified target flow through the use of target distribution functions, f_q^T. This is done by adding an additional

term into Eq. 3.36,

$$f_q(\vec{r} + \vec{c_q}\Delta t, t + \Delta t) - f_q(\vec{r}, t) = -\frac{1}{\tau}(f_q - f_q^{eq}) - \sigma(f_q^{eq} - f_q^T), \tag{4.16}$$

where $\sigma = \sigma_m(\delta/D)^2$ is the absorption coefficient and σ_m is a constant, normally 0.3, D is the width of the buffer region and δ is the distance measured from the beginning of the buffer zone to the position in the buffer zone. Dividing by D normalizes this distance. The ABC is convenient because it is easy to implement, efficient and provides second-order accuracy [37]. Additionally, the ABC allows for a source flow to be created by setting f_q^T equal to a non-zero target velocity at the inlet of the simulation. For these reasons, the ABC was chosen as the open boundary condition used in this thesis.

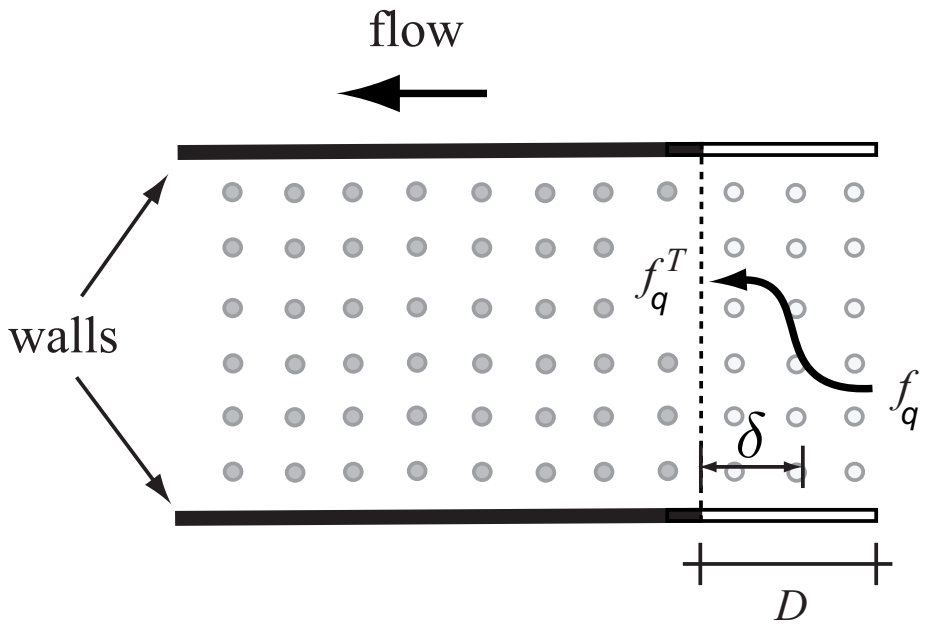


FIGURE 4.6: A schematic displaying the asymtotic target flow for the absorbing boundary condition (ABC) where D is the length of the buffer, δ is the distance from the end of the buffer and f_q^T is the target distribution function. Notice that in this schematic, the target distribution function is used to asymptotically induce a source flow at the inlet of the simulation. Not pictured, it could similarly be used to set an outlet flow at the edge of the domain [9].

4.4 GPU Computing

As will be discussed in Section 5.2, the LBM is expensive in terms of computational cycles, especially as the grid size and resolution increase. However, the LBM is inherently parallelizable due to its simple and explicit algorithm along with the fact that information is largely decoupled, outside of streaming to neighboring cells.

Before looking at the specifics of computing on the graphics processing unit (GPU), its place in the field of parallel programming should be noted. As will be discussed in Section 4.4.2, subroutines that run on the GPU are executed by many threads in parallel. While all threads execute the same code, they operate on different data, different lattice nodes in this case. This is known as fine-grained parallelism where it is advantageous to have adjacent threads operating on adjacent data. This type of parallelism is much different from course-grained parallelism, such as MPI, where the data is partitioned into large segments with each MPI thread operating on the entire data partition [38]. An example of fine-grained parallelism from conventional CPU computing is given below.

Using a conventional CPU Fortran algorithm, elements of two arrays, A and B, can be added with the following do loop:

```
1 do i = 1,5
2 C(i) = A(i) + B(i)
3 enddo
```

LISTING 4.1: A simple example of a series loop in Fortran that can parallelized.

This loop is performed sequentially, with the first index of C taking the value of the first index of A plus the first index of B followed by the second index up through the fifth index. Utilizing parallel computing, instead of each

index of the arrays being added sequentially, they are performed simultaneously allowing for a significant speed up in performance. How this works discussed in Section 4.4.2. First, the architecture of the graphics processing unit is discussed.

4.4.1 Architecture

Almost every computer contains a central processing unit (CPU) that has multiple cores. The reason for more than one core stems from the inability of CPU manufacturers to increase performance in a singe core leading to CPU designs scaling out to multiple core instead of scaling up to higher clock rates [38]. While CPUs have anywhere from a couple to dozens of cores, this pales in comparison to the number of cores available in the graphics processing unit. As graphics processing is an inherently parallel task, GPUs have a highly parallel architecture. Originally, performing general purpose GPU computing (GPGPU) was quite difficult due to the lack of support and restrictive types of algortihms that could be mapped to the GPU. This changed with the creation of NVIDIA's CUDA architecture in 2007 [38]. This allowed for the highly parallel nature of GPGPU to be realized with only slight modifications to the original CPU Fortran code.

At its most basic level, the thread processor is the unit of the GPU that executes individual GPU threads. These thread processors are grouped into multiprocessors which have a small amount of on board shared memory [38]. This hierarchy of GPU components can be seen in Fig. 4.7. As will be discussed in 4.4.2, the computational units in the GPU are analogous with components in the programming model.

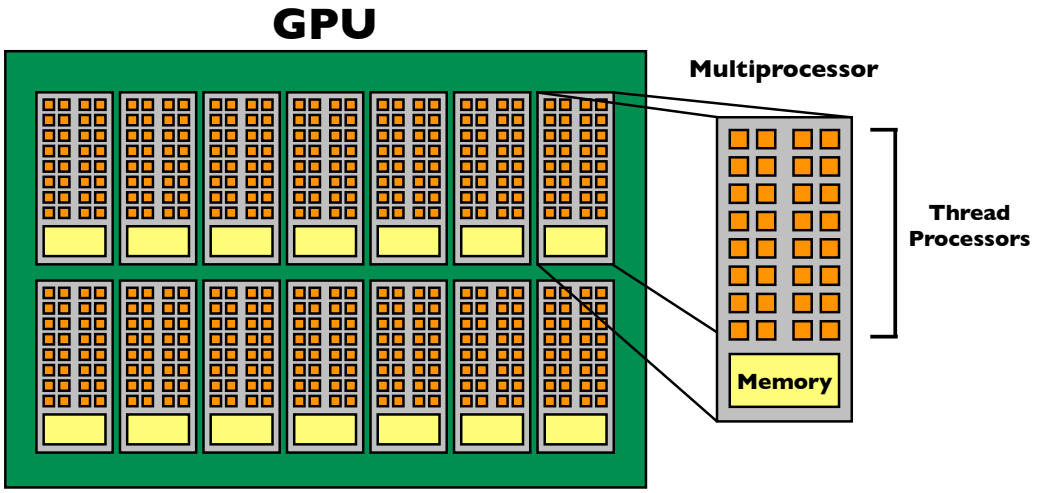


FIGURE 4.7: A graphical representation of the breakdown of the GPU into multiprocessors and thread processors. Notice that the thread processors are grouped into multiprocessors along with a limited amount of shared memory. This breakdown of the GPU computational units is analagous to the breakdown of the programming model into threads and blocks [38].

