Informatics

Simulating Chemical Reactions with a Well-Founded Lattice Boltzmann Approach

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Technische Informatik

eingereicht von

Dylan Baumann, BSc
Matrikelnummer 11905166

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Ao.Univ.Prof.Dipl.-Ing. Dr.techn. Andreas Steininger

Wien, 2. September 2024

Dylan Baumann | Andreas Steininger

TU WIEN Informatics

Simulating Chemical Reactions with a Well-Founded Lattice Boltzmann Approach

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Computer Engineering

by

Dylan Baumann, BSc
Registration Number 11905166

to the Faculty of Informatics

at the TU Wien

Advisor: Ao.Univ.Prof.Dipl.-Ing. Dr.techn. Andreas Steininger

Vienna, September 2, 2024

Dylan Baumann Andreas Steininger

Erklärung zur Verfassung der Arbeit

Dylan Baumann, BSc

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.
Ich erkläre weiters, dass ich mich generativer KI-Tools lediglich als Hilfsmittel bedient habe und in der vorliegenden Arbeit mein gestalterischer Einfluss überwiegt. Im Anhang „Übersicht verwendeter Hilfsmittel“ habe ich alle generativen KI-Tools gelistet, die verwendet wurden, und angegeben, wo und wie sie verwendet wurden. Für Textpassagen, die ohne substantielle Änderungen übernommen wurden, habe ich jeweils die von mir formulierten Eingaben (Prompts) und die verwendete IT-Anwendung mit ihrem Produktnamen und Versionsnummer/Datum angegeben.

Wien, 2. September 2024

Dylan Baumann

Danksagung

Diese Abschlussarbeit war für mich in vielen Belangen äußerst herausfordernd und hätte ohne der Unterstützung einiger Personen wohl niemals in dieser finalen Form geendet. An erster Stelle möchte ich mich bei meinem Betreuuer Andreas bedanken, der es erst ermöglicht hat mich mit diesem spannenden Thema zu beschäftigen und stets ein offenes Ohr für meine Anliegen hat. Obwohl einiges an dieser Arbeit für ihn genauso neu war für mich, konnte er mir stets mit aufschlussreichen Diskussionen helfen mich auf die wichtigen Themen zu fokussieren, und mich nicht auf Nebenschauplätzen zu verlaufen Sein aufmerksames Lesen und wertvolles Feedback haben der Arbeit zu ihrer finalen Form verholfen - in der den Leserinnen einige äußerst holprige Formulierungen und peinliche Fehler erspart werden.

Größten Dank und Anerkennung möchte ich hiermit auch meinen beiden Mit-Betreuuern Matthias Függer und Thomas Nowak aussprechen. Nicht nur haben sie das Thema erst angeregt, sondern mich mit viel Geduld und stetigem Nachbohren, wenn ich etwas nicht befriedigend beantworten konnte, auch auf den richtigen Weg gebracht, ohne mich jemals von meinen eigenen Ideen und Ansätzen abzubringen. Ohne der vielen Diskussionen mit den beiden hätte ich mich in dem für mich neuen Gebiet schlussendlich wohl völlig verirrt. Außerdem möchte ich an dieser Stelle voller Ehrfurcht auf den Einfallsreichtum und die beinahe hellseherischen Fähigkeiten der beiden hinweisen. Es hat viel Aufwand, und zahlreiche *Aha*-Momente über Monate hinweg gebraucht, um zu einigen Erkenntnissen und Schlussfolgerungen zu gelangen, die die beiden fast von Beginn an vermutet haben. Ein derart intuitives und dennoch pragmatisches Verständnis, wie es die beiden an den Tag legen, ist mir bisher nur selten begegnet.

Schlussendlich möchte ich mir hiermit auch bei meinem engsten Familien- und Freundeskreis bedanken. Während meiner Beschäftigung mit dieser Arbeit und anderen zeitgleichen Herausforderungen, habe ich viel zu selten und viel zu wenig die Zeit gefunden die euch eigentlich gebührt. Dennoch habe ich zu keiner Zeit an eurer vollständigen, bedingunslosen, Unterstützung gezweifelt. Meiner Dankbarkeit darüber kann ich hier mit ein paar Worten nicht gerecht werden.

Acknowledgements

This thesis was in many aspects extremely challenging for me and would never have ended up in this final form without the support of a few people.
First and foremost, I would like to thank my supervisor Andreas, who made it possible for me to engage with this exciting topic in the first place, and always had an open ear for my concerns. Although some of this work was just as new to him as it was to me, he was always able to help me focus on the important topics with insightful discussions and not get lost in side issues. Furthermore, it was only his attentive reading and valuable feedback that helped the work to its final textual form, without which readers would have to deal with some very clumsy formulations and embarrassing mistakes.

I would also like to express my sincere thanks and appreciation to my two co-supervisors Matthias Függer and Thomas Nowak. Not only did they propose the topic in the first place, but they also put me on the right track with a lot of patience and constant probing when I couldn't answer something satisfactorily, while never dissuading me from my own ideas and approaches. Without the many discussions with the two of them, I would probably have ended up completely lost in this new area for me. I would also like to take this opportunity to point out the ingenuity and seemingly clairvoyant abilities the two of them possess. It took a lot of effort and numerous eye-opening moments over a period of months, to arrive at some of the insights and conclusions that the two of them suspected almost from the get-go. I have rarely come across such an intuitive, yet pragmatic, understanding as the two of them display.

Finally, I would also like to thank my closest family and friends. During my occupation with this work, and other simultaneous challenges I faced, I have found time to spend with you far to seldom, and if I did, it was only a shadow of what you would deserve. Nevertheless, I have never doubted your complete and unconditional support, and I am endlessly grateful for that.

Kurzfassung

Ein bemerkenswerter Simulator in der numerischen Strömungsmechanik ist die Lattice-Boltzmann-Methode (LBM). Anstatt die partiellen Differentialgleichungen, die das makroskopische Verhalten von Flüssigkeiten beschreiben, direkt mit numerischen Algorithmen zu lösen, erfasst die LBM die Auswirkungen der Wechselwirkungen zwischen den mikroskopischen Partikeln einer Flüssigkeit auf der Ebene der statistischen Mechanik. Das makroskopische Verhalten des betrachteten Fluids ergibt sich dann daraus. Das macht diesen Ansatz auch für andere Probleme interessant, bei denen sich ein komplexes makroskopisches Verhalten aus unzähligen mikroskopischen Wechselwirkungen ergibt.

Ein solches Problem ist die Simulation des dynamischen Verhaltens von biologischen Systemen, welche möglicherweise auch Bakterien enthalten. In dieser Arbeit werden wir erste Untersuchungen zur Verwendung der LBM für solche System durchführen. Bakterien und andere chemische Verbindungen werden dabei auf Populationsebene mit der LBM simuliert. Insbesondere interessieren wir uns für die Simulation der Auswirkungen von Diffusion, Reaktion und Advektion auf die verschiedenen chemischen/biologischen Spezies für Systeme die mehrere unterschiedliche Spezies beinhalten.

Wir werden uns auf die Untersuchung und das Verstehen der bisher in diesem Bereich durchgeführten Forschung konzentrieren, mit besonderem Augenmerk auf die getroffenen Annahmen und darauf, ob diese mit jenen der LBM übereinstimmen. Wir werden für die Hauptkomponenten der LBM, insbesondere die stationäre End-Verteilung der Partikel, das Raster auf dem simuliert wird, sowie den BGK-Kollisionsoperator, eine entsprechende Analyse zur Verfügung stellen. Darüber hinaus werden die Ergebnisse einer Untersuchung über die Anwendbarkeit des BGK-Operators in Multispezies-Szenarien unter Verwendung eines Molekulardynamik-Simulators vorgestellt.

Außerdem werden wir einen Open-Source, LBM-basierten Python-Simulator entwickeln und bereitstellen, der für weitere Forschungen zu diesem Thema genutzt werden kann. Unser Fokus bei diesem Simulator liegt auf der Zugänglichkeit und Erweiterbarkeit, so dass dieser auch ohne tiefgreifende Kenntnisse über die LBM einfach genutzt und in Zukunft erweitert werden kann. Besondere Eigenschaften des Simulators sind die Fähigkeit, sowohl vollständig einheitsbezogene als auch nicht-dimensionale Simulationen durchzuführen, sowie die Unterstützung für mehrere verschiedene Spezies und Reaktionen zwischen diesen Spezies. Die Reaktionsgleichungen können als chemisches Reaktionsnetzwerk spezifiziert

werden. Die korrekte Implementierung des Simulator wird durch das Nachstellen von Szenarien aus der Literatur nachgewiesen.

Abstract

The lattice Boltzmann method (LBM) is a remarkable computational fluid dynamics simulator. Instead of directly using numerical algorithms to solve the partial differential equations that govern the macroscopic behavior of fluids, it captures the effects of interactions between a fluid's microscopic particles on a level of statistical mechanics, with the macroscopic fluid quantities emerging as a result. The approach lends itself to other problems where a rich macroscopic behavior arises from a myriad of microscopic interactions.

One such problem is the simulation of the dynamic behavior of biological system, possibly containing bacteria. In this work we will conduct preliminary investigations into using the LBM for such cases, where bacteria and other chemical compounds are simulated at the population-level using the LBM. In particular, we will be interested in simulating the effects of diffusion, reaction and advection due to a bulk fluid's flow on the different chemical /biological species in a multi-species scenario.

Our focus will be on studying and understanding the research conducted in this area so far, with particular attention to the assumptions made and whether they align with those imposed by the LBM itself. We will investigate the principal components of the LBM, specifically the equilibrium distribution, lattice, and the BGK collision operator, and provide a corresponding analysis. In addition to that, results of an investigation about the feasibility of the BGK operator in multi-species scenarios using a molecular-dynamics simulator will be presented.

Furthermore, we will develop and provide an open-source, LBM-based Python simulator that can be used for further research on this topic. Our focus for this simulator will be on accessibility and expandability, such that it can easily be used without in-depth knowledge about the LBM and be extended in the future. Noteworthy features of the simulator are the capability to perform fully unit-afflicted, as well as non-dimensional, simulations, along with support for multiple distinct species and reactions between them, specified as a chemical reaction network. Furthermore, the correctness of its implementation will be validated by reproducing results from literature.

Contents

CHAPTER 1

Introduction

1.1 Motivation

The Lattice Boltzmann Method (LBM) is a relatively new approach to the simulation of fluid flows. In comparison to other, alternative, simulation schemes that are applied to numerically approximate such flows, it features several properties that make it stand apart.
Arguably the most notable one is its theoretical basis. Whereas a majority of the simulators used in the field of Computational Fluid Dynamics (CFD) are, in first instance, just numerical solvers for generic partial differential equations (PDE)s [1], the LBM specifically originates from *microscopic* fluid models [2], being based on the kinetic theory of gases. This allows it to operate on a completely different scale than, for example, finite element, difference or volume methods. Rather than directly simulating the macroscopic fluid quantities, the LBM operates on the *mesoscopic* populations, which we can consider as sets of fluid particles.
The method originating from microscopic and operating on mesoscopic models, comes with two paramount consequences:

- Performing simulations with the LBM can potentially provide insights into the behavior of the considered fluid on a level of the populations.
- There are systems which might be easier to model on a microscopic, rather than macroscopic, level. The LBM can, with modifications, potentially be used in such cases as well, as its core functionality originates from microscopic models.

Furthermore, in its basic form, the core of the LBM is almost embarrassingly parallel. Thus, it greatly benefits from the current development towards ever-increasing concurrency in general purpose hardware like GPUs.

Additionally, while the LBM was initially created as a numerical solver for fluid flows of a single type of particle (a *species*), over the past four decades researches worked on applying the method to fluids consisting of multiple species as well, in particular also for fluid flows of *reacting* chemical species [3]. These advancements attract attention from different scientific fields, leading to usage of this CFD method for the treatment of exciting new problems.

However, while providing fertile ground for new insights and discoveries, this interdisciplinary usage of the LBM also bears risks While the core ideas stayed the same, most of the simulation scheme's properties and prinicpal components where thoroughly analyzed, refined and extended since them being initially reported. In fact, the whole LBM was initially not even considered as a dedicated CFD scheme but rather as a patch for another method [4].
Therefore, researches from fields foreign to fluid dynamics are, ideally, initially required to properly acquaint themselves with the method, including not only its capabilities but also its limits and especially caveats. This is especially important since the LBM can produce surprisingly convincing results in the event of a wrong implementation or simulation setup (as briefly illustrated in Appendix H of [5]). With the scheme being a numerical solver, naturally used on problems where it is infeasible or possibly impossible to compute exact solutions, this makes it particularly hard to confirm the soundness of results.

Nevertheless, a particularly interesting application of the LBM could be the simulation of biological systems where bacteria inhabit a flowing fluid in which, for example, nutrients, or attracting or toxic compounds are dissolved.
While such problems quite naturally require a CFD solver, classical, generic, numerical schemes that are based on numerically solving the macroscopic laws governing fluid dynamics, are hardly equipped to simulate microorganisms and their behavior as well. However, since nowadays more-and-more is known about the ways certain bacteria act [6], the LBM appears to lend itself to such scenarios.
And indeed, while rare, there are reports of such applications in the context of bacterial chemotaxis [7, 8] or bacterial growth [9], showing some first, promising, results.
But, also a simulation of simpler biological systems, without bacteria, that supports enzymatic reactions could already provide useful insights during, for example, the prototyping of biological circuits, which can operate using these kinds of reactions [10]

1.2 Aims and Contribution

Our primary goal in this work is to provide a basis for harnessing the LBM for future work on biological fluid systems, consisting of multiple distinct species that react with each other and are potentially subject to a fluid flow. In particular, is to elaborate novel insights into how the LBM can be used to simulate the behavior of bacteria on a population scale inside a stirred bio-reactor.

As a means to achieve this, we will provide an introduction to the field of CFD and, naturally, the LBM. We will further review the existing literature on the topic, and also the evolution of the LBM as a solver for chemically reacting flows, and provide a concise and insightful discussions of the studied literature. We expect two things from this:

- An intuition into how the LBM can be modified to accommodate for the distinct properties of reacting species
- Insights into the validity of the schemes reported in literature, in particular if assumptions inherent to the LBM are complied with

A second artifact of this work will be an open-source implementation of the LBM, which we want to use as a baseline for future work on this topic. The primary focus of this simulator will be on its correctness and adaptability, rather than its performance. To achieve this, we want it to be simple to create new simulations using our implementation, without requiring in-depth knowledge about the LBM. Furthermore, we will validate our implementation on different benchmarks and reproduce results from literature in order to provide a measure of its correctness.

1.3 Outline

We will start by introducing some preliminaries in Chapter 2. This includes the basic problem around which computational fluid dynamics revolves, as well as the lattice Boltzmann method and related concepts.

In Chapter 3 we will then, based on related work, provide a discussion about what researchers typically do when simulating chemical reactions. In particular, we also cover applications of the lattice Boltzmann method to biological system involving bacteria.

After the familiarization with the underlying concepts and the state-of-the-art, we will dive deeper into the lattice Boltzmann in Chapter 4, where we will investigate how to introduce species-specific information into the components of the simulation. Furthermore, we will explore how the particle collision operator that is typically used in the lattice Boltzmann method fares in the case of multiple distinct species.

We will then introduce and discuss our implementation of a simulator based on the lattice Boltzmann method, and how we validated it, in Chapter 5.

Finally, we will provide some concluding thoughts and an outlook to future work based on this thesis in Chapter 6.

CHAPTER 2

Preliminaries

Before we are able to dive into this thesis, we need to define the problem of computational fluid dynamics (CFD) and related physical concepts in general, as well as the lattice-based approaches developed to approximately solve it.
Naturally, special focus will be put on the lattice Boltzmann method, as this works revolves around it. In addition to that, we will briefly introduce some concepts of molecular reaction dynamics and systems (advection) reaction diffusion equations. While this chapter aims to cover most of the concepts required to understand this work, all in all it can only be a mere introduction and does by no means compensate studying the literature dedicated to the respective topics, such as [1, 2, 11, 12].

2.1 Notation and Used Symbols

Reading the popular literature soon leads to the impression that there are as many notations as there are books about the LBM and related topics. This work uses the symbols and notation where a majority of the considered literature appears to agree on, except for a few cases where it leads to, in the author's opinion, enhanced clarity and consistency. In cases where such a majority does not exist, the focus is on a notation that is consistent with the remainder of this work.
Readers can find the used symbols, as well as their respective meaning, in Appendix A. Be aware of the *Depends* column of the table of the physical quantity symbols. In order to keep many of the used equations concise and readable, the explicit dependency of physical quantities on other quantities is not shown, if it is clear from the context. However, for completeness' sake said column will express all such dependencies explicitly. For example, most of the time we will write ρ instead of $\rho(\mathbf{x}, t)$ for the position- and time-dependent density.
A notational decision made for this work, that might appear self-inconsistent at first glance, is the one to switch between the Einstein index and the typical vector notation.

As the vector notation is likely more familiar to potential readers, coming from the field of computer engineering (or related fields), it is thus used when introducing important equations. However, the Einstein index notation allows expressing many of the encountered terms and equations in a very concise way, thus simplifying their handling during derivations. It will thus be used on such occasions, and we will dedicate the Greek letters $\alpha, \beta, \gamma, \delta$ to be the respective indices. Readers unfamiliar with this notation might find the introduction provided by Krüger *et al.* in the appendix of [1] useful.
Note that we print vectors in boldface and that latin letters are used to denote their components. For example, the fluid velocity vector field will be referred to as $\mathbf{u} = (u_x, u_y)$. Note how u_x, u_y are not in boldface as they correspond to simple scalars.

2.2 Computational Fluid Dynamics

The general objective of the field of CFD is to use computers to simulate how the dynamic macroscopic properties, such as the density or the velocity vector field, of a fluid evolve. During this work, we consider a *fluid* to be a continuous blob of matter, consisting of distinct particles. We will call each unique type of particle in a fluid a *species*. However, until stated otherwise, we will assume our fluid to consist of a single species only. Furthermore, as defined by Batchelor in [13], our fluids "cannot withstand any tendency by applied forces to deform it in a way which leaves the volume unchanged" (in Batchelor's terminology we would thus consider *simple* fluids). In this sense, gases and liquids are captured by our definition of a fluid, but solids are not (as they resist deformation due to applied forces to some extent). As a result, our fluids can *flow*, i.e. deform in a continuous way.
Careful readers might notice the apparent contradiction between a fluid being continuous and it consisting of distinct particles. However, typically we are interested in quantities of a fluid on a much larger scale than the one used to describe its molecular structure. Due to this we can apply what Batchelor calls the *continuum hypothesis*, i.e., on a macroscopic level we assume our fluids to behave as if they were continuous due to the shear amount of particles in them.
As will become clear in the following, a key property of the LBM is to consider sets of particles, described using statistics, to finally predict the behavior of the fluid on a macroscopic level.

2.2.1 Macroscopic description

Consider a fluid that flows with a macroscopic velocity $\mathbf{u}$ ($\mathbf{u}$ is a time and position dependent vector field). If we look at a specific volume of this fluid, and consider the change of mass in it (i.e., the amount of particles), it is clear that the mass can only change due to a flow of particles into, or out of, the volume (without any flow the volume's mass must be conserved as we do not consider any reactions or external forces for now). The fluid flow thus only *redistributes* the particles. As a consequence, we can formulate the Continuity Equation (CE) (shown in Equation (2.1)), which essentially states that changes of the fluid density, ρ, at location $\mathbf{x}$ over time t, only happen due to in- and outflow of the fluid with the fluid velocity $\mathbf{u}$.

Continuity Equation

$$\partial_t \rho + \nabla (\rho \cdot \mathbf{u}) = 0 \tag{2.1}$$

In addition to the conservation of mass, an enclosed fluid must also conserve its momentum, which leads to the famous Navier-Stokes Equations (NSE) (depicted in Equation (2.2)). Dropping the terms describing the effects of viscosity (the ones containing the shear, respectively bulk, viscosity coefficients η_s and η_b) yields the Euler equation (c.f., (B.1)), which we will also encounter during this thesis.

Navier-Stokes Equation (force-free) as presented in [14]

$$\rho \partial_t \mathbf{u} + \rho(\mathbf{u}\nabla)\mathbf{u} = -\nabla p + \eta_s \nabla^2 \mathbf{u} + \left(\eta_b + \frac{\eta_s}{3}\right) \nabla(\nabla \mathbf{u}) \tag{2.2}$$

As stated in [1] and [2], only in particular cases it is possible to analytically solve these equations in order to determine the macroscopic behavior of a fluid. Therefore, in general, numerical methods have to be used to obtain approximate solutions.

2.2.2 Top-Down Simulations

Equations (2.1), (2.2) are both PDE. Thus, a rather natural approach to approximate solutions is applying a conventional numerical scheme for generic differential equations. The finite difference, volume or element methods are such schemes. What they all have in common is the general idea of discretizing the continuous PDEs and then expressing the original problem as a system of linear equations. Due to this shared idea of taking the macroscopic NSE and CE as starting-point, Wolf-Gladrow classifies them all as being *top-down* [2]. However, while these methods are comparably simple to apply, they all come with their own set of disadvantages as outlined in detail by Krüger *et al.* [1]. We will not further consider such top-down simulation schemes.

2.2.3 Bottom-Up Simulations

A conceptually entirely different approach to simulating the macroscopic properties of fluid flows is what Wolf-Gladrow calls the *bottom-up* approach. Instead of starting from the macroscopic equations of fluid dynamics, one creates an abstraction of the considered fluid's particles and their interactions. Then, these models are used in a simulation in the hope that a macroscopic behavior similar to that of the real fluid develops.
The fundamental idea behind this approach is conceptually as simple as it is elegant. Let us assume that the complex behavior of real fluids is a result of the interactions between distinct particles, and that the macroscopic properties merely arise out of an

unimaginably high number of such interactions (c.f., Section 2.5.1). Then, bottom-up approaches essentially try to mimic the behavior of real fluids using suitable abstractions. If the abstractions are simple enough to allow the simulation of many particles, yet complex enough to capture the bare essence of the physical processes involved, it could potentially approximate how a real fluid behaves.

It is obvious that there must exist a trade-off between the chosen level of abstraction for the particles and their kinetics, and the number of distinct particles one can capture in a simulation. However, there is another trade-off factor involved as well. Ultimately, computers cannot handle continuous quantities like physical space, time and velocity. Therefore, the mentioned quantities must be discretized to a certain extent, which influences the accuracy and performance / memory requirements of such bottom-up simulations.

2.3 Molecular Dynamics

The, so-called, Molecular Dynamics (MD) simulators try to model particles and their interactions in a rather accurate manner (in comparison to other bottom-up simulation schemes). Individual particles typically feature a position and velocity that is as close to the real-world continuous quantities as the machine-precision allows it.

Under the common assumptions that particles behave like hard spheres (with a species-dependent mass and size), only interact via non-reactive collisions and that collisions are elastic (the total kinetic energy of the involved bodies is the same before and after a collision), the evolution of the simulated fluid is driven by collisions between particles only. It is further assumed that binary collisions (i.e., between exactly two particles) dominate and that higher order collisions can be neglected. This assumption implies that we are considering a rarefied gas.
An obvious bottleneck of such MD simulators is finding pairs of particles that will collide, since one needs to check for all particles whether there are particles in their vicinity that could lead to a collision during the next time step. However, while this can be overcome by using a suitable data-structure (e.g. an oct- or quad-tree), such simulators are, on personal computers, still restricted to thousands of particles [15] and limited simulation time [2] due to the (computational) complexity of the individual collisions. However, note that the advancements in machine learning over the past decade allow it to simulate around 100 million particles for times of around 1ns on supercomputers [16, **?**].

2.4 Lattice-Gas Cellular Automata

A type of bottom-up simulator that discretizes the physical space, time and velocity space significantly more than MD simulators do, are so-called Lattice-Gas Cellular Automata (LGCA). These special kinds of Cellular Automata (CA) were introduced to the field of CFD by Hardy, Pomeau and de Pazzis in [17]. Since the LBM is an offspring of these automata, and understanding them is valuable for grasping the LBM, we will now spend some time on the principle LGCAs.

Like any other CA, they consist of

- a lattice of nodes
- an adjacency relation between nodes (we will also call adjacent nodes neighbors)
- a set of node states
- a set of update rules, defining how the state of a node changes depending on the own state and that of its neighbors

For LGCA the lattice nodes correspond to points in discretized physical space and each node can be occupied by one particle *per* adjacent node. This particle occupation information (for m neighbors m bits) marks the state of a node, plus some additional information that allows to indicate whether it is a solid node (necessary for boundary conditions). We will encode the particle occupation information of a node at position $\mathbf{x}$ and time t as a binary vector $\mathbf{n}$.
In each simulation time step, particles are *streamed* to the respective neighbor node, according to their respective lattice velocity vector. Thus, the assignment of particles to adjacent nodes corresponds to mapping them to discrete velocity vectors (particles are moved by the distance between neighbors per time step width). We will call these vectors the *lattice velocity vectors*. As a result, LGCA explicitly discretize the velocity space, physical space and time. Observe how a (small) set of possible lattice velocity vectors implies that there is only a comparably small number of possible binary (and ternary) collisions. This allows for a significantly less complex handling of particle collisions than in MD schemes.

As outlined by Wolf-Gladrow, not every CA is a LGCA [2] (rather, LGCA are a proper subset of CA). The reason being that we demand certain properties of a cellular automaton in order for it to recover the macroscopic properties of the fluid it models. One such property is the conservation of mass and momentum. This is governed by the macroscopic equations, introduced in Section 2.2. Naturally, the LGCA's behavior should adhere to them. Furthermore, we require the update rules of an LGCA to be reversible in order to apply the proof technique that will allow us to show that it can recover the desired macroscopic equations.

Another, rather conceptual, difference between CA and LGCA is that one typically splits the update rule into two separate steps, named the *collision* and *streaming* steps. The collision step updates each node's state based on its current state (i.e. particle occupancy information), modeling the collisions between particles at the corresponding position (or rather volume around this position). Note that this step can be performed at each node completely independent of all other nodes, which invites massively parallel implementations.

The streaming step models the movement of particles in space along the direction of their respective lattice velocity vector. For the automata this means that particles are moved from nodes to adjacent nodes. This step is therefore non-local, but simple and can thus be implemented efficiently.

We can write the whole update procedure concisely as

$$\mathbf{n}_i(\mathbf{x} + \mathbf{c}_i \Delta t, t + \Delta t) = \mathbf{n}_i(\mathbf{x}, t) + \Omega_i(\mathbf{x}, t) \tag{2.3}$$

where $\mathbf{c}_i$ is the lattice velocity vectors towards a node's i-th neighbor, Δt is the time by which a simulation step advances and Ω is the so-called collision operator that contains the collision rule. It returns a value out of the set $\{-1, 0, 1\}$, depending on $\mathbf{n}_i(\mathbf{x}, t)$, and is typically implemented as a matrix where each entry contains the probability of a state transitioning to another state. The matrix thus has the dimensions $2^m \times 2^m$ (recall that m is the amount of state bits per node).

Note that the magnitude $\mathbf{c}$ of the velocity vectors depends on the distance, Δx, between adjacent nodes and the time step. However, often all simulation quantities are normalized such that $\Delta t = \Delta x = \mathbf{c} = 1$. As long as the *relations* between all parameters and quantities remain the same, this normalization does not change the simulated fluid flow.

Example: The FHP LGCA

To illustrate this type of simulator, let us consider an example LGCA. While the automaton introduced in [17] (called the *HPP* model after the authors) is arguably the simplest one, it fails to macroscopically recover the desired NSE [18]. We will therefore consider the refined automaton reported by Frisch, Hasslacher and Pomeau in [18] (called the FHP model), which uses a slightly more complex lattice but can be shown to macroscopically result in the NSE.

The lattice used in the FHP model is a hexagonal one, where each node is connected to six other nodes via velocity vectors of equal length. Thus, each node can hold up to six particles (the authors also report a version with a seventh "rest" particle that is not streamed). Figure 2.1 shows a fragment of such a lattice.

The square symbols highlight the lattice nodes, with the center node being shown in greater detail. A solid arrow head represents a particle being assigned to the respective lattice velocity vector (edges between the nodes) and an empty one a vacant particle slot. The directions of the arrows coincide with the directions of the respective lattice velocity vectors, which are

$$\mathbf{c}_i = c \cdot \left(\cos\left(\frac{i \cdot \pi}{3}\right), \sin\left(\frac{i \cdot \pi}{3}\right)\right) \tag{2.4}$$

where c is the lattice velocity, i.e., the magnitude of the lattice velocity vectors.

For the shown lattice, Frisch, Hasslacher and Pomeau define two types of collisions. Two-particle head-on collisions happen for node states where exactly two opposing particle

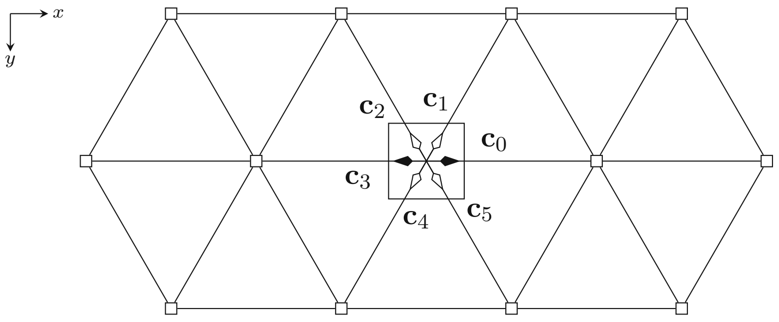


Figure 2.1: The hexagonal FHP lattice

slots are occupied, i.e., for states where it holds that

$$\exists! i : n_i = n_{i+3} = 1 \land \sum_{j=0}^{Q-1} n_j = 2, \quad i \in \{0, \dots, 4\}$$

with Q being the cardinality of the set of lattice velocity vectors (in this example $Q = 5$). Figure 2.2 illustrates the two-particle collision rules defined in [18].

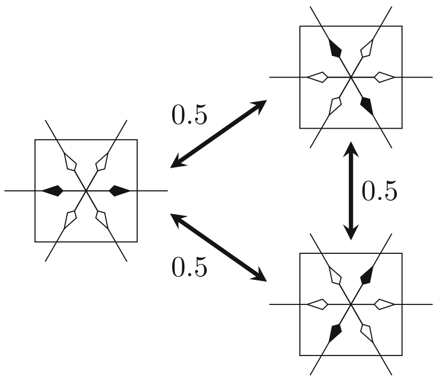


Figure 2.2: FHP two-particle collision rules

Note that a specific collision state does not map to a unique other state, but rather two both-possible post-collision states with an equal probability of 0.5. This is necessary to mitigate an invariant of the modelled fluid which real fluids do not possess (more details can be found in [2]). All other states of a node, where exactly two particles are occupied, are left unchanged by this collision rule.

