CHALMERS

Modelling of electrokinetic flow using the lattice-Boltzmann method

Thesis for the degree of Master of Science

ANDREAS BÜLLING

Department of Mathematical Sciences
Division of Mathematics
Chalmers University of Technology
Gothenburg, Sweden 2012

Modelling of electrokinetic flow using the lattice-Boltzmann method

Thesis for the degree of Master of Science

ANDREAS BÜLLING

Department of Mathematical Sciences
Division of Mathematics
Chalmers University of Technology
Gothenburg, Sweden 2012

Modelling of electrokinetic flow using the lattice-Boltzmann method
Thesis for the degree of Master of Science
ANDREAS BÜLLING

Department of Mathematical Sciences
Division of Mathematics
Chalmers University of Technology
SE-412 96 Gothenburg
Sweden
Telephone +46 (0)31-772 1000

Chalmers Reproservice
Gothenburg, Sweden 2012

Modelling of electrokinetic flow using the lattice-Boltzmann method
Thesis for the degree of Master of Science
ANDREAS BÜLLING
Department of Mathematical Sciences
Division of Mathematics
Chalmers University of Technology

Abstract

The lattice-Boltzmann method is used to model flow in electrokinetic systems. A modelling approach based on the coupling of Navier-Stokes, Nernst-Planck and Poisson's equation of electrostatics is utilised. Three lattice-Boltzmann methods are formulated for the three equations respectively.

The method is implemented in `C++` with the aim of being high performing. Topics as locality, instruction pipelines and parallel computing are considered. The implementation is tested for a number of classic examples with known solutions, e.g. Taylor-Green vortex flow, an Helmholtz equation and an advection-diffusion situation. The computed solutions agree well with the analytic solutions.

The physical systems modelled consists mainly of various charged channel flows of ionic solutions. Electrokinetic effects, such as electroosmosis and the electrovicous effect are studied. This is done in thin channels where the thickness of the electrical double layers is comparable to the channel dimension. The electroviscous effect is shown to slow the flow down and a local minimum is found in the velocity profile for thick enough double layers. Other more complicated systems are also studied; electroosmotic flow in a channel with heterogeneously charged walls and flow in a an array of charged squares.

Keywords: lattice-Boltzmann, electrokinetics, electrohydrodynamics, Nernst-Planck, Poisson-Boltzmann, high performance computing.

Acknowledgements

I would hereby like to express my gratitude to my supervisor Alexei Heintz. You did always have time for questions and discussions even during your spare time and on your vacation. Also the precise amount of guidance and the freedom given is very much appreciated. I am thankful for having been given the opportunity of working on a project where the fields of my interest from physics, mathematics and programming were all present.

Andreas Bülling, Göteborg, January 12, 2013

Contents

List of Figures

List of Abbreviations

API Application Programming Interface

BGK Relaxation type collision operator (Bhatnagar, Gross and Krook)

CE, C-E Chapman-Enskog

CFD Computational Fluid Dynamics

CISC Complex Instruction Set Computing

CPU Central Processing Unit

D2Q9 2D lattice with 9 discrete velocities

EDL Electrical Double Layer

HDD Hard Disk Drive

L1, L2, L3 Level 1,2,3 (cache)

LBE Lattice-Boltzmann Equation

LBM Lattice-Boltzmann Method

LGA Lattice Gas Automata/Automaton

LL Last level (cache)

MLUPS Million Lattice Updates Per Second

MPI Message Passing Interface

NP, N-P Nernst-Planck equation

NS, N-S Navier-Stokes equations

PB, P-B Poisson-Boltzmann model/equation

PE, P-E Poisson's equation of electrostatics

PU Processing Unit

RAM Random Access Memory

RISC Reduced Instruction Set Computing

STP Standard Temperature and Pressure

UNIX A computer operating system

1

Introduction

This thesis deals with modelling of physical problems in the interdisciplinary field of hydrodynamics and electrostatics. The tool used for realising this is the new and promising but somewhat immature lattice-Boltzmann method. This is a method that is still under development but is today used in practical applications both in industry and academy.

1.1 Background

There is currently an ongoing project at the mathematics faculty of Chalmers University in producing a modelling package that should be able to deal with transport of various liquids and particles through complicated structures. The method of choice has fallen upon the lattice-Boltzmann method for its suitable characteristics in the systems of interest.

This work aims to investigate the possibility and procedure for taking electrical effects into account in the modelling of charged fluids. More theoretical questions about the method itself and of the physics involved is of interest as well as how the method may be effectively implemented on a computer.

From both industry and academy, there is a demand on the modelling of this kind of physics. For instance, in medical sciences, accurate modelling of transport of charged fluids is a fundamental ingredient in understanding biological systems and to be able to manipulate them. As a consequence of the always so present desire of more environmental friendly ways of using the planet, automotive industry are now engineering electrical cars. A great challenge is to produce high performing and durable batteries, the ability to accurate model the electrolytes in the batteries is indeed an advantage in achieving this.

1.2 Outline

The text is structured in five main chapters. In chapter 2, the physics involved and the equations of interest are presented. This is followed by chapter 3 where the lattice-Boltzmann method is formulated for the different equations of interest. Also an introduction to the method as well as some discussion on different boundary conditions is given here. In chapter 4, the implementation of the method is discussed together with some general aspects that is important to have in mind in order to produce a high performing code. The implementation is then tested for classic examples with known solutions in chapter 5. Finally some results in electrokintetics are presented and discussed in chapter 6. Here, the focus is rather on the physics of the simulated systems then on LBM aspects of the problems. These aspects, such as grid dimensions, how LBM parameters relate to physical quanities etc. are discussed for the problems in chapter 5.

1.3 Previous work

An extensive treatment of both theory and experiments in the field of electrokinetics is carried out in [1]. Mainly the Poisson-Boltzmann model is used in the modelling but also in some situations, the model used in this work based on the coupling of Navier-Stokes, Nernst-Planck and Poisson's equation of electrostatics is discussed. Also in [27], this modelling approach is used. However, the computational model is not the Lattice-Boltzmann method (LBM).

There are a lot of formulations of the LBM for the Navier-Stokes equations as the method typically is used in the modelling of fluid dynamics. Not so common are formulations for the Nernst-Planck and Poisson's equation. However there are a few, e.g. in [4] and [26] formulations for the Poisson's equation is discussed. In [28] a complete formulation for the three equations are presented together with some example simulations of electrokinetic systems. The formulation presented in [28] will not be completely the same as the one used in this work as is discussed in later chapters of this text.

2

Electrohydrodynamics in microchannels

In this chapter, the fundamental physics behind electrokinetic flow, important for later discussions, will be presented. Particularly, a modelling approach based on the coupling of Navier-Stokes, Nernst-Planck and Poission's equations is given.

2.1 Basic concepts of electrokinetic flow

Electrohydrodynamics involves the study of electric phenomena on fluid flow. How fluids carrying electrical charges (electrolytes) react upon external electrical fields or interact with charged objects are examples of problems that arise in this field.

2.1.1 Electrical double layers

As a charged object is brought into contact with an electrolyte it is, qualitatively, easily deduced that ions with a sign of charge opposite to that of the object will be attracted to the object and ions with the same sign of charge will be repelled. These two distinct categories of ions will from hereon be referred to as counter- and co-ions respectively. In this case, for a neutral electrolyte, a surplus of counter-ions will be present in the direct vicinity of the object and a surplus of co-ions will be present at some other location further from the object.

The area with a surplus of counter-ions in an electrolyte in contact with a charged object is often referred to as an electrical double layer (EDL). Two distinct regions will be formed in this area, thus the name double layer. The two layers are often referred to as the Stern layer (adsorbed ions) and the diffusive layer (mobile ions). The Stern layer is usually several orders of magnitude thinner than the diffusive layer and is therefore seldom considered when it comes to modelling [1].

2.1.2 Electroosmosis

As a fluid carrying a net charge, e.g. in the diffusive layer of an EDL, is under influence of an electric field, the charged particles will move due to the electric forces. As the charge particles move, they will affect the surrounding liquid, causing it to move as well. This liquid motion is often referred to as electroosmotic flow. [1]

2.2 Complete physical model

To model the fluid motion of a charged fluid under influences of electrostatic forces, a coupling between different models is considered.

The electric field and potential in the system are obtained from solving Poisson's equation (PE) for electrostatics (section 2.3) with a given charge density. This charge density is obtained from a set of Nernst-Planck (NP) equations (section 2.4) by including effects on the charge distribution from the electric field previously mentioned, diffusion and advection. One NP equation is solved for each different ion species in the solution. For instance in a 1:1 solution, two equations are solved one for the positive and one for the negative ions respectively. advective charge flux is given from the velocity field in the fluid that is obtatined by solving the Navier-Stokes (NS) equations (section 2.5). Forces due to present electric fields on net charged areas of the fluid also couples the NS equations to the NP equation. More about the force coupling is discussed in sections. 2.6 and 2.7. The coupling between the different equations are visualised in fig. 2.1.

2.3 The potential - Poisson's equation

To be able to model the flow dynamics of liquids in a channel with present EDLs, the potential and charge distribution in the channel must be determined. These quantities are mutually related through Poisson's equation for electrostatics:

$$\nabla^2\psi = -\frac{\rho_e}{\epsilon_r\epsilon_0} \tag{2.1}$$

where ψ is the electrical potential, ρ_e the electrical charge density, ϵ_r is the relative permittivity and ϵ_0 the vacuum permittivity. Under certain assumptions, the charge density may be explicitly determined as a function of the potential distribution, one such result is the so called Poisson-Boltzmann equation, further discussed in section 2.4.2.

2.3.1 Boundary conditions

At the charged boundaries, most physical situations may be covered by either specifying the potential or the surface charge density. The former would be a boundary condition of Dirichlet type:

$$\psi(\mathbf{x}) = \zeta(\mathbf{x}) \ , \ \mathbf{x} \in \Gamma \tag{2.2}$$

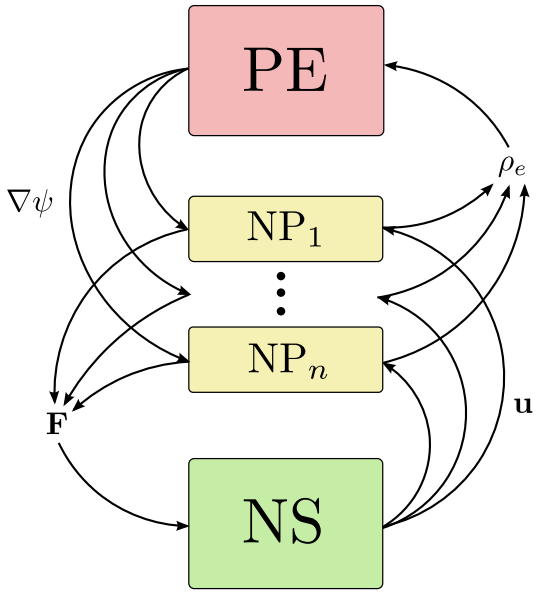


Figure 2.1: Visualisation of the coupling between the three equations present in the model. Poisson's equation (PE), the set of Nernst-Planck equations (NP_1 ... NP_n) for the different ion species and the Navier-Stokes equations (NS). The dependencies have also be marked with arrows indicating what quantities for a certain equation that are needed from an other.

and the latter a boundary condition of Neumann type:

$$\nabla\psi(\mathbf{x}) \cdot \mathbf{n} = -\frac{\sigma(\mathbf{x})}{\epsilon_0 \epsilon_r} \ , \ \ \mathbf{x} \in \Gamma \tag{2.3}$$

where Γ denotes the boundary of the domain and $\mathbf{n}$ is the normal to the boundary surface. [2]

2.4 The transport of charges - Nernst-Planck equation

The charge concentration in an electrolyte is indeed affected by its environment. In the model proposed here, influences from: advection of the electrolyte, diffusion due to concentration gradients and effects from the electric field originating from charged objects placed at the border or in the flow is considered. Charge conservation without any external sources of the ion density, $\mathrm{c}(\mathbf{x}, t)$, gives:

$$\frac{\partial \mathrm{c}}{\partial t} + \nabla \cdot \mathbf{J} = 0 \tag{2.4}$$

where $\mathbf{J}(\mathbf{x}, \mathrm{t})$ is the net flux induced by the effects described above. Explicit expressions for the fluxes due to advection and diffusion respectively are

$$\mathbf{J}_{adv} = \mathrm{c}\mathbf{u} \tag{2.5}$$

and

$$\mathbf{J}_{dif} = -D\nabla \mathrm{c} \tag{2.6}$$

where $\mathbf{u}$ is the advective velocity and D is a diffusion coefficient. The ionic flux due to the presence of an electric potential, $\psi(\mathbf{x}, t)$, is given by the Nernst equation [1]:

$$\mathbf{J}_{ele} = -\frac{zq_eD}{k_BT}\mathrm{c}\nabla\psi \tag{2.7}$$

where z is the relative charge of the ion species, q_e is the fundamental charge, k_B is the Boltzmann constant and T is the temperature of the fluid.

Summing up the fluxes and putting them into eq. (2.4) gives

$$\frac{\partial \mathrm{c}}{\partial t} = \nabla \cdot \left[D\nabla \mathrm{c} - \mathrm{c}\mathbf{u} + \frac{zq_eD}{k_BT}\mathrm{c}\nabla\psi \right] \tag{2.8}$$

which is a known result often referred to as the Nernst-Planck equation. This is the equation for the transport of *one* species of ions, if several are present one NP equation for each species needs to be solved. The advective velocity, $\mathbf{u}$, and the potential gradient, $\nabla\psi$, are obtained from couplings to the Navier-Stokes and Poisson's equation respectively. More about the coupling between the equations is discussed in section 2.2.