4.4.2 Programming

CUDA Fortran is a hybrid programming model which means that sections of the code can operate on the CPU or GPU, or as is more commonly said, on the host or device. The host refers to the CPU and its memory whereas the device refers to the GPU and its memory. A subroutine that is called from the host but executes on the GPU is called a kernel [38]. The use of kernels is the basis of GPU computing with CUDA.

As can be seen in Fig. 4.8, the programming model is made up of threads that are grouped into blocks that make up the execution grid. This hierarchy closely resembles that of the CPU architecture where the analog of the multiprocessor is the thread block. Thread blocks are assigned to multiprocessors when the kernel is called and do not migrate once assigned. Multiple thread blocks can be on a single multiprocessor however this number is limited by the required resources of each thread block [38].

The layout of the execution grid is determined at run time, and it can have up to three dimensions of blocks. Similarly, the blocks of threads within

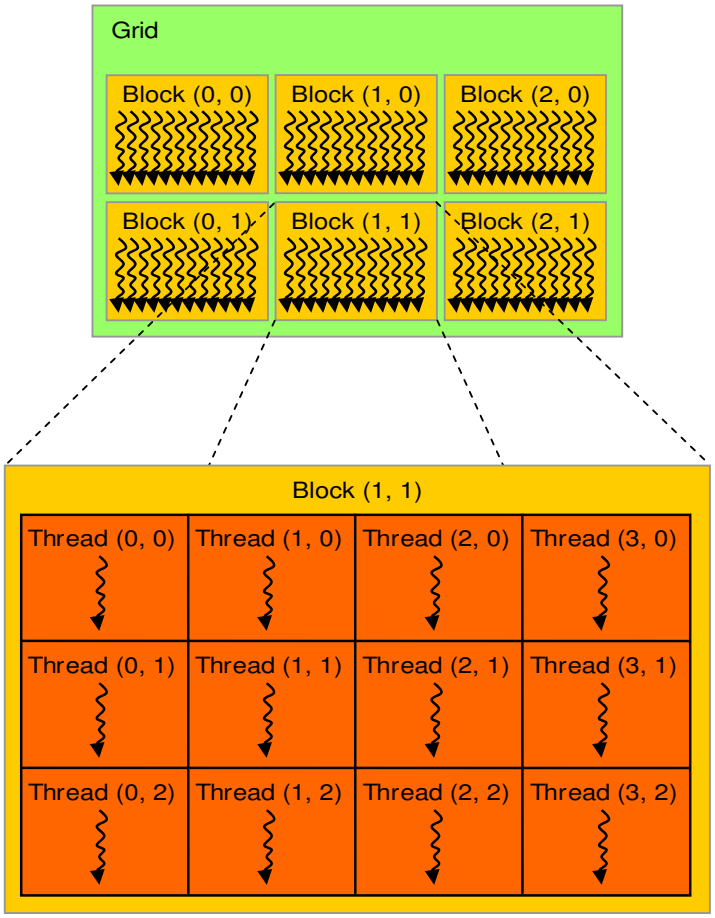


FIGURE 4.8: The breakdown of the programming model where the grid is comprised of thread blocks which are further comprised of threads. Notice that this hierarchy is similar to the architecture of the GPU itself [39].

the grid can also have up to three dimensions, however, they each must be identical in terms of threads per block. Each thread is identified by a unique identifier through the use of the structures, threadIdx and blockIdx, which contain the three fields x, y and z.

The mapping of lattice nodes on to threads on the GPU is rather simplistic. As shown in Fig. 4.9, each lattice node is mapped to one thread with adjacent nodes corresponding to adjacent threads on the GPU. The number of threads per block was chosen in order to maximize the number of concurrent threads running on each multiprocessor.

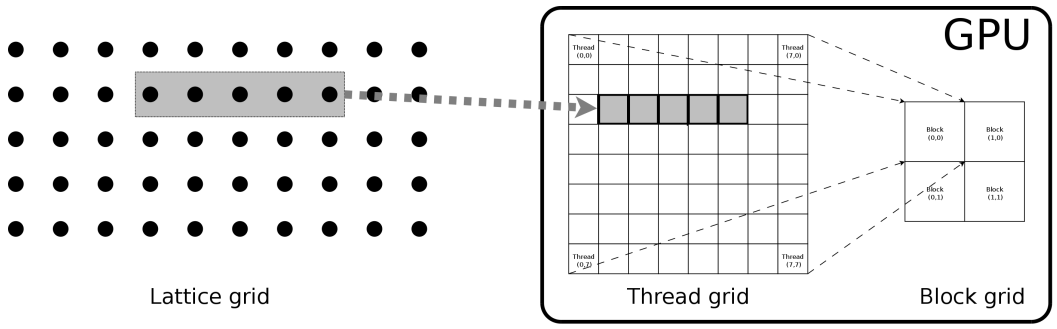


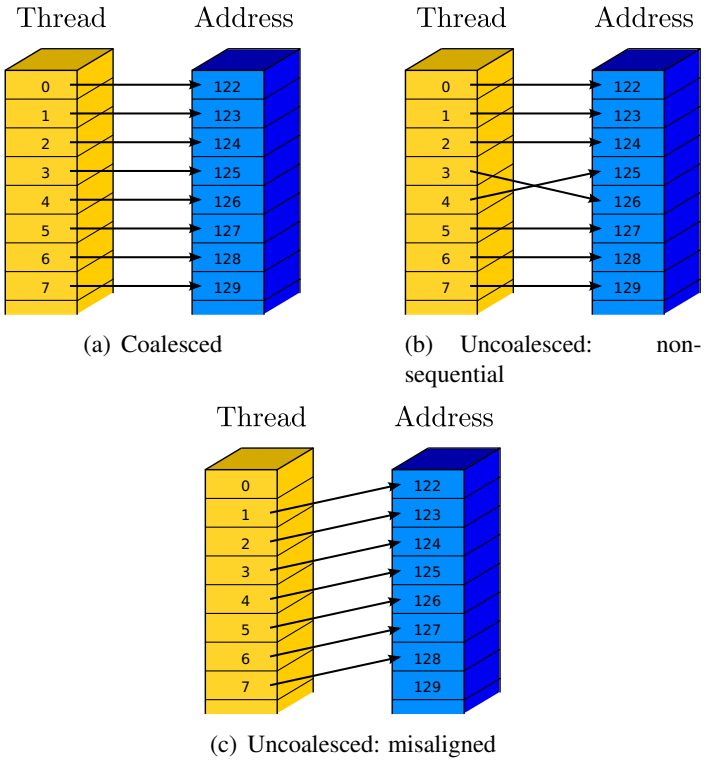
FIGURE 4.9: The mapping of the lattice grid on to the thread grid on the GPU. Notice that each lattie node corresponds with a thread on the thread grid which makes this fine-grained parallelism [10].

4.4.3 Memory Management

With regards to GPU computing, it is necessary to discuss memory arrangement. As mentioned in the previous sections, there are two types of memory: shared memory for intra-block communication and global memory for inter-block communication. Threads are able to efficiently communicate within a block through the use of shared memory, however, inter-block communication is much slower due to its use of global memory. Thus, it is important to limit inter-block communication as much as possible. To give an idea of the importance of proper memory management, optimal memory access patterns can improve the performance by an order of magnitude [40].

One way to do this is to ensure that data are aligned such that the reading and writing of those data is coalesced into a continuous aligned memory access. An example of this and two types of uncoalesced memory access can be seen in Fig. 4.10. As will be discussed in Section 4.5, some amount of uncoalesced data access is unavoidable due to streaming however it is important to limit it.