The other collision rule, illustrated by Figure 2.3, is for the symmetrical three-particle collisions marked by states satisfying

$$\exists! i : n_i = n_{i+2} = n_{i-2} = 1 \land \sum_{j=0}^{Q-1} n_j = 3, \quad i \in \{0, \dots, 4\}$$

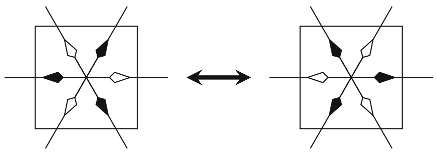


Figure 2.3: FHP three-particle collision rules

To complete the illustration of the update rules, Figure 2.4 shows the streaming step on a fragment of the lattice of an FHP automaton.

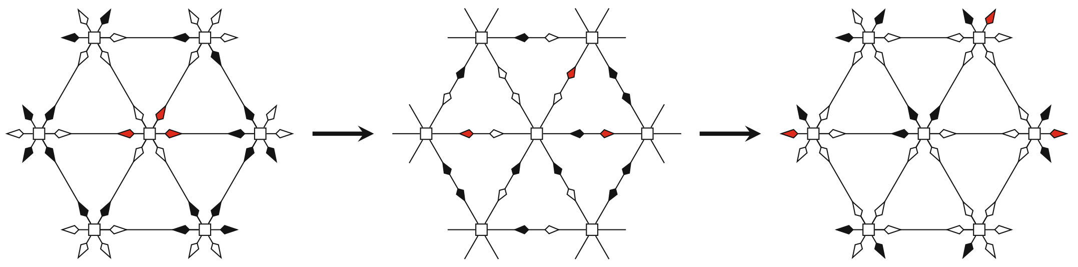


Figure 2.4: Illustration of the FHP streaming step

Observe how all particles are streamed to the same particle slot at the neighboring node, connected via the respective lattice velocity vector. Note that for the sake of this illustration's clarity, it is assumed that nodes outside this fragment do not contain any particles that are streamed into it. Also note that, for increased clarification, the center state shown in the figure depicts an artificial LGCA intermediate step, where the particles are "halfway" along the vector between their origin and target nodes. This intermediate state does not occur in a real simulation though, as particles stream to their new nodes in a single, atomic step. The particles of the center node are drawn in red to highlight their evolution.

Since one is ultimately interested in obtaining the simulated fluid's macroscopic quantities density and velocity (possibly among others), a way to determine them from the state of an LGCA is required. For the density at a (lattice) position $\mathbf{x}$ and time t, this is done by determining the number of particles at the respective lattice node and scaling the result with the respective physical properties. The fluid velocity can be determined quite similar via the momentum density $\mathbf{j} = \rho\mathbf{u}$, which can be obtained by considering the total momentum at a lattice node, and the previously determined density. Expressed as equations this yields

$$\rho = \frac{m}{V_n}\sum_{i=0}^{Q}\mathbf{n}_i, \qquad \mathbf{j} = \rho\mathbf{u} = \frac{m}{V_n}\sum_{i=0}^{Q}\mathbf{c}_i\mathbf{n}_i \tag{2.5}$$

where m is the mass of a single particle and V_n a lattice node's volume in physical space [1].

A typical CFD simulation using such an automaton performs the following steps. First, the nodes are randomly populated, such that the overall amount of particles approximates given initial constraints for the density and velocity field. Then, one alternates between applying the collision and the streaming steps (Equation (2.3)) for a desired amount of iterations. During the simulation the macroscopic properties are computed and stored at some suitable interval. Note that boundary conditions, that define what happens at nodes where the targeted PDEs do not apply, (for example solid, impenetrable, nodes) are accounted for inside the collision step. We will discuss boundary conditions a bit more in-depth later in the section about the LBM, as the way they are handled for LGCA does not relate as well to the LBM as its other properties.

At first glance, this scheme appears to be quite ad-hoc. Stating that this abstract model of a fluid can correctly produce the macroscopic behavior of a real fluid sounds like a bold claim. However, as the authors of [18] illustrate, it is actually possible to show that the behavior produced by an LGCA approximately adheres to the NSE. The way this is typically done is by using ideas from statistical mechanics and a, so-called, *Chapman-Enskog* analysis on the resulting statistical quantities. For example, for the FHP model the proof runs along the following lines. First, one introduces a discrete *particle distribution function* f. As concisely described in [1] it can be viewed as a generalization of the mass density that does not only depend on the position in space and the time, but also on the velocity. In case this function is normalized to sum to 1, it is sometimes also interpreted as the probability of a particle being at a certain position and time with a given velocity. Using the kinetics of the LGCA model one can show that there exists an *equilibrium distribution* for this function, which is a solution to a statistical mechanics description of the automaton's behavior. Using this equilibrium, the particle distribution function is then expressed as a series of perturbations from it due to the macroscopic fluid properties. If the underlying lattice of the automaton satisfies certain properties, it can then be shown that the NSE emerges when using this expansion. For more details, readers be referred to [2] and [19] which cover this proof for the FHP model in great detail.

A big advantage of LGCA over MD simulators is that both, the models of particles and collisions, are a lot simpler due to the extensive discretization, far beyond the discretization imposed by machine precision. This, in conjunction with the massively parallelizable update step, allows for the simulation of hundreds of millions of particles [20].
However, there are some issues with the physical setups that can be simulated [1]. The main problem of LGCA though is noise, which is a result of the discrete nature of the particles and their constant, non-continuous, changes due to the model's collisions and streaming steps [1, 4, 2]. As a countermeasure one typically averages over numerous lattice nodes ([21] reports averaging over up to 1024 neighbors), time and different initial conditions [4] (this is sometimes referred to as ensemble averaging). However, as stated

in [1] this only decreases the effects of the problem, rather than solving it.

2.5 The Lattice Boltzmann Method

In [4] McNamara and Zanetti proposed an alternative to the typical large-scale ensemble averaging used for combating the noise issue of LGCA. Their suggested remedy is to replace the discrete particle occupation information by the particle distribution function, f, briefly mentioned above. We will from now on call the elements of f, meaning the density of particles with a velocity $\mathbf{c}_i$ (denoted by f_i), the populations.
If we keep everything else of the LGCA, i.e., the use of a lattice, the conceptual collision and streaming steps and just replace the particles per lattice velocity vector by a density, we can rewrite Equation (2.3) to become the Lattice Boltzmann Equation (LBE):

Lattice Boltzmann Equation (force-free)

$$f_i(\mathbf{x} + \mathbf{c}_i\Delta t, t + \Delta t) = f_i(\mathbf{x}, t) + \Omega_i(f(\mathbf{x}, t)) \tag{2.6}$$

In essence, this equation does describe the same as the corresponding LGCA equation. The populations at each node are updated by first applying a local collision rule (Ω_i) to the current populations and then streaming the result at each node to its neighbors. Observe that this means that we keep the massively parallelizable update rule of LGCA. However, the state of a node does now no longer consist of the binary particle occupation number, but rather of the real-numbered distribution function. This results in the model not being subject to the inherent noise caused by the discrete particles and their kinetics.

While getting rid of the noise, that plagued the LGCA, was certainly valuable on its own, what McNamara and Zanetti introduced as a fix has a further, paramount, consequence. Namely, it can be shown that Equation (2.6) is a discretized version of the Boltzmann Equation (shown in Equation (2.8)). This fact effectively results in a whole new CFD simulator, to which we nowadays refer to as the LBM, and which has largely replaced LGCA. To properly convey the implications of this relationship to the Boltzmann equation, the following will give a brief introduction to the kinetic theory of gases.

2.5.1 Kinetic Theory of Gases

The kinetic theory of gases assumes that the macroscopic behavior of a gas is the result of the interaction between its particles (and likewise for fluids in general). Under some simplifying assumptions, like the gas being rarefied and only consisting of mono-atomic particles, this boils down to the macroscopic behavior emerging out of two-particle collisions that happen due to the particles' motion. Based on these assumptions, one then tries to provide a kinetic description of the particles, that connects the microscopic to the macroscopic fluid behavior.

We have already encountered the key entity around which the kinetic theory revolves. It is the generalized mass density, which we introduced as the particle distribution function f.
Note that the mass density can be obtained by integrating over all the possible velocities:

$$\rho = \int f d\mathbf{v} \tag{2.7}$$

Analogous to Equation (2.5), the momentum density can be determined via an integration of $\mathbf{v}f$ over $\mathbf{v}$. Note that such integrals over f, weighted by some function, here 1, are called the moments of the particle distribution function.

Naturally, with the particle distribution function being directly related to the targeted macroscopic quantities, a mathematical description of the evolution of f due to the particle behavior is desired. Following the lines of [11] or [22], one can derive the Boltzmann Equation (BE), which essentially states that f changes over time due to the movement of particles and collisions between them:

Boltzmann Equation (force-free)

$$\partial_t f + \mathbf{v}\nabla f = \Omega(f) \tag{2.8}$$

The original collision operator derived by Boltzmann is an unwieldy integral that prohibits the analytical solution of Equation (2.8) for most cases [11, 23]. However, it can be shown that solutions of this equation comply with the CE and NSE[1].

An interesting property of f is what is called the Boltzmann $\mathcal{H}$-Theorem. Essentially, it states that the particle distribution function will eventually reach an equilibrium, f^{eq}, at which it remains (Curious readers be referred to [11]). This equilibrium is a solution to the BE that is stationary in time, i.e.,

$$\partial_t f^{eq} = 0 \tag{2.9}$$

Note that the equilibrium is a consequence of the collision between particles. Thus, Ω is responsible for modeling the evolution of f towards f^{eq}. The Boltzmann collision integral achieves this by capturing the effect of all kinds of binary collisions.
However, this integral is not the only collision operator that captures the evolution towards the equilibrium (while conserving physical quantities) for which it can be shown that the BE recovers the governing macroscopic fluid equations. Indeed, in [23] a collision operator is introduced that conserves mass, momentum and energy and leads to a relaxation of f to f^{eq} (adhering to the mentioned $\mathcal{H}$-Theorem). The resulting collision operator, named after the authors as the Bhatnagar-Gross-Krook (BGK) collision operator, has the simple form

BGK-Collision Operator

$$\Omega^{BGK}(f) = -\frac{1}{\tau}\left(f - f^{eq}\right) \tag{2.10}$$

The relaxation time τ determines how fast f relaxes toward f^{eq} and is related to the properties of the considered species, such as viscosity or diffusivity [1].

The equilibrium distribution, derived by Maxwell and Boltzmann and hence referred to as Maxwell-Boltzmann Distribution (MBD), which a gas with the initial assumption relaxes to, has the following form

Maxwell-Boltzmann Distribution

$$f(\mathbf{v}) = \left(\frac{m_p}{2\pi k_B T}\right)^{3/2} \cdot e^{-\frac{m_p \mathbf{v}^2}{2k_B T}} \tag{2.11}$$

Note how this distribution expresses that, at the same temperature T, heavier particles (higher particle mass m_p) are expected to be slower than lighter particles. k_B is the Boltzmann constant and $\mathbf{v}$ the continuous velocity.

2.5.2 Simulator

The previous section has motivated why the modified LGCA proposed by McNamara and Zanetti is so interesting for CFD. With Equation (2.6) being a discretized version of the BE, and solutions to the BE complying with the NSE, this new automaton actually has a physical foundation and can simulate the desired fluid quantities.
We will now discuss how the above quantities and equations from kinetic theory relate to the LBM, as well as illustrate how a typical computational fluid simulation using this scheme proceeds.

As already mentioned before, the BE can be discretized in physical space, velocity space and time to yield the LBE (Equation (2.6)). This can also be done for the version using the BGK collision operator [24], and results in the following form of the LBE

$$f_i(\mathbf{x} + \mathbf{c}_i\Delta t, t + \Delta t) = f_i(\mathbf{x}, t) + \frac{\Delta t}{\tau}\left(f_i - f_i^{eq}\right) \tag{2.12}$$

where f^{eq} is a discretized version of the MBD. Note how this collision operator is significantly simpler, especially for more populations, than the explicit collision matrices required for LGCA [25].
The discretized equilibrium distribution is given below in a particularly illustrative form (the typically used simpler, but less insightful, version will be shown later in Chapter 4):

Discretized MBD as in [2]

$$f_i^{eq} = \frac{w_i}{\rho_0}\left(\rho + \frac{m_p}{k_B T}\mathbf{c}_i \cdot \mathbf{j} + \frac{m_p}{2\rho k_B T}\left[\frac{m_p}{k_B T}(\mathbf{c}_i \cdot \mathbf{j})^2 - \mathbf{j}^2\right]\right) \tag{2.13}$$

The symbol ρ_0 denotes a typical initial density (scalar). Note that the equilibrium distribution, assuming an isothermal setting where $T = const$, only changes due to the populations' moments ρ, $\mathbf{j} = \mathbf{u}\rho$ (where $\mathbf{j}$ is the *momentum-density* and $\mathbf{u}$ the macroscopic fluid density).
The, so-called, lattice weights w_i are important for ensuring certain symmetry and isotropy (in a certain 2D rotational invariance) properties required to recover the desired macroscopic equations. They are particularly important for lattices with more than a single lattice velocity magnitude. This includes many of the lattices typically used for fluid flow simulations (compare with the lattices listed in [1] and [2]). For example, many lattices feature a population at rest with a velocity of zero. Additionally, many lattices also feature different magnitudes for the non-rest velocities.
One such lattice, which is also used in this work, is the D2Q9 lattice (the DdQq notation introduced in [24] specifies lattices via their dimension d and the cardinality q of the lattice velocity vector set). A fragment of this particular lattice is depicted in Figure 2.5.

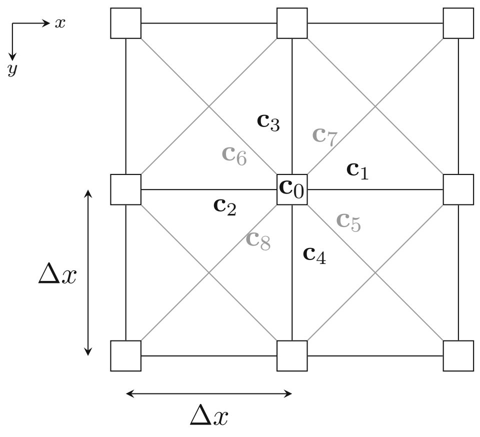


Figure 2.5: The D2Q9 lattice

Note how the magnitudes of the velocities $\mathbf{c}_{\{5,6,7,8\}}$ are by a factor of $\sqrt{2}$ times greater than the ones of the velocities $\mathbf{c}_{\{1,2,3,4\}}$. The reason being that diagonal neighbors are $\sqrt{2}\Delta x$ spaced apart from another, with Δx being the lattice spacing. Also, observe that this lattice features a rest velocity, $\mathbf{c}_0$.

The typical steps of an LBM simulation are similar to the ones of LGCA-based simulations (c.f., Section 2.4):

1. The density and fluid velocity field are initialized for the considered problem (for

LGCA, instead of directly initializing these macroscopic properties, the nodes' particles were set to approximately yield a given initial density and velocity).

2. Optional: Non-dimensionalization of the simulation

3. The equilibrium distribution for this initial density and velocity is determined and used to initialize the populations f. Note that this is a *local* rather than global equilibrium.

4. The (BGK) collision operator is used to update the populations (similar to the LGCA collision step).

5. The post-collision populations are streamed to their neighbors (exactly as in the LGCA streaming step).

6. The time-step is increased by Δt. Also, the macroscopic properties can optionally be dumped.

7. The particle distribution function's moments are updated and, using the results, also the local equilibrium. Then the simulation repeats from step 4 onwards until either some convergence condition is met, or a desired number of iteration steps got performed.

Non-Dimensionalization

We did not mention the non-dimensionalization step so far. As the name suggests, the purpose of this is to scale all quantities that occur during the simulation, such that their afflicted physical units are dropped, while keeping the relations of all quantities unchanged. As described by Krüger *et al.* in [1], the simulated fluid flow problem is invariant to the absolute values of quantities. Only the relations between them are important. While this whole step might not appear very sensible at first glance, there are good reasons to do this.
Most stability heuristics are derived and given for the non-dimensional LBM. This allows to easily apply them to arbitrary simulations. Furthermore, the non-dimensionalization step allows us to apply a nifty trick. By choosing characteristic quantities for the scaling in a smart manner, we can choose the absolute value of some non-dimensional quantities. This allows us to tune the simulation with respect to stability and accuracy without changing the overall considered problem. As long as we perform the inverse of the non-dimensionalization scaling on the simulator output, the obtained quantities will be the ones for the original, dimensional, problem.

Boundary Conditions

Recall that the NSE (Equation (2.2)) describes all fluid flows. We require boundary conditions (and the initialization) to specify the exact problem for which we want to obtain solutions. Such boundary conditions define how the quantities modelled by PDEs

should evolve on boundaries. For example, such a boundary could express that the considered fluid is enclosed in an impermeable container. In this case, suitable boundary conditions would be that some lattice nodes, corresponding to the position of the walls, are modelled to be *solid*. Essentially this expresses that their populations are all zero, as no particles can reside at them. However, to keep the streaming step simple, and to allow complex geometries in a simple manner, populations are also streamed to such nodes. Optional update rules after the streaming step are required to ensure that boundary conditions are respected.
As an example, consider the popular *bounce back* for solid nodes. This scheme simply reverts the direction of populations on solid nodes after the streaming step and streams them back to their original node. However, there exist many alternatives to this scheme, each having its own (dis)advantages and impact on the simulation accuracy and performance [1].
In addition to this simple type of boundary condition, further types exist that allow that certain macroscopic quantities, or their derivatives, satisfy imposed conditions (for example, that a wall is moving with a certain velocity) [26].

Example

In order to make the LBM more tangible, Figure 2.6 shows an illustration of one iteration of the main simulation loop on a D2Q9 lattice of size 5×5. The image depicts how the nodes' populations change due to the collision, streaming and boundary condition steps respectively.
To explicitly distinguish between the discrete particles of the LGCA and the particle distribution function of the LBM, cloud symbols are used to draw them (instead of the LGCA arrow tips). The diameter of the clouds hints at their value (recall that they are floating point numbers). However, be aware that the size of the non-resting populations got increased by a constant factor in order to improve their visibility. Otherwise, especially the populations at the diagonal lattice velocity vectors would hardly be visible next to the rest populations.

The lattice state depicted at the top-left is the initial state where $f = f^{eq}$ for $\mathbf{u} = 0$. Furthermore, observe that the center node got initialized to feature a density twice the one of the other nodes. The black squares mark solid boundary nodes, where the bounce-back scheme is applied.
Observe how the collision step effectively results in the populations being spread at the respective node (top right). This is particularly visible for the center node. Its rest population decreases significantly, while the other populations increase.
The streaming step then distributes differences between the populations of neighboring nodes (bottom right). Also observe how the populations propagate to the boundary nodes (shown in blue) and how there is no propagation from these nodes towards their non-solid neighbors.
Applying the bounce back boundary condition scheme then simply flips the direction of the respective populations at the boundary nodes and streams them back to the respective

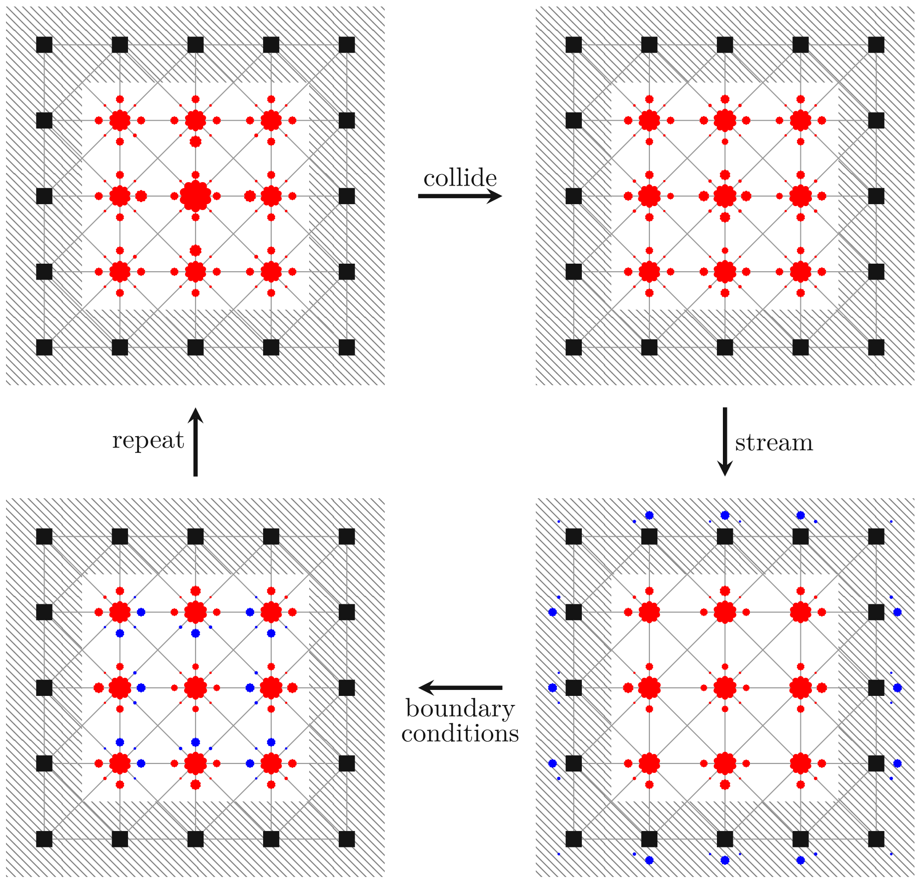


Figure 2.6: Graphical illustration of an LBM simulation iteration

origin node (bottom left; again highlighted in blue). Be aware that this handling of the boundary condition is artificially extended for the purpose of illustration. An efficient implementation would handle the bounce back locally rather than introducing a second explicit streaming step.

Note that the bounce back scheme is considered a, so-called, *link-wise* scheme [1]. Such schemes fit the lattice to the desired geometry such that the geometry's boundaries lie midway *between* lattice nodes. Thus, boundary nodes are completely inside the fluid domain. An alternative scheme, which will be discussed later, is the *wet-node* scheme where boundary nodes coincide with the geometric boundary.

Summary

To conclude this brief introduction to the LBM, we will now summarize some of its most prominent advantages and disadvantages.
First, as was already mentioned above, the LBM lends itself to massively parallel implementations just like LGCA does. The nature of this property lies in the fact, that

the main simulation loop only features comparably hard / computationally expensive operations locally per node (computation of the equilibrium, boundary schemes, collision), whereas the steps that do not work solely on node-local data are rather simple from a computational point of view (streaming).
Another positive property of the LBM is that its basic steps are rather simple to implement, as complexity has its origin in more intricate details like accurate boundary schemes and extending / adapting the method.
Furthermore, an advantage over conventional PDE solvers, which the LBM and LGCA share, is that the populations / particles allow gaining additional insights into the meso- / microscopical dynamics of the considered problem [27].
On the other hand, while alternative schemes like finite difference and LGCA are conceptually rather straightforward, the LBM requires knowledge about statistical mechanics and kinetic theory to be properly understood (this includes giving proofs about the correctness of modified versions of it).

Finally, an interesting property of the LBM is the fact that it recovers the BE due to the used equilibrium distribution, collision rule and lattice. By modifying one or multiple of these parts, one can deploy this simulation method for other PDEs than the ones mentioned so far [1, 2]. Examples are chemical reactions [27, 28, 25, 29] and biological systems [7, 9, 30]. While this is something generic PDE solvers like the finite difference scheme can by design, it is worthwhile to mention as the LBM can keep its proneness to parallel implementations when modified. Furthermore, this property lends itself to combining fluid flow with other partial differential equations, like the ones used for describing chemical and biological reactions, which is why it is of particular interest for this thesis.

2.6 Advection-Reaction-Diffusion Problems

Diffusion is a well-known process where particles of a species move from areas with a high concentration of this species to areas with lower concentration due to random thermal motion. Overall, this results in the local differences in concentration decreasing with time. In addition to diffusion, often systems where species also react are considered. Since reactions also lead to changes in concentration, the two effects are often expressed in a single equation, the Reaction-Diffusion Equation (RDE). Equation (2.14) shows this equation for the concentration $c_\mathcal{X}$ of a species $\mathcal{X}$. As this equation is a well-established way to model the reaction of chemical species in physical space and time, it is of interest during this thesis.

Reaction-Diffusion Equation

$$\partial_t c_\mathcal{X} = D\nabla^2 c_\mathcal{X} + R \tag{2.14}$$

This kind of diffusion is called *Fickian* diffusion and the respective term ($D\nabla^2 c_\mathcal{X}$) has the general form $\nabla\left(D(c_{\mathcal{X}})\cdot\nabla c_{\mathcal{X}}\right)$. However, as stated by Roussel in [31], the diffusion coefficient D for fluids (which are isotropic, meaning invariant to rotations) can be considered a constant, if it depends only weakly on the species concentration. R is a generic reaction term that describes the creation or decay of the species $\mathcal{X}$ due to reactions with other species or the environment. For example, we could use a logistic reaction term, where the environment of the considered species can only sustain a certain concentration K [31]:

$$\partial_t c_{\mathcal{X}} = D\nabla^2 c_{\mathcal{X}} + \frac{r c_{\mathcal{X}}}{1-\frac{c_{\mathcal{X}}}{K}} \tag{2.15}$$

If the diffusion and reaction of all participating species is of interest, the problem is modelled as a system of such RDEs. The respective reaction terms depend on the reacting species' concentrations and the type of chemical kinetics that is chosen to describe the reaction.

Equation (2.14) can further be extended to also contain the effect a velocity field (like one due to a suspension in motion) has on the considered particles by adding an advection term. This results in the Advection-Reaction-Diffusion Equation (ARDE), shown in Equation (2.16), where the term $\nabla\mathbf{u}c_{\mathcal{X}}$ got added (with $\mathbf{u}$ being the fluid velocity).

Advection-Reaction-Diffusion Equation

$$\partial_t c_{\mathcal{X}} = \nabla\mathbf{u}c_{\mathcal{X}} + D\nabla^2 c_{\mathcal{X}} + R \tag{2.16}$$

2.6.1 RDEs and Biological Systems

We concern ourselves with these two equations not only because they are a well-established way to model the reaction of chemical species in physical space and time, but also because it turns out, that they can be used to model biological systems. For example, consider a colony of bacteria in a fluid at rest. It is known from experiments, that individual bacteria will exhibit a random motion [32] (without any further effects in place; compare to, for example, chemotaxis below). Since the effect of diffusion and this random motion is the same on a population scale (differences in the concentration / density vanish over time), the evolution of the density of the bacteria can be expressed as RDE [33]. The reaction term then captures the growth and decay (jointly coined as *proliferation*) of the bacteria.
Regarding the density of bacteria, we will, in addition to their "diffusion" and proliferation, also consider the, so-called, *chemotaxis* of bacteria.
As described in [33], chemotaxis is the movement of biological entities due to concentration differences of chemicals to which they are sensitive. If a chemical attracts a species, it is called *chemoattractant*, if it repels it *chemorepellent*. Without loss of generality, we will in the following only discuss the attraction. Murray further motivates a mathematical

model that captures the effect of chemotaxis of bacteria $\mathcal{B}$ to a chemoattractant $\mathcal{C}$, which is shown below.

$$\nabla\left(\chi_{tx}(c_{\mathcal{B}}, c_{\mathcal{C}})\nabla\mathcal{C}\right) \tag{2.17}$$

χ_{tx} is the chemotactic sensitivity, i.e., a function that models the change of the density of the bacteria due to the chemotaxis towards $\mathcal{C}$. Murray summarizes that a sensitivity of the form

$$\chi_{tx} = \frac{k_1 c_{\mathcal{B}}}{[k_2 + c_{\mathcal{C}}]^2} \tag{2.18}$$

with k_1, k_2 being parameters obtained from experiments is suitable to capture the effect of chemotaxis on a population level.
Observe how Equation (2.17) has the same form as the general model for Fickian diffusion, suggesting that an RDE can be modified to account for the effect of chemotaxis.

However, note that this is only one approach to modelling chemotaxis. In her thesis, Dolak [34] gives a brief overview of different approaches. Interestingly, Dolak also categorizes the models for chemotaxis into two general groups. The macroscopic models describe the evolution of the cell density with respect to macroscopic effects, whereas the kinetic models model cells and their interactions. This is quite reminiscent of the top-down and bottom-up categorization of CFD solvers we discussed earlier.

CHAPTER 3

Related Work

In this chapter we will review the related work and literature. Section 3.1 focuses on reports of the LBM being applied to chemical reactions, which is mostly about recovering the behavior of RDEs. In Section 3.2 works about simulating biological systems, with a focus on the populations of bacteria, using the LBM will be covered.

3.1 LBM for Chemically-Reacting Fluids

The first to consider the simulation of reaction-diffusion problems using the LBM were Dawson *et al.* in [27]. The authors propose a way to numerically approximate solutions for a system of two-dimensional RDEs of the form

$$\partial_t c_{\mathcal{X}} = D_{\mathcal{X}} \nabla^2 c_{\mathcal{X}} + R_{\mathcal{X}}, \quad \mathcal{X} \in \mathscr{S} \tag{3.1}$$

where $\mathscr{S}$ is a set of M distinct species. In order to recover the considered set of macroscopic equations, they modify the LBE (Equation (2.6)) by replacing the typical collision operator with a sum of a non-reactive (Ω_i^{NR}) and a reactive (Ω_i^R) term:

$$f_i(\mathbf{x} + \mathbf{c}_i \Delta t, t + \Delta t) = f_i(\mathbf{x}, t) + \Omega_i^R(f(\mathbf{x}, t)) + \Omega_i^{NR}(f(\mathbf{x}, t)) \tag{3.2}$$

For the non-reactive part they use the BGK-collision operator and for the reactive one the authors settled on the simple term

$$\Omega^R(c_{\mathcal{X}}) = \frac{R_{\mathcal{X}}}{Q} \tag{3.3}$$

where R is the reaction term. It depends on the concentration of all respective reactants. Note that, by using the concentrations, this function must essentially be macroscopic in

its nature. From this viewpoint, the above reaction term simply evaluates the macroscopic reaction kinetics at each node and distributes the result uniformly among all populations. The distribution of the populations among the discrete velocities is not considered. As a result, also noted by the authors themselves, this reactive source term is a strong simplification of the actual microscopic behavior. As for the reaction term itself, its form can vary depending on the simulated system. For example, in [27] the authors chose for it to be the, so-called, *Selkov* model.
It is particularly noteworthy, that Dawson *et al.* show, via a Chapman-Enskog expansion, that their modified LBM does recover the desired macroscopic behavior if τ is related to the diffusion coefficient in a particular manner.
As an alternative to the use of the macroscopic reaction kinetics, they provide a reaction term that incorporates more of the available mesoscopic information. The below equation shows this, arguably more elaborate, term:

$$\Omega^R_{i,\mathcal{X}} = \sum_{S \in 2^{\mathscr{S}_\mathcal{X}}} \sum_V M^S_{i,\mathcal{X}}(V) \prod_{\mathcal{A} \in S} f_{V_\mathcal{A},\mathcal{A}} \tag{3.4}$$

Note that $\mathscr{S}_\mathcal{X}$ denotes the set of all species involved in a reaction with species $\mathcal{X}$. With V we denote a velocity configuration, which is a vector that holds a discrete velocity for each species in S. Thus, the inner sum considers all possible combinations of the populations of the species of the outer sum. These configurations are used to obtain the probability of a collision occurring, involving populations i of $\mathcal{X}$ and the populations according to V of the species in S, from the matrix $M^S_{i,\mathcal{X}}$. This probability is then weighted by the populations.
Obviously, this reaction term is computationally more expensive than the one of Equation (3.3), at least as long as $R_\mathcal{X}$ does not become of similar complexity. The paper does also not explain how these matrices should be obtained, which in general can be assumed to be quite an effort on its own (especially for complex reactions involving many species). Hence, while the authors postulated the term above, they did not use it themselves. Instead, they stuck to Equation (3.3) when validating their scheme. They do so by considering three scenarios: One with pure diffusion, another one with pure reaction and finally one featuring both. The results are then compared to theoretical predictions. It is reported that the numerical solutions capture the expected physical behavior.