2.4.1 Boundary conditions

Depending on the physical situation being modelled, different conditions may be imposed at the boundaries of the domain. Throughout this work, at hard boundaries (walls), the charge flux through the boundary is set to zero, i.e.:

$$\mathbf{J} \cdot \mathbf{n} = 0 \ , \ \ \mathbf{x} \in \Gamma \tag{2.9}$$

where $\mathbf{n}$ denotes the normal to the surface and Γ is the boundary of the domain.

2.4.2 Poisson-Boltzmann equation

Consider a system consisting of an electrolyte in contact with a (flat) charged wall. Under certain assumptions, it is possible to explicitly determine the charge density in eq. (2.8) as a function of the electric potential. E.g. if there is no advection present and if the system has reached a steady state, i.e. $\partial \mathrm{c}/\partial t = 0$ and $\mathbf{u} = \mathbf{0}$ we have:

$$D\nabla \mathrm{c} + \frac{zq_eD}{k_BT}\mathrm{c}\nabla\psi = \mathbf{J}_0 \tag{2.10}$$

where $\mathbf{J}_0$ is a constant flux. Due to the steady state assumption, what the equation above actually says is that the net flux of charge in the system is constant. Since no charges are wanted to flow through the wall boundary, the flux is set to zero on the

wall and since the flux is constant it will therefore be zero everywhere in the liquid, i.e. $\mathbf{J}_0 = 0$.

Considering only a one-dimensional situation with a position variable y varying in a direction out from the wall into the liquid, eq. (2.10) reads

$$\frac{1}{\mathrm{c}}\frac{d\mathrm{c}}{dy} + \frac{zq_e}{k_BT}\frac{d\psi}{dy} = 0. \tag{2.11}$$

The charge density is determined by solving eq. (2.11) for c, i.e. integrating the equation. In order to avoid introducing additional unknown quantities, the equation is integrated to far away from the wall where the potential from the EDL is assumed to have decreased to zero and where the concentrations, c^∞, of the electrolyte is known.

$$\int_y^\infty d\ln(\mathrm{c}(y')) = -\frac{zq_e}{k_BT}\int_y^\infty d\psi(y') \tag{2.12}$$

This gives an expression for $C(y)$:

$$\mathrm{c}(y) = \mathrm{c}^\infty \exp\left(-\frac{zq_e\psi(y)}{k_BT}\right). \tag{2.13}$$

In a general case, there may be several species of ions in the electrolyte, the net charge density, ρ_e, is then given by simply summing up the contributions from the different species:

$$\rho_e = q_e\sum_i z_i\mathrm{c}_i. \tag{2.14}$$

Summarising eqs. (2.1), (2.13) and (2.14) gives the Poisson-Boltzmann equation in one dimension

$$\frac{d^2\psi(y)}{dy^2} = -\frac{q_e}{\epsilon_r\epsilon_0}\sum_i z_i\mathrm{c}_i^\infty \exp\left(-\frac{z_iq_e\psi(y)}{k_BT}\right). \tag{2.15}$$

The Debye–Hückel approximation

Historically, the non-linear nature of eq. (2.15) complicated for those wanting to solve it. This was a major difficulty in the past when the computational power at hands were rather limited. A linearisation is therefore sometimes done, this linear version of the PB equation is often referred to as the Debye–Hückel approximation. The solution of the linearisation gives, something to compare with and is usually used when defining a characteristic length scale of the EDL.

For a 1:1 electrolyte solution with an equal amount of positive and negatively charged ions, eq. (2.15) reduces to

$$\frac{d^2\psi(x)}{dx^2} = \frac{2\mathrm{c}^\infty q_e z}{\epsilon_r\epsilon_0}\sinh\left(\frac{zq_e\psi(x)}{k_BT}\right). \tag{2.16}$$

and the linearised equation is

$$\frac{d^2\psi(x)}{dx^2} = \frac{2\mathrm{c}^\infty q_e^2 z^2}{\epsilon_r \epsilon_0 k_B T}\psi(x) = \kappa^2 \psi(x) \tag{2.17}$$

where κ^{-1} is the Debye length which is where the exponential solution has decayed to e^{-1} of the boundary value. This quantity gives therefore a measure for the characteristic thickness of the EDL.

2.5 The velocity field - Navier-Stokes equations

The Navier-Stokes equations are among the most fundamental corner stones of hydrodynamics. They describe the motion of a fluid under the influence of various internal and external forces.

For later convenience and for reference when it comes to deriving the Lattice-Boltzmann formulation of the NS equation, a brief sketch of a derivation will here be presented. A most general form of the Navier-Stokes equation follows from momentum conservation

$$\frac{\partial(\rho\mathbf{u})}{\partial t} + \nabla \cdot (\rho\mathbf{u} \otimes \mathbf{u}) + \mathbf{Q} = 0 \tag{2.18}$$

where, ρ is fluid density, $\mathbf{u}$ is velocity, $\otimes$ represents the outer product and $\mathbf{Q}$ is a momentum source term (force per volume). Expanding the time derivative and the divergence terms respectively gives

$$\mathbf{u}\left(\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho\mathbf{u})\right) + \rho\left(\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla\mathbf{u}\right) + \mathbf{Q} = 0. \tag{2.19}$$

To assure mass conservation (without sources) we have

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho\mathbf{u}) = 0 \tag{2.20}$$

and eq. (2.19) reduces to

$$\rho\left(\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla\mathbf{u}\right) + \mathbf{Q} = 0 \tag{2.21}$$

which together with eq. (2.20) is a general formulation of the Navier stokes equations.

The force term $\mathbf{Q}$, is determined by the physical properties of the fluid and from its environment. In this work, only incompressible ($\rho =$ constant) Newtonian fluids will be studied. The force contribution to $\mathbf{Q}$ involved in that case is limited to viscous forces, pressure gradients in the fluid and to external force fields. Putting this into eqs. (2.20) and (2.21) gives

$$\nabla \cdot \mathbf{u} = 0 \tag{2.22}$$

and

$$\rho\left(\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u}\cdot\nabla\mathbf{u}\right) = -\nabla \mathrm{P} + \mu\nabla^2\mathbf{u} + \mathbf{F} \tag{2.23}$$

where P is the pressure, μ the kinematic viscosity and $\mathbf{F}$ is the contributions from external forces.

2.5.1 Boundary conditions

At hard boundaries (walls), the boundary conditions to eqs. (2.22) and (2.23) are set on the velocity of either a Dirichlet or Neumann type. In most physical situations the Dirichlet condition is used which corresponds to that there is a friction between the fluid and the wall, usually full friction, i.e. when no relative movement between fluid and wall is present and the velocity at the wall boundary is set to zero, i.e.

$$\mathbf{u} = 0\ ,\ \ \mathbf{x} \in \Gamma \tag{2.24}$$

where Γ denotes the boundary. The Neumann type conditions are used for no-friction walls where the normal component of the derivative of the velocity is specified, usually to zero.

At wet boundaries, inlets and outlets, of the domain various boundary conditions may be set. For instance the pressure or the velocity could be fixed. In the case of a fixed pressure boundary, a flow direction must also be specified for completeness. [3]

2.6 Pressure-driven electrokinetic flow

As a charged fluid is driven by a pressure gradient, a movement of charges, i.e. an electrical current will be induced. Due to the charge flux, a potential gradient will build up along the flow direction. This potential is usually referred to as the streaming potential, $\phi(\mathbf{x})$, and its magnitude is determined from the induced current through Ohm's law

$$\mathbf{J} = -\sigma\nabla\phi \tag{2.25}$$

where σ is the conductivity of the fluid. in a perfectly conducting fluid there will be no potential differences. Also a complete neutral solution will carry no net current and also in this case there will be no potential differences.

Charges under the influence of an electric field will be affected by a force. Charges moving due to this force will, in a liquid, also pull liquid (uncharged) molecules with them. In a macroscopic limit, the force density affecting the charges in the liquid is assumed to affect the liquid as a whole. The volumetric force affecting the fluid from the presence of the streaming potential is then given by:

$$\mathbf{F} = -\rho_e\nabla\phi \tag{2.26}$$

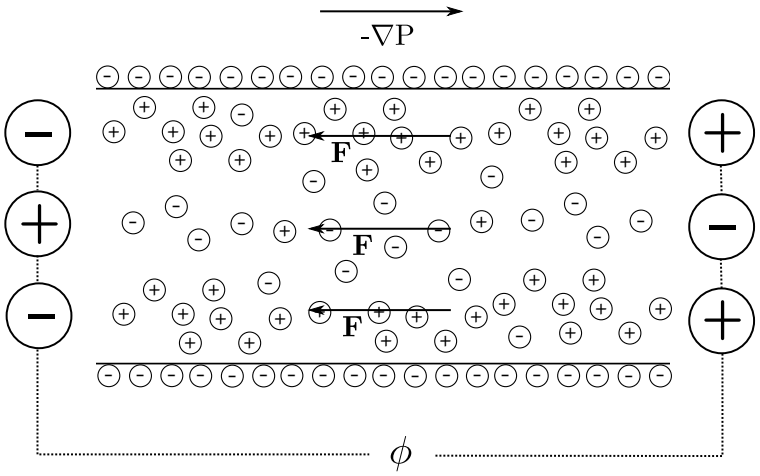


Figure 2.2: Example of an electroviscous system. The fluid is driven by a pressure gradient, ∇P. The directions of the forces on the fluid are always opposite to the flow direction. The force originates from the potential difference, ϕ, that builds up along the channel. The force is always opposite to the flow direction, thus slowing the flow down.

where ρ_e is the charge density. This is an example of how the charge density from the Nernst-Planck equation may couple to the force term in Navier-Stokes equations.

This force will always be affecting the fluid in a direction opposite to the net flux of charge, i.e. the force will slow the fluid down, this is illustrated in fig. 2.2. This effect that a moving net charged fluid is slowed down is called the *electroviscous effect*. The name originates from that a similar effect might be achieved by increasing the viscosity of the fluid.

2.7 Electroosmotic flow

Instead of driving the fluid flow through a pressure drop, a net charged fluid may be driven by an external electric field. This may be seen as the opposite case to that in section 2.6 where a current is induced by a pressure drop.

The volumetric force on the fluid from the external field, $\mathbf{E}_{ext}$, is given by

$$\mathbf{F} = \rho_e \mathbf{E}_{ext} \tag{2.27}$$

where ρ_e is the charge density. If the electric field is constant (or at least has the same direction) everywhere, the sign of the force is not in the same direction for a net charged positive area of the fluid as for a net charged negative. Thus the fluid may be either slowed down or sped up. This is a qualitative difference to pressure driven situation and is illustrated in fig. 2.3.

The electroviscous effect is in the case of pure electroosmotic flow usually neglected

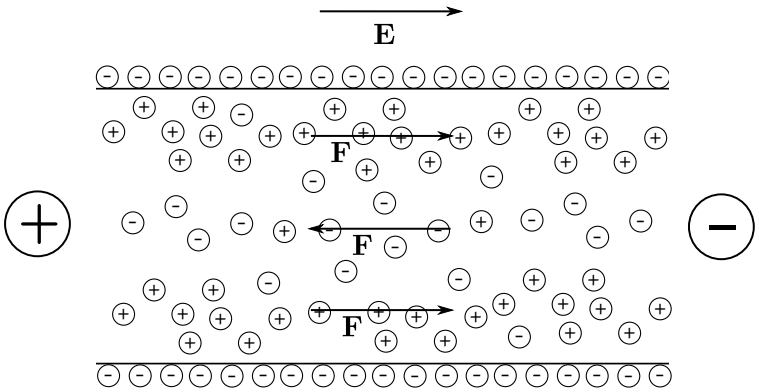


Figure 2.3: Example of an electroosmotic system. The fluid is driven by an external electric field, **E**. The directions of the forces on the fluid from the electric field are indicated with arrows. Note however that the fluid does not necessarily has to flow in the direction of the force, this due to viscous effects in the fluid.

as the field due to the streaming potential is, in most physical cases, small in comparison to the applied external field. [4]

3

The lattice-Boltzmann method

Rather than modelling on a macroscopic or microscopic scale, the lattice-Boltzmann method (LBM) operates at a scale in between those, often referred to as a mesoscopic scale. Nowadays, the method is most frequently used in modelling of fluid dynamics, i.e. computing solutions of the macroscopic Navier-Stokes equations. However, the lattice-Boltzmann method is not limited to this case and may be used to model other macroscopic systems as well. In this chapter a LBM approach will, in addition to the Navier-Stokes equations, also be formulated for the Nernst-Planck and Poission's equation.

3.1 Historical overview

With the introduction of electronic calculating machines came also completely new possibilities of tackling difficult problems. New fields of computational science was born and methods for solving both new and traditional problems were developed.

The idea of using a discrete and simplified version of the Boltzmann-equation dates back to the mid 60's [5] with an experimental attempt to model simple gas dynamics. However, at the time, this kind of statistical computational approaches was not considered a serious alternative for the modelling of more sophisticated and complex systems such as fluid behaviour. It was first in the mid 80's when Frisch, Hasslacher and Pomeau showed that a lattice automaton that conserved mass and momentum in the collisions and with a lattice of certain symmetry, reproduced the Navier-Stokes equations in a macroscopic limit. It was by their work and the always increasing computational power that made the idea of fluid modelling on a mesoscopic scale a serious research topic. [6]

The lattice gas automata (LGA) approach was not perfect and suffered from some notable flaws, e.g. that the boolean nature of the method introduced statistical noise and that lack of symmetry in the lattices used made the advection non-isotropic. The statistical noise was usually dealt with by averaging which resulted in a coarsed domain

and the advection issue was handled by introducing lattices of higher symmetry. An other consequence of the boolean variables is that only one particle per state was allowed which resulted in an equilibrium state from Fermi-Dirac statistics rather than the desired Maxwell-Boltzmann statistics.