In addition to memory access, it is also necessary to discuss the way that information is arranged in the model. As shown in Fig. 4.9, the code is parallelized such that each thread corresponds to one spatial location, $f(x)$. This



(a) Coalesced

(b) Uncoalesced: non-sequential

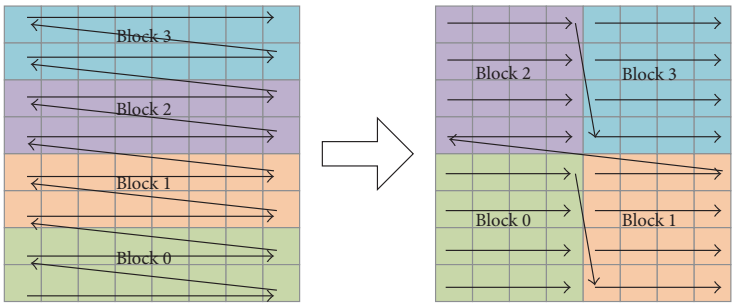
(c) Uncoalesced: misaligned

FIGURE 4.10: The two different types of memory access: coalesced and uncoalesced, where the data of a specific thread is accessed at the address in the GPU memory. Misaligned and non-sequencial memory access can degrade code performance [40].

means that each thread stores the value of f, ρ,u_x,u_y,u_z, δ and whether or not that location is a solid boundary. f is $m+1$ dimensional where it has m spatial dimensions plus an additional dimension corresponding to the streaming directions. For the D3Q19 model, $f(i,j,k,q)$ is 4 dimensional where i,j and k correspond to the x, y and z coordinates and q corresponds to the 19 components of f. It is common practice to flatten multidimensional arrays into a single dimension for memory purposes [40]–[42].

There are two main formats for ordering data in f: Array of Structures (AoS) and Structure of Arrays (SoA). In the AoS arrangement, the q values of the distribution function for each component are stored contiguously for each lattice node. In the SoA arrangement, the $Nx * Ny * Nz$ lattice nodes for each component of the distribution function are stored contiguously, where

Nr is the number of nodes in the r direction. The two arrangements, AoS and SoA, are computationally optimized for different steps of the LBM algorithm, collision and streaming. Hence, while AoS is preferable for CPU implementations, SoA is necessary for better coalesced access to global memory within the GPU [43]. This is because in SoA spatially adjacent nodes are adjacent in memory as well. Additionally, it has been shown that improvements in performance in the streaming step with SoA outweigh performance lost in the collision step when using AoS [42]. It is for these reasons that the ordering of data in this thesis utilized the SoA format.



(a) Original data layout (b) Proposed data layout

FIGURE 4.11: Two different data layout schemes: row major and tiling. While less efficient, the row major scheme is simple and still sufficiently efficient for the purposes of this thesis. For larger simulations, the increased efficiency of the tiling scheme could merit its more difficult implementation [42].

While the AoS vs SoA format addresses how the q components of f are formatted, it does not address how the spatial components of f are organized. This is important because it is beneficial for fewer accesses to global memory to be made meaning that adjacent data should be in the same block whenever possible. For the code presented in this thesis, a rather simplistic row major data layout is used where all nodes in a row belong to the same block. This is a fairly common layout that is used due to its simplicity and efficiency [40], [41]. With a row major data layout utilizing SoA, $f(i,j,k,q)$ is flattened to $f(q * Nx * Ny * Nz + k * Nx * Ny + j * Nx + i)$. It should be

noted, however, that more efficient data layout schemes exist. It was proposed by Tran et al. to switch from a row major scheme to a tiled scheme instead as it proved to be around 28 percent more efficient. The added complexities in implementing the tiling method were outside the desired scope and the performance of the code was deemed sufficient for this work. The row major scheme and tiling scheme are pictured in Fig. 4.11.

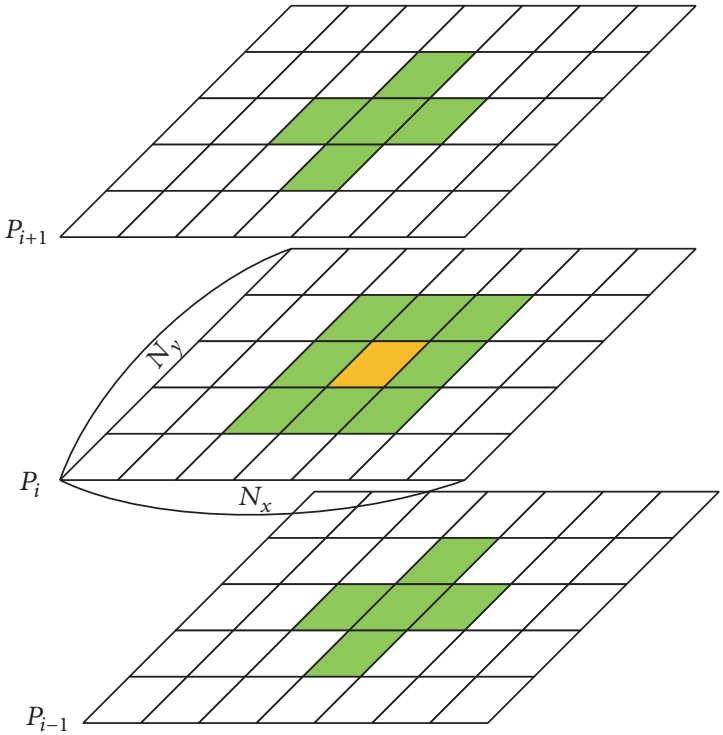


FIGURE 4.12: The data access required for the orange cell during the streaming and collision phases where Nx is the number of cells in the x dimension, Ny is the number of cells in the y direction and P_i is the ith plane of cells, $0 \leq i \leq Nz$ where Nz is the number of cells in the z direction [42].

While the tiling scheme was not used in this thesis, the reasons why it is more efficient are important to gain a better understanding of the LBM using GPU computing. As shown in Eq. 3.36, there are two steps in the LBM, collision and streaming, each representing one side of the equation. The data access required for a single cell is shown in Fig. 4.12. The collision step is local in that $f(l, t + \Delta t)$ depends only on $f(l, t)$ while the streaming step is non-local as can be seen in Eq. 3.36 and discussed in Section 3.3.1. While the

streaming step is indeed non-local, it only depends on cells that are at most one coordinate away in any direction. This means that as long as these cells all belong to the same block, shared memory as opposed to global memory can be used.

4.5 Implementation

With the basics of GPU computing addressed, it is necessary to discuss the implementation of the LBM using GPU computing. While Eq. 3.36 shows the two main steps of the LBM, f^{eq} depends on ρ and $\vec{u}$, hence, another step where these are computed is necessary. As will be discussed in Section 5.2, this code began as a CPU implementation and hence this is a natural place for this discussion to begin. The pseudocode for this algorithm can be seen below:

Algorithm 1 CPU Algorithm

```
1:  for all i, j, k do
2:      Read f_in to f
3:      Calculate f^eq from ρ and u⃗ using Eq. 3.37
4:      Collision between f_in and f^eq
5:      for all q do
6:          Stream f_in(x) to f(x + c_q Δt)
7:      end for
8:      Synchronize across f
9:      Read f to f_in
10:     Apply boundary conditions
11:     Calculate ρ and u⃗ from Eqs. 3.38 and 3.39
12: end for
```

While this algorithm is fine for a serial CPU implementation, there are some locality issues caused by the streaming step that inhibit this algorithm from being performed in a parallel manner. In order for this algorithm to run as written, since a thread operates on each location, the streaming step requires a synchronization across the domain before the boundary conditions

are performed and macroscopic variables are calculated. These synchronizations cause for idle threads, which are to be avoided in trying to achieve better efficiency.

While this synchronization cannot be totally avoided due to the non-local nature of the streaming step, it can be relegated to the end of the kernel where it occurs naturally. This then leads to two reorderings of the algorithm which achieve this: the push and pull algorithm [43]. The main difference between the push and pull algorithms is where the streaming step occurs. For the push algorithm, the streaming step occurs at the end where $f(x,t)$ is pushed to $f(x + c_q\Delta t, t + \Delta t)$. For the pull algorithm, the streaming step occurs at the beginning where $f(x,t)$ is pulled from $f(x - c_q\Delta t, t - \Delta t)$. The two algorithms are given below in Algorithms 2 and 3.