It should be pointed out that Dawson *et al.* assumed the macroscopic velocity of the reacting fluids during most of their tests to be zero, i.e., the fluids to be at rest. However, they also report an instance where they use the LBM to solve the system of RDEs *and* a solvent's flow satisfying the NSE in conjunction. Note that the whole simulation happened on the same lattice, D2Q7 (the use of this lattice can likely be attributed to the year of the publication and the close resemblance to the FHP lattice), and without considering the solvent to be subject to diffusion as well.

In [25], Qian and Orszag use the LBM to obtain solutions of the irreversible chemical reaction $\mathcal{A} + \mathcal{B} \to \mathcal{C}$, where all three species are driven by diffusion and for which no

general analytical solution is known. While the authors argue that they incorporate the effects of chemical reactions on the species' concentrations as a force that acts on the diffusion process, they ultimately end up with the same modified LBE as Dawson *et al.* (using Equation (3.2) with the reaction term shown in Equation (3.3)) and τ incorporating the diffusivity.
To validate their method, Qian and Orszag measure the simulated diffusivity of their LBM for the three lattices D1Q2, D2Q4 and D3Q6 and report agreement with theoretical results within 0.1%.
Furthermore, they also simulate another chemical reaction, the so-called *Schlögl* model, which is a single-species fourth-order reaction where the competition between the effects of diffusion and reaction can result in the formation of patterns. They report a set of parameters for which such pattern formation is expected and can indeed observe it in their simulations.

As noted in [25], diffusion effectively drives the concentration of a species towards an equilibrium. This observation from physics is directly reflected in the above two schemes, where the relaxation time in the BGK operator depends on the diffusion coefficient. However, this is not the only way to account for diffusion in the LBM. A slightly different approach to simulate this physical process within the scheme was first reported by Kingdon and Schofield in [35]. The authors do not use the BGK collision operator (which was only shown to be applicable to the LBM a short time before their letter). Instead, they derive a dedicated diffusion update rule that uses the gradients of the species' concentrations. Note that this makes the update step, which is the most expensive one in terms of computations, non-local. Thus, one of the central properties of the LBM, its aptitude for massively parallel implementations, is no longer given for this model.
After this modified update step, an iteration of their main simulation loop ends with using the concentrations per node to compute the effects of reactions. Since this step does not regard the distinct populations, similar to the simplified reaction term of Dawson *et al.*, the produced microscopic picture is not accurate.
An interesting detail of their approach is the consideration of an advection term, such that their method can be applied to ARDEs. The way this is done is similar to the extension Dawson and co-authors report for their method [27]. It is assumed that the reacting species are, when compared to the moving solute, of low concentration and that the flow is thus only produced by the solute movement alone.

In his PhD thesis, Alemani [36] uses the LBM to simulate complex reaction diffusion systems that model the flux of particles of a metal due to its consumption by *bioanalogical* sensors or microorganisms. While the reported LBE uses the same reactive term as reported by Dawson *et al.* [27], the thesis is noteworthy regarding the used reaction scheme, as the considered problem features diffusion and reaction time scales that differ by several orders of magnitude. Alemani deploys a *time splitting* technique to capture the effects of both processes on the populations in the same simulation in an accurate manner, without requiring the computational resources necessary for treating the whole problem on the smallest time scale. Time splitting, as the name suggests, splits the lattice time step into a sum of smaller time steps, each corresponding to one physical process (for example diffusion and reaction). The populations are then updated by the effect of each such process using the respective time step. Finally, the resulting populations are propagated.

An observation that can be made from the approaches mentioned so far is that the reaction terms typically work on the recovered macroscopic concentrations. This makes sense, considering that most approaches appear to use established chemical reaction models that revolve around reaction rates. These rates describe the effects of up to billions of collisions per second and are thus typically highly averaged. Thus, the very purpose of chemical kinetics is not to deal with distinct collisions and the responsible elementary processes [37].
However, ultimately the LBM is a mesoscopic method that does not operate on the level of the concentrations, but rather on the level of the particle distribution function. Thus, a reaction term that models chemical reactions on the mesoscopic scale as well seems like an objective worth pursuing. This got recognized by some researches, who proposed alternative schemes based on chemical kinetics and collision theory.

Among them are Bresolin and Oliveira, which aim to incorporate chemical reactions by a modified streaming step in [28]. In their modified version, *populations* of one species can probabilistically be converted to the populations of another species, in accordance with the simulated chemical reactions. This is supposed to reflect a model from collision theory that can be used to derive chemical reaction rates.
The authors base their idea on the assumption that chemical reactions happen much faster than diffusion, such that at each propagation step they can flip a complete population in case their algorithm determines that a reaction occurs. This determination is done by sampling a random number for each population of a node per time stamp, and then determining whether this random number is above a certain threshold $P_{reaction}$ or not (the reaction takes place in case it is). In essence, this method aims at mimicking the insight from collision theory that reactions depend on the rate of collisions and the energy involved in a collision. A higher reaction rate is assumed to be the result of more frequent collisions with sufficient energy, which relates to the threshold being lower and population-scale reactions taken place more often.
The authors themselves interpret their scheme as a mixture of the LBM and a Monte-Carlo method.
To test their method, they consider three model examples that are simple enough to be solved analytically as well. While they show that their method can reproduce the analytical solution, they also observe that they require a certain lattice size, especially for low reaction rates, due to introduced statistical noise. This is insight is quite noteworthy, considering that one of the main features of the LBM was to get rid of the LGCA requirement for large lattices in order to mitigate the effects of statistical noise.

In a later work [38], Abdollazadeh *et al.* use the modifications proposed by Bresolin and Oliveira and compare the results obtained by using it with the results produced by a commercial finite element solver and the LBM using the macroscopic reaction term. Their reported results show that the two LBM schemes are in general very similar. However, the scheme with the macroscopic reaction term shows slightly less deviation from the result obtained by the commercial solver, than the one with the collision theory inspired term.

A different approach to an LBM for chemical reactions based on collision theory is the one in the work of Pribec *et al.* [29]. Just like Bresolin and Oliveira, the authors observe that the majority of reactive LBM schemes uses a reactive term that neglects the existing information about the populations' respective velocities and simply distributes the effects of reactions uniformly (c.f., Equation (3.3)).
However, the paper proposes an alternative mesoscopic reactive collision term, that is constructed by applying similar ideas from collision theory as Bresolin and Oliveira did. To be more specific, the authors consider the relative velocity $\mathbf{c}_{rel}(i, j)$ between populations with velocities i, j as a means to incorporate the relative energy of collisions into their term. The relative velocity is then used to indirectly determine a reaction probability. This is done by plugging it into a step function $\sigma_{\mathcal{AB}}$, which mimics the probability of a collision being reactive (similar to the random number and threshold check in [28]).
As for the extent to which a reaction changes a population, a special reaction term is used. For the reaction $\mathcal{A} + \mathcal{B} \to \mathcal{C}$, with reaction rates $R_{\mathcal{A}} = R_{\mathcal{B}} = -R_{\mathcal{C}}$, the reactive term for the population of $\mathcal{X} \in \{\mathcal{A}, \mathcal{B}, \mathcal{C}\}$ with velocity $\mathbf{c}_i$ is

$$R_{i,\mathcal{X}} = \sum_{j,k \in Q} K_{\mathcal{AB},\mathcal{X}}(i, j, k)\sigma_{\mathcal{AB}}(\mathbf{c}_{rel})\mathbf{c}_{rel}(j, k) f_{j,\mathcal{A}} f_{k,\mathcal{B}} \tag{3.5}$$

The matrix K contains coefficients that are set such that the reaction term for each species conserves mass and momentum. This is particularly interesting for the product species $\mathcal{C}$ as, depending on the relative mass of particles of species $\mathcal{A}$ and $\mathcal{B}$, the resulting velocity vectors that conserve momentum do not coincide with the lattice velocity vectors (Figure 2 in [29]). Pribec *et al.* propose to deal with this issue by considering these "off-lattice" velocities as linear combinations of the three closest lattice velocity vectors and put the coefficients of these combinations into $K_{\mathcal{AB},\mathcal{C}}$. Observe how this equation is actually just a special case of Equation (3.2) with the matrix M being given in more detail.
Note that, in comparison to the scheme of Bresolin and Oliveira in [28], no random number generation is necessary and that the matrices $K_{spec\mathcal{AB},\mathcal{X}}$ and the relative velocities can be pre-computed.

The test their scheme, Pribec *et al.* consider the same $A + B \to C$ system as Qian and Orszag did in [25]. Furthermore, in addition to comparing the two different versions of the LBM, they also perform a comparison with the results obtained by numerically solving the macroscopic equations using a finite difference approach.
However, while they could show for the considered cases that their LBM scheme is as capable as the reference from [25], the difference in accuracy was all in all underwhelming

(at best below 1%) when considering that the authors report that their method requires more computation time.
It is further noteworthy that Pribec *et al.* also considered a version of the system affected by a velocity field originating from a NSE compliant fluid flow simulation (again under the assumption of the species being of sufficiently low concentration such that they do not impact the solvent).

A slightly different problem to apply an LBM for ARDEs, is the one considered by Moaty *et al.* in [39]. The authors aim to simulate a 3D bioreactor - a vessel that provides a suitable environment for cell growth - containing cartilage cells placed on porous beds, subject to an oxygen flow.
This is achieved by adding a sink term to the LBE that accounts for the consumption of oxygen by the cells. However, the cells themselves are not simulated. The modified LBM is validated for two simple testcases, and then used to determine the amount of oxygen consumed by the cells, as well as the effect of this on the flow. Furthermore, the authors simulate the stress applied to the cells by the flowing oxygen.

3.2 Simulating Bacteria with the LBM

The growth of bacterial colonies is of major scientific interest for various reasons [40, 9]. As mentioned in Section 2.6, systems of RDE appear like a promising way to model such growth. For example, in [41] Mimura *et al.* were able to reproduce growth patterns, similar to the ones observed in experiments, by numerically solving such systems of RDE using a finite element solver with randomized mesh.
As we have motivated by now, the LBM should be a suitable method to model the behavior of bacteria at the population level, allowing us to gain more insight into the dynamics of the populations than generic PDE solvers. While there are only a few reports of usage of the LBM to simulate the growth and "flow" of bacteria, some researches have made some first investigations in that direction.

In [9], De Rosis *et al.* use the LBM to numerically determine solutions to the same system of reaction diffusion equations as Mimura *et al.* did in [41].
The considered system consists of three species, (1) the $\mathcal{N}$utrient, (2) the active $\mathcal{B}$acteria and (3) the $\mathcal{I}$nactive bacteria. The change over time of these three species' concentrations is described by the following three coupled equations (non-dimensionalized):

$$\partial_t c_{\mathcal{N}} = \nabla^2 c_{\mathcal{N}} - c_{\mathcal{B}} c_{\mathcal{N}} \tag{3.6}$$

$$\partial_t c_{\mathcal{B}} = D\nabla^2 c_{\mathcal{B}} + c_{\mathcal{B}} c_{\mathcal{N}} - a \cdot c_{\mathcal{B}} \tag{3.7}$$

$$\partial_t c_{\mathcal{I}} = a \cdot c_{\mathcal{B}} \tag{3.8}$$

Observe that both nutrient and bacteria are subject to diffusion (in a system of RDE one can normalize such that one diffusion coefficient becomes 1 [31]). The consumption of the nutrient - and the resulting growth of the bacteria population - is modelled via the simple reaction term $c_{\mathcal{B}} c_{\mathcal{N}}$. In addition to the growth due to nutrient consumption, the proliferation of active bacteria entails the conversion to inactive bacteria. This decay happens with the rate a, which has the form

$$a = \frac{1}{(1 + c_{\mathcal{B}}/a_1)\,(1 + c_{\mathcal{N}}/a_2)} \tag{3.9}$$

with a_1, a_2 being constants. Both [9] and [41] use $a_1 = 1/2400, a_2 = 1/120$ without any explanation, other than that the values reproduce the desired results.
Note that the concentration of inactive bacteria changes solely due to active bacteria becoming inactive. As stated by Mimura *et al.* in [41] and Murray in [40], experiments show the existence of bacteria that stop being motile, i.e., lose their ability to move by themselves. The equation does therefore not contain a diffusive term, as diffusion is primarily used to model the movement of bacteria.

Figure 3.1 shows a comparison between the results reported in [41] and [9]. Note how there are different types of patterns, that depend on the relation between the concentration of the nutrients ($\mathrm{C_n}$) and the medium on which the bacteria and nutrients are (agar, $\mathrm{C_a}$). Observe how the results of De Rosis and co-authors are much more symmetrical, as the team around Mimura introduced some randomization into their simulation in the form of a non-uniform and randomized mesh.

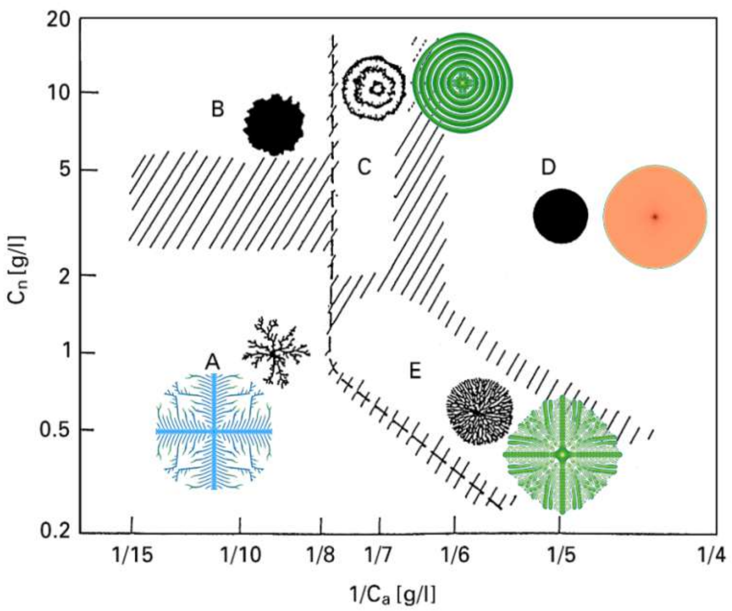


Figure 3.1: Comparison of the simulated growth patterns with the ones presented in [41] (black and white) and [9] (colored)

While the results of De Rosis *et al.* are just a reproduction of the ones reported by Mimura *et al.*, with the LBM as solver instead of the finite element scheme, there is an interesting observation to take away from the way the LBM was applied. Namely, since the PDE one wants to recover is not the NSE, the simulated problem is not the one fluid flow, but rather that of the transport of a scalar like mass or concentration. This means that a lattice with smaller velocity set can be used without losing the ability to recover the macrosopic solutions. In their letter they use the D2Q5 lattice. The reason for that being that the RDE does not consider velocity. Thus, there is also no inherent momentum conservation, which typically depends on the lattice being sufficiently big (with respect to the velocity set) and symmetric in the proof scheme.

Another interesting result by De Rosis *et al.* is the observation that the growth patterns do not emerge when using the D2Q9 (or D2Q13) lattice. This is quite surprising considering that a finer resolution of the velocity space should, intuitively, result in a higher accuracy. The authors did not further investigate the cause of this behavior.

While the results from [9] are inspiring in the rich behavior produced by applying the LBM to a biologically inspired system of RDE, the bacterial model is rather simple. The modelled bacteria are only considered to move randomly and consume the nutrient. Effects like chemotaxis or other behavior produced by the bacteria themselves is not contained in the model (and also not in the one of Mimura *et al.*).
Furthermore, real bacteria are influenced by their environment which is in reality rarely at rest. Both works mentioned above do not consider any velocity field produced by the bacteria (and nutrient's) environment though.
It would thus be interesting to extend the used schemes by the effect of chemotaxis and to account for fluid movement of a solute. The latter could be achieved by harnessing the fact that the LBM is used, lending itself to fluid dynamic simulations and simulate the fluid containing the considered species in conjunction with the system of RDE.

The first to study the application of the LBM to bacterial chemotaxis was Hilpert in [7]. The general system considered in the paper is one where a species of $\mathcal{B}$acteria and a $\mathcal{C}$hemoattractant are present in a liquid. The flow of the liquid is assumed to be independent of the bacteria and the attractant (this is a common assumption that we have already mentioned before). The system is described by the following two coupled PDEs

$$\partial_t c_{\mathcal{B}} + \nabla c_{\mathcal{B}} \left(\mathbf{u} + \mathbf{u}_{tx}\right) = \mu \nabla^2 c_{\mathcal{B}} + R_{\mathcal{B}} \tag{3.10}$$

$$\partial_t c_{\mathcal{C}} + \nabla c_{\mathcal{C}} \mathbf{u} = D \nabla^2 c_{\mathcal{C}} + R_{\mathcal{C}} \tag{3.11}$$

where μ is the motility coefficient of the bacteria, i.e., how movable they are and $\mathbf{u}_{tx}$ is the chemotactic velocity, i.e., the velocity with which bacteria move towards the chemoattractant. Note how the two equations are essentially just ARDEs with the one for the bacteria containing an additional advection term due to the chemotaxis. The fluid velocity $\mathbf{u}$ is assumed to be provided externally, possibly by a dedicated LBM simulation. Hilpert accounts for the effect of chemotaxis on the populations in the reported LBM scheme, by adding the chemotactic velocity to the equilibrium distribution function of the bacteria. Equation (3.12) shows this. The equilibrium distribution function for the chemoattractant is the same as the one of the bacteria without the addition of $\mathbf{u}_{tx}$.

$$c^{eq}_{i,\mathcal{B}} = w_i c_{\mathcal{B}} \left(1 + 3\mathbf{c}_i \cdot (\mathbf{u} + \mathbf{u}_{tx})\right) \tag{3.12}$$

Note how the equilibria only contain linear velocity-dependent terms instead of up to second order ones as in Equation (2.13). The reason is that we are not simulating the NSE and thus first order velocity-dependent terms are sufficient to macroscopically recover the ARDE [2, 1].
Hilpert gives a detailed Chapman-Enskog analysis for the reported scheme in order to show that it does indeed recover the desired macroscopic equations with the included chemotaxis. It is further noteworthy, that the simulated system is parametrized with data from experiments and tested for a few different initial bacteria and chemoattractant concentrations. While the results are qualitatively plausible, a quantitative comparison to experimental results could only be made to a different type of chemotaxis setup, which does not yield the numbers observed in the simulation.

In [8] Long and Hilpert continue the work of Hilpert by modifying the LBM reported in [7] to work for 3D as well. However, the overall system of ARDE remains the same and so does the general LBM approach. In addition to the extension to 3D, they also consider advection in this second work. In the initial publication, the scheme's built-in advection term was not used, as all simulations assumed the solute to be at rest. In [8] they combine the simulation of the bacteria and chemoattractant with the simulation of a solvent fluid's flow and show that their model is capable of simulating a bacterial population being injected into water flowing through a complex geometry.

CHAPTER 4

LBM for Reactions - Analysis

If we consider the related literature in Chapter 3, we can observe something noteworthy. A majority of LBM schemes for systems of RDE appears to handle the simulation of M distinct, reacting, species like M distinct single-species simulations, only related via a reaction step operating on the macroscopic concentrations. Hence, no mesoscopic information appears to be exchanged between different species. While this will, as we already mentioned, approximate the desired macroscopic behavior, it does not make use of the LBM's foundation in kinetic theory and the resulting possibility to gain insights into the mescoscopic (and through that maybe microscopic) dynamics. Thus, one of the key advantages the LBM has over top-down solvers is seemingly wasted.
Note that this also includes the few cases where ARDE systems were considered, as it is commonly assumed that the reacting species are too low in concentration to impact the bulk fluid's motion.

Furthermore, if we ignore the reaction terms for a second, we can make a particularly interesting observation: The simulation of the evolution of M distinct species is in most schemes de-facto *exactly* the same, except for varying BGK relaxation times and initial setups. The kinetics of the populations are, more often than not, invariant to another, as the lattice equilibrium distribution and general collision step are the same for all species. This is quite remarkable, considering that the BGK operator and the discretized Boltzmann distribution as equilibrium were introduced in [23] under the assumption of a single-species system. Furthermore, even if one does decide to apply the BGK model for multiple species, the continuous MBD is explicitly depending on the particle mass m_p (which is in general different for distinct species). Thus, it appears physically rather unjustified to use the exact same discretized version of this distribution as equilibrium for all species.

This chapter is devoted to investigating the consequences of treating all the species the same, and to discussing the overarching question of how an LBM for multiple reacting species can contain particle-specific information. First and foremost, we will be interested in incorporating the different particle masses of the simulated species into the LBM, as this is something barely mentioned - and even if so, only in pass-by (for example [27]).

4.1 A Place for the Particle Mass

It is a well-known fact in the LBM community, that the recovered macroscopic behavior depends on the used equilibrium distribution, collision operator and lattice [2]. Since the MBD tells us that the macroscopic behavior of the species (in the form of its particle distribution) depends on the particle-mass, we will investigate these parts of the LBM with the goal of finding a suitable location for the distinct particle mass.
Note that for the classic, single-species, LBM it is not further surprising that this information is not explicitly contained. First, for a single species it is always possible to pick the non-dimensionalization factors such that the explicit dependency on the particle mass vanishes. Second, it can even be considered desirable that such an explicit dependency does not exist. With the LBM being a mesoscopic scheme, operating on populations rather than individual particles, it should ideally not directly depend on microscopic information.
However, for multiple species the scheme has to account for the different physics of particles with different mass. Thus, this information must be contained at some place.

4.1.1 Relationship between Lattice and Mass

Naturally, the discretized equilibrium distribution is the primary candidate for a dependence on the particle mass, considering that such a dependency exists for the continuous version. We will thus analyze f^{eq} in a first step.

Careful readers might be asking themselves what all the fuss is about at this point, as the initially shown discretized equilibrium (Equation (2.13)) clearly depends on the m_p. However, the discretized equilibrium distribution is almost exclusively used in a form where it got simplified according to the used lattice. For example, Equation (4.1) shows the typically used equilibrium distribution for the D2Q9 lattice, where this explicit dependency is no longer observable.
Note that we denote non-dimensionalized quantities (i.e., normalized by a factor that is afflicted with the same physical unit) with an asterisk in the superscript.

Discretized MBD as in [1]

$$f_i^{eq} = w_i \rho^* \left(2 + 6 \cdot \frac{\mathbf{c}_i \mathbf{u}}{c^2} + 9 \frac{(\mathbf{c}_i \mathbf{u})^2}{c^4} - 3 \frac{\mathbf{u}^2}{c^2} \right) \tag{4.1}$$

Considering that Equations (2.13) and (4.1) are both second order discretizations of the MBD, which **only** depends on m_p in an isothermal setting, we would also expect both versions to explicitly contain m_p. However, while this is not the case for Equation (4.1),

it is clear that this dependency must be contained in one of the other parameters.
A seemingly natural place for this information is the non-dimensionalization factor used to obtain ρ^* from ρ. For multiple species one would then assume that the same factor is used for all macroscopic densities, thus resulting in different scalings of f^{eq}. However, as can be seen in, for example, [7], this is not the case. There, Hilpert uses distinct non-dimensionalization factors per species, such that all densities are within the same range.
Furthermore, from a physical point of view, the mass dependency being solely contained in the non-dimensionalization factor of the density is not sensible. Note that with the density's unit containing a unit of mass in its numerator, the scaling factor would need to be inversely proportional to m_p. Further, such a factor would clearly affect all populations the same. Hence, a comparably higher particle mass would result in all populations being reduced the same. However, since each population corresponds to a certain discrete speed, and we know that the distribution is a discretized MBD, we would expect populations with a higher speed to decrease, while populations with a lower speed should increase. We can conclude that a mass dependence is required, that also depends on the respective population's speed.

To find this dependence, the following will derive Equation (4.1) from Equation (2.13) and analyze where the particle mass ends up. Since the steps are virtually the same for all lattice velocity vectors, only the case of $\mathbf{c}_1 = (c, 0)$, i.e., the velocity vector along the x-axis, will be considered.

A first simplification is to account for the specific lattice velocity vector:

$$\begin{aligned} f_1^{eq} &= \frac{w_i}{\rho_0}\left[\rho + \frac{m_p}{k_BT}(c,0)\cdot(\mathbf{j}_x,\mathbf{j}_y) + \frac{m_p}{2\rho k_BT}\left(\frac{m_p}{k_BT}((c,0)\cdot(\mathbf{j}_x,\mathbf{j}_y))^2 - (\mathbf{j}_x,\mathbf{j}_y)^2\right)\right] && (4.2)\\ &= \frac{w_i}{\rho_0}\left[\rho + \frac{m_p}{k_BT}c\cdot\mathbf{j}_x + \frac{m_p}{2\rho k_BT}\left(\frac{m_p}{k_BT}(c\cdot\mathbf{j}_x)^2 - (\mathbf{j}_x^2+\mathbf{j}_y^2)\right)\right] && (4.3)\end{aligned}$$

Next, using the definition of the momentum density $\mathbf{j}$ (Equation (2.5)), the mass density ρ can be factored out (and normalized via ρ_0 to become ρ^*), which leaves the components of the fluid velocity vector in their stead. Furthermore, the lattice speed c can be factored out of the lattice velocity vector components (essentially non-dimensionalizing them). Below these steps are applied.

$$\begin{aligned} f_1^{eq} &= \frac{w_i}{\rho_0}\left[\rho + \frac{m_p}{k_BT}c\rho\mathbf{u}_x + \frac{m_p}{2k_BT}\rho\left(\frac{m_p}{k_BT}\mathbf{u}_x^2 - (\mathbf{u}_x^2+\mathbf{u}_y^2)\right)\right] && (4.4)\\ &= w_i\rho^*\left[1 + \frac{m_p}{k_BT}c\mathbf{u}_x + \frac{m_p}{2k_BT}\left(\frac{m_p}{k_BT}\mathbf{u}_x^2 - (\mathbf{u}_x^2+\mathbf{u}_y^2)\right)\right] && (4.5)\\ &= w_i\rho^*\left[2 + 2\frac{m_p}{k_BT}c^2\mathbf{u}_x^* + \frac{m_p}{k_BT}c^2\left(\frac{m_p}{k_BT}\mathbf{u}_x^{*2} - (\mathbf{u}_x^{*2}+\mathbf{u}_y^{*2})\right)\right] && (4.6)\end{aligned}$$

Note that also a factor of $\frac{1}{2}$ was factored out and moved into the, so far undefined, lattice weights (which will be considered in detail later). Then, we compare the second term of the sum in 4.6 with the corresponding one in Equation (4.1):

$$2\frac{m_p}{k_B T}c^2\mathbf{u}'_x = 6\frac{\mathbf{c}_1\mathbf{u}}{c^2} \tag{4.7}$$

$$\frac{m_p}{k_B T}c^2\mathbf{u}'_x = 3\frac{c^2\mathbf{u}'_x}{c^2} \tag{4.8}$$

$$\frac{m_p}{k_B T} = \frac{3}{c^2} \tag{4.9}$$

$$c^2 = 3\frac{k_B T}{m_p} \tag{4.10}$$

By replacing $\frac{m_p}{k_B T}$ by $\frac{3}{c^2}$ in 4.6 one immediately obtains Equation (2.13). Therefore, the particle mass is contained in the lattice speed. Note that this agrees with the relationship between particle mass and lattice speed stated in [2].
In an isothermal setting, the relation in 4.10 states that species with different particle masses require different lattice velocities. It is therefore of interest to take a closer look at this parameter and to consider the consequences of this relation between lattice speed and particle mass.

When we recall the LBM's streaming step (refer to Section 2.5), populations are modelled to move by the lattice spacing Δx per time step Δt. The velocity with which populations move inside the simulation, c, is therefore simply a consequence of this propagation and can be expressed as

$$c = \frac{\Delta x}{\Delta t} \tag{4.11}$$

Remark: Note that some lattices, according to the above relation, feature multiple lattice speeds. An example for such a *mutli-speed* lattice is the D2Q9 lattice (Figure 2.5), where populations being propagated along the diagonals actually travel $\sqrt{2}\Delta x$ per timestep Δt. Furthermore, also the population at rest has a distinct lattice speed of zero. This is related to the lattice weights, which we will discuss later.

By combining Equation (4.10) and 4.11, we obtain an interesting relationship between the discretization parameters of the lattice and the particle mass:

$$\frac{\Delta x}{\Delta t} = \sqrt{3\frac{k_B T}{m_p}} \tag{4.12}$$

Intuitively this makes sense. The MBD states that lighter particles will, on average, move faster than heavier particles (for the same temperature and total kinetic energy).

We can interpret Equation (4.12) as expressing that a lattice with a certain resolution is only capable of capturing the evolution of a single type of species accurately.
An interesting consequence of this observation is, that one cannot simply simulate two (or more) species $\mathcal{A}, \mathcal{B}$ with different particle masses $m_{p,\mathcal{A}} \neq m_{p,\mathcal{B}}$ and the same equilibrium distribution 4.1 on the same lattice without violating Equation (4.12).
Note that, so far, we only considered the case of all species using the same f^{eq}. In Subsection 4.1.3 we will discuss an alternative approach, where the equilibrium is species-dependent, but the lattice is not. However, in the following the focus will be on the approach where each species has a distinct lattice speed.