As the flaws of the LGA approach was resolved one by another, the method evolved into what we today know as the lattice-Boltzmann method, with the crucial refinement of using continuous distributions over boolean variables. [6]

Today, the lattice-Boltzmann method is in many situations indeed a competitor to more traditional CFD methods. For example with advantages when it comes to parallelisation or implementing boundary conditions in complex geometries. One major downside with the LBM is the lack of theoretical work done and lack of literature compared to the case with more traditional methods such as finite element/volume methods. [7]

3.2 Statistical background

Consider one litre of air. At STP, the volume will contain in the order of 10^{22} molecules. In order to model this system microscopically, 6 variables per molecule will be needed to describe the microstate of the system. Just to store the state of the system in a computer would require more space than the estimated size of the whole world wide web times one million [8]. Thus, for these kind of systems, the microscopic approach is somewhat impractical.

Statistical approaches have been developed for these types of problems. A fundamental quantity used for describing the system is a continuous probability density distribution, f. This distribution may be regarded as an average over the microstates. Consider a volume of $d^3\mathbf{x}d^3\mathbf{p}$ in phase space, the number of molecules, dN in this volume is then given through the density distribution, f as

$$dN(\mathbf{x}, \mathbf{p}, t) = f(\mathbf{x}, \mathbf{p}, t)\mathrm{d}^3\mathbf{x}\mathrm{d}^3\mathbf{p}. \tag{3.1}$$

Thus $f(\mathbf{x}, \mathbf{p}, t)$ is a measure of the number of particles at location $\mathbf{x}$ with momentum $\mathbf{p}$ and at time t. Macroscopic variables are obtained by summing, e.g. the particle density, n, is obtained from

$$n(\mathbf{x}, t) = \int f(\mathbf{x}, \mathbf{p}, t)\mathrm{d}^3\mathbf{p} \tag{3.2}$$

and the macroscopic momentum density of the system is determined by

$$\rho(\mathbf{x}, t)\mathbf{u}(\mathbf{x}, t) = \int \mathbf{p} f(\mathbf{x}, \mathbf{p}, t)\mathrm{d}^3\mathbf{p} \tag{3.3}$$

where $\rho = mn$ is the mass density. Multiplying f by a power k of $\mathbf{p}$ or $\mathbf{v} = \mathbf{p}/m$ and integrating is often referred to as taking the k:th moment of f and is a term that will be used throughout this chapter.

In the late 19th century, Boltzmann developed a model for the time evolution of f. To do this he had to make several assumptions. First, only collisions between two particles are considered, this makes the equation mostly applicable to dilute gases. Second, the two particles colliding are assumed to be uncorrelated before the collision. Third, external forces are assumed not to affect the collisions [6]. The equation is named after its father to the Boltzmann (transport) equation and reads:

$$\partial_t f + \frac{\mathbf{p}}{m} \cdot \nabla_{\mathbf{x}} f + \frac{\mathbf{F}}{m} \cdot \nabla_{\mathbf{v}} f = Q(f, f) \tag{3.4}$$

where f is the distribution function for a single species collection of particles of mass m, $\mathbf{F}$ is external forces, $\mathbf{p}$ is momentum, $\nabla_{\mathbf{x}}$ and $\nabla_{\mathbf{v}}$ are the gradients in location and velocity space respectively. The right-hand side contains the so called collision term which in the general case is expressed as an integral. This integral states how the distribution function changes after a two particle collision. However, the structure of this integral is in most physical situations too complicated to be used directly. Therefore, a number of simplifications have been proposed during the years.

When designing these approximations, at least two main properties of the collision integral must be kept. [6]

1. The same quantities that are conserved under collisions in the collsion integral must also be conserved in the approximation.

2. Boltzmann's H-theorem must be fulfilled for the approximated collision operator.

Without being to specific, the H-theorem states that the entropy computed from f is always increasing with time and that the maximum entropy is obtained for a so called Maxwellian distribution in momentum/velocity space. Boltzmann used an other quantity denoted by H closely related to entropy, thus the name of the theorem. The Maxwellian distribution in two dimensions that f tends towards is given by

$$f^{(M)}(\mathbf{x}, \mathbf{v}, t) = n \left(\frac{m}{2\pi k_B T} \right) \exp \left(-\frac{m}{2k_B T} (\mathbf{v} - \mathbf{u})^2 \right) \tag{3.5}$$

where $\mathbf{u}(\mathbf{x}, t)$ is the mean velocity of the particles in the system and $n(\mathbf{x}, t)$ is the number of particles at location $\mathbf{x}$. In section 3.4, one of the most widely used approximations of the collision integral will be presented.

3.3 Basic idea of the LBM

As previously noted, the lattice-Boltzmann method is a mesoscopic method. This means that the modelling is neither done on a microscopic (molecular) level nor by direct solving of the macroscopic equations involved. The aim, in most situations with the lattice-Boltzmann method is indeed to solve some macroscopic equation but not direct. Instead a statistical model is used with various mesoscopic variables that, in some limit,

reproduces the macroscopic variables. It is also possible to ensure that these variables (to some extent) fulfil a certain macroscopic equation by using a certain scheme.

Basically the lattice-Boltzmann method solves a discretised version of eq. (3.4) for the distribution functions from which the macroscopic quantities may be determined. Both the spatial positions and the velocity space is discretised allowing the distributions to "sit" only at certain positions and to stream to neighbouring locations only in certain directions. A naive way to visualise the evolution of f is to consider the distribution functions at the lattice nodes as pseudo particles that move along the lattice and collide.

Usually in two dimensions the velocity space is discretised into 9 distinct velocities, more about the choice of lattice is discussed in section 3.5. In this case 9 distribution functions are needed per node which might correspond to one or two macroscopic variables but is indeed fewer than the number of variables needed for a microscopic approach.

The discretised Boltzmann equation is referred to as the lattice-Boltzmann equation (LBE) and is one of the fundamental corner stones in the lattice-Boltzmann method, it reads:

$$f_i(\mathbf{x} + \mathbf{c}_i\delta_t, t + \delta_t) - f_i(\mathbf{x}, t) = \Omega_{ij}(\mathbf{x}, t) \tag{3.6}$$

where f_i denotes the distribution function for direction $\mathbf{c}_i$, δ_t is the time step and $\Omega_i j$ is the (for now non-specified) collision operator. An implicit sum over the second velocity index j is assumed. Various forms of collision operators exist and will be further discussed in section 3.4.

3.3.1 Computational algorithm

In order to solve eq. (3.6) the distribution functions must be set to some initial value. The choice of initial value is in most cases crucial with respect to stability and accuracy of the method. More about the initialisation will be discussed in later sections of this chapter.

When a proper initialisation has been performed, the time evolution of the distribution functions is determined iteratively by the explicit scheme in the LBE, eq. (3.6). The update in each time step is usually divided into two computational tasks. First, the new value that later will be propagated to a neighbouring node is computed, i.e.

$$f_i^*(\mathbf{x}, t + \delta_t) = f_i(\mathbf{x}, t) + \Omega_{ij}(\mathbf{x}, t) \tag{3.7}$$

This step will be referred to as the collision step since it is here the "collision" is computed. The second step consists of propagating the distribution functions to the neighbouring node in its corresponding direction, i.e.

$$f_i(\mathbf{x} + \mathbf{c}_i\delta_t, t + \delta_t) = f_i^*(\mathbf{x}, t + \delta_t) \tag{3.8}$$

This step will be referred to as the streaming step.

In the case of a finite domain, certain rules (boundary conditions) must be specified at the boundaries. Typically the distribution functions that are going to be streamed

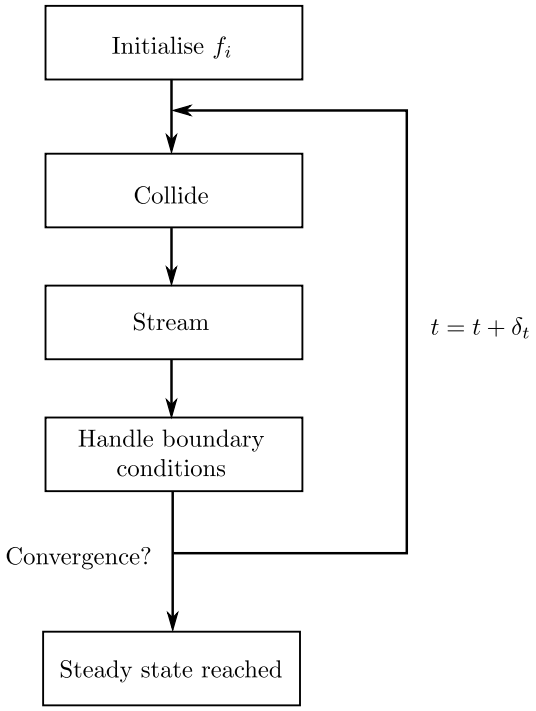


Figure 3.1: Flowchart of the most fundamental parts in an implementation of the LBM. The convergence is usually tested for a macroscopic variable.

out of the domain is used to define the unknown ones that will "enter" the domain. More about boundary conditions in section 3.11. Thus, at each time step, the boundary conditions must also be handled.

This is broadly the whole computational algorithm behind the LBM, in fig. 3.1, a flow scheme of the algorithm is shown.

3.4 The BGK collision operator

The collision term in the LBE is the main ingredient in what determines the physics of the system that is being modelled. Here the desired interaction of the pseudo particles is stated. In section 3.2, two necessary properties to approximations of the full collision integral was stated.

One of the simplest collision operators that fulfil conditions (1) and (2) in section 3.2 is the BGK operator (BGK from its creators: Bhatnagar, Gross and Krook). It was proposed in 1954 and is today one of the most commonly used collision operators both in the case of the lattice-Boltzmann and the continuous Boltzmann equation. It is based on the principle of relaxing f towards a Maxwellian distribution. The relaxation is also performed in such a way that the collision invariants are preserved. In the discrete case,

eq. (3.6), the operator is given by:

$$\Omega_{ij} = \Omega_i = -\omega \left[f_i(\mathbf{x}, t) - f_i^{(eq)}(\mathbf{x}, t) \right] \tag{3.9}$$

where ω is a parameter determining the relaxation rate and $f_i^{(eq)}$ should be an equilibrium distribution that makes sure that the necessary conditions are fulfilled. In the discrete case, a truncated expansion of eq. (3.5) is typically used [6]. This gives for instance

$$f_i^{(eq)} = w_i \rho \left[1 + \frac{\mathbf{c}_i \cdot \mathbf{u}}{c_s^2} + \frac{(\mathbf{c}_i \cdot \mathbf{u})^2}{2c_s^4} - \frac{\mathbf{u}^2}{2c_s^2} \right] \tag{3.10}$$

where w_i is a lattice specific weight, ρ is the zeroth moment of f_i and $\mathbf{c}_i$ is a unit velocity in the discretised velocity space.

The BGK operator is due to its simplicity both when it comes to theoretical treatment and implementation a popular choice. However in some physical situations, e.g. multi-phase or high Reynolds-number flows, more sophisticated alternatives are required [6]. Throughout this work, the BGK operator will be used.

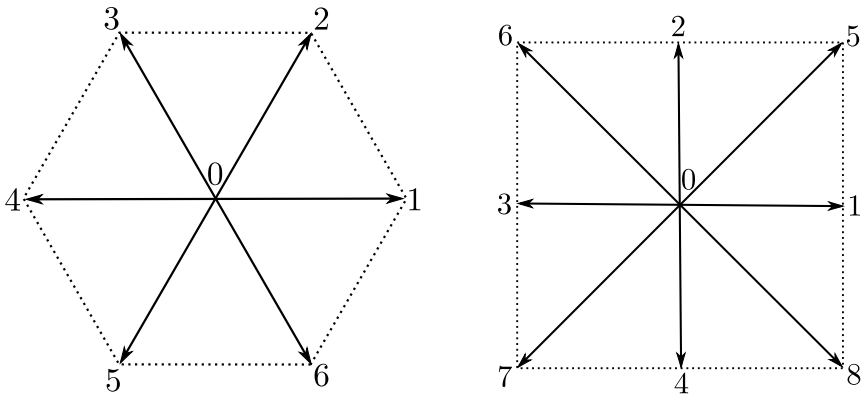
3.5 The lattice

The discrete spatial coordinates together with the discrete velocity coordinates forms a lattice. Spatial coordinates are neighbourwise connect through the discrete velocities. The discretisation in the LBM must be performed in such a way that the lattice may be produced from translating a single unit cell. This gives that the spatial resolution is constant for the whole domain. In other approaches, e.g. finite element methods the mesh may locally be refined at interesting regions of the domain which is a strength to these methods over the LBM.

There exist a convention for naming different lattices. A lattice of dimension d and with q distinct velocities is denoted by DdQq. For example a lattice with 9 velocities in dimension 2 is denoted D2Q9. An example of two different lattices is presented in fig. 3.2. The D2Q9 lattice is also the one that will be used in this work, the velocities $\mathbf{c}_i$ are given by

$$\{\mathbf{c}_i\} = \left\{ \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \begin{bmatrix} -1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ -1 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \begin{bmatrix} -1 \\ 1 \end{bmatrix}, \begin{bmatrix} -1 \\ -1 \end{bmatrix}, \begin{bmatrix} 1 \\ -1 \end{bmatrix} \right\} \tag{3.11}$$

To be able to retrieve the desired equations in the macroscopic limit of the LBE, the lattice used must possess a certain degree of isotropy. Thus the choice of lattice is not arbitrary. For instance, for obtaining the Navier-Stokes equations, the lattice velocities must at least posses the following properties:



(a) D2Q7 **(b)** D2Q9

Figure 3.2: Two different unit cells for lattices used in the LBM in two dimensions. In (a) the D2Q7 seven speed lattice is shown and in (b) the nine speed D2Q9 lattice. The numbering at the edges is the usual naming convention for the different velocities.

$$\sum_i w_i = 1 \tag{3.12a}$$

$$\sum_i w_i c_{i\alpha} = 0 \tag{3.12b}$$

$$\sum_i w_i c_{i\alpha} c_{i\beta} = c_s^2 \delta_{\alpha\beta} \tag{3.12c}$$

$$\sum_i w_i c_{i\alpha} c_{i\beta} c_{i\gamma} = 0 \tag{3.12d}$$

$$\sum_i w_i c_{i\alpha} c_{i\beta} c_{i\gamma} c_{i\delta} = c_s^4 (\delta_{\alpha\beta}\delta_{\gamma\delta} + \delta_{\alpha\gamma}\delta_{\beta\delta} + \delta_{\alpha\delta}\delta_{\beta\gamma}) \tag{3.12e}$$

where the sums are over the discrete velocities $\mathbf{c}_i$, w_i are lattice specific weights c_s is the speed of sound for the lattice and δ_{ij} is a Kroenecker delta.