Algorithm 2 Push Algorithm

for all i, j, k **do**
  for all q **do**
    Create a local copy of f: $f_{in} = f$
  end for
  Apply boundary conditions
  Calculate ρ and $\vec{u}$ from Eqs. 3.38 and 3.39
  for all q **do**
    Calculate f^{eq} from ρ and $\vec{u}$ using Eq. 3.37
    Collision between f_{in} and f^{eq}
    Stream $f_{in}(x)$ to $f(x + c_q\Delta t)$
  end for
end for

As mentioned earlier, uncoalesced data can degrade the performance of the code, however, due to the non-local nature of the streaming step some amount of uncoalesced memory access is unavoidable. The difference between the push and pull algorithms then is whether the misaligned data is read to or written from memory. It has been shown on multiple different GPUs that the former is more efficient than the later [41]–[43]. It is for these reasons that a variant of the pull algorithm is used for this thesis, which will be discussed next.

Algorithm 3 Pull Algorithm

for all i, j, k **do**
  for all q **do**
    Stream $f_{in}(x,t)$ from $f(x - c_q\Delta t, t - \Delta t)$
  end for
  Apply boundary conditions
  Calculate ρ and $\vec{u}$ from Eqs. 3.38 and 3.39
  Calculate f^{eq} from ρ and $\vec{u}$ using Eq. 3.37
  for all q **do**
    Collision between f_{in} and f^{eq}
  end for
end for

4.6 LBM Procedure

As discussed in the previous section, the algorithm used for this thesis was based on the pull algorithm. The main difference between the pull algorithm in Algorithm 3 and the one used in these thesis (Algorithm 4) is the relocation of the calculation of the macroscopic variables of ρ and $\vec{u}$. This was done because of the difficulty in performing Eqs. 3.38 and 3.39 in parallel. By placing this step at the end of the kernel, it allowed for an inherent synchronization to occur at the end of the kernel similar to having the streaming step at the end of the push algorithm, seen in Algorithm 2. This however required downloading and uploading ρ and $\vec{u}$ at each time step which reduced the efficiency of the code. This should be avoidable, and reorganizing the algorithm to make achieve this will be left for future work.

Algorithm 4 LBM Algorithm

for all i, j, k **do**
  for all q **do** Stream $f_{in}(x,t)$ from $f(x - c_q\Delta t, t - \Delta t)$
  end for
  Calculate f^{eq} from ρ and $\vec{u}$ using Eq. 3.37
  for all q **do**
    Collision between f_{in} and f^{eq}
  end for
  Apply boundary conditions
  Calculate ρ and $\vec{u}$ from Eqs. 3.38 and 3.39
end for

Overall, the code operates as follows. First, the domain is created and all constants are given their respective values. Next, $\rho, \vec{u}, f$ and f^{eq} are initialized. This is important because an inaccurate initialization can cause instabilities. Since the velocity and pressure fields are homogeneous at the start of simulation, an equilibrium initialization can be applied where $f_q = f_q^{eq}$ at $t = 0$.

At this point, the for loop is entered that iterates over t. Within the loop, the kernel, which is shown in Algorithm 4, is called. As mentioned, at every time step ρ and $\vec{u}$ are downloaded. At certain time steps, these values are written to a file for post processing. This occurs until a set number of time steps is completed.

4.7 Conclusion

While the code and suggestions for implementation thereof for the LBM were presented in several references, there were some specific challenges faced in creating the code. Spurious pressure oscillations at the edge of the domain were fixed through implementing a ghost layer. Further, the stability of the simulation was improved through the use of the MRT scheme. In order to increase the performance of the code, the code was translated from running on the CPU to running on the GPU which required a number of other optimizations as well. Nevertheless, there remain a few challenges to overcome. In Chapter 5, the benchmarking and several results of simulations run in an attempt to accurately model the flute described in Chapter 2 are presented. Then later, these challenges and suggested improvement for future work are discussed in Chapter 6.

Chapter 5

Numerical Analysis of Pipes and Flutes

5.1 Introduction

This chapter will present the results, modifications, and different iterations of the code that have been run throughout this thesis work.

5.2 Evolution of the code

The first iteration of the code was a D2Q9 code written in Matlab. Due to its status as a very high level programming language along with the author's familiarity with the language, it was a deemed a natural starting point for the code. Multiple different simulations were run on this code. These ranged from basic benchmark tests such as Poiseuille flow in a pipe and a lid driven cavity to more advanced simulations such as quiescent flow in a pipe and flow in a flute. These simulations were run using a D2Q9 LBM-MRT model. For the more basic tests, where a high grid resolution and many time steps were not necessary, the Matlab CPU code was more than satisfactory. However, as the simulations became more complicated, the grid resolution had to be increased, leading to more required time steps that caused for the simulation to take a long time to complete. Knowing that this was only a 2D

code and that the ultimate goal of the thesis was to develop a 3D code, it became apparent that the Matlab code would need to be abandoned in favor of Fortran.

The second iteration of the code was subsequently a D2Q9 code written in Fortran. Fortran, due to its nature as a lower level programming language than Matlab, allowed for the simulation time to be reduced for the same parameters, grid size, resolution and time steps. This version of the code was purely a CPU code and many of the same simulations run on the Matlab D2Q9 code were performed on the Fortran code. While converting the code from Matlab to Fortran did provide a marginal speed up in performance, the large grid size and high resolution required meant that additional changes to the code were necessary. Using CUDA Fortran, the code was converted into a Fortran code that ran on the GPU as described in Section 4.4. By utilizing the GPU, the code was sped up by 11 times, which made it so 3D simulations would be feasible. The difference in code speed can be seen in Table 5.1. The table also shows the million lattice updates per second (MLUPS) of both the CPU and GPU codes, which is the way LBM code performance is compared and is defined as,

$$\text{MLUPS} = \frac{Nx * Ny * Nz * N_t}{t_s}, \tag{5.1}$$

where N_t is the number of time steps and t_s is the total time of the simulation. 15.58 MLUPS for the GPU model places it in line with the 18.86 MLUPS achieved by Shi and the 11.23 MLUPS by Kühnelt [4], [10].

	Time per iteration (s)	MLUPS
CPU model	0.0395	1.39
GPU model	0.0035	15.58

TABLE 5.1: Performance of the D2Q9 Fortran CPU model compared to the same GPU model on a 500 x 110 lattice. The 15.58 million lattice updates per second (MLUPS) places it in line with other codes used in musical acoustics [4], [10].

As a 2D CPU code, the change from Matlab to Fortran equated to just

translating the code without the addition of any structural or memory arrangement changes. This allowed for the D2Q9 LBM-MRT code to be created without too many issues. However, when switching from CPU to GPU, the changes made as described in Sections 4.4 and 4.5 meant that the GPU Fortran code remained using the SRT model. Being an upgraded version of the D2Q9 LBM-SRT model on the GPU, the D3Q19 code additionally is a SRT model. It will be subject of future work to upgrade this code to a D3Q19 LBM-MRT model on the GPU. Unless otherwise specified, any reference to code below is referring either to the D2Q9 or D3Q19 LBM-SRT model on the GPU written in Fortran.

5.3 Poiseuille Flow in a Pipe

A common benchmark test for computational fluid dynamic codes is that of Poiseuille flow in a pipe. The simulation itself is relatively simple, consisting of a pipe that is not too wide or short. The Hagen-Poiseuille equation can be derived from the Navier-Stokes equations under the assumptions that the flow is in steady-state, axisymmetric, fully developed and that the radial and azimuthal components of the velocity are zero.