4.1.2 Multiple Species on different Lattices

As a consequence of Equation (4.10) we have to consider each species $\mathcal{X}$ to inhabit a dedicated lattice with $c_\mathcal{X} = \frac{\Delta x_\mathcal{X}}{\Delta t_\mathcal{X}}$ in multi-species scenarios. However, since we are ultimately interested in simulating reactions between the distinct species, we must relate these species-specific lattices somehow. They can either be aligned in space (by fixing Δx) or in time (by fixing Δt). We will discuss and interpret both of these possibilities in the following.
For the purpose of illustration we will restrict ourselves to two species, $\mathcal{A}, \mathcal{B}$, with particle masses $m_{p,\mathcal{A}} = k \cdot m_{p,\mathcal{B}}$ and $k > 1$ that react with each other. However, extending these thoughts to an arbitrary amount of species is straightforward.

Fixing Δt

By fixing the lattice time step Δt and using the relation in 4.12 we can observe that 4.13 must hold.

$$\frac{\Delta x_\mathcal{B}}{\Delta x_\mathcal{A}} = \sqrt{\frac{m_{p,\mathcal{A}}}{m_{p,\mathcal{B}}}} \tag{4.13}$$

This relation between the two lattice spacings essentially defines two interleaved lattices with a single spacing each, or, as an alternative interpretation, a single lattice with two different lattice spacings. Figure 4.1 illustrates this for $k = 4$ (i.e. the lattice spacing for the heavier particles of $\mathcal{A}$ will be half the one of species $\mathcal{B}$). The lattice nodes of the $\mathcal{A}$-lattice are drawn as circles, the ones of the $\mathcal{B}$-lattice as rectangles. Furthermore, the lattice velocity vectors of the $\mathcal{B}$-lattice are dashed, whereas the ones for the $\mathcal{A}$ lattice are drawn solid and red.

The populations of the interleaved lattices are streamed on the respective lattice just as they would in a single-species simulation. After each such streaming step, the collision step (BGK) is performed per species. Next, the reaction step is carried out for nodes where both lattices coincide (as only there particles of both species are present).
Consider a system containing more than two species and assume that any kind of two-species collision can result in a reaction. Particles of a lighter species are then, in total,

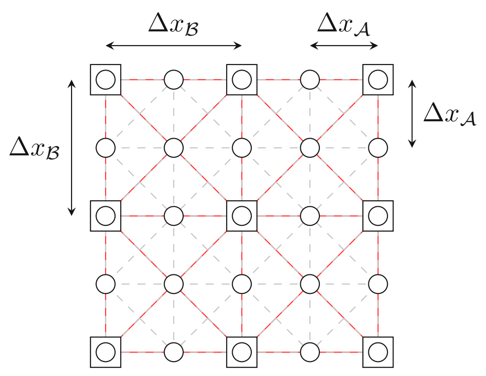


Figure 4.1: Interleaved lattices for the fixed Δt approach with $k = 4$

involved in more potentially reactive collisions than the populations of a heavier species. This is tangible from a physical point of view, as the lighter particles are more likely to hit other particles due to their higher speed.

However, while this approach does incorporate the different particle masses into the simulation in a way that is consistent with Equation (4.10), it comes with some drawbacks, which we will now discuss.
Note that in order to perform reactions, our scheme relies on nodes of the different species' lattices to eventually coincide. Due to the masses being directly related to the lattice spacings, this requirement has consequences for their relation as well. Specifically, we require the ratio of the masses, k, to be a square over $\mathbb{Q}$. Obviously, this severely restricts the possible applications of this scheme. We will discuss this in more detail in 4.1.2.
Another problem of this approach is the initialization of the two lattices' populations. For example, consider the scenario of a reacting front reported by Quian and Orszag in [25]. There the simulation domain is initialized such that one half contains only particles of species $\mathcal{A}$ and the other half only of $\mathcal{B}$. Using just a single lattice, this is nothing remarkable since all nodes are properly initialized. However, using the fixed Δt scheme, some nodes of the heavier species' lattice (smaller lattice spacing) are vacant in the half containing the lighter particles (think about the single lattice with multiple lattice spacings interpretation). Physically speaking, such nodes would have a density of zero and thus model a vacuum which is clearly not desirable.
While this can be fixed by introducing an additional species, with the same Δx as the heavier species, as an inert bulk fluid, this increases the computational cost without increasing the accuracy of the model.
Another disadvantage of simulating multiple lattices is that the obtained results are only accurate for the coarsest spatial discretization (biggest Δx), while the computational complexity is dominated by the lattice with the smallest spacing. Therefore, the simulation time needed by our scheme scales with $\sqrt{\frac{m_{p,\mathcal{A}}}{m_{p,\mathcal{B}}}}$.
Finally, we observed instabilities in the simulation which relate to the spacing of the

largest lattice.

It must be pointed out that we are not the first to consider the relationship between the lattice speed and the particle mass and its consequences on multi-species simulations. In particular, in [42] McCracken and Abraham also draw the conclusion that one way to account for different particle masses is by allowing different lattice spacings. Indeed, they propose a scheme, which they fittingly call *Different Lattice Speed* (DLS), in which species with different particle masses are streamed for distances related via 4.13.
However, instead of only performing interactions between species at coinciding lattice nodes, they deploy a second order interpolation scheme to compute the missing populations of the other species.
The authors state that this interpolation is beneficial for the numerical stability and also relaxes requirements on the resolution / discretization of the LBM. Furthermore, due to this interpolation, their scheme is not restricted to special ratios between the particles masses. It is therefore applicable to a greater set of problems. However, it should also be pointed out that the interpolation is a non-local computation (and due to its second order nature more expensive than the streaming), thus handicapping the LBM's possible parallelization.

Fixing Δx

As mentioned before, instead of fixing Δt we could also opt for a single Δx. As a consequence of 4.12 each species $\mathcal{X}$ would then require a distinct time step $\Delta t_{\mathcal{X}}$. We can write down the dual relation to the one of 4.13:

$$\frac{\Delta t_{\mathcal{B}}}{\Delta t_{\mathcal{A}}} = \sqrt{\frac{m_{p,\mathcal{B}}}{m_{p,\mathcal{A}}}} \tag{4.14}$$

This expresses that a smaller time step is needed for lighter particles, which is sensible considering that the MBD states that lighter particles move, on average, faster than heavier ones.
While the two approaches, fixing either Δt or Δx, appear very similar, they are conceptually quite different due to their duality with respect to space and time. In the fixed Δt scheme, we always know where the different species' populations would be in the simulation domain. It is just that the sets of nodes a species can possibly occupy, differ. In the fixed Δx scheme, all species share a single set of nodes, but their populations will not be at elements of this set at the same points in time. We can combat this by using an approach similar to having a single lattice with multiple Δx. We can understand the fixed Δt approach as having a micro lattice spacing Δx_μ and a species' nodes being distanced by certain multiples of Δx_μ. Interspecies interaction happens at nodes with the same distance, in multiples of Δx_μ, from an arbitrary reference node where all species coincide.
As the dual scheme for the fixed Δx method, we define a micro time step Δt_μ that divides all $\Delta t_{\mathcal{X}}$ (we will discuss later under what requirements such a Δt_μ exists) and

use it to advance the simulation time. At each step, the simulation then decides per species whether its time step is a divider of the current time. If it is, the collision and streaming steps for the respective species' populations are performed. After that, the reaction step between these species and *all* other species is performed. Species that are not streamed do not *cause* reactions.
Figure 4.2 illustrates the scheme by depicting the timeline between $t, t + 3\Delta t_{\mathcal{A}}$ for the case of $k = 4$ (i.e., $\Delta t_{\mathcal{A}} = 2\Delta t_{\mathcal{B}}$). The ticks of this timeline mark the distinct simulation steps. For each such step it is shown which species stream and collide and if a reaction between $\mathcal{A}, \mathcal{B}$ is performed.

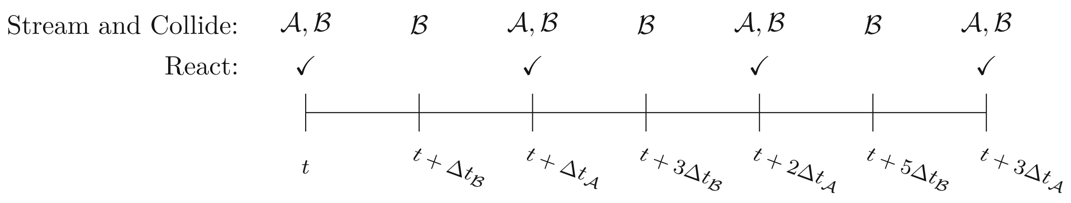

Stream and Collide:
React:
$\mathcal{A}, \mathcal{B}$
$\mathcal{B}$
$\mathcal{A}, \mathcal{B}$
$\mathcal{B}$
$\mathcal{A}, \mathcal{B}$
$\mathcal{B}$
$\mathcal{A}, \mathcal{B}$
✓
✓
✓
✓
t
$t + \Delta t_{\mathcal{B}}$
$t + \Delta t_{\mathcal{A}}$
$t + 3\Delta t_{\mathcal{B}}$
$t + 2\Delta t_{\mathcal{A}}$
$t + 5\Delta t_{\mathcal{B}}$
$t + 3\Delta t_{\mathcal{A}}$

Figure 4.2: Timeline section for the fixed Δx scheme with $k = 4$

In this scheme the streaming and collision steps are performed more often on lighter particles than on heavier ones. The same holds for the reaction step. For the streaming this is quite natural, considering that, according to the MBD, lighter particles are (in an isothermal setting) on average faster than slower ones. Thus, if all particles move by the same Δx per time step, faster particles need to stream more often.

Due to the underwhelming results and drawbacks of the fixed Δt approach, and the duality of it to this scheme, we did not further pursue the fixed Δx method.

Requirements on the Relations between Masses

In both of the above two schemes, we introduced a micro lattice parameter. For both, fixed Δx and fixed Δt, this parameter was assumed to coincide with the smallest lattice time step, respectively spacing, of any species. The reason being, that this is the smallest value for the respective parameter by which the simulator can sensibly work.
However, while such a micro-lattice parameter allows illustrating both schemes in a very concise way, it severely restricts the possible combinations of particle masses that can be simulated together. Essentially, the existence of such a micro parameter requires the respective lattice parameter of all other species to be a multiple of it. Recalling the relations in 4.13 and 4.14 this means that k would need to be a square over $\mathbb{N}$.

However, in order for the different lattices' nodes, or time steps, to eventually coincide, the existence of such a micro parameter is not strictly necessary. For example, consider Figure 4.3 which shows a similar interleaved lattice for the fixed Δt approach as Figure 4.1, but with $k = 9/4$, i.e., $2\Delta x_{\mathcal{B}} = 3\Delta x_{\mathcal{A}}$, instead of $k = 4$. Obviously, there are still coinciding nodes and the scheme can be applied as before.

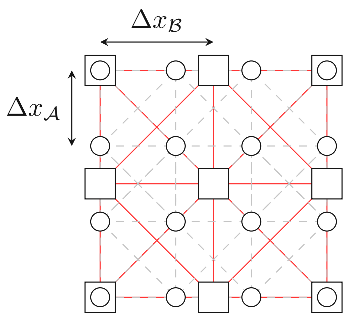


Figure 4.3: Interleaved lattices for the fixed Δt approach with $k = 3/2$

Similarly, a micro time step for the fixed Δx scheme is not strictly required. Instead, the simulator could order the distinct Δt values in an increasing manner and always advance to the next multiple of the different time steps, increasing the multiplicity after each pass through this ordered list.
In both cases, the requirement on the lattice parameters is that their relation is an element of $\mathbb{Q}$, i.e., that there exist multiples of them that coincide. This relaxes the requirement on the mass ratio k slightly, as it is now required to be a fraction of squares over $\mathbb{N}$ instead of being such a square itself:

$$\frac{\Delta x_{\mathcal{B}} \Delta t_{\mathcal{A}}}{\Delta x_{\mathcal{A}} \Delta t_{\mathcal{B}}} = \sqrt{\frac{m_{p,\mathcal{A}}}{m_{p,\mathcal{B}}}} = \sqrt{k} = \frac{q}{r}, \quad q, r \in \mathbb{N} \tag{4.15}$$

However, this is obviously still a fatal restriction that renders both of these schemes practically useless.

4.1.3 Multiple Species on the same Lattice

With both, our approaches and reports from literature, showing severe problems with combining different lattices, an approach that uses a single lattice, thus retaining the LBM's key algorithm and properties, would be much desired. Naturally, the question whether such a single-lattice method exists arises. We will attempt to answer this question in the following.

The Equilibrium Distribution

Consider a multi-species setup with varying particle masses, where we want to use a single lattice, i.e., a single combination of $\Delta x, \Delta t$ (and thus lattice speed) for *all* species. In the previous subsection we have argued that doing that is not possible. However, this

was based on the assumption that the equilibrium distribution of all species is the same. If we use distinct, species-dependent, equilibrium distributions, we might be able to incorporate different masses. From a physical point of view this is absolutely reasonable, as this ultimately what we assume to happen in the “real” world as well: In an isothermal setting, each species’ particles relax to a *distinct* Maxwell-Boltzmann distribution.
However, as the discretized form(s) of the MBD (Equations (2.13), (4.1)) do not bear too much resemblance to their continuous pendant at first glance, we will briefly discuss its derivation in the hope of gaining insights into possible places for a particle mass dependence.

First we want to note though, that there is more than just a single way of deriving the discretized MBD. Whereas some authors [43] argue that the most rigorous way of doing so is by using the similarity of the continuous MBD to the weight function of the, so-called, *Hermite polynomials* (interested readers be referred to [1]), we will follow Wolf-Gladrow’s approach [2], which we found to be more insightful, in this work.

Wolf-Gladrow starts by defining a global equilibrium distribution per population, $f_i^{eq} = w_i$, for the case where the fluid is macroscopically at rest (i.e. $\mathbf{u} = 0$). In order to determine this global equilibrium, we require it to “behave the same” as the MBD in the proof that shows that the LBM does indeed recover the NSE. As Viggen elaborates in his PhD thesis [44], by this we mean, that the zeroth up to the third moments of the discrete equilibrium distribution (sums of the form $\sum_{i=0}^{Q} \mathbf{c}_{i,\alpha} \cdot \mathbf{c}_{i,\beta} \cdots f_i^{eq}$) must match the corresponding ones of the continuous MBD. This results in a set of constraints for w_i, shown in Equations (4.16)-(4.18). Note that f^{MB}denotes the continuous MBD.

$$\sum_{i=0}^{Q} f_i^{eq} = \int f^{MB}(\mathbf{v})d\mathbf{v} = \rho_0 \tag{4.16}$$

$$\sum_{i=0}^{Q} \mathbf{c}_\alpha \mathbf{c}_\beta f_i^{eq} = \int f^{MB}(\mathbf{v})d\mathbf{v} = \rho_0 \frac{k_B T}{m_p} \delta_{\alpha\beta} \tag{4.17}$$

$$\sum_{i=0}^{Q} \mathbf{c}_\alpha \mathbf{c}_\beta \mathbf{c}_\gamma \mathbf{c}_\delta f_i^{eq} = \int f^{MB}(\mathbf{v})d\mathbf{v} = \rho_0 \left(\frac{k_B T}{m_p}\right) (\delta_{\alpha\beta}\delta_{\gamma\delta} + \delta_{\alpha\gamma}\delta_{\beta\delta} + \delta_{\alpha\delta}\delta_{\beta\gamma}) \tag{4.18}$$

In addition to the above we require sufficient symmetry of the lattice velocity vectors such that the moments of odd order vanish (all lattices used for the LBM satisfy this).
Using these constraints, and a set of lattice velocity vectors (to be exact, we just require the directions, not lattice speed c), we can determine the values of w_i, which will call the *lattice weights*. Furthermore, as we will see, we will also be able to obtain the relation shown in 4.10 in a more natural way. To illustrate this, we will now do this for the D2Q9 lattice.

Recall the D2Q9 lattice, shown in Figure 2.5. Its velocity vectors can be defined as in (4.19)

$$\mathbf{c}_i = \begin{pmatrix} 0 & 1 & -1 & 0 & 0 & 1 & 1 & -1 & -1 \\ 0 & 0 & 0 & -1 & 1 & -1 & 1 & 1 & -1 \end{pmatrix} \tag{4.19}$$

Using these vectors, the general constraints of (4.16)-(4.18) and the equilibrium $f_i^{eq} = w_i$, we can formulate the lattice-specific constraints, shown in Equations (4.20)-(4.23). Note that we introduced a handy observation in order to reduce the number of unknowns: We know that f^{eq} must, within the discretization, correspond to the MBD. As this distribution is invariant to the orientation of a particle's velocity (the isothermal MBD just depends on the speed), we can conclude that the discrete velocities $c_{1,2,3,4}$ and $c_{5,6,7,8}$ must each share the same weight (we will refer to them as w_1, respectively w_2).

$$w_0 + 4w_1 + 4w_2 = \rho_0 \tag{4.20}$$

$$2c^2 w_1 + 4c^2 w_2 = \rho_0 \frac{k_B T}{m_p} \tag{4.21}$$

$$4c^4 w_2 = \rho_0 \left(\frac{k_B T}{m_p}\right)^2 \tag{4.22}$$

$$2c^4 w_1 + 4c^4 w_2 = 3\rho_0 \left(\frac{k_B T}{m_p}\right)^2 \tag{4.23}$$

Solving this set of equations results in the following:

$$c^2 = 3\frac{k_B T}{m_p}, \quad w_0 = \frac{4\rho_0}{9}, w_1 = \frac{\rho_0}{9}, w_2 = \frac{\rho_0}{36} \tag{4.24}$$

Note that (4.10) can therefore also be considered to be a constraint on the discretization of the velocity space such that the bare essence of the MBD is captured by its discretized version.

While the lattice weights appear a bit arbitrary at first glance, we can, to some extent, attempt to interpret them from a physical point of view: Note how the discretized equilibrium distribution for a resting fluid essentially just re-distributes the density according to the different populations. The shares each population gets during this re-distribution depends on the lattice weights, which in turn depend on the respective population's associated lattice velocity. Now, if the discretized equilibrium is to be *Maxwellian*, these weights have to result in this distribution adhering to the MBD.

If we consider the weights of the D2Q9 lattice in (4.24), we can observe that populations corresponding to a lattice velocity vector with higher magnitude get assigned less of the

total amount of the populations. In the case of a single species, and c being constrained to satisfy 4.10, this directly reflects the MBD, as c is then directly related to the velocity at which the MBD has its peak. Therefore, the diagonal speeds of magnitude $\sqrt{2}c$ in D2Q9 are “less probable”, resulting in lower lattice weights.
However, also note the comparably high weight of the rest population. In order to somehow interpret this in a reasonable manner, we consider the weights as a partitioning of the area under the MBD, and the respective populations are associated to these areas via their velocity magnitude. For example, the rest population then relates to the area under the distribution from $\mathbf{v} = 0$ until $\mathbf{v}'$ such that $\int_0^{\mathbf{v}'} f^{MB} dv = w_0$ (for the non-rest velocities we would multiply the right-hand side of this equation by the amount of lattice velocity vectors sharing the particular weight).

A Naïve Approach

Considering the above, it might appear sensible to constraint the lattice speed for one species, and then use the masses of the other species when deriving their respective equilibrium distributions in order to obtain species-specific lattice weights.
For example, for a setup on D2Q9 with two species $\mathcal{A}, \mathcal{B}$ and $m_{p,\mathcal{B}} = k m_{p,\mathcal{A}}$ we would set the lattice speed to $c = \sqrt{\frac{3k_BT}{m_{p,\mathcal{A}}}}$ and then compute lattice weights for $\mathcal{B}$, $w_i^{\mathcal{B}}$, via the following:

$$w_0^{\mathcal{B}} + 4w_1^{\mathcal{B}} + 4w_2^{\mathcal{B}} = \rho_0^{\mathcal{B}} \tag{4.25}$$

$$2c^2 w_1^{\mathcal{B}} + 4c^2 w_2^{\mathcal{B}} = \rho_0^{\mathcal{B}} \frac{k_BT}{k m_{p,\mathcal{A}}} \tag{4.26}$$

$$4c^4 w_2^{\mathcal{B}} = \rho_0^{\mathcal{B}} \left(\frac{k_BT}{k m_{p,\mathcal{A}}}\right)^2 \tag{4.27}$$

$$2c^4 w_1^{\mathcal{B}} + 4c^4 w_2^{\mathcal{B}} = 3\rho_0^{\mathcal{B}} \left(\frac{k_BT}{k m_{p,\mathcal{A}}}\right)^2 \tag{4.28}$$

However, when attempting to solve this over-determined system of equations, one quickly ends up with a contradiction unless $k = 1$. We can conclude that, on a lattice constrained for a species with a different particle mass, the typically used f^{eq} is **not** the discretized MBD of the respective species! However, this is exactly what a lot of reported works do. At this point we ought to remark that we are not the first ones to observe this [42, 45, 43].

Remark: Note that this is also the case if we use a simpler lattice, say D2Q5, which only contains two different types of velocity vectors (and thus weights).

Species-Specific Equilibrium Distribution

Before we continue with the discussion of incorporating the particle mass in species-specific equilibrium distributions, let us briefly extend f^{eq} from the resting case to the one where $\mathbf{u} \neq 0$.

In [2], Wolf-Gladrow does this by proposing an ansatz for the equilibrium distribution that is based on observations from LGCA and has the form given in Equation (4.29), where A_i, B_i, C_i, D_i are coefficients to be constrained such that the desired macroscopic PDE is recovered.

$$f^{eq} = w_i \left(A_i + B_i \mathbf{c}_i \cdot \mathbf{u} + C_i \left(\mathbf{c}_i \cdot \mathbf{u} \right)^2 + D_i \mathbf{u}^2 \right) \tag{4.29}$$

Wolf-Gladrow uses this ansatz in a Chapman-Enskog multi-scale analysis to determine how the coefficients need to be constrained such that an LBM using the BGK operator with this f^{eq} results in the NSE. The result is the the equilibrium distribution given in Equation 4.1.
Note that this ansatz still results in the equilibrium for a fluid at rest with $\mathbf{u} = 0$. Therefore, the naïve approach will also result in a constraint violation if we use this distribution.

As we observed above, the ultimate culprit of modifying the equilibrium for different species is that the resulting system of equations is over-determined. Therefore, we might just need to introduce more degrees of freedom in order to solve it.
This is exactly what Looije *et al.* do in [43]. In their publication, they suggest using the D2Q13 lattice instead of D2Q9, which extends the velocity space by introducing four additional velocity vectors (of the same magnitude). Figure 4.4 depicts this lattice, where the additional velocity vectors are drawn in bold and red, whereas the ones already present in D2Q9 are dashed and gray.

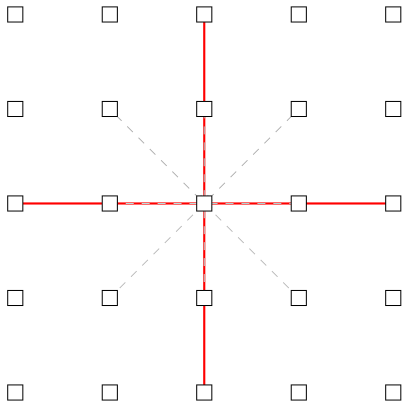


Figure 4.4: The D2Q13 lattice

By introducing these additional lattice velocities, we gain an additional lattice weight which we will denote as w_3.
Now assume that we constrain a lattice for a species $\mathcal{A}$, such that its equilibrium is the one in Equation 4.1 and that $c^2 = 2\frac{k_B T}{m_{p,\mathcal{A}}}$ (note that the D2Q13 lattice has a slightly different lattice speed). The lattice weights for $\mathcal{A}$ are then $w_0 = 3/8, w_1 = 1/12, w_2 = 1/16, w_3 = 1/96$.

For a species $\mathcal{B}$ with $m_{p,\mathcal{B}} = xm_{p,\mathcal{A}}$, we now assume an equilibrium of the form suggested by Looije *et al.*, shown in (4.30), which is just a special form of the ansatz proposed by Gladrow (c_s is the, so-called, lattice speed of sound which we will discuss later).

$$f^{eq} = w_i\rho\left(A_i + B_i\frac{\mathbf{c}_i\cdot\mathbf{u}}{c^2} + \frac{(c_{i,\alpha}c_{i,\beta} - c_s^2\delta_{\alpha\beta})\mathbf{u}_\alpha\mathbf{u}_\beta}{2c^4}\right) \tag{4.30}$$

If we now set up a system of equations, based on a version of the constraints from (4.16)-(4.18), that accounts for **u** (shown in (4.31)-(4.34)), we can actually derive coefficients for $\mathcal{B}$'s equilibrium distribution without violating the constraints for achieving Maxwellian behavior of f^{eq}.

$$\sum_{i=0}^{Q} f_i^{eq} = \rho \tag{4.31}$$

$$\sum_{i=0}^{Q} \mathbf{c}_\alpha f_i^{eq} = \rho\mathbf{u}_\alpha \tag{4.32}$$

$$\sum_{i=0}^{Q} \mathbf{c}_\alpha\mathbf{c}_\beta f_i^{eq} = \rho\left(c_{s,\mathcal{B}}^2\delta_{\alpha\beta} + \mathbf{u}_\alpha\mathbf{u}_\beta\right) \tag{4.33}$$

$$\sum_{i=0}^{Q} \mathbf{c}_\alpha\mathbf{c}_\beta\mathbf{c}_\gamma\mathbf{c}_\delta f_i^{eq} = \rho c_{s,\mathcal{B}}^2\left(\mathbf{u}_\alpha\delta_{\beta\gamma} + \mathbf{u}_\beta\delta_{\alpha\gamma} + \mathbf{u}_\gamma\delta_{\alpha\beta}\right) \tag{4.34}$$

Specifically, if we apply the above constraints to the equilibrium distribution of $\mathcal{B}$, we get the system of equations in (4.35)-(4.39), plus some additional constraints which are satisfied by the symmetry of the D2Q13 lattice (and virtually any other LBM lattice for that matter).

$$\frac{3A_0}{8} + \frac{A_1}{3} + \frac{A_2}{4} + \frac{A_3}{24} = 1 \tag{4.35}$$

$$\frac{A_1}{3} + \frac{A_2}{4} + \frac{A_3}{24} = \frac{k_BT}{m_{p,\mathcal{B}}} \tag{4.36}$$

$$\frac{B_1}{3} + \frac{B_2}{2} + \frac{B_3}{6} = \frac{k_BT}{m_{p,\mathcal{A}}} \tag{4.37}$$

$$\frac{2B_1}{3} + B_2 + \frac{4B_3}{3} = 3\frac{k_BT}{m_{p,\mathcal{A}}}\cdot\frac{k_BT}{m_{p,\mathcal{B}}} \tag{4.38}$$

$$\frac{B_2}{4} = \frac{k_BT}{m_{p,\mathcal{A}}}\cdot\frac{k_BT}{m_{p,\mathcal{B}}} \tag{4.39}$$

Solving this system, we can arrive at the mass-dependent coefficients reported by Looije and co-authors, shown in (4.40)-(4.43), where the parameter $r = 3 - 2x$ contains the mass-dependence.

$$A_0 = \frac{4}{9-3r}\left(\frac{4+8r-3r^2}{2+3r+r^2}\right) \qquad B_0 = 0 \tag{4.40}$$

$$A_1 = \frac{12}{6+7r-r^3} \qquad B_1 = \frac{6-4r}{3-r} \tag{4.41}$$

$$A_2 = \frac{12r}{6+7r-r^3} \qquad B_2 = \frac{2}{3-r} \tag{4.42}$$

$$A_3 = \frac{12r^2}{6+7r-r^3} \qquad B_3 = \frac{2r}{3-r} \tag{4.43}$$

Admittedly, these coefficients look rather unwieldy. And indeed, Looijen *et al.* reported that they were not able to obtain numerically stable solutions for a relative mass difference of more than four. Nevertheless, it should be mentioned that Van der Akker and co-auhors reported success in applying this scheme to multi-species scenarios with limited relative difference in mass (below 3), showing good agreement between their simulation and analytical solutions [45].

However, while we can conclude that equilibrium distributions with an explicit dependence on the particle mass are possible, and allow multiple distinct species on the same lattice, the possible use cases are limited by the supported range of relative masses. Considering that our targeted biological systems might require a wide range of particle masses, we will therefore need to continue our investigations in the future.

4.2 Non-Reactive Interspecies Collisions

As we motivated in the introduction to this chapter, the interactions between species in the considered related work are typically reflected by the reactive term only. This term typically is, with a few exceptions, a macroscopic reaction model applied at the mesoscopic nodes. It must be pointed out that these reaction terms apply to mixtures of species that are in thermodynamic equilibrium, i.e., that are well-mixed such that their macroscopic concentrations have no tendency to change spatially. This is definitely not the case for the problems we desire to treat with the LBM, since we are interested in the dynamic behavior before the thermodynamic equilibrium is reached. Furthermore, an external velocity field that models, for example, a stirring will also cause spatial differences in the concentrations.
To accommodate for this, while still being able to apply the macroscopic reaction kinetics, it is often assumed that, instead of the whole domain, each lattice node is well-stirred [28, 25] and that there are sufficient particles per node to justify the use of macroscopic laws *per* node. However, since the node concentrations are the most fine-grained spatial information contained in the LBM, the assumption of well-stirred nodes is arguably somewhat axiomatic - but not per se dismissible. Ideally, one would argue on a case-to-case basis if the resolution of the chosen lattice, and the amounts of reacting species contained in the simulation, justify this assumption.

In addition to these assumptions, there is another catch with using the macroscopic reaction models. Usually, such reaction kinetics only aim to quantify the effects *reactive* collisions have on the concentrations of the reacting species (for example mass action kinetics, Section 5.2.5). However, it is known that not all interspecies collisions do end up reacting [12, 37] and, recalling the premises of kinetic theory (Section 2.5.1), that such non-reacting collisions between particles of different species will have an effect on the post-collision trajectories and velocities. As the collisions between particles result in a relaxation towards their equilibrium distribution, we would expect for these effects of non-reactive collisions between species to precipitate in this relaxation, and maybe also the final distribution itself.
As this behavior is modelled by the non-reactive collision operator, we expect it to account for interspecies collisions in a multi-species scenario.

In this section, we will investigate whether this expectation holds true for the BGK collision operator, in conjunction with the discretized MBD as equilibrium distribution, as this is what is used in a majority of cases.