The weights are introduced to compensate for the fact that different velocity vectors are of different length. See for example the D2Q9 lattice in fig. 3.2b where three lengths are present. In this case, three different weights will be needed. From the relations in eqs. (3.12) follow that these weights are:

$$w_i = \begin{cases} 4/9 & \text{if } i = 0 \\ 1/9 & \text{if } i = 1, 2, 3, 4 \\ 1/36 & \text{if } i = 5, 6, 7, 8 \end{cases} \tag{3.13}$$

The quantity c_s which often is referred to as the speed of sound may be thought of as an effective propagation velocity of the lattice and is determined, for the D2Q9 lattice, to

$$c_s = c/\sqrt{3} \tag{3.14}$$

where $c = |\mathbf{c}_{1,2,3,4}| = \delta_x/\delta_t$.

3.6 Asymptotic analysis

Methods from asymptotic analysis will, in this section, be used to investigate the macroscopic limit of the general LBE. More detailed and specific analyses for the three different equations considered will be presented in sections 3.7.1, 3.8.1 and 3.9.1 respectively.

Asymptotic analysis is basically about describing mathematical objects in some limit, e.g. how a function behaves for large or small values of some variable or parameter. Consider for example the series S_ϵ:

$$S_\epsilon = a^{(0)} + a^{(1)}\epsilon + a^{(2)}\epsilon^2 + a^{(3)}\epsilon^3 + \mathcal{O}(\epsilon^4) \tag{3.15}$$

It is clear that for sufficiently small values of ϵ, the terms of higher order is of negligible magnitude to those of lower order and the series may be truncated at some point and still be a good approximation of S_ϵ. For example if ϵ is small, then $S_\epsilon \approx a^{(0)}$ and we say that if $a^{(1)} \neq 0$ that this approximation is of first order accuracy.

There are different approaches to go from the discrete LBE to a continuous macroscopic equation. The most frequently applied one to obtain the Navier-Stokes equations is the Chapman-Enskog method [9], which will reproduce the compressible equations. Another method, often employed by M. Junk and his associates, e.g. in [7], is a method based on regular asymptotic expansions, this is also the method that will be utilised in this work and will in the case of Navier-Stokes reproduce the incompressible equations. A brief discussion of the differences between the Chapman-Enskog and the regular expansion approaches will be carried out at the end of this chapter.

The basic idea behind the analysis is to expand the distribution function f_i in some small parameter, ϵ. Also this parameter will be related to the spatial and time scales. The macroscopic limit is obtained by taking the Taylor expansion of the discrete LBE and comparing terms of equal order in ϵ. Together with the fact that certain quantities are invariant under collisions, macroscopic differential equations are obtained. Now follows the part of the analysis which is common for the three equations, the more equation specific analysis is carried out in sections 3.7.1, 3.8.1 and 3.9.1 respectively.

3.6.1 Motivation of the choice of expansion parameter

A most desired property of the expansion parameter is that it should be a small and dimensionless number. If the lattice is dense enough with respect to the characteristic length scale of the system, a suitable choice is the Knudsen number, ϵ, which is defined as the ratio of the mean free path, δ_x, and the characteristic length of the system under consideration, ℓ_0, i.e. $\epsilon = \delta_x/\ell_o$. To be able to perform the asymptotic analysis we must also relate the time scale to this parameter. From the fact that the lattice speed $c = \delta_x/\delta_t$ and by introducing a characteristic speed, $u_o = \ell_0/t_0$, we have

$$\epsilon = \frac{\delta_x}{\ell_0} = \frac{c}{u_0}\frac{\delta_t}{t_0} \tag{3.16}$$

It is now clear that what determines the relation between the timescale and the parameter ϵ is the ratio of the characteristic speed and the lattice speed which is usually referred to as the Mach number, Ma. In our particular case we will operate in the incompressible limit, i.e. $\mathrm{Ma} \ll 1$ and a suitable choice is a small number, thus $\mathrm{Ma} = \epsilon$ is chosen [9]. The discretisation of the space and time step is then related through

$$\delta_x'^2 = \delta_t' = \epsilon^2 \tag{3.17}$$

where the primes denote dimensionless variables. This particular scaling is usually referred to as diffusive scaling.

3.6.2 Expanding the LBE

The LBE, eq. (3.6), with dimensionless variables and the BGK collision operator reads:

$$f_i(\mathbf{x}' + \epsilon\mathbf{c}_i', t' + \epsilon^2) - f_i(\mathbf{x}', t') = -\omega\left[f_i(\mathbf{x}', t') - f_i^{(eq)}(\mathbf{x}', t')\right]. \tag{3.18}$$

The primes denoting dimensionless variables will, for readability reasons, from hereon be dropped. If nothing else is stated we always consider dimensionless variables.

To obtain a differential equation, the difference equation in eq. (3.18) is Taylor expanded, which gives

$$\epsilon(\mathbf{c}_i\cdot\nabla f_i)+\epsilon^2(\partial_t f_i+(\mathbf{c}_i\cdot\nabla f_i)^2/2)+\epsilon^3(\partial_t(\mathbf{c}_i\cdot\nabla f_i)+(\mathbf{c}_i\cdot\nabla f_i)^3/6)+\mathcal{O}(\epsilon^4) = -\omega\left[f_i - f_i^{(eq)}\right] \tag{3.19}$$

Expanding also f_i and $f_i^{(eq)}$ in the parameter ϵ:

$$f_i = f_i^{(0)} + \epsilon f_i^{(1)} + \epsilon^2 f_i^{(2)} + \epsilon^3 f_i^{(3)} + \mathcal{O}(\epsilon^4) \tag{3.20}$$

$$f_i^{(eq)} = f_i^{(eq,0)} + \epsilon f_i^{(eq,1)} + \epsilon^2 f_i^{(eq,2)} + \epsilon^3 f_i^{(eq,3)} + \mathcal{O}(\epsilon^4) \tag{3.21}$$

and inserting these expressions into eq. (3.19) gives an equation with terms of varying orders of ϵ. Separating this equation in equations of common orders allows for an analysis of what happens at different scales of ϵ. For the four leading orders in ϵ we have:

$$\epsilon^0: \quad 0 = -\omega\left[f_i^{(0)} - f_i^{(eq,0)}\right], \tag{3.22}$$

$$\epsilon^1: \quad \mathbf{c}_i\cdot\nabla f_i^{(0)} = -\omega\left[f_i^{(1)} - f_i^{(eq,1)}\right], \tag{3.23}$$

$$\epsilon^2: \quad \mathbf{c}_i\cdot\nabla f_i^{(1)} + \partial_t f_i^{(0)} + (\mathbf{c}_i\cdot\nabla f_i^{(0)})^2/2 = -\omega\left[f_i^{(2)} - f_i^{(eq,2)}\right] \tag{3.24}$$

and

$$\epsilon^3: \quad \mathbf{c}_i\cdot\nabla f_i^{(2)}+\partial_t f_i^{(1)}+(\mathbf{c}_i\cdot\nabla f_i^{(1)})^2/2+\partial_t(\mathbf{c}_i\cdot\nabla f_i^{(0)})+(\mathbf{c}_i\cdot\nabla f_i^{(0)})^3/6=-\omega\left[f_i^{(3)}-f_i^{(eq,3)}\right]. \tag{3.25}$$

The idea is now that for an equation of a particular order in ϵ, use collision invariants and eliminate unknown $f_i^{(n)}$ by using equations of lower order in ϵ. This will in the end, result in differential equations for macroscopic variables, given by moments of the f_i:s.

3.7 LBM for the Nernst-Planck equation

The method presented here is based on representing the Nernst-Planck equation, eq. (2.8), as an equation of advection-diffusion type. Considering the quantity:

$$\bar{\mathbf{u}}=\mathbf{u}-\frac{zq_eD}{k_BT}\nabla\psi \tag{3.26}$$

as an effective advective velocity, we have:

$$\frac{\partial\rho}{\partial t}+\nabla\cdot(\bar{\mathbf{u}}\rho-D\nabla\rho)=0 \tag{3.27}$$

which is a mass conservation equation with fluxes from diffusion and from advection respectively. The letter c for denoting the charge concentration has in this section been replaced by the letter ρ to avoid the risk of confusing it with the lattice velocities which traditionally are denoted by $\mathbf{c}_i$.

A collision operator of BGK type, eq. (3.9) will be used together with a D2Q9 lattice. The lattice-Boltzmann equation then reads:

$$f_i(\mathbf{x}+\mathbf{c}_i\delta_t,t+\delta_t)-f_i(\mathbf{x},t)=-\omega\left[f_i(\mathbf{x},t)-f_i^{(eq)}(\mathbf{x},t)\right] \tag{3.28}$$

with $\{\mathbf{c}_i\}_{i=0}^{Q-1}$ for the D2Q9 lattice as in eq. (3.11). The equilibrium function, $f_i^{(eq)}$, is chosen as [10]:

$$f_i^{(eq)}=w_i\rho\left(1+\frac{\mathbf{c}_i\cdot\bar{\mathbf{u}}}{c_s^2}\right) \tag{3.29}$$

with the weights, w_i, as in eq. (3.13). The charge density and charge flux density is obtained by taking the zeroth and first moments of the distribution function respectively, i.e:

$$\rho=\sum_i f_i \tag{3.30}$$

and

$$\mathbf{j}=\sum_i f_i\mathbf{c}_i \tag{3.31}$$

The diffusion constant, D, is related to the relaxation parameter ω through

$$D = c_s^2 \left(\frac{1}{\omega} - \frac{1}{2} \right). \tag{3.32}$$

3.7.1 Asymptotic analysis

To motivate the appearance of the above suggested method for solving eq. (3.27) and for showing under what premises the method is valid, the macroscopic limit of the discrete scheme will now be analysed in an asymptotic manner. Note that for the advection diffusion equation mass is but flux is not conserved.

From the expansion of f_i in eq. (3.20) and from eqs. (3.47) and (3.48) follow the expansions of the mass density and flux respectively as

$$\rho = \rho^{(0)} + \epsilon\rho^{(1)} + \epsilon^2\rho^{(2)} + \epsilon^3\rho^{(3)} + \mathcal{O}(\epsilon^4) \tag{3.33}$$

and

$$\mathbf{j} = \mathbf{j}^{(0)} + \epsilon\mathbf{j}^{(1)} + \epsilon^2\mathbf{j}^{(2)} + \epsilon^3\mathbf{j}^{(3)} + \mathcal{O}(\epsilon^4) \tag{3.34}$$

The advective velocity is also expanded as:

$$\bar{\mathbf{u}} = \bar{\mathbf{u}}^{(0)} + \epsilon\bar{\mathbf{u}}^{(1)} + \epsilon^2\bar{\mathbf{u}}^{(2)} + \epsilon^3\bar{\mathbf{u}}^{(3)} + \mathcal{O}(\epsilon^4) \tag{3.35}$$

By plugging these expansion into the equilibrium distribution eq. (3.29), the expansion in eq. (3.21) is obtained. The terms of order zero is used in the zeroth order equation of the LBE, eq. (3.22), which gives

$$f_i^{(0)} = w_i\rho^{(0)} \left(1 + \frac{\mathbf{c}_i \cdot \bar{\mathbf{u}}^{(0)}}{c_s^2} \right). \tag{3.36}$$

However, since we are only considering advection in the low Mach limit, i.e. $|\bar{\mathbf{u}}| \sim \epsilon$, we will in this analysis assume that $\bar{\mathbf{u}}^{(0)} = 0$. $\bar{\mathbf{u}}$ will then be of order ϵ to leading order. It is possible to show [7] that this assumption holds if $\bar{\mathbf{u}}$ is initialised properly, i.e. small and if no major momentum sources are present. Thus the expression for $f_i^{(0)}$ reduces to

$$f_i^{(0)} = w_i\rho^{(0)}. \tag{3.37}$$

We now continue to the equation of order one in ϵ, eq. (3.23). Taking the zeroth moment gives the equation $0 = 0$ which indeed is true but not very useful. Note that the right hand side vanishes due to mass conservation. $f_i^{(1)}$ will be needed in the next step and is, by using eq. (3.37):

$$f_i^{(1)} = -\frac{1}{\omega}(\mathbf{c}_i \cdot \nabla)(w_i\rho^{(0)}) + w_i \left(\rho^{(1)} + \rho^{(0)} \frac{\mathbf{c}_i \cdot \bar{\mathbf{u}}^{(1)}}{c_s^2} \right). \tag{3.38}$$

Taking the first moment of $f_i^{(1)}$ gives the leading order in the flux ($\mathbf{j}^{(0)} = 0$ since $\bar{\mathbf{u}}^{(0)} = 0$):

$$\mathbf{j}^{(1)} = \rho^{(0)}\bar{\mathbf{u}}^{(1)} - c_s^2/\omega\nabla\rho^{(0)} \tag{3.39}$$

Continuing to the equation of order two in ϵ, eq. (3.24) and taking the zeroth moment of the equation gives

$$\nabla \cdot \mathbf{j}^{(1)} + \partial_t\rho^{(0)} + c_s^2/2\nabla^2\rho^{(0)} = 0 \tag{3.40}$$

and by inserting the expression for $\mathbf{j}^{(1)}$ we end up with

$$\partial_t\rho^{(0)} + \nabla \cdot \left[\rho^{(0)}\bar{\mathbf{u}}^{(1)} - c_s^2\left(\frac{1}{\omega} - \frac{1}{2}\right)\nabla\rho^{(0)}\right] = 0 \tag{3.41}$$

which is an advection diffusion equation with a diffusion constant as in eq. (3.32). Since $\rho^{(0)}$ fulfils the equation of interest, the solution ρ that we get from the lattice-Boltzmann method is at least of first order accuracy. To determine the exact order of accuracy, higher order terms, $\rho^{(k>0)}$, must be determined. If also all those terms would be zero the method would be exact, unfortunately that is not the case. The analysis of higher order terms will not be performed here, but it is possible to show that $\rho^{(1)}$ is zero only under certain premises, i.e. for a proper initialisation [10]. $\rho^{(2)}$ is however in general non-zero and the obtained solution is thus second order accurate.