In three dimensions using cylindrical coordinates, the axial momentum becomes,

$$\frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial u}{\partial r}\right) = \frac{1}{\mu}\frac{dp}{dx}, \tag{5.2}$$

where μ is the dynamic viscosity of the fluid. Since the left hand side is a function of r and the right hand side is a function of x, this implies that both sides are equal to a constant. This allows for the equation to be solved as,

$$u = -\frac{\Delta p}{L}\frac{r^2}{4\mu} + A\ln(r) + B, \tag{5.3}$$

where Δp is the change of pressure from the inlet to the outlet of the pipe, L

is the length of the pipe and A and B are constants. Since u exists at $r = 0$, $A = 0$ and the bounce back condition at the walls mean that $u = 0$ at $r = R$ implying that $B = (\Delta p/L)(R^2/4\mu)$ where R is the radius of the pipe. Thus u can be written as,

$$u = \frac{\Delta p}{L}\frac{1}{4\mu}(R^2 - r^2), \tag{5.4}$$

yielding the parabolic velocity profile as expected.

In two dimensions for plane flow in an infinitely long pipe, the Navier-Stokes equations reduce to,

$$\frac{d^2u}{dy^2} = \frac{dp}{dx}\frac{1}{\mu}, \tag{5.5}$$

where dp/dx is a constant. With $u = 0$ at $y = 0$ and $y = 2R$, u can be solved as,

$$u = \frac{-dp}{dx}\frac{1}{2\mu}y(2R - y), \tag{5.6}$$

where h is the height of the tube. For $L >> 2R$, u be approximated as,

$$u = \frac{\Delta p}{L}\frac{1}{2\mu}y(2R - y). \tag{5.7}$$

5.3.1 D2Q9

The Poiseulle flow in a 2D pipe was the first benchmark test performed on the code. The D2Q9 model as described in Section 4.2 was employed with the fully developed flow being shown in Fig. 5.1. The speed is as follows: yellow corresponds to high speed flow to green to dark blue which corresponds with low speed or zero net flow. The yellow and dark blue blocks on the ends of Fig. 5.1 are the inlet and outlet sources.

For this simulation, rather simple velocity boundary conditions were employed. At the top and bottom solid boundaries, which correspond to the walls of the pipe, the no slip boundary condition was imposed. On the

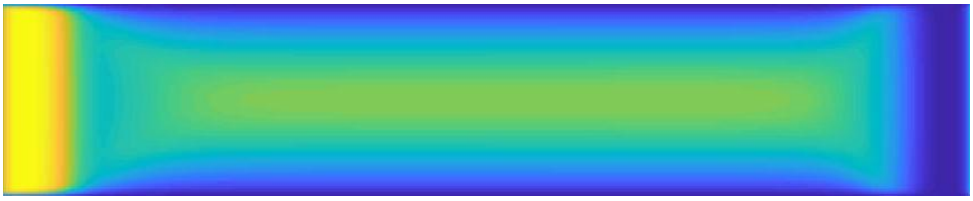


FIGURE 5.1: Poiseuille flow in a pipe from the D2Q9 model where the yellow on the left side indicates the inlet velocity and the blue on the right side indicates the outlet condition of zero flow. Additionally, notice that the velocity at the walls is zero and maximum in the middle as expected.

left and right ends of the simulation, which correspond to the open ends of the pipe, the ABC boundary condition was imposed with a buffer distance, $D = 58$. To start the simulation, a source flow and an outlet flow of zero was initiated through the use of the target velocity, u^T. In terms of the geometry of the pipe, the height was 100 cells and the length was 500 cells which was deemed sufficient so Eq. 5.7 would be valid.

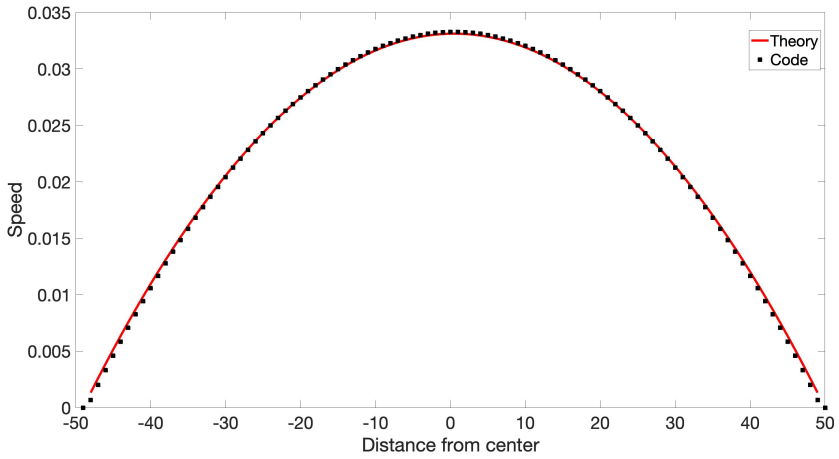


FIGURE 5.2: Comparison of the theoretical results for the Poiseuille flow from Eq. 5.7 and the results from the D2Q9 code. Notice that the simulated results agree well with the theory curve.

At an arbitrary length along the pipe, a vertical slice of the velocity was taken and that velocity compared to the theoretical velocity from Eq. 5.7 is shown in Fig. 5.2. Since the flow is fully developed, the exact place where the slice was taken is not important. As can be seen in Fig. 5.2, the simulation

velocity was in agreement with the theoretical velocity showing the validity of the model.

The numerical error is given by,

$$\text{Error} = \sqrt{\sum_i \sum_j (\vec{u}(i,j) - \vec{u_a}(i,j))^2 / P}, \tag{5.8}$$

where $\vec{u}$ is the velocity from the simulation, $\vec{u_a}$ is the analytical velocity from Eq. 5.7 and the sum is over all P nodes in the domain other than the buffer regions. The calculated error from this simulation was 0.0001, additionally showing the validity of the model.

5.3.2 D3Q19

The Poiseulle flow in 3D cylindrical pipe was the first 3D benchmark test performed on the code. The D3Q19 model as described in Section 4.2.3 was employed. For this simulation, similar conditions to the ones for the Poiseulle flow in a 2D pipe were employed. In terms of the geometry of the pipe, the radius was 40 cells and the length was 500 cells. Due to the rectangular nature of the discretized domain, a simplistic zig-zag scheme was used for the walls. This scheme approximates a circle at larger radii, and the 40 cells was deemed sufficient for this simulation. It is left for future work to implement an interpolation scheme for the solid boundaries. The no slip boundary condition was used for the walls of the pipe. At the open ends of the simulation, the ABC boundary condition was used with a buffer distance of, $D = 58$. A source flow and outlet flow of zero was initiated through the use of the target velocity, u^T.

At an arbitrary length along the pipe with the y-axis set to the center of the pipe, a slice of the velocity was taken and compared to the theoretical velocity from Eq. 5.4. This comparison is shown in Fig. 5.3, and shows that the velocity was in agreement with the theoretical velocity showing the validity

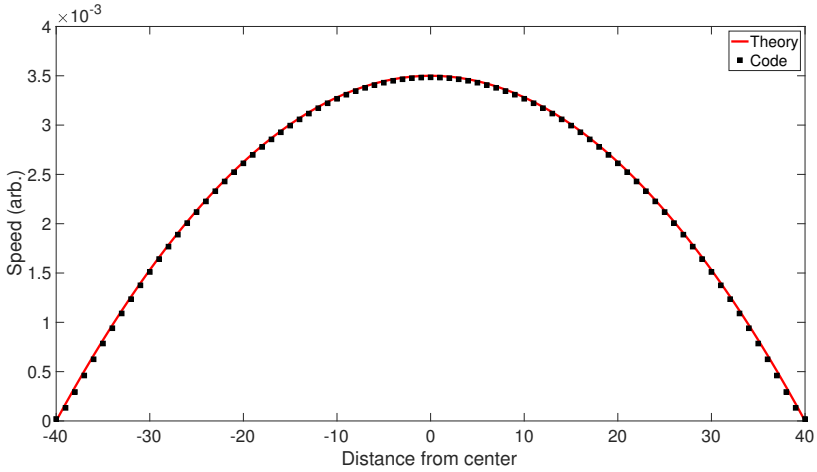


FIGURE 5.3: Comparison of the theoretical results for the Poiseuille flow from Eq. 5.4 and the results from the D3Q19 code. Notice that the simulated results agree well with the theory curve.

of the 3D model. Additionally, at an arbitrary length along the length, a planar slice was taken. This contour plot shows the equivelocity lines from the simulation with yellow corresponding to high speed flow and purple corresponding to low to zero flow, and is shown in Fig. 5.4. It can be seen from the figure that there is high speed flow at the center of the pipe and close to zero flow at the walls of the pipe. Additionally, the concentric circle nature of the equivelocity lines are what is expected from Eq. 5.4.