4.2.1 Investigation of the BGK Operator using an MD Simulator

As already mentioned, in the context of multi-species LBMs, one often observes the BGK operator being applied on a per-species basis, seemingly indifferent to the existence of other species. The relaxation time τ then only depends on properties of the respective species and the equilibrium distribution is simply the discretized form of the MBD. This models an invariance of the dynamic behavior of each species' populations to non-reacting interspecies collisions.
We found this approach to be worthy of investigation, particularly due to the BGK operator assuming a single species, and also the MBD only depending on the properties of a single type of particle. In order to perform this investigation, we implemented a dedicated multi-species MD simulator. Note that this simulator is capable of operating on completely unit-afflicted quantities by using the unit system *Pint* [46].

MD Simulator

Recall from Chapter 2 that MD schemes try to simulate the interactions between particles with the level of discretization kept low, and a collision model, compliant with classical mechanics, being applied to distinct particles.
Our simulator operates in two-dimensional space and supports circular particles of arbitrary many species. Each particle has an individual position, velocity, as well as a species-dependent, mass and radius. All these quantities can take arbitrary values within machine precision.

The simulator advances the simulation time by an arbitrary, user-defined, time step. Per such step, it checks for all pairs of particles, whether they collide within this time step (by comparing the distance between their next positions due to their movement within a time step). If a collision takes place, we assume it to be perfectly elastic, i.e., to keep the total kinetic energy and momentum of the colliding particles unchanged. However, the two particles themselves can, and typically will, have a different kinetic energy and momentum after their collision. Essentially, collisions between particles of mass $m_{p,\mathcal{A}}$, moving with $\mathbf{v}_1$, respectively of $m_{p,\mathcal{B}}$ and moving with $\mathbf{v}_2$ satisfy the following two equalities

$$m_{p,\mathcal{A}}\mathbf{v}_1 + m_{p,\mathcal{B}}\mathbf{v}_2 = m_{p,\mathcal{A}}\mathbf{v}'_1 + m_{p,\mathcal{B}}\mathbf{v}'_2 \tag{4.44}$$

$$\frac{m_{p,\mathcal{A}}\mathbf{v}_1^2}{2} + \frac{m_{p,\mathcal{B}}\mathbf{v}_2^2}{2} = \frac{m_{p,\mathcal{A}}{\mathbf{v}'_1}^2}{2} + \frac{m_{p,\mathcal{B}}{\mathbf{v}'_2}^2}{2} \tag{4.45}$$

where the post-collision velocities are primed. After each such collision step, the simulator moves all particles according to the specified time step and their respective velocity.
Note that our simulator is based around a quadtree for maintaining the particles, in order to reduce the amount of required collision checks. Thus, an iteration of the simulator completes with splitting and merging of quadtree nodes to ensure that the tree corresponds to the state of the particles after their movement.

Simulation Results

We validated our MD simulator by considering scenarios where the simulation domain is initialized to contain a few hundred particles of a single species around a user-defined temperature. We use this temperature, and the relation between it and the average kinetic energy of particles according to kinetic theory (Equation (B.2)), to determine the respective average speed. Then, our simulator populates the simulation domain by pseudo-randomly sampling from a uniform distribution over [0,1), in order to determine random particle positions, speeds and orientations. This is done in a way such that the mean particle speed is the one dictated by the specified temperature. However, note that due to random initialization this temperature is just a reference value. The real temperature will be determined once the simulation got initialized.
Figure 4.5 shows the speed distribution of particles, as obtained by our MD simulator, after initialization and after 300 time steps.
We set up the simulation to contain 250 particles with a mass of one atomic unit and a radius of 2 Bohr. The simulation domain was set to be rectangular with dimensions of 200×200 Bohr and the time step to be 0.1 ps. The temperature after the random initialization was determined to be around 126 K. We track the speed distributions using a histogram with 40 bins. As we want to compare the obtained results to the probability according to the MBD, we normalized the bin data to sum up to one.
Note that the figure shows the average probability of particles being in the respective speed range over ten consecutive iterations.

Observe how the particle speed distribution quickly evolves towards the MBD, as one would expect. Note that we were able to observe this behavior for different particles masses, temperatures, domain sizes and numbers of particles.

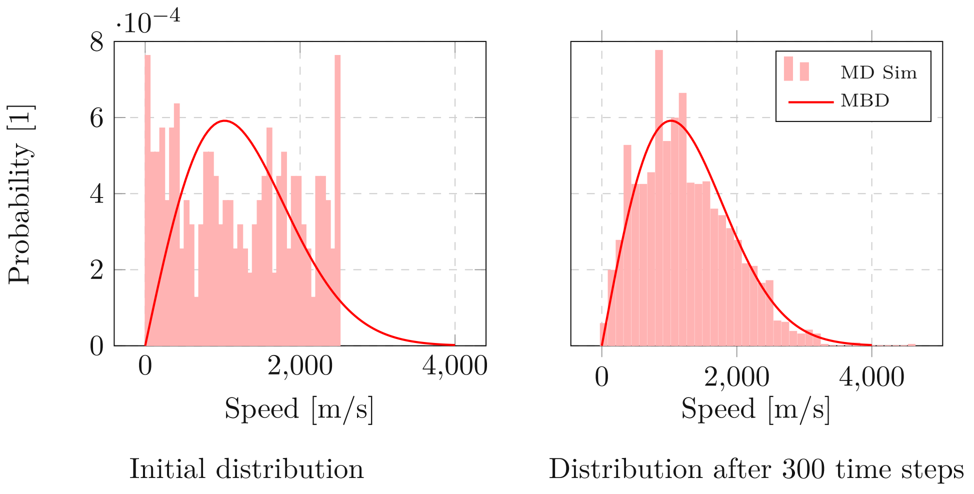


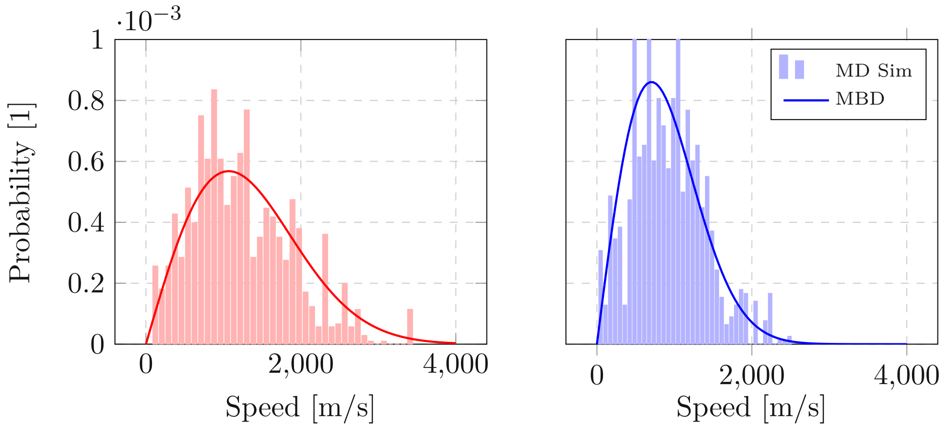
Figure 4.5: Single species particle speed distribution of our MD simulator and the MBD

After validating that the implemented particle model adheres to the MBD, we simulated multi-species systems. The species were configured to feature different particle masses and radii, as we were interested in investigating whether the size of the species' particles has an impact on the evolution of the distributions. This is important to us, as we are ultimately interested in biological systems containing bacteria, which are significantly bigger than, for example, nutrients.

Figure 4.6 shows the particle speed distribution of a two-species simulation after 300 time steps plotted against the respective MBDs. We kept the settings similar to the ones for the single-species case reported above, but instead of using 250 particles of the same species, we used 125 particles of a species $\mathcal{A}$ with $m_{pm\mathcal{A}} = 1u$ and 125 of a species $\mathcal{B}$ with $m_{p,\mathcal{B}} = 2u$. The radii of the two species were both set to 2 Bohr. Observe how the two distributions relax to the respective MBD.

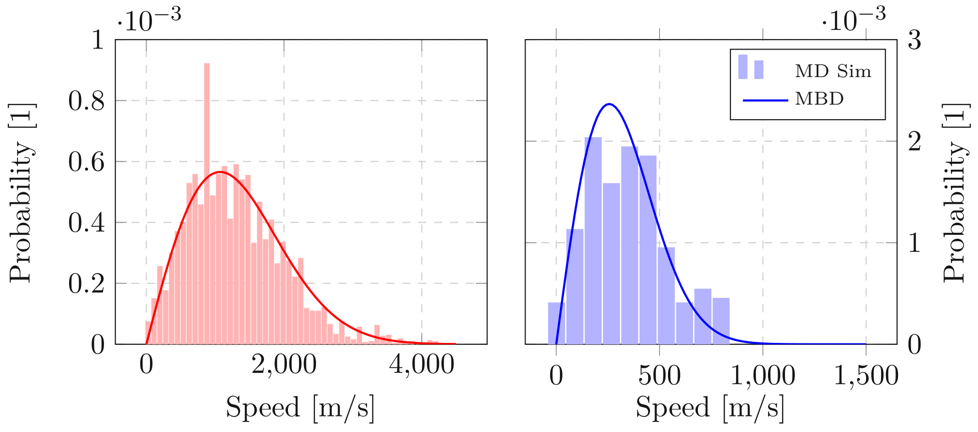
Again, Figure 4.6 is only exemplary for multiple different setups we simulated, with varying parameters. However, the ultimate result of all these simulations was that a species' particles relax towards the respective MBD.

Next, we considered cases where the distinct species' particles have different radii. However, in all our simulations the species did end up to relax to the MBD just as before. To mitigate repetition, we therefore omit printing corresponding figures.
Finally, we considered cases where there are many small, light particles and, in comparison, few heavy, big ones, as this is in principle reminiscent of the biological systems we are ultimately interested in. Figure 4.7 shows the resulting particle speed distributions (and for illustration also an image of the simulated particles) after 100 time steps. The simulation contained 400 particles of a species $\mathcal{A}$ with $m_{p,\mathcal{A}} = 1u$ and a radius of 2 Bohr and 25 particles of species $\mathcal{B}$ with $m_{p,\mathcal{B}} = 15u$ and a radius of 15 Bohr. Remarkably, the particles, although differing by an order of magnitude in their masses and radii, and by two orders of magnitude in their numbers, relax towards the equilibrium of their species.



$m_{p,\mathcal{A}} = 1u$ $m_{p,\mathcal{B}} = 2u$

Figure 4.6: Particle speed distributions in a multi-species setup after 300 time steps



$m_{p,\mathcal{A}} = 1u$, radius 2 Bohr $m_{p,\mathcal{B}} = 15u$, radius 15 Bohr

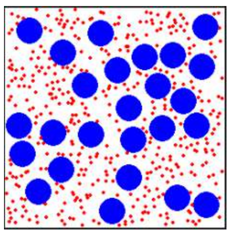


Illustration of the MD simulation domain

Figure 4.7: Particle speed distributions after 100 time steps for $\frac{m_{p\mathcal{A}}}{m_{p\mathcal{B}}} = \frac{1}{15}$ and different radii

While we conclude from our simulations above, that taking the MBD as equilibrium for multi-species scenarios does indeed appear to be reasonable, we have not discussed the other component of the BGK collision operator, namely τ, so far.
As the relaxation time expresses how fast the speed distribution of a species' particles converges towards its equilibrium due to collisions, we would expect it to be affected by collisions with particles of other species.
To investigate this, we need a way to quantify the time it takes for a distribution to relax to its equilibrium. For this, we apply the χ^2 likelihood test on the distributions obtained by the simulator and the respective continuous MBD evaluated at the histogram's bin speeds. We perform this likelihood test per simulation step and record the results.
Equation (4.46) shows the formula used to compute χ^2, where t denotes the simulation time, $\texttt{sim}_t[j]$ the (normalized) value of bin j of the simulated particles' speed distribution at time t, and $f(\mathbf{v}[j])$ the continuous MBD evaluated at the mean speed of bin j. N is the total number of bins.

$$\chi^2(t) = \sum_{j=0}^{N} \frac{(\texttt{sim}_t[j] - f(\mathbf{v}[j]))^2}{f(\mathbf{v}[j])} \tag{4.46}$$

In Figure 4.8 the decrease of this likelihood for single-species setups with different particle masses is shown. For all depicted cases, we simulated 400 particles with the masses stated in the plot's legend, a radius of 2 Bohr and a simulation domain of 250×250 Bohr. Each iteration of the simulator corresponds to a time step of 0.1 ps. All the plotted results are the average of three equivalently parametrized setups with different random initialization.

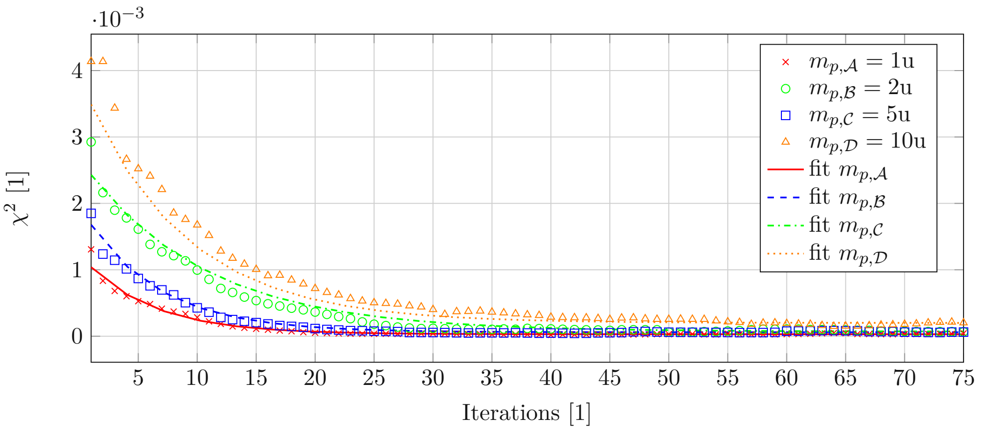


Figure 4.8: χ^2 during single-species relaxation to equilibrium for various m_p

Observe how the χ^2 values decrease towards a stationary value within a few dozen iterations, increasing in correspondence to the respective particle mass. This makes sense if we consider the MBD, as heavier particles will move slower on average, and thus collide less frequently. With collisions being responsible for the evolution towards the equilibrium, fewer collisions should result in a slower relaxation.

Furthermore, our recorded results suggest an exponential relaxation towards the equilibrium. To back up this observation, we fitted an exponential decay function to each data series. The results are also contained in Figure 4.8. Observe how well the fitted exponential functions and the recorded data align. This is in itself a noteworthy result, as the BGK collision operator ultimately models the effect of the collisions as an exponential decay towards f^{eq}. We were thus able to observe the validity of the BGK approximation in numerical experiments, based on the assumptions of kinetic theory.

We then continued to use our simulator to investigate whether the relaxation of a species' particles *towards* its equilibrium is sensitive to collisions with particles of another species. Figure 4.9 depicts the obtained results for a similar setup as for Figure 4.8, but with the 400 particles being partitioned into 200 particles of a species $\mathcal{A}$ with $m_{p,\mathcal{A}} = 1u$ and 200 particles of a species $\mathcal{B}$ with varying particle mass $m_{p,\mathcal{B}}$.

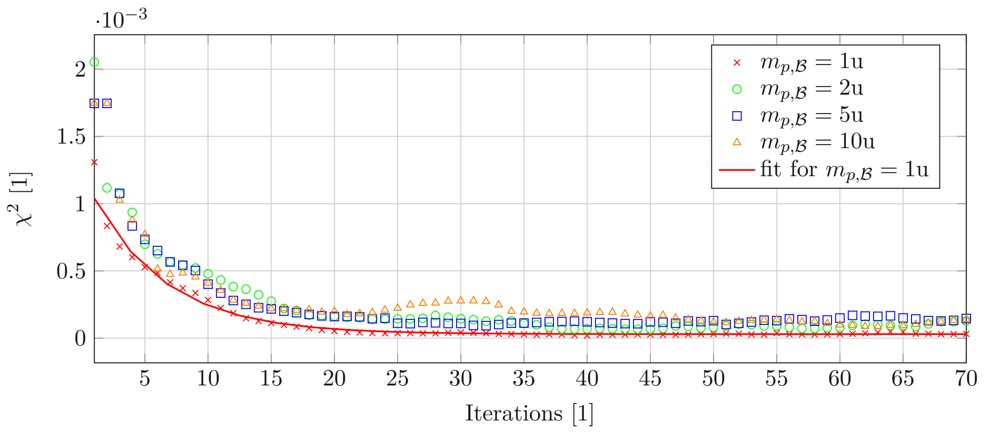


Figure 4.9: χ^2 for the particle speed distribution of $\mathcal{A}$ for various $m_{p,\mathcal{B}}$

Note that we included an exponential fit for the case of $m_{p,\mathcal{B}} = 1$u as a reference, as this corresponds to the single-species setup of Figure 4.8. Much to our surprise, while, in general, the relaxation time of the particles of $\mathcal{A}$ appears to be affected *slightly* by the different mass of the other half of the simulation's particles, the relaxation time was observed to be invariant to different values of $m_{p,\mathcal{B}} \neq 1u$.
However, this is one only side of the coin. Figure 4.10 shows the decrease of χ^2 for $\mathcal{B}$, plotted against the fit of the respective single-species relaxations, for the respective single-species scenarios of Figure 4.8. We can clearly observe that the relaxation time has increased when compared to the single-species case.

We also considered cases where the particles of the distinct species have different radii. As we already mentioned multiple of times before, kinetic theory assumes the collisions to be the primary source of the relaxation of a particle speed distribution towards its equilibrium.
Figure 4.11 shows the obtained evolution of χ^2 for the 200 particles of a species $\mathcal{A}$ with a particle mass of 1 u and a particle radius of 2 Bohr, in the presence of a different amount of particles of a species $\mathcal{B}$ with a particle mass of 1 u and varying radii.

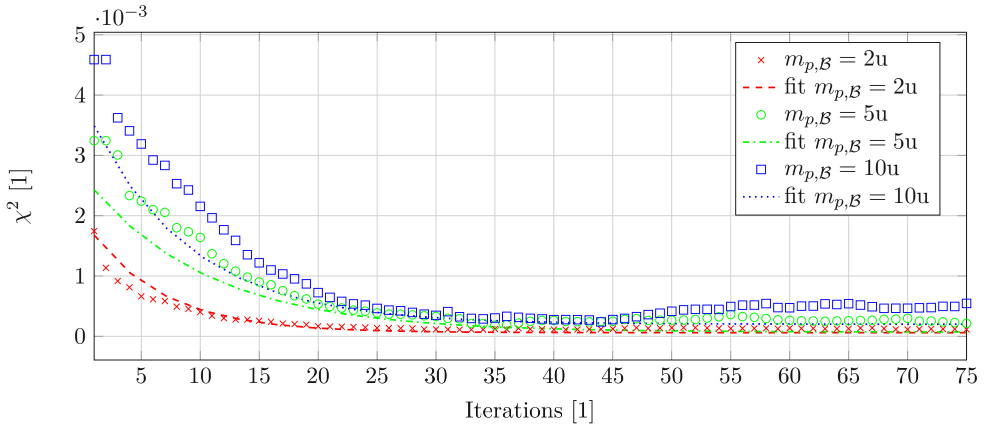


Figure 4.10: χ^2 for the particle speed distribution of $\mathcal{B}$ for various $m_{p,\mathcal{B}}$

Note that we changed the size of the simulation domain and the amount of particles of $\mathcal{B}$, in order to keep the different scenarios comparable, by parametrizing the simulation such that the relative share of the simulated domain occupied by particles was always approximately the same (for the shown cases $\approx 14\%$). This way, the probability of lighter particles hitting another particle should stay similar in all cases.
The figure's legend provides the respective simulation parameters, where N is the amount of particles of species $\mathcal{B}$ and D the simulation's domain size in Bohr.

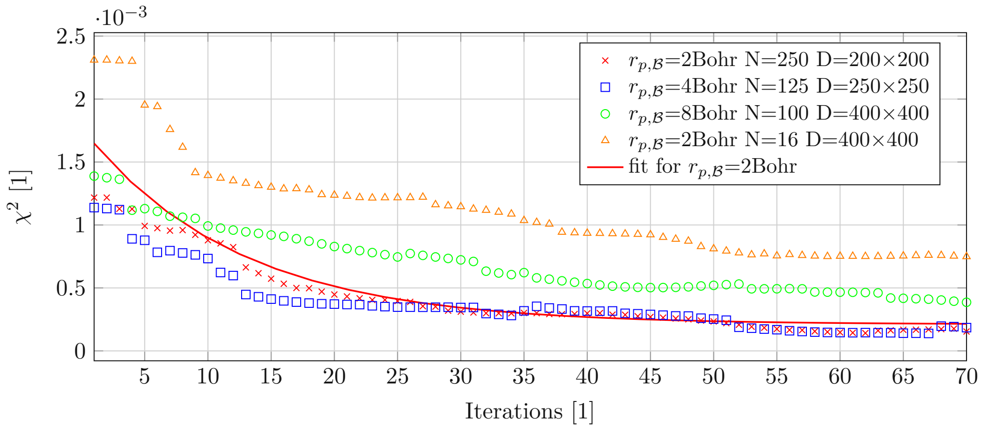


Figure 4.11: χ^2 for the particle speed distribution of $\mathcal{A}$ for various particle radii of $\mathcal{B}$

Note that the case of $r_{p,\mathcal{B}} = 2$ Bohr essentially corresponds to a single-species case where particles of $\mathcal{A}$ cover $\approx 14\%$ of the simulation domain.
We can observe from the data that the varying radii do appear to have a rather significant impact on the relaxation times of the distinct species. However, further work will be required in the future to properly quantify this effect.

Finally, we briefly considered systems where the particles are different in size *and* mass.

Figure 4.12 shows results for the same scenarios as considered in Figure 4.11, but with $m_{p,\mathcal{B}}$ scaling in lockstep with the radius of $\mathcal{B}$'s particles. Note that the figure's legend lists only the magnitudes of the particle radius and mass. However, the physical units are Bohr, respectively u. It is obvious that the relaxation of $\mathcal{A}$'s particles towards their equilibrium distribution is affected by their interactions with the particles of species $\mathcal{B}$.

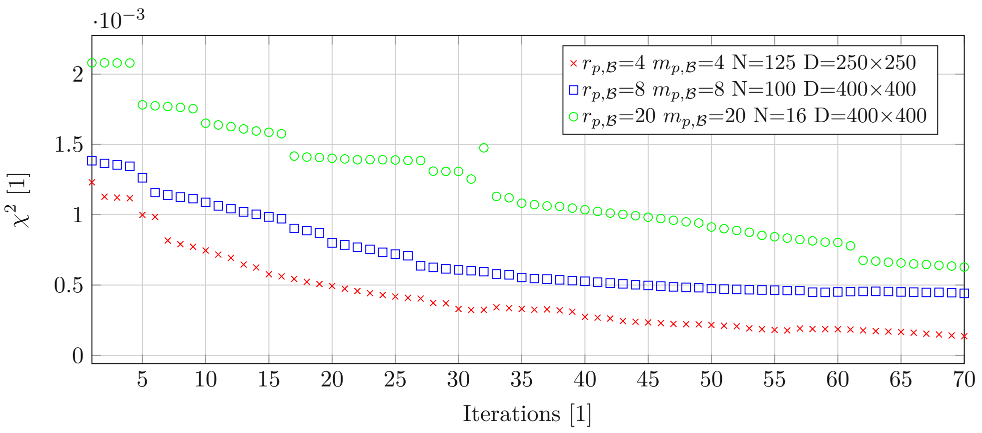


Figure 4.12: χ^2 for the particle speed distribution of $\mathcal{A}$ for various $m_{p,\mathcal{B}}$ and $r_{p,\mathcal{B}}$

Conclusions

From our experiments with the MD simulator, we conclude that the use of the basic BGK operator in a multi-species scenario, without paying heed to the mixture of species is only a feasible approximation if the different particles are similar in mass and size. For example, for same sized particles with a mass ratio of two, Figures 4.9 and 4.10 show that the overall impact on the relaxation time can likely be considered negligible for most cases. However, with an increase in the different relative masses of the particles, we can observe a relaxation time that increasingly differs from the single-species one. Thus, when the BGK operator is used with a τ for the single-species case, the obtained results also become increasingly worse.
An interesting exception, that deserves a further investigation in future work, emerges if one is only interested in the evolution of the lightest particle, in which case the relaxation appears to be remarkably invariant to other particles (Figure 4.9).
Furthermore, we could observe severe differences between the single- and multi-species relaxations for varying particles radii. A further investigation of the impact different particle sizes have on the relaxation times in multi-species setups could provide valuable information. In particular, a quantification of these observations, in addition to relating them to the physical process of cross-diffusion, as well as a generalization to particle shapes other than circular ones, would be quite interesting.
Furthermore, investigating whether an explicit relationship between the time constant of the exponential χ^2 and BGK's τ exists could be a fertile ground for future work, as it would allow using relaxation times that are can be obtained by simulating the microscopic

behavior of particles under the assumptions of kinetic theory.

Finally, we conclude that the use of the MBD as equilibrium distribution in multi-species systems is a sensible choice.

CHAPTER 5

LBM for Reactions - Implementation

A significant part of this thesis was the implementation of an LBM capable of simulating reactions between multiple species. This chapter provides an overview of the result of this effort, as well as of how the resulting simulator was validated.

5.1 Design Considerations and Requirements

In order to keep the entry barrier for using the simulator low, Python got chosen as target programming language since it is, as of writing this thesis, wide-spread among the sciences. A further requirement the implemented LBM is to be easy to modify. The reason for that being, that the simulator can in first instance be understood as a basis for further experiments in the future. This is also accommodated by using Python.
Due to the desire to allow modifications of the simulator with as little effort as possible, performance was not considered a key issue for now. Hence, the simulator will not be implemented in a way that harnesses the massively parallel nature of the LBM in the most efficient way. Instead, the initial implementation will be restricted to a simple single-core CPU implementation (albeit making use of the Numpy package [47] and its vectorized implementation).

Regarding the simulator's capabilities, in addition to supporting multiple species and reactions between them, the following features are considered paramount:

- Possibility to switch between a completely unit-afflicted, dimensional, and non-dimensional simulation core
- Interfacing with the simulator should happen via physical quantities and not require detailed knowledge about the LBM

- Support for creating and loading simulation checkpoints

The first of the above features is a quite powerful one for a simulator of physical processes. It allows to easily validate whether the programmed terms and equations are physically sound with respect to their “syntax” (i.e., if the physical units in the equations are consistent). This is especially important for the LBM, which is typically applied to the non-dimensional lattice units. Mixing up dimensional and non-dimensional quantities can cause subtle and hard-to-find errors. As written by Krüger *et al.* in [1], such errors are bound to happen for most LBM practitioners eventually, so relieving users of our simulator from handling non-dimensional quantities at all is desirable. This, in combination with providing implementations for different types of particles, lattices and boundary conditions satisfies the second stated feature.
Finally, creating and loading checkpoints of simulations allows to easily share and continue simulations that evolve in an interesting way. Furthermore, this also allows it to, for example, pre-compute the results of fluid flow simulations for different ARDE scenarios within the same environment. Ideally, one could create a set of readily available velocity fields for different applications (for example for the flow in a channel or a stirred reactor).

5.2 Software Architecture

The simulator and its components are developed in an object-oriented manner. Specializations / modifications of components are achieved by inheritance from a respective base class and extending or overriding its default functionality.
A (multi-species) simulation in the new simulator involves the simulation core (responsible for running the simulation), the unit system (hiding the non-dimensional quantities from users), a set of lattices and species each, as well as a definition of the reactions in the system. The following will provide a quick overview about the implementation of each of these parts, respective considerations and the interplay of these functionalities. Note that the full implementation can be found on Github [48].

5.2.1 Simulation Core

The simulation core is implemented in the `MultiLBM` class. This class contains the main simulation loop and is responsible for maintaining and interfacing with all the other parts.
A new simulation can be created by inheriting from this abstract class providing an implementation of the abstract `setup` method. Inside this method all *required* information for defining the simulation scenario is provided. This includes the characteristic quantities for the unit system, the set of species, parameters, and further custom user variables.
By overriding further methods, the functionality of the simulator can easily be extended or modified. For example, by providing an implementation of the abstract `dump` method users can specify the data being periodically dumped during a simulation.

Figure 5.1 illustrates the steps involved in a simulation. Each box corresponds to a conceptual step, dashed boxes mark optional steps.

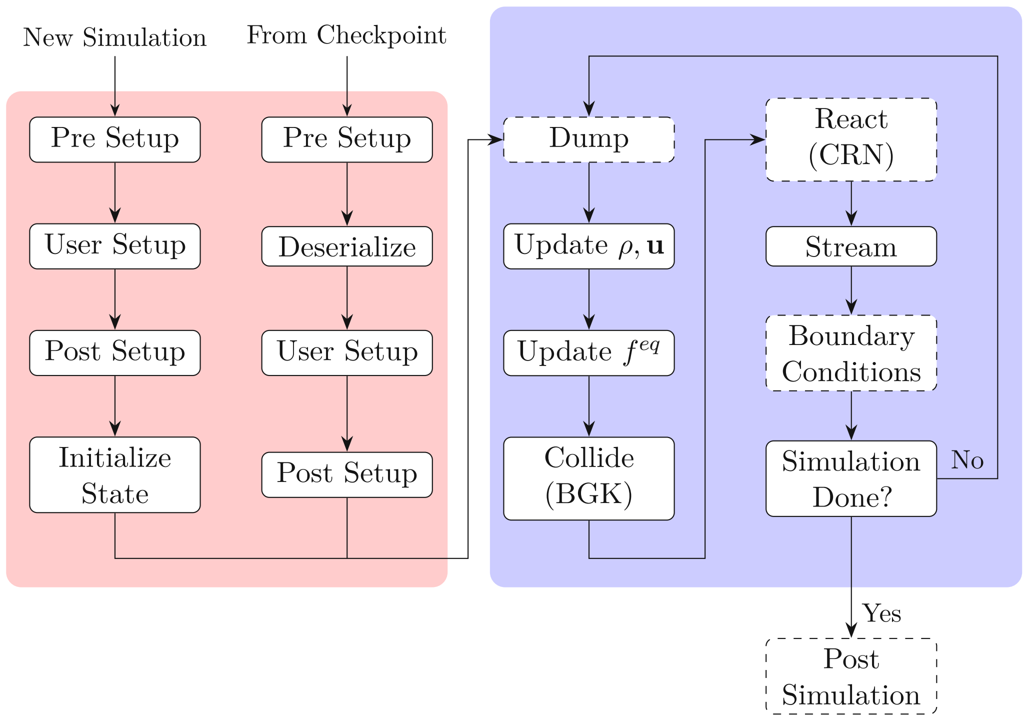


Figure 5.1: Simulator core loop illustration

The major simulation loop (contained in the right box; highlighted in blue) is very similar to the one introduced in Section 2.5, except for the optional reaction step. In this step our implementation computes the effect of inter-species reactions on the populations. The reactions of a system are defined as a Chemical Reaction Network (CRN) (we will discuss this in more detail in Subsection 5.2.5).