3.8 LBM for the incompressible Navier-Stokes

The most frequent use of the LBM is to solve the Navier-Stokes equations. In this work only the incompressible case, eqs. (2.22) and (2.23) will be considered. The LBM may however be used to reproduce weak compressibility [6]. The incorporation of forces will be treated in section 3.8.2. We recall the incompressible Navier-Stokes equations without external forces present from chapter 2 as:

$$\nabla \cdot \mathbf{u} = 0 \tag{3.42}$$

and

$$\rho\left(\frac{\partial\mathbf{u}}{\partial t} + \mathbf{u}\cdot\nabla\mathbf{u}\right) = -\nabla\mathrm{P} + \rho\nu\nabla^2\mathbf{u} \tag{3.43}$$

where ν is the kinematic viscosity related through the dynamic viscosity, μ, by

$$\mu = \rho\nu. \tag{3.44}$$

A LBM will now be formulated for eqs. (3.42) and (3.43). The LBE with a BGK collision operator is given by

$$f_i(\mathbf{x} + \mathbf{c}_i\delta_t, t + \delta_t) - f_i(\mathbf{x}, t) = -\omega\left[f_i(\mathbf{x}, t) - f_i^{(eq)}(\mathbf{x}, t)\right] \tag{3.45}$$

where

$$f_i^{(eq)} = w_i \rho \left[1 + \frac{\mathbf{c}_i \cdot \mathbf{u}}{c_s^2} + \frac{(\mathbf{c}_i \cdot \mathbf{u})^2}{2c_s^4} - \frac{\mathbf{u}^2}{2c_s^2} \right] \tag{3.46}$$

where w_i are the weights in eq. (3.13).

The density, ρ, and the mass flux, $\rho\mathbf{u}$, is determined from f_i by taking the zeroth and first moments respectively:

$$\rho = \sum_i f_i \tag{3.47}$$

and

$$\rho\mathbf{u} = \sum_i f_i \mathbf{c}_i. \tag{3.48}$$

The kinematic viscosity is related to the relaxation parameter, ω

$$\nu = c_s^2 \left(\frac{1}{\omega} - \frac{1}{2} \right). \tag{3.49}$$

3.8.1 Asymptotic analysis

Partially based on [7], an asymptotic analysis of the above suggested method will be performed. In most literature available, when reproducing the Navier-Stokes equations in the macroscopic limit, a Chapman-Enskog expansion is performed. Note that this is *not* what is done here.

From the expansion of f_i in eq. (3.20) follows the expansion of the macroscopic mass and velocity as

$$\rho = \rho^{(0)} + \epsilon\rho^{(1)} + \epsilon^2\rho^{(2)} + \epsilon^3\rho^{(3)} + \mathcal{O}(\epsilon^4) \tag{3.50}$$

and

$$\mathbf{u} = \mathbf{u}^{(0)} + \epsilon\mathbf{u}^{(1)} + \epsilon^2\mathbf{u}^{(2)} + \epsilon^3\mathbf{u}^{(3)} + \mathcal{O}(\epsilon^4). \tag{3.51}$$

These expansions are plugged into the equilibrium distribution in eq. (3.46) and from the equation of order zero in ϵ, eq. (3.22) gives:

$$f_i^{(0)} = w_i \rho^{(0)} \tag{3.52}$$

Here $\mathbf{u}^{(0)}$ has been assumed to be zero by the same argumentation as in the Nernst-Planck analysis, section 3.7.1. Continuing to the equation of order 1 in ϵ, eq. (3.23) and taking the first moment gives

$$\nabla\rho^{(0)} = 0. \tag{3.53}$$

Further, by using this fact that $\rho^{(0)}$ is constant in space and that

$$f_i^{(1)} = -\frac{1}{\omega}\mathbf{c}_i \cdot \nabla f_i^{(0)} + f_i^{(eq,1)} \tag{3.54}$$

where

$$f_i^{(eq,1)} = w_i \left[\rho^{(1)} + \rho^{(0)} \frac{\mathbf{c}_i \cdot \mathbf{u}^{(1)}}{c_s^2} \right] \tag{3.55}$$

gives when taking the zeroth moment of the equation of order two in ϵ, eq. (3.24)

$$\partial_t \rho^{(0)} + \nabla \cdot (\rho^{(0)} \mathbf{u}^{(1)}) = 0. \tag{3.56}$$

This equation states the conservation of mass for $\rho^{(0)}$. Since we are considering only systems in the incompressible limit, $\rho^{(0)}$ will be assumed to also be constant in time and we have

$$\nabla \cdot \mathbf{u}^{(1)} = 0. \tag{3.57}$$

Taking the first moment of eq. 3.24 gives

$$\nabla \rho^{(1)} = 0. \tag{3.58}$$

It can be showed [7] that if $\rho^{(1)}$ is initialised properly, then it does not change in time. Therefore, if $\rho^{(1)}$ is initialised to zero, it will also remain zero for all time.

As we now continue to the equation of order three in ϵ, things will get a bit more technical as we will make use of the fourth order lattice isotropy, eq. (3.12). The zeroth moment of the equation gives

$$\nabla \cdot \mathbf{u}^{(2)} = 0. \tag{3.59}$$

which may be used to show that $\mathbf{u}^{(2)} = 0$ [7], it will not be shown here. Instead we continue by taking the first moment of the equation, as it gets a bit more technical now, some intermediate steps in the calculation will explicitly be written out. First from the equation of order two in ϵ we have that

$$f_i^{(2)} = -\frac{1}{\omega} \mathbf{c}_i \cdot \nabla f_i^{(1)} - \frac{1}{\omega} \partial_t f_i^{(0)} - \frac{1}{2\omega} (\mathbf{c}_i \cdot \nabla f_i^{(0)})^2 + f_i^{(eq,2)} \tag{3.60}$$

where

$$f_i^{(eq,2)} = w_i \left[\rho^{(2)} + \rho^{(0)} \frac{(\mathbf{c}_i \cdot \mathbf{u}^{(1)})^2}{2c_s^4} - \rho^{(0)} \frac{\left(\mathbf{u}^{(1)}\right)^2}{2c_s^2} \right] \tag{3.61}$$

where the fact that $\mathbf{u}^{(2)} = 0$ have been used. Inserting the expression for $f_i^{(2)}$ into eq. (3.25) gives, in index notation

$$\begin{aligned} c_{i\alpha} c_{i\beta} \partial_\beta \left[-\frac{\rho^{(0)}}{\omega c_s^2} c_{i\gamma} \partial_\gamma c_{i\delta} u_\delta^{(1)} + w_i \rho^{(2)} + \frac{\rho^{(0)} w_i}{2c_s^4} \left(c_{i\gamma} c_{i\delta} u_\gamma^{(1)} u_\delta^{(1)} - (u^{(1)})^2 \right) \right] + \\ + \partial_t w_i \rho^{(0)} u_\alpha^{(1)} + \frac{\rho^{(0)}}{2c_s^2} c_{i\alpha} c_{i\beta} \partial_\beta c_{i\gamma} \partial_\gamma c_{i\delta} = 0 \end{aligned} \tag{3.62}$$

For brevity, the terms that in the end will equal zero have been omitted. The indices α, β, γ and δ may take the x or y component respectively. Summing eq. (3.62) over the directional index i and by using the isotropity properties of the lattice, eq. (3.12) gives:

$$\begin{aligned} c_s^2\rho^{(0)}\left(\frac{1}{2}-\frac{1}{\omega}\right)\partial_\beta\left(\delta_{\alpha\beta}\delta_{\gamma\delta}+\delta_{\alpha\gamma}\delta_{\beta\delta}+\delta_{\alpha\delta}\delta_{\beta\gamma}\right)+c_s^2\partial_\beta\delta_{\alpha\beta}\rho^{(2)}+ \\ +\frac{\rho^{(0)}}{2}\left((\delta_{\alpha\beta}\delta_{\gamma\delta}+\delta_{\alpha\gamma}\delta_{\beta\delta}+\delta_{\alpha\delta}\delta_{\beta\gamma})u_\gamma^{(1)}u_\delta^{(1)}-\delta_{\alpha\beta}(u^{(1)})^2\right)+\rho^{(0)}\partial_t u_\alpha^{(1)}=0 \end{aligned} \tag{3.63}$$

On the form it is written now, it is not totally clear whether the above equation is the incompressible Navier-Stokes momentum equation or not. A simplification is in order and for brevity only the simplification of one term is shown here.

By noting that $(u^{(1)})^2$ may be written as $\delta_{\gamma\delta}u_\gamma^{(1)}u_\delta^{(1)}$ we have

$$\begin{aligned} \partial_\beta\left[(\delta_{\alpha\beta}\delta_{\gamma\delta}+\delta_{\alpha\gamma}\delta_{\beta\delta}+\delta_{\alpha\delta}\delta_{\beta\gamma})u_\gamma^{(1)}u_\delta^{(1)}-\delta_{\alpha\beta}(u^{(1)})^2\right]= \\ =\partial_\beta\left[(\delta_{\alpha\beta}\delta_{\gamma\delta}+\delta_{\alpha\gamma}\delta_{\beta\delta})u_\gamma^{(1)}u_\delta^{(1)}\right] \end{aligned} \tag{3.64}$$

and by summing over γ and δ the above expression reduces to:

$$\partial_\beta\left[u_\alpha^{(1)}u_\beta^{(1)}+u_\beta^{(1)}u_\alpha^{(1)}\right]=2\partial_\beta u_\alpha^{(1)}u_\beta^{(1)} \tag{3.65}$$

and from incompressibility ($\partial_\beta u_\beta^{(1)}=0$), eq. (3.57) we end up with

$$2u_\beta^{(1)}\partial_\beta u_\alpha^{(1)}. \tag{3.66}$$

By analogous simplification of the remaining terms in eq. (3.63) we get

$$\rho^{(0)}\left(\partial_t u_\alpha^{(1)}+u_\beta^{(1)}\partial_\beta u_\alpha^{(1)}\right)=-\partial_\beta(c_s^2\rho^{(2)})+\rho^{(0)}\nu\partial_\beta\left(\partial_\beta u_\alpha^{(1)}+\partial_\alpha u_\beta^{(1)}\right) \tag{3.67}$$

where ν is defined as in eq. (3.49), and if the quantity $c_s^2\rho^{(2)}$ is interpreted as the pressure, the above equation is indeed the incompressible Navier-Stokes momentum equation, eq. (3.43). It is satisfied by $\mathbf{u}^{(1)}$ in the expansion of $\mathbf{u}$, since $\mathbf{u}^{(2)}=0$, the error will be of order ϵ^3 and we say that the LB formulation is of second order accuracy in velocity. $\rho^{(0)}$ satisfies the mass equation and since $\rho^{(1)}=0$ we say that the method is second order in density as well [7]. Since the pressure is included as a term in the density expansion, problems with keeping the incompressibility might arise if large enough pressures are present.

3.8.2 Forcing schemes

Several methods have been proposed to add the forcing term in eq. (2.23) to the LBE. In [11] five of these methods have been compared. To briefly summarise the article; for single phase flows, the methods achieve comparable.

Due to its simplicity, a method formulated by Shan and Chen [12] was used in this work. It consists of modifying the equilibrium distribution by computing it with the modified velocity

$$\mathbf{u}^{(eq)} = \frac{1}{\rho}\left(\sum_i f_i \mathbf{c}_i + \frac{\mathbf{F}\delta_t}{\omega}\right) \tag{3.68}$$

where $\mathbf{F}$ is the external force. The physical velocity, $\mathbf{u}$, is computed by

$$\mathbf{u} = \frac{1}{\rho}\left(\sum_i f_i \mathbf{c}_i + \frac{\mathbf{F}\delta_t}{2}\right). \tag{3.69}$$

3.9 LBM for Poisson's equation

As Poisson's equation, eq. (2.1), is considered, a fundamental difference to the Nernst-Planck and Navier-Stokes equations is imediatley noted. It is not a differential equation of parabolic but of elliptic type. If we consider what has been said about the LBM so far in this chapter, it seems that it is a method that deals with parabolic equations and not elliptic. However, by introducing a time derivative to the Poisson's equation and only considering the steady state solution the LBM may be used even for this equation. A diffusion-like equation with a source term is then obtained

$$\partial_t \rho + \nabla^2 \rho = \mathrm{R} \tag{3.70}$$

where R is the right-hand side of the Poisson's equation.

A LBM for this equation will now be formulated. The LBE to be solved in this case is, an equation of the form:

$$f_i(\mathbf{x} + \mathbf{c}_i\delta_t, t + \delta_t) - f_i(\mathbf{x}, t) = -\omega\left[f_i(\mathbf{x}, t) - f_i^{(eq)}(\mathbf{x}, t)\right] + \mathrm{G_i(R)} \tag{3.71}$$

where $\mathrm{G_i(R)}$ is an addition to the BGK collision operator to account for the source term R in eq. (3.70). The equilibrium distribution is given by

$$f_i^{(eq)} = w_i \rho \tag{3.72}$$

where w_i are the weights in eq. (3.13).