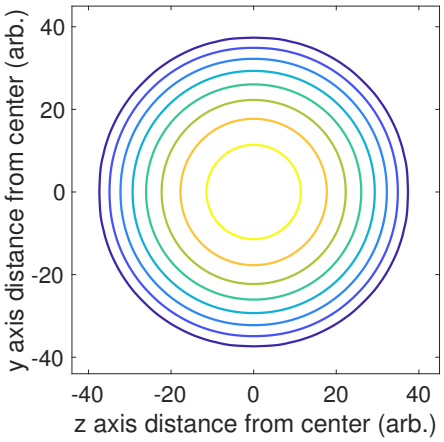


FIGURE 5.4: Equivelocity lines from the results of the D3Q19 code at an arbitrary length. Notice that the concentric circles are what is expected from Eq. 5.4.

5.4 Acoustics of a Pipe with No Mean Flow

As discussed in 2.5, the end of a pipe is an important area of research in musical acoustics. As an interest of the current work is to study the flow at the end of the organ pipe, a tangential objective was to examine the end corrections of a pipe. End corrections are notoriously difficult to determine theoretically leaving most of the work to experimentation. However, these end corrections can also be determined computationally [9], [10]. These results can then be compared to the exact result determined by Levine and Schwinger for a pipe with zero mean flow [22]. Using the D2Q9 model, the goal for this thesis was to show that the propagation of the wave pulse was as expected. The length correction, l, can be determined in the D3Q19 model and compared to the exact result, however, this will be left to future work.

5.4.1 D2Q9

For the D2Q9 simulation, the dimensions were 1000 by 1000 cells with the pipe length equal to 510 cells and the radius of the pipe, a, equal to 10 cells. A free slip boundary condition was applied to the walls of the pipe with the ABC being used for the open boundaries with a buffer width, D, equal to 60 cells. In order to excite the system, a perturbation was implemented in the form of a Hanning impulse as,

$$\begin{aligned} \rho^T &= \rho_0 + \rho' \left(0.5 + 0.5 \cos\left(\frac{2\pi x}{T_0} + \pi \right) \right) \\ u_x^T &= \frac{\rho' c_0}{\rho_0} \left(0.5 + 0.5 \cos\left(\frac{2\pi x}{T_0} + \pi \right) \right), \end{aligned} \tag{5.9}$$

where $\rho_0 = 1$ is the initial density, $\rho' = 0.001$ is the density perturbation and $T_0 = 10$ was the length of the impulse. For $t > T_0$, $\rho^T = \rho_0$ and $u_x^T = 0$.

The use of this smooth impulse is required to reduce the production of high frequency noise which aids in the stability of the simulation [9].

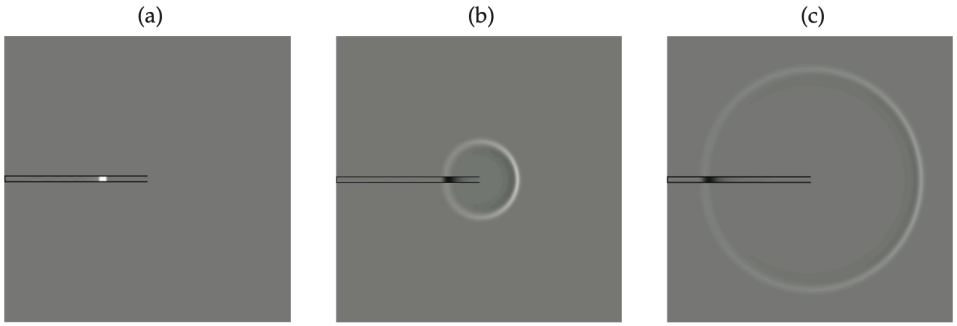


FIGURE 5.5: Snapshots of flow from a pipe with an initial perturbation from [9]. From a to b, the pulse reaches the end of the pipe where part of it reflects and part of it is radiated outwards. From b to c, the pulse continues to radiate outwards and propagate in the pipe.

This is similar to some of da Silva's simulations, with those results shown in Fig. 5.5. The results of this simulation are shown in Fig. 5.6, which show the propagation of the wave pulse at times similar to that of Fig. 5.5. In the first snapshot, the initial propagation is traveling to the right and has yet to reach the end of the pipe. In the second snapshot, some of the pulse has reflected at the end and is now traveling to the left in the pipe while the remainder of the pulse is propagated outside the pipe. In the final snapshot the wave continues to propagate outwards as the reflected pulse continues to travel to the left. A three dimensional model is necessary to accurately

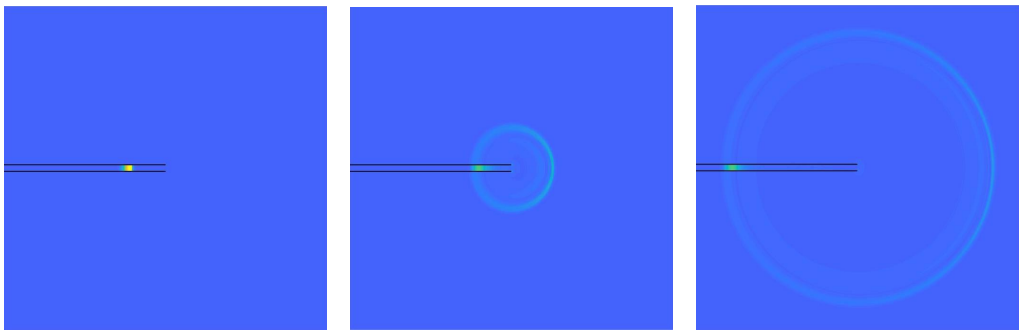


FIGURE 5.6: Results from the D2Q9 code for a pipe with an initial perturbation. Notice that the flow at the three different snapshots are similar to that of 5.5.

calculate the end corrections of a pipe, so these similarities between Figs. 5.5 and 5.6 gave confidence to move on to the D3Q19 model. It will be the subject of future work to calculate the end corrections of the pipe using the D3Q19 model.

5.5 Flow in a Flute

This brings us to the flute. The acoustics of the flute family, of which the organ pipe and recorder are instruments, is discussed more thoroughly in Section 2.3. While a lot of research has been performed both experimental and computationally on the flute, most of this research has focused on the flow around the labium. Subsequent work will study the flow exiting the end of the instrument for different resonator geometries, mainly circular and rectangular flutes. Since these geometries could not be examined by using a two dimensional model, the purpose of the D2Q9 model was to see if the behavior of the flow was as expected. It will left for future work to use the D3Q19 model to examine the flow exiting the instrument for circular and rectangular geometries.