The left box, highlighted in red, contains the two alternative ways to start a simulation. This can either happen by starting a completely new one, or by loading a checkpoint from a previous run. Naturally, the involved steps are quite similar, with the difference being that a new simulation needs to initialize the populations (and macroscopic moments), whereas loading from a checkpoint takes the previous values for these quantities instead.

5.2.2 Unit System

The simulator comes with a dedicated unit system in the `UnitSystem` module. This functionality is encapsulated in a module rather than in a class, as it is ideally available globally and maintaining multiple unit systems in a single session is bound to cause

problems due to mixing up non-dimensionalized quantities from different instances.
The unit system, and handling of physical quantities and units in general, is implemented by extensive use of the *Pint* Python package [46]. The package provides extensive sets of existing units and respective conversion rules allowing to extend them both. Furthermore, a particularly powerful feature, the `Quantity` objects used by Pint provide overloads for many common operations, allowing to do computations with quantities while not only maintaining but also checking for the correctness with respect to physical units. A particularly noteworthy overload is the one for Numpy arrays and a subset of common operations involving them.

The implemented unit system adheres strongly to the description of non-dimensionalization for the LBM provided in Chapter 7 of [1]. In essence, one makes use of the fact that the dimensionality of all mechanical quantities is a product of powers of the dimension of length, time and mass. Thus, by providing three quantities with independent dimensions, one can non-dimensionalize any other mechanical quantity.
Naturally, when dealing with further quantities than just the mechanical ones, one can extend this. For example, when simulating bacteria one might use *colony-forming-units* (cfu) to express the bacterial concentration. Since the cfu unit cannot be expressed via a combination of mechanical base units, one requires an additional quantity for the non-dimensionalization.

The unit system hides the non-dimensionalization from users by requiring them to provide characteristic quantities at the beginning of a simulation, and to use the function `Q`, provided by this module, to create quantities. These characteristics must be independent and provided in a dimensional form, i.e., afflicted with physical units (which they are by construction when created via `Q`). Both of these requirements are checked automatically when setting the characteristic quantities.
After providing them, all further uses of `Q` will by default return non-dimensionalized versions of the passed quantities. This is achieved by using Pint's implementation of the *Buckingham π Theorem* on the characteristic quantities and the quantity to be non-dimensionalized, in order to obtain a combination of these quqntities that results in a non-dimensional result. Likewise, the simulator's internal quantities can be dimensionalized to a desired target unit.

As stated, this automatic non-dimensionalization is the default behavior. However, it is possible to change the unit system's conversion mode such that `Q` returns the desired quantity as unit-afflicted Pint Quantity. Since all the internal simulator code uses this function whenever applicable, this conversion mode allows performing simulations where each intermediate computation is checked for correctness with respect to the dimensionalities of the involved quantities.

5.2.3 Lattices

The implemented simulator comes with a `Lattices` sub-module, containing two built-in lattices, D2Q5 and D2Q9. With this thesis being restricted to simulations of 2D fluid flows ARDE systems, D2Q5 and D2Q9 suffice for the considered scenarios.

Additional lattices can be implemented by inheriting from the abstract `Lattice` base class. Such child classes must define their lattice velocity vectors (consisting of weights and directions). Per default, the lattice speed, speed of sound and related terms are provided by the abstract parent class. In case the desired lattice is more exotic than accounted for by this default implementation, it must be overridden.
In addition to these parameters, lattices must also provide the functionality to perform the streaming of the species' populations. Furthermore, implementations of the lattice-specific boundary conditions also belong into the specific lattice class.

Our D2Q9 lattice implementation comes with a wet-node scheme, the so-called Non-Equilibrium Bounce Back (NEBB) method [49, 50]. In the literature this scheme is sometimes coined by, referring to its authors, *Zou-He* boundary conditions. We settled on implementing this method as it features a higher order accuracy when compared to other wet-node schemes. In [1] Krüger *et al.* state that this scheme is accurate enough to recover exact parabolic solutions (for example for the Poiseuille flow we will consider in Section 5.3), independent of the value of τ (which is often required to be, in its non-dimensional form, one in order for the LBM to accurately recover second order solutions).
The method works for solid, straight boundaries, parallel to the x-, or y-axis, where either a density or a velocity profile (for the component aligned with the wall) is imposed on the respective boundary. A velocity profile can be used to model a moving boundary, whereas a density profile can be applied when describing a pressure gradient [50].

The principle idea is the following: After the streaming step, all populations at such a boundary, except for three, are known (the five pointing into or along the solid boundary must be zero). In addition to that, imposed profile plus the velocity component perpendicular to the wall (zero due to impermeability) are known as well.
This information, together with the assumption made by Zou and He that the *non-equilibrium* populations $f^{neq} = f - f^{eq}$ are simply bounced back at solid boundaries, is sufficient to determine the three missing populations plus the missing macroscopic quantity.
For illustration, Figure 5.2 depicts the known and unknown quantities for the case of a solid left wall, moving with a constant velocity $\mathbf{v}_{wall}$ in the y direction. The unknown populations (f_1, f_5, f_7 and node density (ρ_{wall}) are highlighted in red). The other populations and wall velocity components are known.

For the D2Q5 we implemented the common bounce-back scheme (see Section 2.5). While this schemes does not come with boundaries conditions as sophisticated as the Zou-He ones, it suffices for our purpose. We merely use the D2Q5 lattice for the simulation of the reacting species with an imposed velocity field. Thus, the boundary conditions required for the development of the respective fluid flow are not needed for the particles suspended in the bulk, as long as we take care that the velocity field does not introduce undesired boundary effects into the domain of the reacting species. We can, for example,

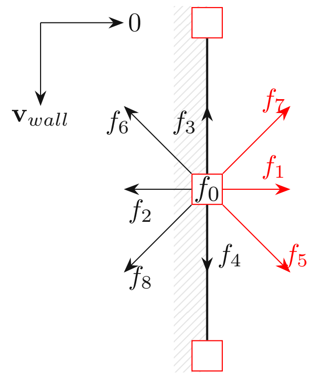


Figure 5.2: Illustration of known and unknown quantities at left boundary

achieve this by applying a filter function (like a Gaussian) to the velocity field such that the boundaries themselves are suppressed.

5.2.4 Species

Much of the behavior of a simulation depends on the specifics of the considered species. Thus, each simulation must be equipped with a set of the ones contained in the considered scenario. The implementation of each species is a specialization of the abstract `Species` class. Our implementation comes with a few readily available species in the `Species` submodule, which we required for validation and reproduction of results from literature.

The mentioned abstract class already defines the member variables for the populations, equilibrium distribution and post-collision populations, as well as the macroscopic moments of the populations (density and velocity). Furthermore, the abstract class also already contains default implementations for updating the moments, self-collision (BGK) and computing the local equilibria. For the equilibrium distribution, three versions of the discretized MBD are implemented. Namely, the ones where the bulk is at rest ($\mathbf{u} = 0$), the one only depending on terms linear in the fluid velocity and the second order one shown in Equation (4.1).
By inheriting from the abstract class, individual species essentially only need to provide the information which equilibrium distribution they use, a relaxation time for the BGK collision operator (or an altogether different operator) and any further species' specific behavior (for example, in the LBM reported in [7] a species must be equipped with the equilibrium that contains the chemotactic velocity).

5.2.5 Reactions

The `Reactions` module provides users with means to define reactions between the species contained in the simulation. This is done by specifying the reaction system as

CRN [51].

As defined by Feinberg in [51], a CRN consists of

- a finite set of species $\mathscr{S}$
- $\mathscr{C}$, a finite set of vectors in $\overline{\mathbb{R}}_+^{\mathscr{S}}$ ($\overline{\mathbb{R}}_+$ is the set of non-negative real numbers) called *complexes*
- a reaction relation $\mathscr{R} \subset \mathscr{C} \times \mathscr{C}$ where

$$\forall y \in \mathscr{C} : (y, y) \notin \mathscr{R} \land \exists y' \in \mathscr{C} : (y, y') \in \mathscr{R} \lor (y', y) \in \mathscr{R}$$

Essentially, complexes are just a way to encode linear combinations of species that occur in the system. The coefficients of the species in such complexes are referred to as the *stoichiometric* coefficients.
The reaction relation states that each complex must participate in at least one reaction, i.e., either at least produce another complex or be produced by one. Note that for $(y, y') \in \mathscr{C}$ y is coined as the *reactant* complex and y' as the *product* complex.

Note that the above definition does not contain any information about how reactions are to be modelled. This happens by defining the *kinetics* of a CRN, which is a function that assigns each reaction in $\mathscr{R}$ a rate function. This rate function takes the current concentrations of all species of the reaction's complexes as parameters and returns a reaction rate. A particularly popular kinetics is the *mass action* kinetics, which defines the rate of a reaction to be the product of the reactant complex's species raised to their respective stoichiometric coefficient, times a positive proportionality constant (either from estimation or experiments [51]).

Let us consider the simple $\mathcal{A} + \mathcal{B} \to \mathcal{C}$ example mentioned in Chapter 3 in order to illustrate the above notation. The set of species is $\mathscr{S} = \{\mathcal{A}, \mathcal{B}, \mathcal{C}\}$, the set of complexes $\mathscr{C} = \{\mathcal{A} + \mathcal{B}, \mathcal{C}\}$ and the reaction relation $\mathscr{R} = \{(\mathcal{A} + \mathcal{B}, \mathcal{C})\}$. The respective reaction rate, according to mass action kinetics, is $kc_{\mathcal{A}}c_{\mathcal{B}}$ where k is the mentioned constant. Note that $\mathcal{A}, \mathcal{B}$ both feature a stoichiometric coefficient of 1. Hence, their concentrations only occur to the power of 1.
Given this example, we can also discuss the idea behind this kinetics: Naturally, a reaction of this type can only happen if particles of the two species $\mathcal{A}, \mathcal{B}$ collide. It is assumed that the occurrence of this is directly proportional to the amount of particles at a certain volume, which gives the product of the concentrations. The proportionality factor can be used to model the likeliness of such collisions to be reactive (recall, that not all inter-species collisions are).

Defining a CRN for the simulation happens by programmatically adding all reactions. The set of species and complexes is constructed automatically. This is done by specifying the reactant and product complexes, as well as a rate function per reaction. The underlying implementation will compute the effect on the populations for each reaction, in the specified order, during the simulation's reaction step (refer to Figure 5.1). Note that by specifying a rate function rather than a constant, different kinetics than mass action can easily be used. An alternative would be *Monoid* kinetics, which can be used for modeling two-species reactions where one species acts as a catalyst. For example, in [7] Hilpert makes use of this kind of kinetics.

5.2.6 Checkpoints

As motivated at the beginning of this chapter, storing checkpoints of a simulation is a desirable feature. This way results can easily be reused (desirable for ARDE scenarios where the fluid flow is not dependent on the other species), continued if they prove interesting after a short initial run (this would be nice for automatic sweeps of species or simulation parameters and versions) and shared.
The way this is implemented is via the abstract class `Serializable`. Other classes can be equipped with the checkpoint feature by inheriting the respective functionality from this base class. Users than simply specify a list of class members they wish to store. In the majority of cases nothing more is required. The default implementation of `Serializable` will handle the (de-)serialization of the desired members in a hierarchical manner and store them in a human-readable form inside a given checkpoint directory. In essence, for each serialized object a JSON file is created that contains everything we consider a *primitive* member (for example int, float, string, scalar Pint quantities) and the information on how to obtain non-primitive members. For example, Numpy arrays are stored in dedicated files, allowing for simple reuse and memory-efficient storage. Thus, the respective entry in the JSON file contains the location of the stored array.
Dependencies between objects are handled by assigning each object during serialization a unique identifier that is stored as well. During serialization the identifiers of already serialized objects are tracked, such that hierarchical dependencies can be expressed by referring to the identifiers.
During de-serialization a corresponding scheme is deployed. The identifiers of all de-serialized objects will be tracked, allowing objects to obtain the correct non-primitive member.

5.2.7 Example

We will now give an example of setting up a simulation in our LBM implementation. This highlights the simplicity and conciseness of such setups, as well as how modifications can be introduced.
As a model problem we will configure our simulator for the scenario considered by Hilpert in [7]. There, bacterial chemotaxis of the bacteria *Pseudomonas putida* (abbreviated as PpG7) towards *naphtalene* (specifically, the version used for the bacterial ring) is simulated. Listing 5.1 shows the required setup code for our simulator.

```python
class Hilpert05(MultiLBM):
    serialize_members = ["ks", "q", "attr_init", "bac_init"]

    def setup(self) -> None:
        self.dx, _, self.attr_init, self.bac_init = characteristics(Q(0.1, 'mm'),
                                                                    Species.PpG7.swim_speed,
                                                                    Q(2.83E-2, 'kg/m**3'),
                                                                    Q(4E12, 'cfu/m**3'))

        self.x = Q(10, 'mm')
        self.y = Q(10, 'mm')
        self.dt = self.dx / Q(Species.PpG7.swim_speed)

        self.ks = Q(1.3E-4, 'g/L')
        self.q = Q(7.9E-16, 'g/(cfu*s)')

        attr = self.add_species("Naphtalene", "attractant", "D2Q9",
                                init_quantity=self.ca_init)
        bac = self.add_species("PpG7", "bacteria", "D2Q9",
                               init_quantity=Q(0, 'cfu/m**3'), attractant=attr)

        self.add_reaction([1 * bac, 1 * attr], [1 * bac], self.attr_consumption, q=self.q,
                                                          ks=self.ks, dt=ca.lattice.dt)

#       self.add_species("BasicParticle", "fluid", "D2Q9",
#                        init_quantity=Q(.9982, 'kg/m**3'), viscosity=Q(1.0034, 'mm**2/s'))
#       self.bulk_velocity = self.fluid.velocity
#       width, height = int(mag(self.x / self.dx)), int(mag(self.y / self.dx))
#       self.solid = np.pad(np.zeros((height-2, width-2), dtype=int), pad_width=1, constant_values=1, mode="constant")

    @staticmethod
    def attr_consumption(reactants, externals, products, q, ks, dt):
        return q * dt / (reactants[1].density + ks)

    def init_state(self) -> None:
        x, y = self.width//2-1, self.height//2-1
        circle_filled(x, y, int(non_dim(Q(1, 'mm'))), self.bacteria.density, self.bac_init)

#   def boundaries(self) -> None:
#       velocity = Q(np.zeros((2, self.width)), 'm/s')
#       velocity[0] = Q(0.1, 'm/s') * (1 - math.exp(-self.runs**2 / (2 * 1E04)))
#       self.wall_boundary(self.fluid, 0, 0, self.width, self.fluid.lattice.bc_top, velocity_profile=velocity)

    def dump(self, directory: str) -> None:
        dump_as_img(self.bacteria.density, directory, "bac_density_" + str(self.dump_count))
        dump_as_img(self.attractant.density, directory, "attr_density_" + str(self.dump_count))

sim = Hilpert05()
sim.run(100, 10, "sim_out/HilpertEx")
sim.serialize("chkpts/results")
```

Listing 5.1: Example Code for reproducing [7]

Initially, we define the lattice spacing (`dx`), bacterial swimming speed and the two species' densities as characteristic quantities for the unit system (note how we require four quantities due to the non-mechanical *cfu* unit). We continue with providing the remaining parameters to completely define the lattice (`x`,`y` together with `dx` determines the lattice size), featuring a time step of one. After that, we add the bacteria and chemoattractant species to the simulation. Note that we define the used lattice for each species (this allows to simulate ARDEs on D2Q5 in conjunction with the respective bulk fluid flow on D2Q9).

With the species in place, we can add the system's reaction. As outlined in Subsection 5.2.5, this is done by providing the reactant and product complexes. In this case, the two complexes are simply $\mathcal{B} + \mathcal{C}$ respectively $\mathcal{B}$ (with $\mathcal{B}$ being the bacteria and

$\mathcal{C}$ the chemoattractant). We account for the Monod kinetics used by Hilpert via the reaction function `ca_consumption`, which is essentially the respective Monod equation, except for the naphtalene concentration being absent in the numerator. This is added automatically by the underlying reaction system.
Finally, we initialize the density of the bacteria, which Hilpert just sets to the initial density at node in the lattice center. The remainder of the code is responsible for running the simulation for 100 iterations, dumping the two species' densities every 10 steps. Furthermore, we create a checkpoint once the simulation is done, which can be used for continuation or post-processing of the simulation data.

The commented lines of code serve the purpose of highlighting how simple modifying and extending simulations is. By uncommenting them, the species become enclosed inside a solid, rectangular container, an additional species for the solvent is added and the bacteria and naphtalene become subject to the advection due to the solvent's motion. The solvent itself is excited by the top wall being driven with an increasing velocity.

5.3 Validation

Naturally, validating the soundness of the results produced by a simulator such as the one we implemented is paramount. This holds especially true since there exist no analytical solutions to the considered problems, except for a few simple cases. However, this is also means that it is non-trivial to determine if the results produced by the simulator are indeed correct (to some extent, it is a numerical scheme after all), or just *look* like they are. For example, consider Appendix H in Flatscher's thesis [5], where it is shown how a slight deviation in a single parameter can cause significant errors in the results, while still appearing feasible at first glance. And indeed, during this work we have experienced similar situation where subtle mistakes would only cause a loss in accuracy or stability for certain parameters, edges cases and / or after thousands of iterations. This, combined with the fact that debugging has to happen on the level of populations, makes it vital to carefully ensure the validity of an implementation of the LBM. In this section, we will therefore validate our LBM implementation.

5.3.1 Classic CFD Benchmarks

This first subsection focuses on our implementation's capabilities for classical single-species CFD problems. We will apply our LBM implementation to some of the few fluid flow problems with known analytical solutions, and compare it to results reported in literature for a problem where no such solutions are known.

Couette Flow

The first classic fluid dynamics benchmark we will consider is the, so-called, *Couette* flow. This fluid flow problem consists of two parallel walls, separated by some distance d, where one wall is at rest and the other one moves with a constant velocity $\mathbf{v}_{wall}$.

The reason why the Couette flow is an interesting benchmark for CFD simulators, is the fact that it is simple enough to be solved analytically. Under the assumption that there is no slip between the fluid and the walls (i.e., particles at the walls move with the same speed as the wall), a simple, linear, fluid velocity develops [1]. Equation (5.1) shows this analytical solution for the case where the walls are parallel to the x-axis and the top wall is moved along the x-direction.

$$\mathbf{u}_x(y) = -\frac{\mathbf{v}_{wall}}{d} \cdot y \tag{5.1}$$

Furthermore, Figure 5.3 illustrates the setup and also the resulting fluid velocity.

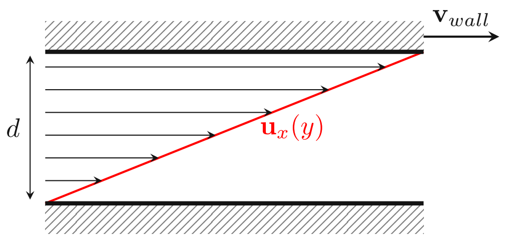
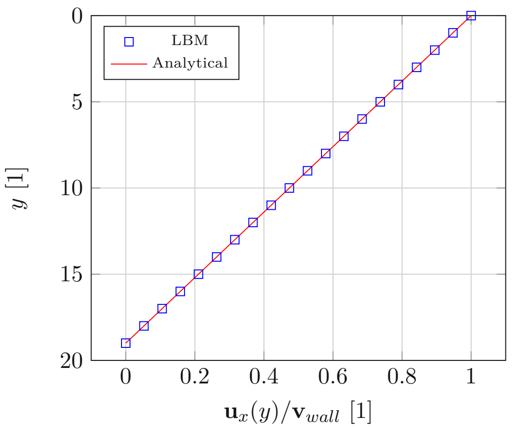


Figure 5.3: Couette flow setup and solution

As stated by Krüger *et al.*, the LBM (with a suitable boundary condition scheme) should be able to recover the solution of this simple linear flow up to machine precision, disregarding the BGK relaxation time τ.
Figure 5.4 shows the non-dimensionalized center velocity profile (with respect to the x axis) obtained by our implementation and by the analytical solution.



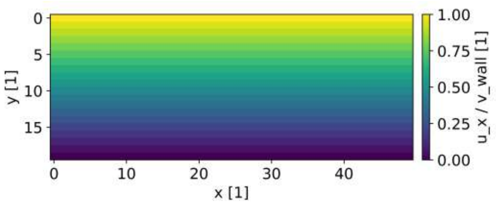


Figure 5.4: Center velocity profile (x=25)

Figure 5.5: Fluid velocity field

For the simulation we used the NEBB scheme at the bottom (velocity zero) and top (velocity set to the lattice speed) boundaries. For the left and right boundaries we deployed periodic streaming. The lattice was parametrized to consist of 50×20 (width$\times$height) nodes and to $\Delta x = \Delta t = 1$. The initial fluid velocity was set to zero, the non-dimensionalized density uniformly to one. We chose a viscosity that resulted in a relaxation time of $\tau = 2$. The simulation was run until the result converged towards the analytical solution. As a requirement for convergence we checked whether the deviation between the computed fluid velocity and the analytical solution is within twice the machine precision (around $\epsilon = \texttt{2.22E} - \texttt{16}$) for ten consecutive iterations. This happened after around 300 iterations. As we can observe in the figure, our implementation does indeed accurately recover the Couette flow, shown in Figure 5.3 for the fluid velocity's x-component across the whole domain.

Since the LBM is quite sensitive towards its parameters, especially the relaxation time, we conducted multiple simulations with different values for τ and also lattice sizes. Table 5.1 shows the respective simulation parameters.
The column *Re* shows the *Reynolds* number of the respective case, which is a commonly used dimensionless number in CFD to characterize a fluid flow. It is computed via $Re = \frac{U \cdot L}{\nu}$, where U, L, ν are a characteristic velocity, length, respectively viscosity.
The final column shows the number of iterations of the simulation loop that were required for the macroscopic velocity to converge towards the analytical solution within 5ϵ.
Note how all the tested cases reproduce the analytical solution with an accuracy in the order of the machine precision, thus providing some feedback about the correctness of our implementation of the basic LBM and the used boundary conditions (in this case the straight wall, velocity-driven, NEBB ones).

Lattice Size	τ	Re	Iterations
50x10	0.8	100.0	2994
50x10	1.4	33.3	1166
50x10	2.0	20.0	625
50x20	0.8	200.0	13757
50x20	1.4	66.6	4385
50x20	2.0	40.0	2807
50x30	1.4	99.9	10917
50x30	2.0	60.0	7579

Table 5.1: Couette flow simulation setups

Poiseuille Flow

The next benchmark to which we applied our implementation is the *Poiseuille* flow. Similar to the Couette flow, the problem consists of two parallel walls, spaced apart by some distance d (or a tube / pipe with some inner diameter d). However, for the Poiseuille flow both walls are stationary. Instead, a fluid flow will develop due to a pressure gradient, Δp, along the axis parallel to the walls. As for the Couette flow, an analytical solution is known [1], shown in Equation (5.2) (for the case of the walls being parallel to the x-axis and pressure gradient constant). Note that this equation is quadratic, rather than linear, in y.

$$\mathbf{u}_x(y) = -\frac{1}{\eta_s} d_x p \cdot y(y-d) \tag{5.2}$$

Figure 5.6 illustrates the Poiseuille flow setup and velocity profile according to the analytical solution.

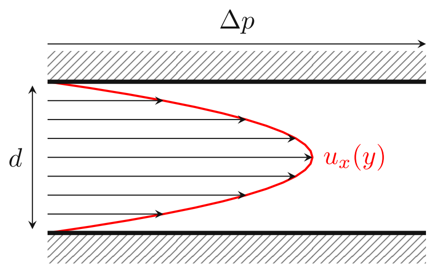


Figure 5.6: Poiseuille flow setup and solution

This classical CFD benchmark is particularly interesting for the LBM, as the method is a second-order accurate solver. Therefore, it is in principle capable of recovering the exact solution for the fluid velocity field of the Poiseuille flow. However, as Krüger *et al.* demonstrate, this depends on suitable parameters and boundary conditions.
As for the Couette flow, we use the NEBB scheme for the boundaries. The top and bottom walls are modelled to be impermeable ($\mathbf{u}_y = 0$), the left and right boundaries are responsible for the encoding the pressure gradient. However, as our implemented boundary conditions do not support imposing pressure gradients, and we are just interested in validating whether our implementation works properly, we can also excite the potential development of a Poiseuille flow by applying the velocity profile corresponding to a fully developed Poiseuille flow (Equation (5.2)) at one boundary, and a constant density at the opposing one. We set the left wall to the velocity profile and the right to the constant initial density. Furthermore, we also deploy the NEBB scheme at the corners of the rectangular simulation domain.
We can interpret this as considering a segment of a pipe in which the Poiseuille flow has already developed before our considered segment. The required pressure gradient is then given implicitly via the inflow at one end of the segment, and the constant density at the other one.

Figures 5.7 and 5.8 show the resulting center velocity profile, respectively the x-component of the fluid velocity field, for a developed Poiseuille flow. Note that $\mathbf{v}_{max}$ is the maximum velocity of the analytical solution (at the centerline of the pipe) and used here for non-dimensionalization.

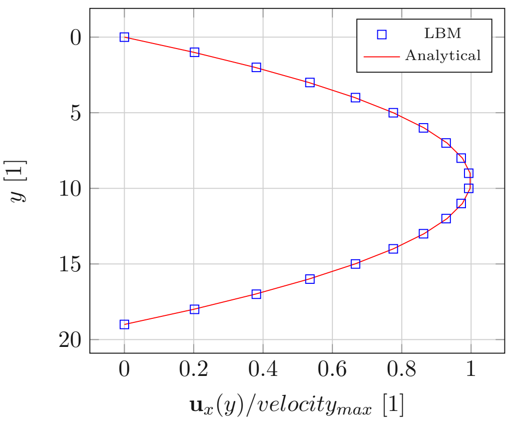


Figure 5.7: Poiseuille center velocity

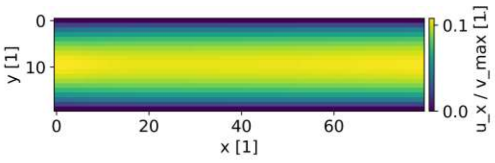


Figure 5.8: $\mathbf{u}_x$ field of the Poiseuille flow

For the simulation we used the D2Q9 lattice with a width of 80 lattice nodes and a height of 20. The Reynolds number was set to 50 and the relaxation time to the non-dimensional value of 0.66. The simulation was set to either end if a certain maximum of iterations is surpassed, or if the centerline velocity profile deviates from the analytical solution by less than 1% for ten consecutive iterations. For the shown case this happened after around 2100 simulation steps.
Note that this convergence criterion is way less strict than the one for the Couette flow, as the obtained solutions do not exhibit the same amount of accuracy as the ones for the simple first order problem. However, we verified that our achieved accuracy is on par with another open-source LBM implementation [52].
Finally, we provide some convergence results (according to the above criterion) for different simulation setups in Table 5.2. Note that the case with Re=150 on the 80×20 lattice, featuring the smallest τ, did not result in a stable simulation due to the relaxation time being too close to 0.5 and the lattice being comparably small. Observe however, how other setups with the same Reynolds number, but different τ and lattice dimensions did converge.

Lattice Size	τ	Re	Iterations
80x20	0.66	50	2103
80x20	0.58	100	2278
80x20	0.55	150.0	-
80x30	0.74	50	2223
80x30	0.62	100	2339
80x30	0.58	150	2390
80x40	0.82	50	2281
80x40	0.66	100	2370
80x40	0.61	150	2427

Table 5.2: Poiseuille simulation setups and iterations until convergence

Lid-Driven Cavity

Another classic CFD benchmark to which we apply our simulator is the Lid-Driven Cavity (LDC), which is also relevant for some industrial applications [53]. A fluid, initially at rest and with uniform initial density, is inside a rectangular enclosure with impermeable walls. One of these these walls is moving tangentially to itself with a constant velocity $\mathbf{v}_{lid}$. Figure 5.9 shows this setup and a stylized velocity field resulting from the movement of the wall. The expected behavior is the development of a vortex at the corner of the driven wall and the one perpendicular to it in its movement's direction. The shape and position of the fully developed vortex depends on the enclosed fluid's viscosity, the speed of $\mathbf{v}_{lid}$ and the size of the enclosure.

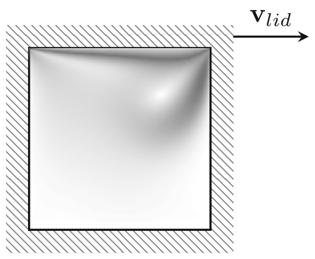


Figure 5.9: Lid-driven cavity setup

Note that the LDC is the first benchmark that we consider for which no analytical solution exists [54]. However, due to the problem's popularity for the validation of CFD simulators [55, 54, 56, 57], there exist many publications reporting results.
We validated our implementation by comparing the center domain velocity profiles for different simulation parameters with the results reported in literature. First and foremost, we compared to the results provided by Ghia *et al.* in [55], as this appears to be the only reference to contain proper values for the velocity profiles, rather than just plots of them. Furthermore, it appears as if these values would be what most other publications use as a baseline anyway (for example, [56, 57, 58, 53]).

Figures 5.10 and 5.11 show the velocity profiles in the x, respectively y, direction along the cavity's centerlines for Reynolds numbers of 100, 400 and 1000. We used the D2Q9 lattice, which was set to be composed of 129×129 nodes for the Re=100 and Re=400 cases, and to 257×257 for the Re=1000 case. As boundary condition scheme we deployed NEBB along all the cavity's walls and at the corners. Note that the top wall is set to be the one moving in the results reported in the figure.
For comparison, we also plot the results reported by Ghia and co-authors. Observe how well our results fit the reported reference, although the authors use a different CFD scheme (for more details we refer to their paper).