The quantity ρ which in this work is interpreted to electric potential in the steady state is determined by

$$\rho = \sum_i f_i \tag{3.73}$$

and the quantity which, to us, is of even more interest, i.e. the electric field, is given by

$$\nabla\rho = -\frac{\omega}{c_s^2\delta_x}\sum_i f_i \mathbf{c}_i. \tag{3.74}$$

Finally, the addtional source term in the collision operator, $\mathrm{G_i(R)}$, is given by

$$\mathrm{G_i(R)} = w_i c_s^2 \left(\frac{1}{2} - \frac{1}{\omega}\right) \mathrm{R}. \tag{3.75}$$

3.9.1 Asymptotic analysis

The given method for solving eq. (3.70) will now be motivated by preforming an asymptotic analysis of the suggested LBE.

f_i is expanded as in eq. (3.20) which gives an expansion of ρ as in eq. (3.33). Further, the source term R is expanded:

$$\mathrm{R} = \mathrm{R}^{(0)} + \epsilon \mathrm{R}^{(1)} + \epsilon^2 \mathrm{R}^{(2)} + \epsilon^3 \mathrm{R}^{(3)} + \mathcal{O}(\epsilon^4) \tag{3.76}$$

which also gives an expansion for the LBE source term as

$$\mathrm{G_i} = \mathrm{G_i}^{(0)} + \epsilon \mathrm{G_i}^{(1)} + \epsilon^2 \mathrm{G_i}^{(2)} + \epsilon^3 \mathrm{G_i}^{(3)} + \mathcal{O}(\epsilon^4). \tag{3.77}$$

The equilibrium distribution is expanded by plugging in the expansion of ρ from eq. (3.33). This gives for the zeroth order equation in ϵ, eq. (3.22) that

$$f_i^{(0)} = w_i \rho^{(0)} + \frac{1}{\omega} \mathrm{G_i}^{(0)} \tag{3.78}$$

However, taking the first moment of the equation gives $\mathrm{G_i}^{(0)} = 0$ and the above expression for $f_i^{(0)}$ reduces to

$$f_i^{(0)} = w_i \rho^{(0)}. \tag{3.79}$$

Coninuing to the equation of first order in ϵ, eq. (3.23) and taking the zeroth moment gives that also $\mathrm{G_i}^{(1)} = 0$. Taking the first moment of the same equation gives

$$\nabla \rho^{(0)} = -\frac{\omega}{c_s^2} \sum_i f_i^{(1)} \mathbf{c}_i \approx -\frac{\omega}{c_s^2 \delta_x} \sum_i f_i \mathbf{c}_i \tag{3.80}$$

Here, for the approximation, the expansion of f_i is used together with the fact that $\epsilon = \delta_x$ in dimensionless variables.

The next equation in ϵ, i.e. the one of order two, will eventually give us what we are looking for. Taking the zeroth moment gives

$$-\frac{c_s^2}{\omega} \nabla^2 \rho^{(0)} + \partial_t \rho^{(0)} + \frac{c_s^2}{2} \nabla^2 \rho^{(0)} = \sum_i \mathrm{G_i}^{(2)}(\mathrm{R}^{(2)}) \tag{3.81}$$

From this equation, it is deduced that in order to obtain Poisson's equation in a steady state situation, i.e. when $\partial_t \rho^{(0)} = 0$ the right-hand side must fulfill the follwong condition

$$\sum_i \mathrm{G_i}^{(2)}(\mathrm{R}^{(2)}) = c_s^2 \left(\frac{1}{2} - \frac{1}{\omega}\right) \mathrm{R}^{(2)} \tag{3.82}$$

One such $\mathrm{G_i}^{(2)}(R)$ is the one described in eq. (3.75).

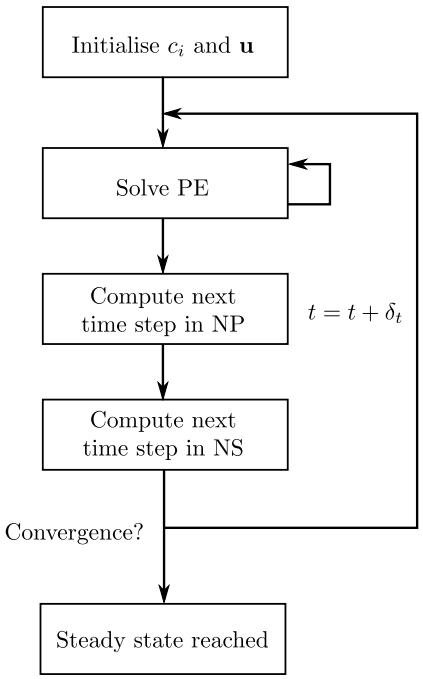


Figure 3.3: Flow scheme of the algorithm for solving the coupled equations.

3.10 Algorithm/Scheme for solving the coupled equations

In this section, the scheme to solve the coupled Poisson's, Nernst-Planck and Navier-Stokes equations is described. The dependencies between the equations are shown in fig. 2.1.

Following an initialisation of the velocity field and charge density, an iteration in time is performed. At each time step, a potential and electric field is computed by solving Poisson's equation for the present charge density. This is followed by updating the charge density from the new potential and then the velocity field is updated with the prescence of a force related to the charge density. The main ingredients in the iterative algorithm are shown in fig. 3.3.

3.11 Boundary conditions

In all physical situations, when solving a differential equations on a domain, conditions for what is happening on the boundary of the domain must be specified. So far, only the update rule for f_i on the interior of this domain has been treated. In this section we will also define rules for the boundaries of the domain. The nodes in the interior and the boundary will be referred to as *interior nodes* and *boundary nodes* respectively. Typically, a boundary condition in a macroscopic variable, e.g. velocity, is specified

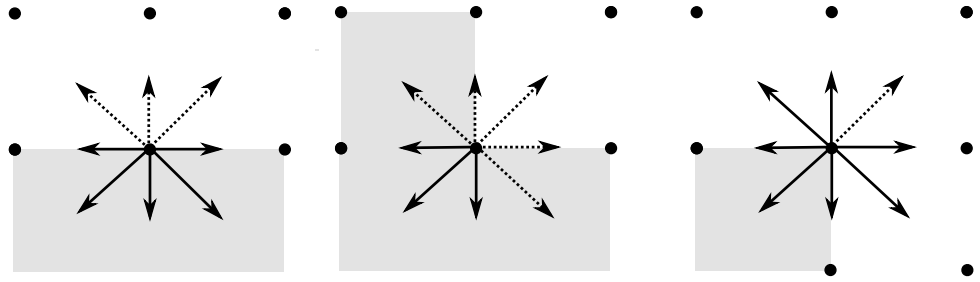


Figure 3.4: Three distinct boundary situations that in general has to be treated differently. From left to right, a straight boundary, a corner and an edge. Grey areas are outer of the domain. The solid arrows corresponds to known directions of the distribution function and dotted lines to unknown.

from the physical problem. This condition must be translated into a condition for the distribution function, f_i, on the statistical level. In this section some, to this work, useful boundary conditions will be formulated and discussed.

3.11.1 Bounce-back boundaries

The lattice-Boltzmann approach is often praised for its rather straight-forward easiness of implementing boundary conditions. One of the simplest and most commonly used is the bounce-back rule. It is a mesoscopic rule for implying a Dirichlet condition on the first moment. In the case of specifying boundary conditions for Navier-Stokes, it may be used to set a velocity at a boundary, typically at wall boundaries the velocity is set to zero.

The bounce-back rule for setting the first moment to zero reads

$$f_i^{(bb)} = f_{i^*} \tag{3.83}$$

where $\mathbf{c}_{i^*}$ is the direction opposite to $\mathbf{c}_i$. This justifies the name of the rule, all pseudo particles are propagated in a direction opposite to where they came from, i.e. bounced back. If the boundary nodes are updated between the collision and streaming step, all directions should be updated with its opposite counterpart. However, if the update is performed after the streaming step, only the unknown directions are to be updated, see fig. 3.4. These two schemes are referred to full-way and half-way bounce-back. If a non-zero Dirichlet condition is desired, it is possible to add some momentum to suitable directions.

It is possible to show that, in the case of a straight boundary, the actual boundary is with second order accuracy located half a node-node distance into the computational domain [9]. Thus, the boundary will not be located at the boundary node.

As suggested earlier in this section, this type of local boundary conditions allows for easy implementation of more or less arbitrary boundaries. In fig. 3.5, an example

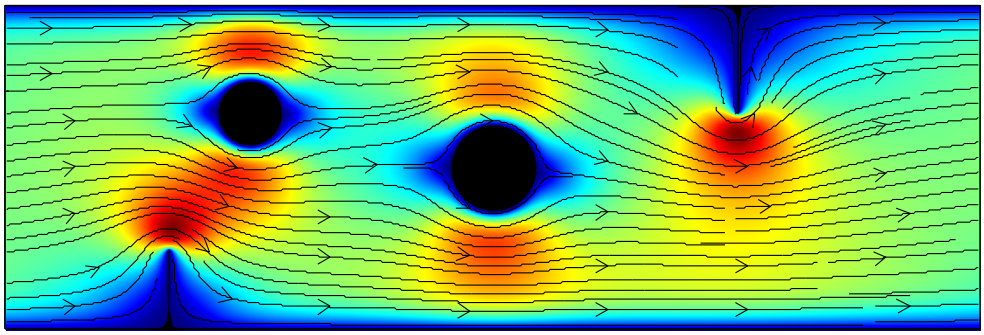


Figure 3.5: Example of flow in a non-trivial geometry where bounce-back boundaries are used.

of 2D flow in a complicated domain is shown. At the walls in the figure, bounce-back conditions are used.

3.11.2 Slip boundaries

A zero friction (slip) boundary condition may be imposed by the following rule for the straight boundary situation in fig. 3.4. Instead of making all pseudo particles "bounce back", they are reflected in a mirror-like manner, see fig. 3.6b. This gives that the component normal to the boundary of the first moment is zero while the tangential component is unchanged. In this work, this condition is used to set the ion flux through the boundary to zero, eq. (2.9).

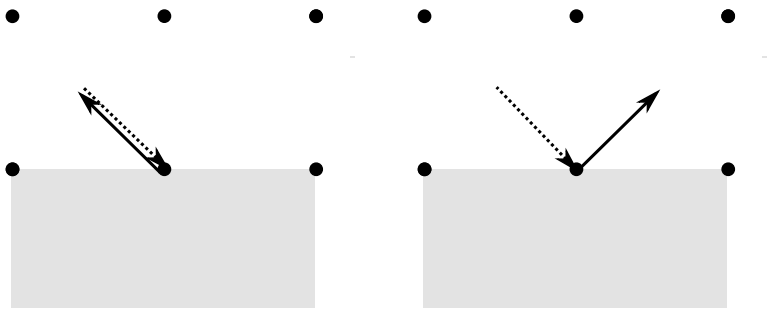
It is not as straight-forward to define rules at corner and edge nodes, as for the straight boundary nodes. From tests, it has been shown that adequate accuracy may be obtained by treating the corner or edge as two perpendicular straight boundaries and mirror reflect half of the distributions at each plane respectively.

With momentum addition

In the case with the fixed surface charge boundary condition, eq. (2.3), we do not wish to set the normal component to zero but to some value of the surface charge. This is realised by adding some "momentum" in the three directions pointing in to the domain. The total surface charge may be divided between the directions in different proportions, in this work it is however evenly distributed, i.e. one third per direction.

3.12 Physical and lattice units

A physical system may in its physical units contain quantities whose numerical values may differ a lot from each other. It is therefore, from a numerical point of view, a very bad idea and in the case with lattice-Boltzmann it will in general not work to just solve



(a) Bounce-back **(b)** Mirror reflection

Figure 3.6: Intuitive scheme for an incoming pseudo particle for two common boundary conditions in the LBM. The bounce-back rule (a) and mirror reflection (b).

the equation in physical units. The equation is therefore scaled to dimensionless form by introducing characteristic quantities. For example a quantity A is scaled by A_0 to a dimensionless quantity A' through

$$\mathrm{A}' = \mathrm{A}/\mathrm{A}_0 \tag{3.84}$$

Usually the dimensionless quantities are wanted to be of order one.

In the case with Navier-Stokes equation, three characteristic quantities are needed to put it on dimensionless form. I.e. one for the density (ρ_0), velocity (u_0) and length (ℓ_0). The non-dimensional incompressible N-S will contain one dimensionless parameter, i.e. the Reynolds number $\mathrm{Re} = \rho_0 u_0 \ell_0 / \mu$.

For the advection-diffusion equation, in addition to a characteristic length and velocity, a characteristic concentration (c_0) is needed. One dimensionless parameter arise when non-dimensionalising the equation, i.e. the Peclet number, $\mathrm{Pe} = u_0 \ell_0 / D$ where D is the diffusion coefficient.

Poisson's equation is non-dimensionalised by using a length and a characteristic voltage (V_0). The RHS then gets multiplied by factor ℓ_0^2/V_0 in the non-dimensional form.

When the non-dimensional form of the equations are determined, the domain is discretised. The non-dimensional length of the system is now typically 1 and $\delta_x = 1/N$ where N is the number of lattice cells used. From the diffusive scaling used, $\delta_t = \delta_x^2$ is usually a good choice.

3.13 Chapman-Enskog vs. regular expansion analysis

When reading literature and articles about the LBM, different approaches may be used to derive the macroscopic behaviour of the LBM. The aim of this section is to bring some clarity and briefly explain the main differences between two of these.

The method used in this work is referred to as regular (error) expansion analysis [13]

and is used with one time scale. In the case of analysing the macroscopic behaviour of the LBM for the Navier-Stokes equations, the incompressible equations are obtained. The mass and momentum equations are exactly satisfied by $\rho^{(0)}$ and $\mathbf{u}^{(1)}$ from the regular expansions of ρ and $\mathbf{u}$ respectively.

An other approach is by using the so called Chapman-Enskog analysis [6]. This is a traditional method in kinetic theory and is the most frequently used in litterature about the LBM. Here two time scales are used, on faster (convective) and one slower (diffusive). This gives that in the macroscopic limit of the LBE, the compressible Navier-Stokes equations are obtained. In the C-E analysis, the full quantities ρ and $\mathbf{u}$ are showed to satisfy, not the exact, but the N-S equations with higher order error terms.