5.5.1 D2Q9

For the D2Q9 model, the dimensions of the simulation were 700 by 1000 cells. The dimensions of the flute are as follows: the length and height of the flue was 84 by 9 cells; the distance between the labium and flue exit was 35 cells; the height of the labium was 16 cells and the angle was 15 degrees; and, the length and height of the resonator was 344 by 52 cells. No-slip boundary conditions were used for the walls of the flute and the ABC was used for the open boundaries with a buffer width, $D = 58$, and an outlet velocity condition of $u_x^T = 0$. Similar to the simulation performed in Section 5.4, a

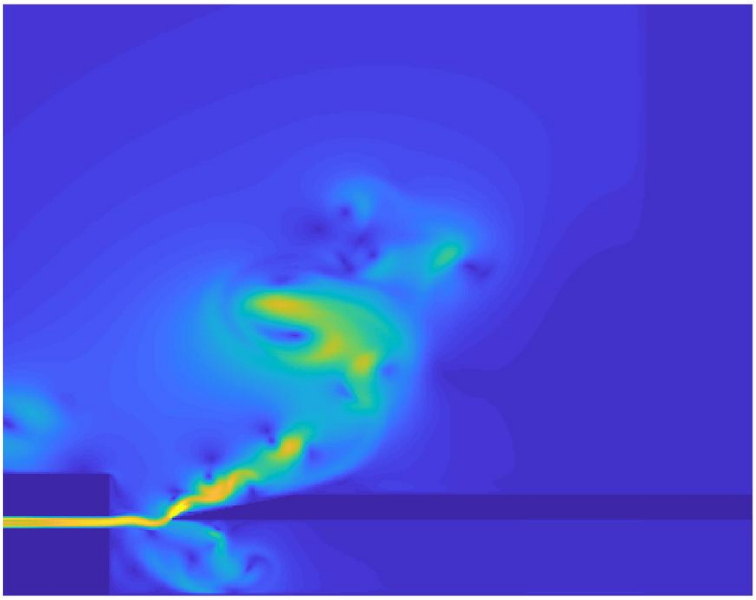


FIGURE 5.7: Flow from a flute from the D2Q9 code. Notice the presence of oscillations and vorticies about the labium.

smooth velocity inlet was used of the form,

$$u_x^T = u_{in} \sin^2(\frac{\pi t}{2T_0}), \tag{5.10}$$

where $T_0 = 5000$ and $u_{in} = 0.071$. The results of the simulation can be seen in Fig. 5.7 where yellow to dark blue represents high to low/zero speed flow. As can be seen in the figure, the flow oscillates above and below the labium as expected and the creation of vorticies can be seen.

5.6 Conclusion

In this chapter, the results from the simulations performed in this thesis were presented along with the different models of the code used. Chapter 6 will describe the improvements possible and ideas for future work.

Chapter 6

Conclusion and Future Research

6.1 Conclusion

In this thesis, a lattice Boltzmann method code was developed from scratch with the intent of studying the flow inside woodwind instruments. The code started in Matlab with a D2Q9 model. As the required domain of the simulations increased, it was realized that it would be necessary to re-implement the code in Fortran. This CPU Fortran code allowed for a marginal speedup ratio but it was still not enough for a 3D model. This led to the final implementation of the code: a D3Q19 GPU CUDA Fortran model.

The simulations reported in this thesis are from the D2Q9 and D3Q19 GPU Fortran code. The first simulation was that of Poiseuille flow in a pipe that was used as a first, easy to implement benchmark test. This allowed for the validity of the code to be established through both qualitative and quantitative means. Moving from here, this simple test of a pipe was extended to an open pipe where the acoustic wave was able to propagate outside the pipe. The results were compared qualitatively to other simulations and followed what experimentally occurs. The final simulations performed were that of an organ pipe. The 2D model simulation results were used as a qualitative assessment of the code.

6.2 Future Work

As referenced in Chapter 1, the work in this thesis represents a starting point that is meant to be built upon. With that in mind, there are numerous improvements and advancements that can be made.

In terms of simulations to be run, there are two simulations discussed in Chapter 5 that can be run without any major changes to the code: 3D end corrections of a pipe and the 3D flute. Calculating the end corrections of a pipe would provide an additional quantitative means to show the validity of the model by comparing the results to those by Levine and Schwinger. Additionally, these results would be relevant because the purpose of the simulations of the 3D flute would be to analyze the flow leaving the end of the flute. This leads us to the second simulation that can be run. There has been some experimental evidence that the flow leaving the end of the flute changes depending on its geometry. One advantage of the LBM is that changing the geometry of what is being simulated does not require extensive changes to the code. This means that flutes of different geometries, rectangular and cylindrical for instance, can be simulated with relative ease. Moving from here, it would also be possible to use this code to simulate the reed-mouthpiece system of the clarinet. This, however, will require some more major changes to the code in the form of moving boundaries among other matters.

Other than additional simulations that can be run on the model, there are improvements that can make the code more efficient and accurate. As discussed, the final D2Q9 and D3Q19 Fortran codes were SRT models. In observing the flow at the end of the 3D flute, it will likely be necessary for the code to be upraded to an MRT model for improved stability. Additionally in considering the flute, it would be beneficial to implement an interpolated bounce back condition at boundaries that fall between nodes. This would also be necessary for the moving boundaries in the case of the reed for the

clarinet. This type of scheme is described in [44]. Both the MRT model and interpolation scheme would help improve the accuracy of the code. In terms of efficiency, there are a few areas where the code can be further optimized to improve its performance. To start, instead of the row major layout that is currently used, the data layout could be switched to a tiling scheme as discussed in Section 4.4.3. Additionally, there are other optimization techniques that were not employed in this thesis that could be considered from [40]–[42]. Finally, the calculation of the macroscopic variables of $\vec{u}$ and ρ could be reworked so that the values did not have to be downloaded and re-uploaded to the GPU at every time step. Of all the optimization proposed here, this last one would lead to the greatest increase in performance.

Bibliography

[1] W. Hui, "Exact solutions of the unsteady two-dimensional navier-stokes equations", *Journal of Applied Mathematics and Physics*, vol. 38, pp. 689–702, 1987.

[2] D. Wang, "Global solutions of the navier–stokes equations for viscous compressible flows", *Nonlinear Analysis: Theory, Methods and Applications*, vol. 52, no. 8, pp. 1867–1890, 2003, ISSN: 0362-546X. DOI: https://doi.org/10.1016/S0362-546X(02)00280-8. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0362546X02002808.

[3] A. Mohamad, *Lattice Boltzmann Method: Fundamentals and Engineering Applications with Computer Codes*, ser. SpringerLink : Bücher. Springer London, 2011, ISBN: 9780857294555.

[4] H. Kühnelt, "Studying the vortex sound of recorder and flute-like instruments by means of the lattice boltzmann method and helmholtz decomposition", PhD thesis, University of Music and Performing Arts Vienna, 2016.

[5] N. Giordano, "Direct numerical simulation of a recorder.", *The Journal of the Acoustical Society of America*, vol. 133 2, pp. 1111–8, 2013.

[6] N Giordano, "Simulation studies of a recorder in three dimensions", *The Journal of the Acoustical Society of America*, vol. 135, no. 2, pp. 906–916, 2014. DOI: 10.1121/1.4861249. eprint: https://doi.org/10.1121/1.4861249. [Online]. Available: https://doi.org/10.1121/1.4861249.

[7] M. Sukop and D. Thorne, *Lattice Boltzmann Modeling: An Introduction for Geoscientists and Engineers*. Springer Berlin Heidelberg, 2007, ISBN: 9783540279822.

[8] P. Skordos, "Modeling flue pipes-subsonic flow, lattice boltzmann, and parallel distributed computers", PhD thesis, Massachusetts Institute of Technology, 1995.

[9] A. R. da Silva, "Numerical studies of aeroacoustic aspects of wind instruments", PhD thesis, McGill University, 2008.

[10] Y. Shi, "A numerical framework for fluid-acoustic-structure interaction in clarinet-like instruments", PhD thesis, McGill University, 2016.