Note that we did not perform any comparisons for higher Reynolds numbers. The reasons for that are being two-fold: First, we are not very keen on simulating increasingly

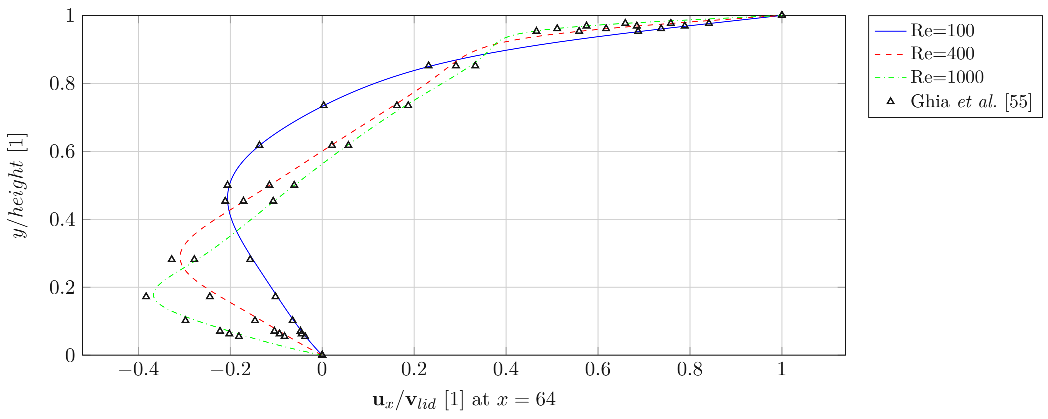


Figure 5.10: Comparison with [55] of non-dimensional x-velocity along vertical center line

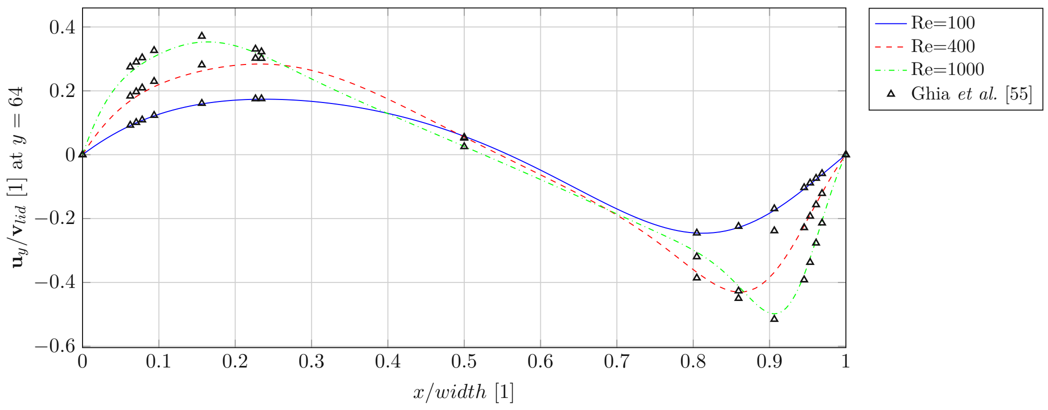


Figure 5.11: Comparison with [55] of non-dimensional y-velocity along horizontal center line

turbulent high-Reynolds flows, as our ultimate interest lies within biological systems which are rather viscous. Second, due to us not being interested in turbulent flow for now, our implementation is currently only supporting the basic BGK collision operator, which requires an increasingly higher amount of lattice nodes to cope with Reynolds numbers in the range of thousands, without leading to instabilities in the LBM. Indeed, we observed the simulation to fail due to such instability for Re=1000 on the 129×129 grid.

Note that we did not provide further quantitative comparisons here, as in general researchers appear to be interested into simulating this problem for higher rather than lower Reynolds numbers. However, we did perform a qualitative comparison with the Re=10 and Re=200 velocity profiles given in Figure 3 of [54] and could observe a good correspondence as well.
Finally, we provide the simulated velocity field for the Re=1000 case in Figure 5.12 (we have plotted the magnitude of the fluid velocity normalized by the magnitude of $\mathbf{v}_{lid}$).

Observe how the expected vortex develops, showing good correspondence to the result shown in [52]

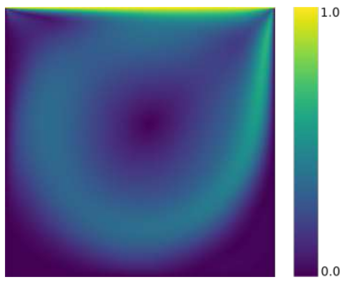


Figure 5.12: Lid-driven cavity velocity field for Re=1000

5.3.2 ARDE Validation

After the application of our implementation to the classic CFD benchmarks above, we continued by validating its capabilities as a RDE solver. To do this, we reproduce results from publications mentioned in Chapter 3.

Reacting Front

The first multi-species reaction-diffusion system on which we use our LBM is the $\mathcal{A}+\mathcal{B} \rightarrow \mathcal{C}$ one treated by Qian and Orszag in [25].
The authors apply mass-action kinetics for the reactive term and use their LBM to compute solutions for "reacting front" scenarios of the reaction system. In these scenarios the simulation is initialized to contain only particles of species $\mathcal{A}$ in one half of its domain, and only particles of $\mathcal{B}$ in the other one (note that, initially, $\mathcal{C}$ is not present). The relation between the initial concentrations of the species is responsible for the dynamic behavior of the system.
In their work, Qian and Orszag report their results for two different setups. In the first, the initial concentrations of $\mathcal{A}$ and $\mathcal{B}$ are the same. In the second one, the concentration of $\mathcal{A}$ is assumed to be twice as high as the one of $\mathcal{B}$. Figure 5.13 illustrates the setup for the two cases considered by Qian and Orszag. Left, the initial concentrations of the species $\mathcal{A}$ (blue circles) and $\mathcal{B}$ (red squares) are set to be the same.

For both cases, one can observe species $\mathcal{C}$ being produced at the interface of the two reactant species. Over time, as the reactants increasingly diffuse into the simulation domain's half where they were initially not present, and their concentrations slowly wane due to the reaction taking place, a Gaussian-like production of $\mathcal{C}$, spreading with time, can be observed. However, depending on the relation between the initial concentrations, this Gaussian is spreading stronger towards the half with the initially lower concentration.

In order to reproduce their results, we mostly adhered to the non-dimensional parameters reported by the authors. That is, a reaction rate of 0.01 and initial concentrations of $c_{0,\mathcal{A}} = c_{0,\mathcal{B}} = 1$. Instead of their one-dimensional simulation domain of a width of 201 on

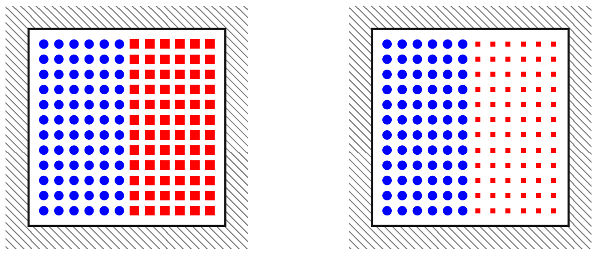


Figure 5.13: Illustration of the reacting front setups for the cases $c_{0,\mathcal{A}} = c_{0,\mathcal{B}}$ $c_{0,\mathcal{A}} = 2c_{0,\mathcal{B}}$

a D1Q2 lattice, we use a D2Q5 lattice of 201×201 nodes, where we non-dimensionalized the quantities such that $\Delta x = \Delta t = 1 = c$. The only parameter left worth noting is the diffusivity coefficient D (which we assume to be the same for all species). In the original paper, this was related to the BGK relaxation time via the same relation as given in [2], Equation (5.3).

$$D = \frac{c^2}{2Q}\left(2\tau - \Delta t\right) \tag{5.3}$$

However, since our implementation uses Equation (5.4) per default, and the two forms should ultimately express the same relation, we converted the relaxation time given in [25] to a diffusivity coefficient suiting our code, which is $D = 0.5$.

$$D = c_s^2\left(\tau - \frac{\Delta t}{2}\right) \tag{5.4}$$

In addition to the mentioned parameters, we used bounce-back boundary conditions alongside all boundaries of the rectangular domain. Figure 5.14 shows the results we obtained at $y = 100\Delta x$, plotted atop the ones reported by Qian and Orszag, for the case of equal initial concentrations after 200, 400, 600, 800 respectively 1000 time steps (the concentrations wane with the time). Note that we only show every second data point in order to keep the reference underneath visible.

An excellent correspondence between their and our results can be observed, suggesting no major mistakes in our multi-species LBM for RDE. Furthermore, the figure also shows the “production rate” of $\mathcal{C}$, which is essentially just the result of the reactive term. Again, observe how well the obtained data fits the reference from [25]. Furthermore, note how $\mathcal{C}$ implicitly has a mass of $m_{p,\mathcal{C}} = 2 \cdot m_{p,\mathcal{A}} = 2 \cdot m_{p,\mathcal{B}}$ if the system is to be mass-conserving (which we strongly assume), although this fact was at no point acknowledged by Qian and Orszag.

We were further able to achieve an equally good correspondence to the results reported for the second case, where $c_{0,\mathcal{A}} = 2c_{0,\mathcal{B}}$. This is shown in Figure 5.15. Observe how the

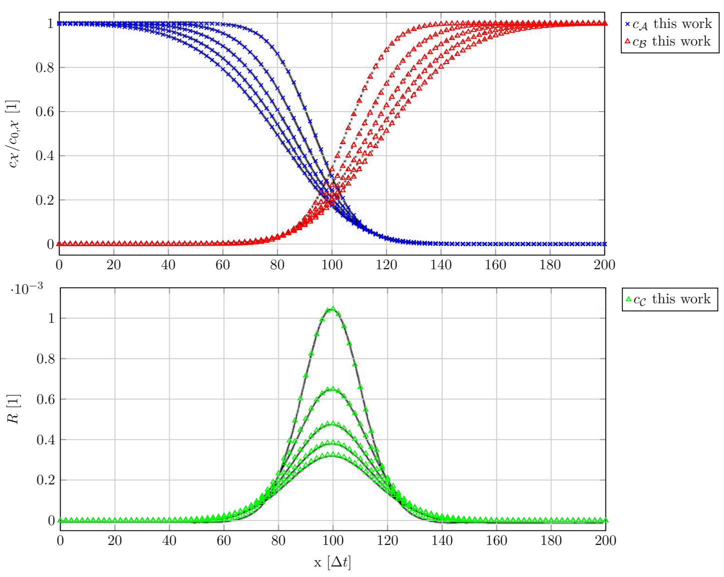


Figure 5.14: Overlay of our results for the reacting front with initial $c_{0,\mathcal{A}} = c_{0,\mathcal{B}}$ atop the result in Fig. 4 of [25]

higher concentration of species $\mathcal{A}$ leads to a stronger diffusion of $\mathcal{A}$ towards the right half of the domain than of species $\mathcal{B}$ towards the left half. As a result, the production rate of $\mathcal{C}$ is increasingly spreading into the right half of the simulation's domain ($x > 100\Delta x$).

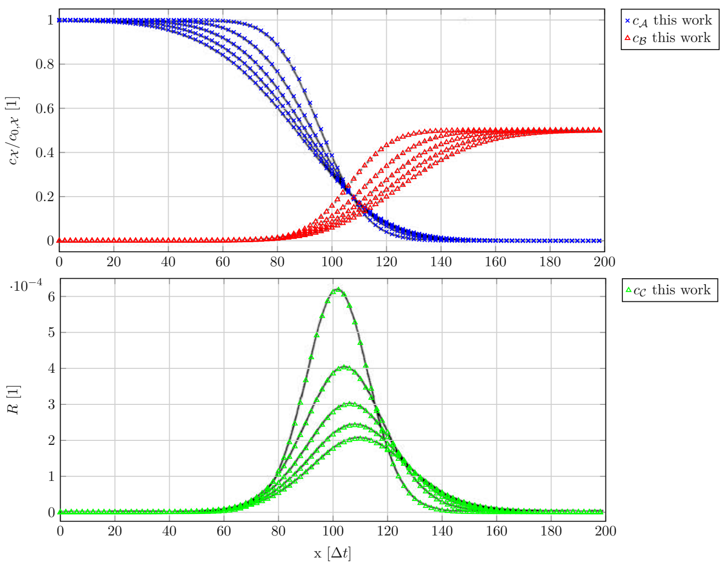


Figure 5.15: Overlay of our results for the reacting front with initial $c_{0,\mathcal{A}} = 2c_{0,\mathcal{B}}$ atop the result in Fig. 4 of [25]

Bacterial Growth Patterns

A slightly more complex reaction system benchmark than the simulation of the $\mathcal{A}+\mathcal{B} \rightarrow \mathcal{C}$ system is the one reported by De Rosis *et al.* in [9]. This RDE system, shown in Equations (3.6)-(3.8), consists of three species and two reactions. However, note that the bulk fluid is considered to be at rest, i.e. it is assumed that $\mathbf{u} = 0$.
In their letter, and the corresponding code repository [59], the authors made their simulation parameters available. This allowed us to reproduce their work as a further means of validating our implementation. As in the original letter, we used the D2Q5 lattice with a simulation domain of 2000×2000 lattice nodes for all species. We used periodic boundary conditions.
We ran each of the four cases considered by De Rosis and co-authors until the bacterial populations started to reach the boundaries, except for the "DLA" case (third row in Figure 5.16). The reason for that is, that for all cases except the DLA the results reported in the letter were clearly obtained without the bacteria (active and inactive) reaching a boundary. However, in the DLA case the reported result is only reproducible by letting the simulation continue after bacteria spread to the boundary.
Figure 5.16 shows a side-by-side comparison of the non-dimensional total bacteria concentration (active and inactive) obtained using our simulation (left), and the original bacteria concentration reported by De Rosis *et al.* (right). Observe how the same growth patterns emerge.

Reacting Lid-Driven Cavity

In [29] Pribec and co-authors reported on an interesting simulation, where they combined the LDC with the reacting front setup for the $\mathcal{A}+\mathcal{B} \rightarrow \mathcal{C}$ RDE system of [25], in order to demonstrate that their scheme is capable of combining the effects of advection, diffusion and reaction. While they could ultimately, due to its novelty, not compare their results to any other similar simulations, we can perform a qualitative comparison between our implementation's results and the ones reported by them.
We set our simulation up as specified by them, i.e, we used a 150×150 D2Q9 lattice and a non-dimensional reaction rate of 0.01. The top wall velocity was set to 0.1 (non-dimensional). Note that the velocity field is produced by a single bulk fluid only, that is not influenced by the reacting species. The viscosity coefficient of this bulk fluid, as well as the diffusivity coefficient of the reacting species, were computed by using the characteristic non-dimensional numbers given in their paper.
One difference between our and their scheme are the boundary conditions. Pribec *et al.* use the bounce back scheme, except for the driven wall of course, where they impose $\mathbf{v}_{lid}$. However, we deploy the NEBB scheme for all boundaries (for *all* species).
Figure 5.17 shows the evolution of the different species concentrations over the course of 20000 time steps just as in Figure 11 of [29].

First, note how the velocity field that develops is virtually the same. However, the species' concentrations do evolve differently. With the reaction system being exactly the same as in the validation of the reacting front (Subsection 5.3.2), there are two sources for this

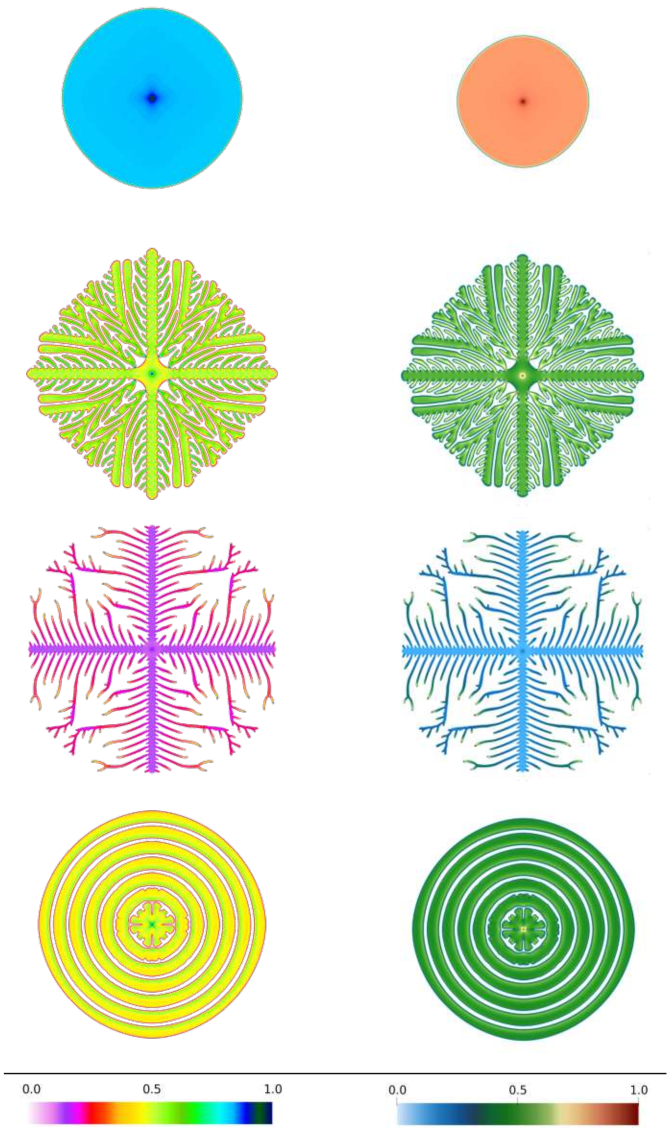


Figure 5.16: Side-by-side comparison of our reproduced growth patterns and the original ones of [9]

difference. First and foremost, Pribec *et al.* use a different reaction term (c.f., Chapter 3). Furthermore, we also impose the velocity boundary constraint on the reacting species - which we find sensible - whereas Pribec and co-authors do not do that, but rather apply a bounce back scheme. Tests from us suggest that this only causes a small deviation though, the majority originating from the reaction term. Therefore, we can ultimately only perform a qualitative comparison.
However, ultimately we conclude that our method is also capable of combining the effects

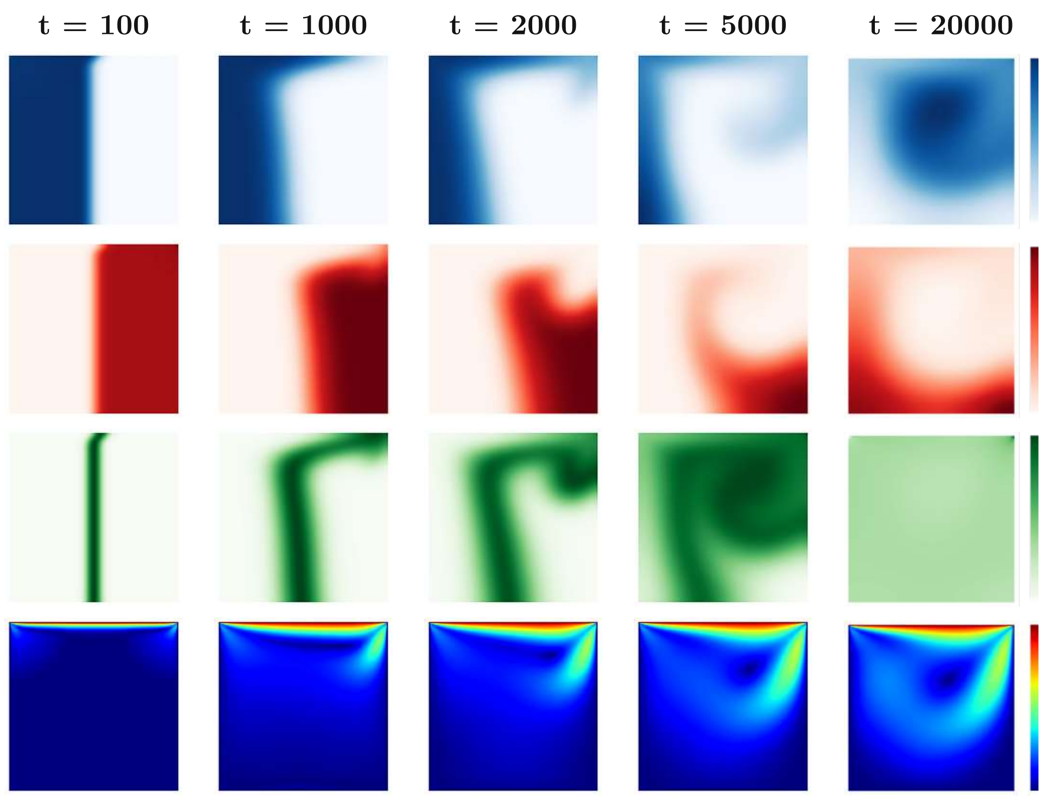


Figure 5.17: Evolution of the species and bulk velocity of the reacting LDC

of advection, diffusion and reaction within a system as we are in principle able to end up with similar results, and have successfully validated the respective reaction system and fluid problem.

Chemotaxis

Finally, after performing all the above validation steps successfully, we attempted to use our implementation to reproduce a traveling bacterial ring due to chemotaxis, as simulated by Hilpert in [7].
To do so, we created two dedicated species for the chemoattractant ($\mathcal{C}$) and bacteria ($\mathcal{B}$) which we parametrized using the same experimentally determined values as Hilpert (Table 1 in his publication). Furthermore, we also use the D2Q9 lattice and periodic boundary conditions, just like in the original paper. To parametrize the lattice, we set the simulation domain to the 10×10mm rectangle reported by Hilpert, and then use varying values Δx, resulting in different numbers of lattice nodes. Note that, like in [7], we use the lattice spacing as one of the characteristic quantities (refer to Subsection 5.2.2), next to the bacterial swimming speed and the initial concentrations of the chemoattractant and bacteria. Finally, the lattice time step is determined via the lattice spacing and the bacterial swimming speed.

We then used our CRN implementation to add the required Monod kinetics reaction for the attractant consumption to the system. Lastly, we added the computation of the chemoattractant gradient, which is required to determine the chemotactic velocity of the bacteria. This is done by having a buffer per lattice node that can hold the concentrations of its neighbors. Then, whenever we update the equilibrium distributions of the two species, we first perform an "inverse" streaming step on the concentrations of $\mathcal{C}$. This way, we obtain the current concentrations of all adjacent nodes (recall that the macroscopic moments are updated before f^{eq}, Figure 5.1). We then compute the gradient $\nabla c_{\mathcal{C}}$ of the chemoattractant concentration via a finite difference scheme, similar to Hilpert:

$$\nabla c_{\mathcal{C}}(\mathbf{x}, t) = \frac{1}{6} \sum_{i=1}^{8} \frac{c_{\mathcal{C}}(\mathbf{x} + \mathbf{c}_i \cdot \Delta t, t)}{\mathbf{c}_i \cdot \Delta t} \tag{5.5}$$

However, note that Hilpert's expressions is slightly different, as he sums up terms of the form $\mathbf{c}_i c_{\mathcal{C}}(x + \mathbf{c}_i, t)$, which is obviously only correct if $c = \Delta t = 1$ (which it is for Hilpert's non-dimensionalized case). Note though, that the scheme reported by Hilpert is in general not sensible. This can be noted when considering the unit equation of the non-dimensionalized form of his expression, resulting in a spatial gradient given in $m \cdot [c_{\mathcal{C}}]$. Naturally, the unit should be $\frac{[c_{\mathcal{C}}]}{m}$ (which it is for our version).

Figure 5.18 shows the best results we were able to obtain for the attractant, respectively bacteria, concentration. Note that both concentrations are given in their non-dimensional form (normalized by their respective initial concentration) and that we used a 100×100 lattice ($\Delta x = 0.1mm$). In correspondence to Hilpert we provide the dimensionalized simulation time for each concentration (in the format HH:MM:SS).

Observe that we were ultimately not able to reproduce the results reported by Hilpert, despite considerable time and effort put into this simulation. While our results show the initial formation of a bacterial ring, we could not achieve the "pinch-off" of the bacterial populations at the ring's center documented by Hilpert. Instead, eventually the substrate concentration at the center becomes so small, that the diffusion appears to dominate the chemotaxis, resulting in the initial signs of a ring formation slowly smearing out.
This behavior seems to be a result of the reaction dynamics not behaving as intended. To remedy this, we attempted the simulation on multiple modified versions of our implementation of this setup. Furthermore, we thoroughly investigated all the terms and parameters involved. However, ultimately all of this was to no avail.
Below, are the steps we have taken:

- Validated the feasibility of all our expressions using unit-afflicted quantities and the Pint unit system
- Used different expressions for the reactive term. Hilpert's scheme incorporates R slightly different than most other schemes, as often one finds the term to be scaled with $\frac{\Delta t}{w_i}$. However, Hilpert uses a factor of $\frac{f_{i,\mathcal{C}}(\mathbf{x},t)}{c_{\mathcal{C}}}$.

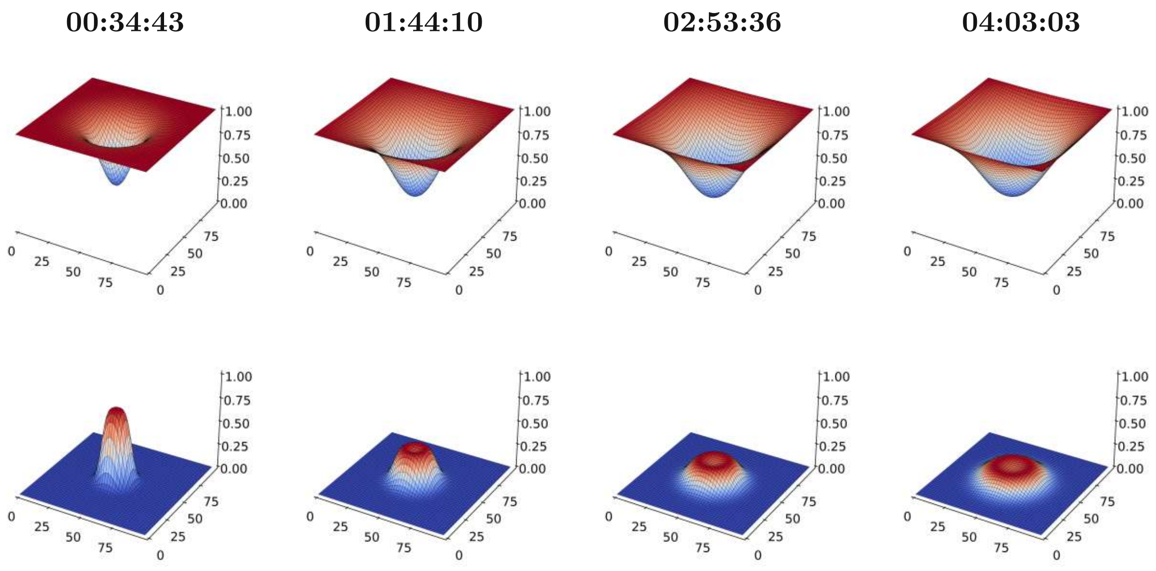


Figure 5.18: Evolution of the non-dimensional attractant (top) and bacteria (bottom) concentrations

- As stated above, our gradient expression is in general slightly different (although mathematically the same for $c = \Delta t = 1$). We also tried the original term, but could not observe a difference for the used non-dimensionalization.
- We cross-checked all the species' parameters with a follow-up work in [8] and the parameters' original source [60].
- We tried versions of the gradient that are be computed across the boundaries, and ones where the gradient is explicitly set to zero at boundaries.
- We varied the dimensional lattice spacing, as Hilpert's work does not explicitly define the one used. Our attempts ranged from small 50×50 to rather large 1000×1000 lattice nodes.
- It was also ensured that the total mass of bacteria is constant, i.e., not mistakenly affected by the reaction.

While our failure in a reproduction of the results remains slightly worrying, we can at this point also not rule out the possibility of a (minor) deviation between the scheme reported by Hilpert and the one actually used when simulating the travelling bacterial ring. However, while we would ideally be able to observe the same behavior as Hilpert, it is ultimately not vital to us. In the future we intend to use our own model for bacterial chemotaxis in any case. Thus, this failure is not really an obstacle for our future endeveaours, and setting up Hilpert's simulation is still a good showcase for the conciseness and simplicity with which comparable scenarios can be set up in our implementation.

CHAPTER 6

Conclusion

The LBM is quite remarkable as a simulator for the dynamics of particles, as it is wound up from assumptions and models of the microscopic interactions between them and manages the recovery of the macroscopic behavior as a result from the entirety of these interactions. It is especially striking how simple the scheme can be for the case of a simple single-species scenarios, where all the possible complexity resulting from particle collisions, can be hidden behind the simple BGK operator, which merely captures the overall relaxation towards an equilibrium distribution of the particle speeds. This is especially surprising when considering that the MBD is used as equilibrium distribution, which is continuous in the particle speed, while a discrete equilibrium distribution that defines only three discrete speeds is sufficient to recover behavior compliant with the NSE.

However, as simple, elegant and tangible as the method is for the single-species case, as ad hoc it can appear for scenarios involving multiple species. The reason for that is that every single element of the original LBM was conceived under the assumption of a single species. If one is interested in the dynamic behavior of a multi species system, this original single-species nature and its consequences cannot be neglected. We motivated that in Chapter 4. There we showed how the lattice speed and the mass of the simulated particles depend on each other. Simply using the same equilibrium distribution and lattice for all species is therefore not correct.
We continued our investigation on the consequences of this interdependence between the lattice and particle mass by discussing two possible approaches to deal with the matter. One approach was to use distinct lattices per species, resulting in different lattice speeds. However, in order for the populations, that are assigned to the different lattices, to be able to interact - which we require if we want to simulate, for example, reactions between species - the lattices must be related by some means. This can either happen in space, or, the dual approach, in time. However, as we argued, both methods bear unpleasant side effects and can hardly be applied due to severe restrictions on the relations between

particle masses.
The other approach is to use a single lattice, and to instead modify the equilibrium distributions of the distinct species.

Furthermore, we also investigated the feasibility of using the BGK operator in a multi-species system. For that purpose, a simple MD simulator was implemented and used to simulate the relaxation towards an equilibrium distribution of a set of particles, where the particles had varying masses and radii.
Much to our surprise, we discovered that the equilibrium distribution always appears to be the continuous MBD - irrespective of the co-existence of particles with other properties. However, this is only one part of the BGK operator, the other being the relaxation time τ that is used to model an exponential decline of the populations towards their equilibrium. Using our MD simulator we were not only able to capture this exponential relaxation in numerical simulations, but also to observe that the rate of this relaxation appears to be strongly affected by the presence of particles with distinct properties.

Finally, in Chapter 5, we presented the key properties and considerations of our open-source Python implementation of the LBM, on which we will base our future work on this topic. While we did not target a performant implementation, the created simulator comes with its own benefits, like the possibility to work with completely unit-afflicted quantities, a unit system for automatic conversion between dimensional and non-dimensional quantities, and an easy expandability.
We concluded this chapter by providing results for multiple benchmarks. This included classic CFD benchmarks like the Poiseuille and Couette flows, and the lid-driven cavity, for which analytical solutions, respectively established reference data in the literature, exist, as well as the reproduction of publications on LBMs for ARDE systems.

6.1 Future Work

In retrospect, the goal of this work was an ambitious one, and the scope of a master's thesis is, at the end, limited. Thus, it does hardly come as a surprise that there are some loose ends which will require further work in the future. Furthermore, our investigations brought up new questions and possibly even fertile ground for entirely new work. We will briefly discuss both in the following.