4

High performance computing and the LBM

To benefit as much as possible from a numerical method, it is crucial that it is implemented in an efficient manner. In this chapter, some important aspects of this is discussed. In order to write a code that allows for efficient execution, some principles behind how a computer operates must be considered in detail. This varies considerable between various computer architectures. In this chapter, the discussion is limited to computers that implement the x86(-64) architecture which is a very common alternative among today's workstations.

4.1 The pipeline

To execute an instruction, several parts of the hardware is typically involved. The basic idea of pipelining is to allow these different parts of hardware to operate simultaneously. For example at the same time as an instruction is executed the instruction that will be executed in the next clock cycle may be decoded. In principle all modern computer architectures are pipelined.

The x86 architecture is a CISC (Complex instruction set computing) arcitecture which means that it consists of a rather large number of specific instructions. The opposite case is a RISC (Reduced instruction set computing) architecture that only has a very few but general instructions. Due to the pipelining, an important property of RISC is that the instructions execute in one clock cycle which is a great performance benefit of the RISC architectures. The modern CISC architectures is however said to be RISC-like and has also the ability of pipelining. Since The CISC pipelining is rather complicated, e.g. the Intel Xeon processor in the computer that this text is written on has a 20 stage pipeline [14]. The principle is however the same as in the RISC case, therefore a brief explanation of the classical RISC pipeline is given here. The RISC

pipeline consists of the following 5 stages:

1. Fetch of instruction
2. Decoding of fetched instruction
3. Execution of instruction
4. Memory access
5. Write result to register

Each of these tasks takes one clock cycle to perform. Thus if a single instruction, e.g. an addition is to be computed, it will take 5 clock cycles. This is the start-up time of the pipeline and when it is filled, one instruction per clock cycle is executed.

In order to achieve as many instructions computed per time as possible, the objective must be to keep the pipeline filled. However, situations may arise when this seems difficult. For instance when there are dependencies between stages, e.g. that the output from execution E1 is the input to execution E2. This means that E2 may have to wait for E1 to finish and the pipeline is stalled for a number of clock cycles. An other situation is the following branch:

```
...
if(a == 3)
  a = 2
else
  a++
end if
...
```

Here as the instruction for the if statement has been fetched, it is not clear what instruction will be fetched in the next clock cycle. The if statement must first be evaluated before such a decision can be made. Thus the pipeline is stalled also in this case. From this example we conclude that having branch statements in places where they are often executed, e.g. in loops, is a performance loss and should be avoided. Sometimes it is not avoidable and therefore, modern compilers have a way of dealing with this issue. A usual approach is to guess which branch that will be taken and continue to fill the pipeline with its content. Some compilers always "guess" the branch following the first conditional statement, in this case it is important for the programmer to know this in order to put the most likely case here. [15]

The conclusion from this section is that unnecessary branches and data dependency in loops must be avoided and when this may not be the case, one must make sure that the compiler does its best to optimise these situations.

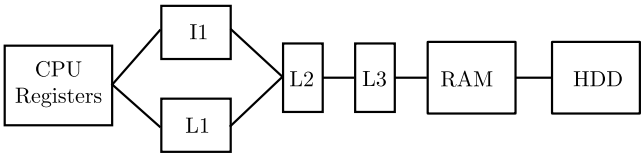


Figure 4.1: Sketch of memory layout in a modern computer. I1 is the instruction cache and L1 the smallest data cache. Memories to the left are faster and smaller but more expensive.

4.2 Locality

In a modern computer, memory is arranged in a subsequent manner, different types of memory are used in order to keep cost down and speed up. Closer to the CPU small and fast memory types are used those are often referred to as cache memories and are an intermediate memory level between the CPU registers and the RAM. In today's computers, usually several layers of cache memories are used. A schematic image of this is shown in fig. 4.1.

When data are to be fetched from the memory into the registers of the CPU, it is first looked for in the L1 data cache if it is not present there, the next level cache (L2) are tried and so on. If the data is not present in any level of the cache it has to be read from the main memory, this is often referred to as a cache miss and is the opposite to a cache hit. To fetch data from the main memory takes, for a modern CPU, the same amount of time in which the CPU may execute about 100 instructions [16]. It is much slower than to fetch from the cache.

As a cache miss occurs, not only the value at the specific address that was wanted but also data that is adjacent to that value is fetched and placed in the cache. This collection of data is called a cache line. In order to keep down memory latency, it may be a good idea if the other data in the cache line could be used before it is overwritten. This is the concept of locality, to keep data adjacent in memory that are to be used adjacent in time. It is a major topic in high performance computing and great performance losses may arise if the principle of locality is not considered.

4.2.1 Locality and the LBM

In an implementation of the lattice-Boltzmann method, the main quantity that is used in computation is the distribution function f_i. It is in the implementation realised as an array of dimension $N_x \times N_y \times Q$ where Q is the number of directions in the discretised velocity space. Basically, two models for arranging f_i in memory exist:

A. All directions for a certain node are placed adjacent in memory.

B. All nodes for a certain direction are placed adjacent in memory.

To decide which model that is chosen, they must be examined with respect to locality and the algorithm used. The algorithm in the LBM consists of two main parts, i.e.

the collision step and the streaming step. In a BGK collision, to update f_i, i.e. the distribution function for direction i, the distribution functions for all other directions from this node must be used in order to compute the equilibrium distribution. Thus for the collision step, model A would give better locality. On the other hand, in the streaming step, to stream f_i, the streaming must start at the nodes where, i is directed out of the domain. Otherwise will unstreamed distributions be overwritten. Thus it is not directly possible to stream all directions for a certain node adjacent in time but instead all nodes for a certain direction. This suggests that model B is more favourable for the streaming step. There is an approach proposed in [17] where two arrays are used for f_i and in the streaming step, the streaming is done from one array to the other and nothing is overwritten. This requires however twice as much memory and it has not been tested in the implementation done in this work.

Q is typically much smaller than the number of nodes $\mathrm{N_x} \times \mathrm{N_y}$. This gives that when a cache line is fetched for model A, same directions from neighbouring nodes will also be fetched into the cache and a cache miss will not occur for each node update in the streaming. In model B, different directions will typically not be fetched for the same node and in the collision step, a cache miss will probably occur for each direction. The conclusion is that model A is better with respect to locality for the "worst" part of the algorithm and is therefore chosen. Tests performed on a laptop with an Intel Core 2 Duo processor also shows that model A in general gives better locality.

4.3 Parallelisation

Parallel computing has, during the last decade, become a major topic not only in high-performance computing but in other fields of computing as well. As it gets tougher and tougher to squeeze in more circuits in a CPU, the idea of having several CPUs (or CPU cores) is today not only utilised in super computers but in most modern user workstations as well.

There are mainly two different hardware setups that has to be distinguished between in parallel computing. Either the memory is shared between all or distributed locally at each processing unit (PU). The former is often referred to as a shared memory and the latter to a distributed memory computer. An example of a shared memory computer would be a workstation having two CPU cores sharing the same RAM. A system using distributed memory could for instance be a super computing cluster where several computers (nodes) are connected in a network.

Obvious differences arise in how the processing units may access data and communicate data between each other. In the case with shared memory computers, common address spaces may be used for communication and for distributed, messages are passed over the network. As a programmer there is a major difference in how this is done. Since this is a common task, there exist software for simplifying the parallelistation of a code. In distributed parallel computing an interface is defined for how the communication over the network may be done, this interface is called MPI (Message Passing Interface). There are several implementations of MPI, one example is Open MPI [18]. For shared memory

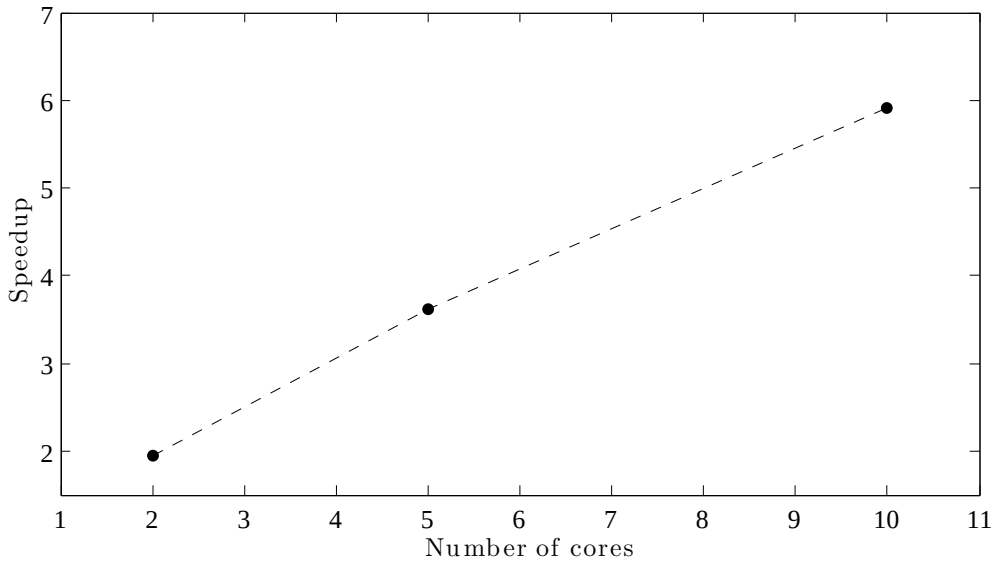


Figure 4.2: Speedup of the parallelised lattice-Boltzmann code. Only the collision step with a BGK collision operator and not the streaming step is parallelised. The computation was done on a computer with two 12 cores `Intel(R) Xeon(R) X5650` CPUs.

computers OpenMP [19] is an API that allows for a rather effortless way of parallelising a code.

An algorithm may in a varying degree be suited for parallelising. If there is a lot of data dependence or dependencies in general. It may be difficult to divide the computation in independent tasks on different PUs. In the case with the algorithm in lattice-Boltzmann, the algorithm is very well suited for parallelisation. This is often considered a great strength of the lattice-Boltzmann method. The main part of the computation is done in the collision step, and a collision on a node is computed independently of data from other nodes. In parallelisation of the streaming step however, communication between nodes are needed. The win in performing the streaming step in parallel is much smaller than for the collision step. By this reason and the fact that a shared memory parallelisation using OpenMP is chosen, only the collision step is parallelised in this work.

The parallelisation was tested for a sample system of 100×100 nodes with periodic boundary conditions and large number of time steps. The speedup was measured for different number of parallel processes used. The result using 2, 5 and 10 respectively cores is shown in fig. 4.2. The computer used had more than 10 CPU cores.

4.4 LBM implementation

The choice of computer language for implementing the LBM methods fell on `C++`. The main reasons for this choice are that `C++` allows both for good possibilities of structuring the good using object oriented features of the language and good performance. Both of these are typically highly desired properties of a computer code. Other alternatives of compiled languages that are known to allow for high performing code are `C` or Fortran, but due to the lack of object orientation in `C` and the lack of previous experience in Fortran neither of those were chosen. Both OpenMP and MPI are also available in `C++`.

This particular implementation was kept general with respect to allowing for an extension to three dimensions without having to change to much of the code. However, due to lack of time, the three dimensional code was never realised. Sometimes when keeping this general approach, performance sacrifices had to be made. If the performance loss was to large, the general approach was put aside and a 2D specific implementation was used. For a concrete example see section 4.5.

One main aim in designing the code interface, was to make it simple to add new classes, representing e.g. collision operators or boundary conditions without having to rewrite any existing code. Another was to make it easy and straightforward to use the code to solve actual problems. To illustrate how the code may be used in such a case, in appendix A follows a snippet of how the implementation is used to model Poiseuille flow.

The performance of the implementation was tested for the different implementations but is here only presented for the Navier-Stokes case, section 3.8, for which it exist comparative work for reference. A common measure in the case of lattice-Boltzmann performance analysis is million lattice updates per second (MLUPS). A crucial parameter in these tests is the lattice dimension, whether the whole f array fits in the last level cache or not may affect the results drastically. The results for a single thread execution and for different grid sizes are presented in fig. 4.3. Here we see a sudden drop about when the grid size passes the cache size. For large enough grids the update frequency seems to converge to about 4.5 MLUPS. In most physical systems the f array will not fit in the cache and if any number should be presented for the performance of the code, this is it.

The update frequency is a very hardware independent measure, it is therefore difficult to compare implementations directly unless you find a benchmark using the exact same hardware setup as you use. However, there are some tests carried out and even if the numbers are not to be compared exactly, it may give a good hunch about the performance of the implementation. In [20], LBM implementations in different computer languages are compared, the C++ implementation is measured to 3.15 MLUPS. Other results are 4.46 MLUPS [21], $1.0 - 4.6$ MLUPS [22] and 6.18 MLUPS (D3Q19) [23]. The numbers here are of comparable magnitude with the one in this work of 4.5 MLUPS.