[11] P. Carroll, *Baroque Woodwind Instruments*. Taylor and Francis Ltd, 1999.

[12] Yamaha, *The structure of the clarinet*, https://www.yamaha.com/en/musical_instrument_guide/clarinet/mechanism/, (accessed: 02.17.2018).

[13] J.-P. Dalmont, J. Gilbert, and S. Ollivier, "Nonlinear characteristics of single-reed instruments: Quasistatic volume flow and reed opening measurements", *The Journal of the Acoustical Society of America*, vol. 114, no. 4, pp. 2253–2262, 2003. DOI: 10.1121/1.1603235. eprint: http://dx.doi.org/10.1121/1.1603235. [Online]. Available: http://dx.doi.org/10.1121/1.1603235.

[14] L. Fuks, "Acoustical, physiological and perceptual aspects of reed wind instrument playing and vocal-ventricular fold phonation", PhD thesis, Kungl Tekniska Högskolan, 1998.

[15] W. L. Coyle, P. Guillemain, J. Kergomard, and J.-P. Dalmont, "Predicting playing frequencies for clarinets: A comparison between numerical simulations and simplified analytical formulas", *The Journal of the Acoustical Society of America*, vol. 138, no. 5, pp. 2770–2781, 2015. DOI:

10.1121/1.4932169. eprint: http://dx.doi.org/10.1121/1.4932169. [Online]. Available: http://dx.doi.org/10.1121/1.4932169.

[16] C. Richard, J. Gabriel, and W. Coyle, "Comparison of the vacuum and compression artificial mouth systems in clarinet acoustics", *Student-Faculty Collaborative Research Annual Reports*, 2017.

[17] M. L. Facchinetti, X. Boutillon, and A. Constantinescu, "Numerical and experimental modal analysis of the reed and pipe of a clarinet", *The Journal of the Acoustical Society of America*, vol. 113, no. 5, pp. 2874–2883, 2003. DOI: 10.1121/1.1560212. eprint: http://dx.doi.org/10.1121/1.1560212. [Online]. Available: http://dx.doi.org/10.1121/1.1560212.

[18] P. Dickens, R. France, J. Smith, and J. Wolfe, "Clarinet acoustics: Introducing a compendium of impedance and sound spectra", vol. 35, pp. 17–24, Apr. 2007.

[19] E. M. von Hornbostel and C. Sachs, "Classification of musical instruments: Translated from the original german by anthony baines and klaus p. wachsmann", *The Galpin Society Journal*, vol. 14, pp. 3–29, 1961, ISSN: 00720127.

[20] A. Benade, *Fundamentals of Musical Acoustics*, ser. Dover Books on Music Series. Dover Publications, 1990, ISBN: 9780486264844.

[21] L. L. Beranek, *Acoustics*, ser. Electrical and electronic engineering. American Institute of Physics, 1986, ISBN: 9780883184943.

[22] H. Levine and J. Schwinger, "On the radiation of sound from an unflanged circular pipe", *Phys. Rev.*, vol. 73, pp. 383–406, 4 1948. DOI: 10.1103/PhysRev.73.383. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRev.73.383.

[23] Y. Ando, "Experimental study of the pressure directivity and the acoustic centre of the circular pipe horn loud speaker", *Acta Acustica united with Acustica*, vol. 20, no. 6, pp. 366–369, 1968, ISSN: 1610-1928. [Online]. Available: https://www.ingentaconnect.com/content/dav/aaua/1968/00000020/00000006/art00009.

[24] A. Norris and I. Sheng, "Acoustic radiation from a circular pipe with an infinite flange", *Journal of Sound and Vibration*, vol. 135, no. 1, pp. 85 –93, 1989, ISSN: 0022-460X. DOI: https://doi.org/10.1016/0022-460X(89)90756-6. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0022460X89907566.

[25] R. Munt, "Acoustic transmission properties of a jet pipe with subsonic jet flow: I. the cold jet reflection coefficient", *Journal of Sound and Vibration*, vol. 142, no. 3, pp. 413 –436, 1990, ISSN: 0022-460X. DOI: https://doi.org/10.1016/0022-460X(90)90659-N. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0022460X9090659N.

[26] J. M. Tyler and T. G. Sofrin, "Axial flow compressor noise studies", in *SAE Technical Paper*, SAE International, Jan. 1962.

[27] D. Tong, *Kinetic theory*, Lecture Notes, 2012.

[28] D. A. Wolf-Gladrow, "5. lattice boltzmann models", in *Lattice Gas Cellular Automata and Lattice Boltzmann Models*, Springer, 2000, pp. 159–246.

[29] D. Tong, *Classical mechanics, chapter 4: The hamiltonian formulation*, Lecture Notes, 2015.

[30] S. Harris, *An Introduction to the Theory of the Boltzmann Equation*. Holt, Rinehart and Winston, 1971.

[31] 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, 3 1954. DOI: 10.1103/PhysRev.94.511. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRev.94.511.

[32] W. Kausel, *A Musical Acoustician's Guide to Computational Physics, Concepts, Algorithms and Applications*. Jan. 2003, pp. 123–132, ISBN: 3900914052.

[33] P. Lallemand and L.-S. Luo, "Theory of the lattice boltzmann method: Dispersion, dissipation, isotropy, galilean invariance and stability", NASA, Tech. Rep., 2000.

[34] D. d'Humieres, *Rarefied Gas Dynamics: Theory and Simulation*. American Institiute of Aeronautics and Astronautics, Inc., 1994, pages 450-458.

[35] Y.-H. Qian, D. d'Humières, and P. Lallemand, "Lattice bgk models for navier-stokes equation", *EPL (Europhysics Letters)*, vol. 17, no. 6, p. 479, 1992.

[36] D. d'Humieres, I. Ginzburg, M. Krafczyk, P. Lallemand, and L.-S. Luo, "Multiple-relaxation time lattice boltzmann models in 3d", NASA, Tech. Rep., 2002.

[37] E. W. Kam, R. M. So, and R. C. Leung, "Non-reflecting boundary conditions for one-step lbm simulation of aeroacoustics", in *12th AIAA/CEAS Aeroacoustics Conference (27th AIAA Aeroacoustics Conference)*, 2006, p. 2416.

[38] M. Fatica and G. Ruetsch, *CUDA Fortran for Scientists and Engineers*. Elsevier, 2014.

[39] *Programming guide, version 10.2.89*, NVIDIA CUDA, 2019.

[40] M. Astorino, J. Sagredo, and A. Quarteroni, "A modular lattice boltzmann solver for gpu computing processors", *SeMA Journal*, vol. 59, pp. 53–78, 2012.

[41] M. J. Mawson and A. J. Revell, "Memory transfer optimization for a lattice boltzmann solver on kepler architecture nvidia gpus", *Computer Physics Communications*, vol. 185, no. 10, pp. 2566 –2574, 2014, ISSN: 0010-4655. DOI: https://doi.org/10.1016/j.cpc.2014.06.003. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0010465514002070.

[42] N.-P. Tran, M. Lee, and S. Hong, "Performance optimization of 3d lattice boltzmann flow solver on a gpu", *Scientific Programming*, vol. 2017, pp. 1–16, Jan. 2017. DOI: 10.1155/2017/1205892.

[43] G. Wellein, T. Zeiser, G. Hager, and S. Donath, "On the single processor performance of simple lattice boltzmann kernels", *Computers and Fluids*, vol. 35, no. 8, pp. 910 –919, 2006, Proceedings of the First International Conference for Mesoscopic Methods in Engineering and Science, ISSN: 0045-7930. DOI: https://doi.org/10.1016/j.compfluid.2005.02.008. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0045793005001532.

[44] P. Lallemand and L.-S. Luo, "Lattice boltzmann method for moving boundaries", *Journal of Computational Physics*, 2002.