6.1.1 BGK for Multiple Species

The observation of the exponential relaxation towards the equilibrium distribution in the MD simulations was much too our own surprise, as we had initially been mainly focused on the equilibrium distribution itself. Especially the observation that the relaxation of species with different radii appears to adhere to an exponential function is deemed as noteworthy by us. This opens up a completely new avenue for future research.
If we can further back this observation, quantify the exponential's time constant in terms of the participating species' size and link this to the BGK relaxation time, we might be

able to elegantly extend the BGK operator to multiple species. This would be much desired, considering the operator's simplicity and resulting computational efficiency. In addition to that, obtaining the parameters required to simulate systems corresponding to real world applications and experiments, is non-trivial by itself. Just consider that among all the literature referenced in Chapter 3, only Hilpert and Long in [7, 8] explicitly mentioned the use of experimental real-world data and its origin.
However, there obviously exists some data that is more readily available as other. To be specific, "microscopic' parameters, like the mass and size, of atoms, molecules and also bacteria are in most cases well-known and documented. However, cross-species diffusion coefficients between bacteria and various chemical compounds, or other "macroscopic" transport parameters are in most cases likely not. By using the microscopic parameters, and the MD simulator, to obtain the mesoscopic ones required by the LBM though, one could possibly use the MD simulator on a small scale to set up the larger scale LBM without requiring macroscopic parameters. After all, recall that the LBM is based on microscopic models alone.

First steps towards that direction would be to extend our MD simulator such that it supports dynamic rather than static time steps, in order to get accurate measurements of the time constant rather than just the general form of the relaxation.
With a dynamic time step, the simulator could determine by itself how much it must advance the simulation for the next collision to occur (since all particles' positions and velocities are known this is possible). This allows the simulator to efficiently capture all collisions as accurate as possible (limited by the machine precision for the used data types for position and velocity).
Currently, a simulation time step must be specified. While this time step can be picked to be very small in order to gain the high resolution required for accurately capturing collisions, this results in a long run time for simulations containing a few thousand particles until they reach their equilibrium. Furthermore, even if we pick a comparably small, fixed, time step, we might still not be able to capture all collisions accurately in time, as according to the MBD particles can have an arbitrary speed. Therefore, theoretically, the time between a collision can also become arbitrary small (with decreasing probability though).

Additionally, extending the collision detection and handling to accommodate more shapes than just circular particles could be worthwhile, considering that some bacteria are, for example, rod-shaped. However, this would also require to equip the simulator's particle models with a rotational speed as well.

6.1.2 Chemotaxis and Bacterial Growth Patterns

The bacterial growth patterns produced by De Rosis *et al.* (and reproduced by us), shown in Figure 5.16, are arguably among the visually most stunning results of the (admittedly limited) applications of the LBM to the simulation of bacteria. However, the results should not blind us to the observation that the underlying system, that produces this rich behavior, is remarkably simple (c.f., (3.6)-(3.8)).

Ultimately, it would be very desirable to treat more elaborate systems with our simulator. Examples are the growth of *S.typhimurium* or *E.coli* colonies shown in Chapter 5 of [40]. Especially the *E.coli* system would be quite interesting, as it features not only bacterial chemotaxis and proliferation, as well as diffusion of *all* species, but also a coupling between this effects due to the bacteria producing the chemoattractant themselves. Furthermore, Murray already suggests an RDE system, consisting of three equations, with striking resemblance to the one used by De Rosis and co-authors.
Murray further summarizes many of the required parameters, which were already determined experimentally.

However, in order for us to simulate this system, we would need to deal with handling the chemotaxis of bacteria inside the LBM first. While the respective term proposed by Murray essentially looks like Fickian diffusion with a cross-diffusion diffusion coefficient (meaning the coefficient depends on the local concentrations of the respective species), this is something our implementation is currently not capable of. And while we are aware that researchers have already approached the problem of cross-diffusion in a different context, for example [61], studying their work and implementing it was ultimately not within the scope of this thesis. However, we might pursue the use of cross-diffusion for this in the future.
Other interesting ways to simulate the *E.coli* system could be to either get the chemotaxis model of Hilpert in [7] to work and to experiment with using it for this particular system or, preferred by us, use an entirely different model for chemotaxis. However, reviewing the available models and selecting one that is suitable for our purposes is a task left for the future.

6.1.3 Split Simulations of Fluid Flow and Reaction System

An insight we gained during this work, is that the fluid flow can be considered invariant to the chemically reacting species (assuming that the species are sufficiently diluted). However, this means that we could ultimately use a dedicated open-source fluid simulator, ideally but not necessarily LBM-based, like for example the parallel LBM implementation Palabos [62] to create the fluid flow fields we require. This would save a lot of implementation and validation effort, while likely also resulting in more accurate results for the fluid flow. We could then concentrate our efforts on the simulation of the reacting species rather than on the underlying CFD problem.

Another way to cut short on the effort of the CFD part could be to use simple geometries and boundary conditions for approximating the desired setup.
For example, since we are ultimately interested in simulating a stirred bio-reactor, we would as a first approximation need a fluid flow field featuring a vortex at the simulation domain's center. In some very first, initial experiments we were able to produce something akin to this by simulating a square lid-driven cavity where two opposing walls are driven in opposite directions with equal speed. By fiddling around with the parameters we could get the two vortices two merge in the center. The fully developed flow field was then used in a simulation of the disk scenario of the system in [9], where we used a two-dimensional

Gaussian function to "suppress" the parts of the field that did not fit the desired vortex. Figure 6.1 shows some results of these first tries (for the concentration of active bacteria). However, further experiments and comparison to results from experiments our literature would be interesting.

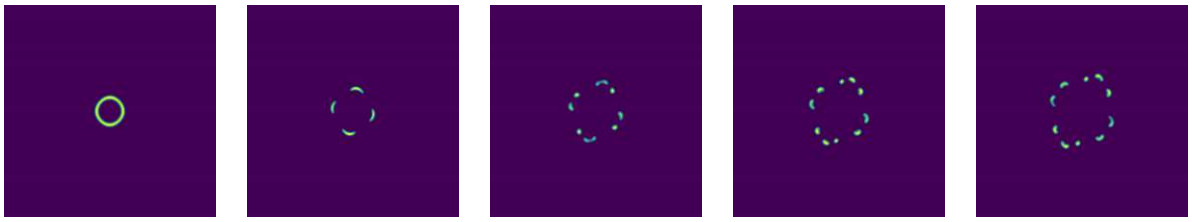


Figure 6.1: Evolution of the bacterial concentration of the disk case of [9] subject to an external velocity field

6.1.4 Efficient Implementation

As discussed in Chapter 5, performance was not a primary concern for us during this work. However, in the future we would like to speed up our implementation by making use of the LBM's inherent parallelizability. To stress this desire, be noted that the simulation of the bacterial growth patterns in Figure 5.16 already took a few hours on the 2000×2000 node lattice to finish the roughly 10^5 simulation steps required to reproduce the "DLA" pattern (third row in the figure) for three species.

Parallel Implementation on General Purpose Hardware

The most straightforward way to parallelize our implementation is to make use of concurrent general purpose hardware like GPUs (and nowadays virtually all CPUs in modern mainstream devices).
In Chapter 13 of their book, Krüger *et al.* discuss parallel implementations of the basic LBM written in C++. For an implementation based on multicore CPUs they use OpenMP and assign the computation of each "column" of the used two-dimensional lattice to a thread. The only non-local step in the basic LBM, the streaming, is handled by using double buffering and letting each thread *read* the populations of the neighbors for each of its nodes, rather than writing its own populations to the respective memory locations. For their GPU implementation, Krüger *et al.* target NVIDIA GPUs using the company's proprietary CUDA framework, assigning each lattice node to a thread. All in all, they show that the LBM can indeed greatly profit from parallel hardware, but also that harnessing its power requires careful thoughts about the used memory and accesses to it.

We want to keep our implementation in Python though, for the reasons given in Chapter 5. To make our Python LBM leverage multiple CPU cores, we performed some tests with the *Numba* package, which does not only perform just-in-time compilation for the Python code, but is to some extent also capable of parallelizing loops. However, we could not get a version of our code with Numba to surpass the performance of the basic version

using Numpy (which makes uses of vector operations). But as Numba is in its core not a package designed for parallelization, a multicore implementation should use something different anyway. For example, *mpi4py* could be worthwhile to consider, allowing to use multiple processes (thus mitigating the multi-threading limitations due the global interpreter lock in the most common Python implementation). However, one would first need to investigate whether using it causes more speed-up than overhead.
Nevertheless, when improving our Python code with respect to performance, we would prefer using a GPU in any case, due to its significantly higher core count and the LBM.'s core computations being simple and virtually branch-free. A suitable way of doing this could be by using PyOpenCL and writing GPU kernels for the collision and reaction steps, as well as for updating the macroscopic quantities. The populations would then need to reside on the GPU as well, such that costly transfers between the CPU and GPU memories are mitigated. In general, as mentioned in [63], the memory bandwidth is the bottleneck of GPU-based implementations and effectively prohibits to use the complete parallelization potential of GPUs.

LBM Hardware Accelerator

An alternative to using general purpose hardware for parallizing our implementation would be the development of a dedicated hardware accelerator based around a Field Programmable Gate Array (FPGA).
Such an accelerator could make use of memory blocks embedded into the programmable logic of the FPGA to store the populations, as well as density and fluid velocity next to the hardware responsible for performing the computation on this data. This can potentially overcome the bandwidth-bottleneck of GPU-based implementations. To really speeds things up, one would further require to use an FPGA with hard digital signal processors embedded into the programmable logic, as the LBM requires floating point computations. Without dedicated hardware in the FPGA, this functionality would need to be implemented in the programmable logic, causing a lot of area overhead.

The available literature suggests that such an accelerator is, in principle, possible. For a concise summary we refer to [63].
In general, there are two ways of accelerating an LBM using an FPGA. First, one can accelerate the non-local collision step and macroscopic quantity update by letting them be performed by custom co-processors on an FPGA [64]. In parallel to this, a CPU can take care of the non-local streaming step. Note that the CPU and co-processors require a shared memory for this approach. However, the authors of [64] concluded that, while achieving a speedup in comparison to a pure CPU implementation, the bandwidth of the shared memory limited the possible performance increase.
Another approach is to perform all the computation on the FPGA, including the streaming. In [65] Sano and Yamamoto report on an FPGA implementation that uses custom streaming processors to compute all steps of the LBM on the FPGA. In its simplest form, each such processor performs the computation for a single lattice node, where the processor receives the node's populations (in the paper they use the D2Q9 lattice, hence

9 single-precision words) and lattice coordinates (x, y). However, the authors also provide a version that works on multiple nodes concurrently, receiving the mentioned input for n distinct nodes. These processors are than chained together to form a pipeline, thus computing multiple simulation time steps at once. However, for the FPGA they used their design was ultimately worse than a GPU implementation they compared their work to. Nevertheless, the authors estimate that their accelerator surpasses the performance of this GPU implementation when implemented on a more potent FPGA.

CHAPTER 7

Overview of Generative AI Tools Used

During this thesis, generative AI tools were utilized to assist with various aspects of the work. Specifically, the following tools were used:

- **ChatGPT (GPT-4)**: [66]: Developed by OpenAI, ChatGPT is a language model designed to assist with generating human-like text based on input prompts.
- **Kagi Search** [67]: Kagi Search is a search engine that offers ad-free, privacy-focused search results and also contains dedicated generative AI tools for specific purposes such as *FastGPT* and *Quick Answer*.
- **Gemini** [68]: Developed by Google Deepmind and Google AI, Gemini is, akin to ChatGPT, a language model designed to assist with generating human-like text based on input prompts.

In its overwhelming majority the use of such tools was restricted to the following tasks:

- As a complement to conventional search engines for increased efficiency (ChatGPT, Gemini, Kagi).
- Summarizing material in order to quickly determine its relevance (ChatGPT, Kagi).
- Assisting in writing tedious code snippets, such as the MD simulator's visualization part (ChatGPT).
- Querying the documentation of the used Python libraries (ChatGPT, Kagi)

Note that all the content generated by the AI tools was evaluated thoroughly before use in this work and that not a single passage of text was taken without substantial changes.

List of Figures

List of Tables

Bibliography

[1] Timm Krueger, Halim Kusumaatmaja, Alexandr Kuzmin, Orest Shardt, Goncalo Silva, and Erlend Magnus Viggen. *The Lattice Boltzmann Method: Principles and Practice*. Graduate Texts in Physics. Springer, 2016.

[2] Dieter Wolf-Gladrow. *Lattice-Gas Cellular Automata and Lattice Boltzmann Models - An Introduction*, volume 308. Springer, Berlin, 2000.

[3] Keerti Vardhan Sharma, Robert Straka, and Frederico Wanderley Tavares. Lattice Boltzmann Methods for Industrial Applications. *Industrial & Engineering Chemistry Research*, 58(36):16205–16234, September 2019. Publisher: American Chemical Society.

[4] GR McNamara and Gianluigi Zanetti. Use of the boltzmann equation to simulate lattice-gas automata. *Physical review letters*, 61:2332–2335, 12 1988.

[5] Tobit Flatscher. The lattice-boltzmann method for multi-component flows in porous media. Master's thesis, Graz University of Technology, 2018.

[6] B. Alberts, D. Bray, K. Hopkin, A.D. Johnson, J. Lewis, M. Raff, K. Roberts, and P. Walter. *Essential Cell Biology*. CRC Press, 2015.

[7] Markus Hilpert. Lattice-boltzmann model for bacterial chemotaxis. *Journal of Mathematical Biology*, 51(3):302–332, September 2005.

[8] Wei Long and Markus Hilpert. Lattice-boltzmann modeling of contaminant degradation by chemotactic bacteria: Exploring the formation and movement of bacterial bands. *Water Resources Research*, 44(9), 2008.

[9] Alessandro De Rosis, Ajay B. Harish, and Weiguang Wang. Lattice Boltzmann modelling of bacterial colony patterns. *Computational Mechanics*, July 2024.

[10] Matthias Függer, Manish Kushwaha, and Thomas Nowak. *Digital Circuit Design for Biological and Silicon Computers*, pages 153–171. 04 2020.

[11] Tamas I. Gombosi. *Gaskinetic Theory*. Cambridge Atmospheric and Space Science Series. Cambridge University Press, 1994.

[12] Raphael D. Levine. *Molecular Reaction Dynamics*. Cambridge University Press, Cambridge, 2005.

[13] G. K. Batchelor. *An Introduction to Fluid Dynamics*. Cambridge Mathematical Library. Cambridge University Press, 2000.

[14] F. Mattioli. *Principles of Fluid Dynamics*. Franco Mattioli, 2010.

[15] Aleksandar Donev, Salvatore Torquato, and Frank H. Stillinger. Neighbor list collision-driven molecular dynamics simulation for nonspherical hard particles. I. Algorithmic details. *Journal of Computational Physics*, 202(2):737–764, January 2005.

[16] Linfeng Zhang, Jiequn Han, Han Wang, Roberto Car, and Weinan E. Deep potential molecular dynamics: A scalable model with the accuracy of quantum mechanics. *Phys. Rev. Lett.*, 120:143001, Apr 2018.

[17] J. Hardy, Y. Pomeau, and O. de Pazzis. Time Evolution of a Two-Dimensional Classical Lattice System. *Physical Review Letters*, 31(5):276–279, July 1973. Publisher: American Physical Society.

[18] U. Frisch, B. Hasslacher, and Y. Pomeau. Lattice-Gas Automata for the Navier-Stokes Equation. *Physical Review Letters*, 56(14):1505–1508, April 1986. Publisher: American Physical Society.

[19] Stephen Wolfram. Cellular automaton fluids 1: Basic theory. *Journal of Statistical Physics*, 45(3):471–526, November 1986.

[20] Gary D. Doolen. Preface. In Gary D. Doolen, Uriel Frisch, Brosl Hasslacher, Steven Orszag, and Stephen Wolfram, editors, *Lattice Gas Methods For Partial Differential Equations*, volume 4 of *Santa Fe Institute Studies in the Sciences of Complexity*. Addison-Wesley, 1990.

[21] Tsutomu Shimomura, Gary D. Doolen, Brosl Hasslacher, and Castor Fu. Calculations using lattice gas techniques. *Los Alamos Science Special Issue*, pages 201–210, 1987.

[22] Gyun Eun Lee. The boltzmann equation. `https://gyu-eun-lee.github.io/academic/gso_boltzmann.pdf`. Accessed: 2024-13-02.

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

[24] Y. H Qian, D D'Humières, and P Lallemand. Lattice BGK Models for Navier-Stokes Equation. *Europhysics Letters (EPL)*, 17(6):479–484, February 1992.

[25] Y. H. Qian and S. A. Orszag. Scalings in diffusion-driven reaction a+b→c: Numerical simulations by lattice bgk models. *Journal of Statistical Physics*, 81:237–253, Oct. 1995.

[26] Albert Puig Arànega. *Review of boundary conditions and investigation towards the development of a growth model: a Lattice Boltzmann Method approach.* Phd thesis, Universitat Rovira i Virgili, Tarragona, Spain, October 2016.

[27] Silvina Ponce Dawson, Suhao Chen, and G Doolen. Lattice boltzmann computations for reaction-diffusion equations. *Journal of Chemical Physics*, 98:1514, 01 1993.

[28] C. S. Bresolin and A. A. M. Oliveira. An algorithm based on collision theory for the lattice Boltzmann simulation of isothermal mass diffusion with chemical reaction. *Computer Physics Communications*, 183(12):2542–2549, December 2012.

[29] Ivan Pribec, Anže Hubman, Tomaz Urbic, and Igor Plazl. A discrete reactive collision scheme for the lattice Boltzmann method. *Journal of Molecular Liquids*, 332:115871, 2021.

[30] A. A. Moaty Sayed, M. A. Hussein, and T. Becker. An innovative lattice Boltzmann model for simulating Michaelis-Menten-based diffusion-advection kinetics and its application within a cartilage cell bioreactor. *Biomechanics and Modeling in Mechanobiology*, 9(2):141–151, April 2010.

[31] Marc R Roussel. Reaction–diffusion equations. In *Nonlinear Dynamics*, 2053-2571, pages 13–1 to 13–16. Morgan & Claypool Publishers, 2019.

[32] So Kitsunezaki. Interface dynamics for bacterial colony formation. *Journal of the Physical Society of Japan*, 66(5):1544–1550, 1997.

[33] James D. Murray. *Mathematical Biology.* Springer, Berlin, Heidelberg, 1993.

[34] Yasmin Dolak. *Advection-dominated models for chemotaxis.* Phd thesis, Technische Universität Wien, Vienna, Austria, October 2004.

[35] R D Kingdon and P Schofield. A reaction-flow lattice boltzmann model. *Journal of Physics A: Mathematical and General*, 25(14):L907, jul 1992.

[36] Davide Alemani. *A Lattice Boltzmann numerical approach for modelling reaction-diffusion processes in chemically and physically heterogeneous environments.* Phd thesis, Universitè de Genève, Geneva, Switzerland, 2007. Available at `https://access.archive-ouverte.unige.ch/access/metadata/0d8b6162-edb5-49c5-8e6d-1fa21e385bf7/download`.

[37] S.W. Benson. *The Foundations of Chemical Kinetics*, chapter 1. McGraw-Hill series in advanced chemistry. R.E. Krieger, 1982.

[38] Yousef Abdollahzadeh, Zahra Mansourpour, Hamed Moqtaderi, Seyed Nader Ajayebi, and Mahyar Mohaghegh Montazeri. A molecular collision based Lattice Boltzmann method for simulation of homogeneous and heterogeneous reactions. *Chemical Engineering Research and Design*, 136:456–467, August 2018.

[39] A. Moaty, Mohamed Hussein, and T Becker. An innovative lattice boltzmann model for simulating michaelis–menten-based diffusion–advection kinetics and its application within a cartilage cell bioreactor. *Biomechanics and modeling in mechanobiology*, 9:141–51, 08 2009.

[40] J.D. Murray. *Mathematical Biology II: Spatial Models and Biomedical Applications*. Interdisciplinary Applied Mathematics. Springer New York, 2011.

[41] Masayasu Mimura, Hideo Sakaguchi, and Mitsugu Matsushita. Reaction–diffusion modelling of bacterial colony patterns. *Physica A: Statistical Mechanics and its Applications*, 282(1):283–303, 2000.

[42] Michael E. McCracken and John Abraham. Lattice Boltzmann methods for binary mixtures with different molecular weights. *Physical Review E*, 71(4):046704, April 2005.

[43] N. Looije, J.J.J. Gillissen, S. Sundaresan, and H.E.A. Van den Akker. Introducing a variable speed of sound in single-component lattice boltzmann simulations of isothermal fluid flows. *Computers & Fluids*, 167:129–145, 2018.

[44] Erlend Magnus Viggen. *The lattice Boltzmann method: Fundamentals and acoustics*. PhD thesis, Norwegian University of Science and Technology, Trondheim, 2014.

[45] Harry E.A. Van den Akker, Renske Donkers, Githin T. Zachariah, and Orest Shardt. On using variable molecular masses in multicomponent lattice boltzmann simulations. *Journal of Computational Science*, 54:101432, 2021.

[46] Hernan E. Grecco and Pint Developers. Pint. https://pint.readthedocs.io/en/stable/. Version: 0.23.

[47] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020.

[48] bio-lbm. `https://github.com/Bawady/bio-lbm`, 2024.

[49] Xiaoyi He and Qisu Zou. Analysis and boundary condition of the lattice Boltzmann BGK model with two velocity components, July 1995.

[50] Qisu Zou and Xiaoyi He. On pressure and velocity boundary conditions for the lattice Boltzmann BGK model. *Physics of Fluids*, 9(6):1591–1598, 06 1997.

[51] Martin Feinberg. *Foundations of Chemical Reaction Network Theory*. Applied Mathematical Sciences. Springer International Publishing, 01 2019.

[52] Jonathan Viquerat. lbm. `https://github.com/jviquerat/lbm/commits/master/`, 2020. Commit 64023fc.

[53] T. Lemée, G. Kasperski, G. Labrosse, and R. Narayanan. Multiple stable solutions in the 2d symmetrical two-sided square lid-driven cavity. *Computers & Fluids*, 119:204–212, 2015.

[54] Shuling Hou, Qisu Zou, Shiyi Chen, Gary Doolen, and Allen C. Cogley. Simulation of cavity flow by the lattice boltzmann method. *Journal of Computational Physics*, 118(2):329–347, 1995.

[55] U Ghia, K.N Ghia, and C.T Shin. High-re solutions for incompressible flow using the navier-stokes equations and a multigrid method. *Journal of Computational Physics*, 48(3):387–411, 1982.

[56] Xiuqiao Xiang and Baochang Shi. Flow simulation of lid-driven rectangular cavity by using lattice boltzmann method, 11 2022.

[57] Charles-Henri Bruneau and Mazen Saad. The 2d lid-driven cavity problem revisited. *Computers & Fluids*, 35(3):326–348, 2006.

[58] Chai Zhen-Hua, Shi Bao-Chang, and Zheng Lin. Simulating high reynolds number flow in two-dimensional lid-driven cavity by multi-relaxation-time lattice boltzmann method. *Chinese Physics*, 15(8):1855, aug 2006.

[59] Alessandro De Rosis, Ajay B. Harish, and Weiguang Wang. Bacterial colony. `https://github.com/SIG-LBM-Multiphysics-Modelling/BacterialColony`, 2024. Commit d89c5c4.

[60] Randall B. Marx and Michael D. Aitken. Quantification of chemotaxis to naphthalene by pseudomonas putida g7. *Applied and Environmental Microbiology*, 65(7):2847–2852, 1999.

[61] Chengjie Zhan, Zhenhua Chai, and Baochang Shi. A lattice Boltzmann model for the coupled cross-diffusion-fluid system. *Applied Mathematics and Computation*, 400:126105, 2021.

[62] Jonas Latt, Orestis Malaspinas, Dimitrios Kontaxakis, Andrea Parmigiani, Daniel Lagrava, Federico Brogi, Mohamed Ben Belgacem, Yann Thorimbert, Sébastien Leclaire, Sha Li, Francesco Marson, Jonathan Lemus, Christos Kotsalos, Raphaël Conradin, Christophe Coreixas, Rémy Petkantchin, Franck Raynaud, Joël Beny, and Bastien Chopard. Palabos: Parallel Lattice Boltzmann Solver. *Computers & Mathematics with Applications*, 81:334–350, 2021.

[63] Wesson Altoyan and Juan. J. Alonso. Accelerating the lattice boltzmann method. In *2023 IEEE Aerospace Conference*, pages 1–20, 2023.

[64] Kentaro Sano, Oliver Pell, Wayne Luk, and Satoru Yamamoto. Fpga-based streaming computation for lattice boltzmann method. In *2007 International Conference on Field-Programmable Technology*, pages 233–236, 2007.

[65] Kentaro Sano and Satoru Yamamoto. Fpga-based scalable and power-efficient fluid simulation using floating-point dsp blocks. *IEEE Transactions on Parallel and Distributed Systems*, 28(10):2823–2837, 2017.

[66] OpenAI. ChatGPT (GPT-4). `https://openai.com/chatgpt/`. Version: GPT-4o.

[67] Kagi Inc. Kagi search. `https://kagi.com/`. Version: Not publicaly provided.

[68] Google DeepMind, Google AI. Gemini. `https://gemini.google.com/app`. Version: Not publicaly provided.

APPENDIX A

Acronyms and Symbols

Acronyms

ARDE	Advection-Reaction-Diffusion Equation.
BE	Boltzmann Equation.
BGK	Bhatnagar-Gross-Krook.
CA	Cellular Automata.
CE	Continuity Equation.
CFD	Computational Fluid Dynamics.
CRN	Chemical Reaction Network.
FPGA	Field Programmable Gate Array.
LBE	Lattice Boltzmann Equation.
LBM	Lattice Boltzmann Method.
LDC	Lid-Driven Cavity.
LGCA	Lattice-Gas Cellular Automata.
MBD	Maxwell-Boltzmann Distribution.
MD	Molecular Dynamics.
NEBB	Non-Equilibrium Bounce Back.
NSE	Navier-Stokes Equations.
PDE	Partial Differential Equation.

RDE	Reaction-Diffusion Equation.

Symbols of Physical Quantities

Symbol	Description	Depends	Unit
D	Diffusion coefficient		m^2/s
N	Number of particles		1
R	Generic reaction term	$\mathbf{x}$,t, species	3D: $mol/(m^3s)$
T	Temperature		K
V	Volume		m^3
$\bar{\mathbf{v}}_p$	Average particle velocity		m/s
χ_{tx}	Chemotactic sensitivity of $\mathcal{B}$ to $\mathcal{C}$	$c_\mathcal{B}, c_\mathcal{C}$	m^2/s
η_b	Bulk viscosity		$kg \cdot m/s$
η_s	Shear visocisty		$kg \cdot m/s$
m_p	Mass of a single particle		kg
μ	Bacterial motility coefficient		m^2/s
ρ	Mass density	$\mathbf{x}$, t	kg/m^3
$\mathcal{A}, \mathcal{B}, \mathcal{C}, \ldots$	Species $\mathcal{A}, \mathcal{B}, \mathcal{C}, \ldots$		
τ	BGK collision operator relaxation time		s
$\mathbf{j}$	Momentum density	$\mathbf{x}$, t	$kg/(m^2s)$
$\mathbf{u}_{tx}$	Chemotactic velocity; 2D: $\mathbf{u} = (u_{c,x}, u_{c,y})$	$c_\mathcal{X}, \nabla c_\mathcal{X}$	m/s
$\mathbf{u}$	Macroscopic fluid velocity; 2D: $\mathbf{u} = (u_x, u_y)$	$\mathbf{x}$, t	m/s
$\mathbf{v}$	Velocity; 2D: $\mathbf{v} = (v_x, v_y)$		m/s
$\mathbf{x}$	Cartesian coordinates; 2D: $\mathbf{x} = (x_x, x_y)$		m
$c_\mathcal{X}$	Concentration of species $\mathcal{X}$	$\mathbf{x}$,t	3D: mol/m^3
f^{eq}	Equilibrium particle distribution function	$\mathbf{x}$, t, $\mathbf{v}$	kgs^3/m^6
f^{neq}	Non-equilibrium particle distribution function	$\mathbf{x}$, t, $\mathbf{v}$	kgs^3/m^6
f	Particle distribution function	$\mathbf{x}$, t, $\mathbf{v}$	kgs^3/m^6
k_B	Boltzmann constant, k_B=1.3806		J/K
m	Mass		kg
p	Pressure	$\mathbf{x}$, t	kg/ms^2
t	Time		s
f^{MB}	Maxwell Boltzmann Distribution	$\mathbf{v}$,T	1

Miscellaneous Symbols

Symbol	Description
Ω	Collision operator
$\alpha, \beta, \gamma, \delta$	Einstein index convention indices
Ω^{BGK}	BGK collision operator [23]

Symbol	Description
$\delta_{\alpha\beta\dots}$	Generalized Dirac delta
$\mathscr{C}$	Set of chemical complexes
$\mathscr{R}$	Set of chemical reactions
$\mathscr{S}$	Set of chemical species
$\overline{\mathbb{R}}_+$	Set of non-negative real numbers
∇	The del operator; 2D: $\nabla = (\partial_x, \partial_y)$

Symbols of Lattice Quantities

Symbol	Description	Depends	Unit
D	Lattice dimensions		1
Q	Cardinality of the lattice velocity vector set		1
Δt	Lattice time step		s
Δx	Lattice spacing		m
f_i^{eq}	Equilibrium particle distribution function for velocity $\mathbf{c}_i$	$\mathbf{x}$,t	2D: kgs^2/m^4
$\mathbf{c}_i$	Lattice velocity vectors; 2D: $\mathbf{c}_i = (c_{ix}, c_{iy})$		m/s
$\mathbf{n}_i$	Particle occupation information of lattice velocity vector i	$\mathbf{x}$,t	1
f_i	Particle distribution function for velocity $\mathbf{c}_i$	$\mathbf{x}$,t	2D: kgs^2/m^4
c_s	Lattice speed of sound		m/s
c	Lattice speed		m/s
w_i	Lattice weights	$\mathbf{x}$,t	1

APPENDIX B

Physical Laws and Equations

Euler Equation (force-free) as in [14]

$$\rho\partial_t \mathbf{u} + \rho(\mathbf{u}\nabla)\mathbf{u} = -\nabla p \tag{B.1}$$

Average Kinetic Particle Energy (2D)

$$\frac{m_p \bar{\mathbf{v}}_p}{2} = k_B T \tag{B.2}$$

Ideal Gas Law

$$pV = N k_B T \tag{B.3}$$