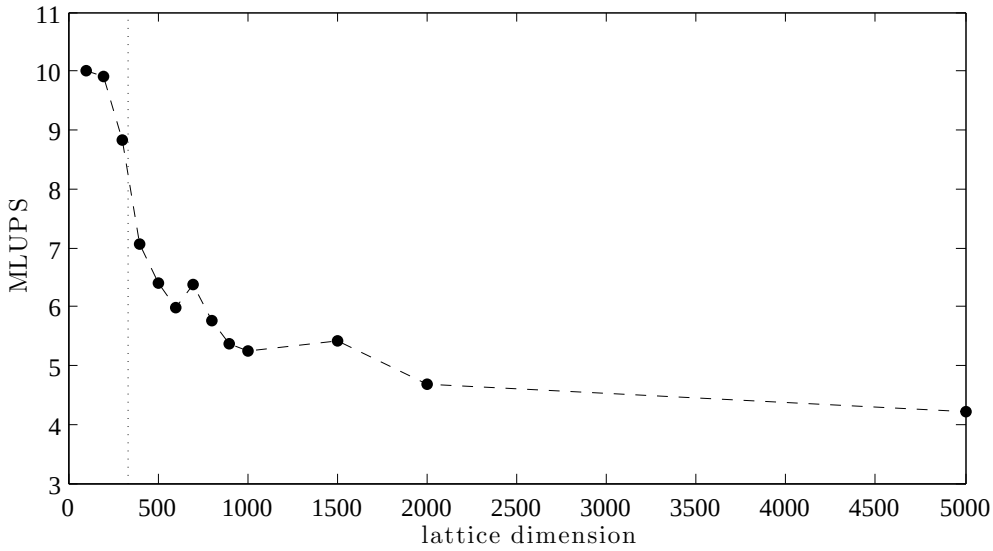


Figure 4.3: Performance of code measured in MLUPS (Million Lattice Updates Per Second) for different grid sized. The dotted vertical line denotes the grid size where the f array has the same size as the L3 cache. These computations were carried out on a `Intel(R) Core(TM) i7-3770 CPU @ 3.40GHz`.

4.5 Profiling

In the development phase of optimising a code, a most valuable tool is a profiler. Basically it is a program that allows for analysing the program as it runs, e.g. to give information of which instructions are executed, if the CPU stalls or if and when cache misses are present. This information is also connected to the source code which gives easy access for the developer to correct eventual performance issues.

A popular profiler used on UNIX systems is `gprof` [24]. In this work, it is used to determine which routines that most contribute to the total execution time. It is in those routines that a code optimisation is most profitable. In order to be able to profile with `gprof`, the code must be compiled with additional compiler flags (`-pg` with `gcc`). This profilable program will run slower than a program compiled without profiling flags, it is thus important to not to forget to compile the "finished" code without profiling flags. Below is an example of some output given by `gprof` when profiling the Navier-Stokes LBM implementation:

```
  %   cumulative   self
time   seconds   seconds    calls  name
40.98      1.61     1.61 40000000  BGKNS::get1moment(int, int, double*)
30.54      2.81     1.20 40000000  BGKNS::fEq(double, double*, double*)
10.69      3.23     0.42 40000000  BGKNS::get0moment(int, int)
10.18      3.63     0.40     1000  StreamD2Q9::stream()
```

for brevity, only the output of the most time consuming routines are shown. These four routines stands together for about 90 % of the total execution time. The main part of this time is for computing the equilibrium distribution which includes computing the zeroth and first moments of the distribution function. The reason why the first moment is four times as time consuming as the zeroth moment, is that the implementation is for a general lattice using the lattice vectors instead of just hardcoding it for the D2Q9 lattice which would have given a computation two times as time consuming as the zeroth moment. This is an example of when less general approach may give a performance boost on the cost of, in this case, to have to define separate routines for each lattice rather than having one general routine.

4.5.1 Memory specific profiling

Even though `gprof` is a good tool in many profiling situations, there may be cases where more low level information such as cache misses or certain instruction counts are desired. Fortunately there exist other more specialised tools to deal with these situations. One popular set of tools for memory profiling/debugging is `valgrind` [25]. Basically it works by simulating a virtual CPU and e.g. memory accesses and cache misses are counted as a program is run on this virtual CPU. In this work the `cachegrind` tool of `valgrind` is used to count cache misses, here is an example output for the Navier-Stokes implementation:

```
I   refs:      33,986,419,147
I1  misses:             1,828
LLi misses:             1,771
I1  miss rate:           0.00%
LLi miss rate:           0.00%

D   refs:      17,284,324,380  (12,564,330,224 rd   + 4,719,994,156 wr)
D1  misses:       364,260,481  (   354,685,981 rd   +     9,574,500 wr)
LLd misses:       175,534,108  (   165,973,036 rd   +     9,561,072 wr)
D1  miss rate:            2.1% (           2.8%     +           0.2%  )
LLd miss rate:            1.0% (           1.3%     +           0.2%  )

LL refs:          364,262,309  (   354,687,809 rd   +     9,574,500 wr)
LL misses:        175,535,879  (   165,974,807 rd   +     9,561,072 wr)
LL miss rate:             0.3% (           0.3%     +           0.2%  )
```

where `I`, `D` and `LL` is the instruction, L1 and last level (L3 in this particular case) caches respectively. The grid is chosen large enough to keep the size of the f array much larger ($\sim$ 300 MiB) than the 8 MiB L3 cache. We see that about 1% of the data memory lookups result in cache misses. A more thorough investigation using `cg_annotate` of in what routines the cache misses arise, gives that about two thirds are in the streaming step and about one third in the collision step. This behaviour is in accordance with the memory model chosen, see section 4.2.1.

5

Verification of model and implementation

Prior the interconnection of the three solvers, they are in this chapter evaluated independently. This is done by comparing computed solutions to known solutions of some classic systems.

5.1 Poiseuille flow

First out for evaluating the LBM solver of the Navier-Stokes equations is the situation with Poiseuille flow. This is a classic example and one of the easiest situations were the NS equations are exact solvable. Consider a 2D channel of length l and width H. If the flow in this channel is driven by a constant force, e.g. a constant pressure drop, the velocity profile will adopt a parabolic shape in the steady state situation. Here follows a brief derivation of the exact expression for the velocity profile.

Consider the (non-dimensional) Navier-Stokes eqs. (3.43) and (3.42) in 2D. Let x be the direction along the channel and y the direction across the channel. In the case of a pressure gradient in the x direction and no other external forces involved we deduce that the y component of the velocity is zero. Thus eq. (3.43) reduces to an equation for the x component of the velocity

$$\frac{\partial \mathrm{u_x}}{\partial t} - \mathrm{u_x}\frac{\partial \mathrm{u_x}}{\partial x} = \nu\frac{\partial^2 \mathrm{u_x}}{\partial y^2} - \frac{\partial \mathrm{P}}{\partial x}. \tag{5.1}$$

Under the assumption of a system in steady state, i.e. $\partial \mathrm{u_x}/\partial t = 0$. Further if the flow is fully developed $\partial \mathrm{u_x}/\partial x = 0$, this also follows from (3.42). We now have together with writing the constant pressure gradient as $\Delta \mathrm{P}/l$:

$$\frac{\partial^2 \mathrm{u_x}}{\partial y^2} = \frac{\Delta \mathrm{P}}{\nu l}. \tag{5.2}$$

Solving this equation with no slip boundary conditions, $\mathrm{u_x}(0) = \mathrm{u_x}(H) = 0$ gives an expression of the velocity profile

$$\mathrm{u_x}(y) = \frac{\Delta \mathrm{P}}{2\nu l} y(y - H). \tag{5.3}$$

In this benchmark, the pressure gradient is incorporated as a force. Other possibilities would be to drive the fluid by imposing fixed pressures/velocities at the inlet and outlet. In this case, with a driving force, periodic boundary conditions are imposed at the inlet and outlet. At the channel walls, the bounce-back boundary condition described in section 3.11.1 is used. The actual boundary will then be located half a node-node distance into the fluid.

A grid with 50 nodes across the channel and 3 in the flow direction is used. The driving force is set to $\Delta \mathrm{P}/l = 1 \cdot 10^{-4}$ and the viscosity is $\nu = 0.2778$ from eq. (3.49) with relaxation parameter $\omega = 0.75$. The solution that was obtained after 20000 iterations is presented in fig. 5.1. The analytical solution, eq. (5.3), is plotted for comparison.

The agreement between computed and analytical solution is satisfying. With an RMS error of $9.271 \cdot 10^{-5}$ l.u. and a maximum absolute error of $1.198 \cdot 10^{-4}$ l.u. Note that the actual boundary is located half a node-node distance into the computational domain, this is due to the implementation of the bounce-back scheme, see section 3.11.1.

5.2 Taylor-Green vortex

A more sophisticated system than the Poisseuille flow is the decaying Taylor-Green vortex flow. Also this system is one of the few where an analytical solution to the Navier-Stokes equation is possible to find. In this section, the Taylor-Green flow is used to benchmark the 2D LBM proposed in section 3.8. A 2D test of the force implementation is also carried out in section 5.2.1.

The Taylor-Green vortex flow in two dimensions is defined by the following pressure and velocity fields [7]:

$$\begin{aligned} u_x(\mathbf{x}, t) &= -\frac{1}{a} \cos(ax) \sin(by) \exp(-\nu(a^2 + b^2)t) \\ u_y(\mathbf{x}, t) &= \frac{1}{b} \sin(ax) \cos(by) \exp(-\nu(a^2 + b^2)t) \\ P(\mathbf{x}, t) &= -\frac{1}{4} \left(\frac{1}{a^2} \cos(2ax) + \frac{1}{b^2} \cos(2by) \right) \exp(-2\nu(a^2 + b^2)t) \end{aligned} \tag{5.4}$$

where ν is the viscosity, $\mathbf{x} \in [0, 2\pi]^2$ and a and b are two real constants. It is straightforward to verify that these quantities satisfy the incompressible Navier-Stokes, eqs. (2.22) and (2.23). The constants a and b are chosen as $a = b = 2\pi/N$ where $N = N_x = N_y = 100$ is the grid resolution. This particular choice of a and b allows us to use lattice

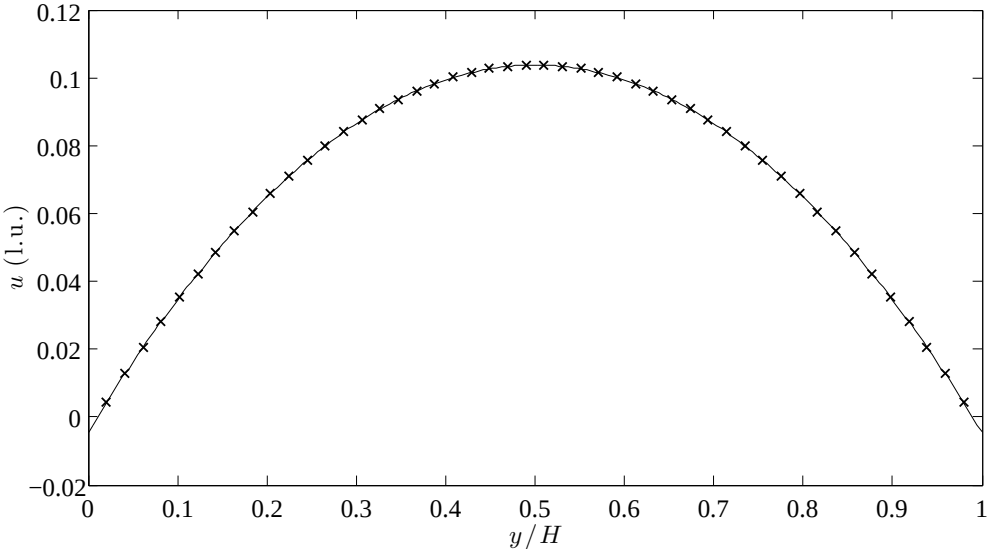


Figure 5.1: Computed velocity profile of Poiseuille flow (×) compared to the analytical solution (solid line). A grid of 3×50 nodes were used. There was also no variation of the velocity field in the flow direction (not shown in this figure). The velocity is given in lattice units.

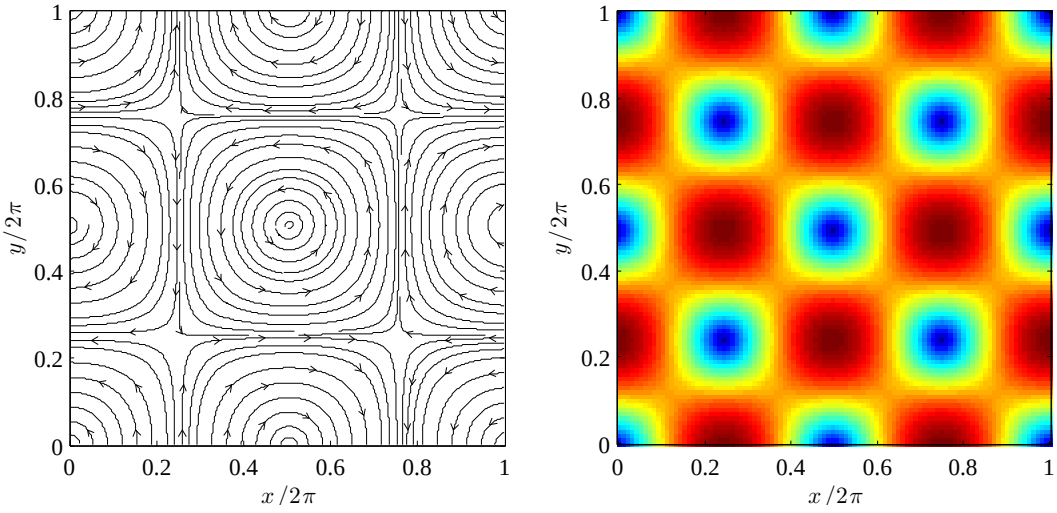
units with $\delta_x = \delta_t = 1$ in the simulation. The maximum velocity in the vortex at $t = 0$ is chosen as $u_0 = 1/a = 1/b$.

The simulation is performed with $\nu = 0.05$ l.u. on a 100×100 lattice. Following the initialisation with the analytical solution, eq. (5.4), at $t = 0$, the decay is studied and compared with the analytical solution. Initialisation of the velocities is done by setting $f_i = f_i^{(eq)}(\rho, \mathbf{u})$ where ρ includes both the constant density (order ϵ^0) and the pressure (order ϵ^2), compare eq. (3.67).

The result is presented in fig. 5.2 where the velocity field and the magnitude of the velocity field is visualised. This is at a time, $t_{1/2}$, where the maximum magnitude of the velocity has decreased to half of the initial value, i.e. $t_{1/2} = \log(2)/(2\nu(2\pi/N)^2)$. In fig. 5.2, the analytical and the computed solutions is compared on a section of the domain at $t = t_{1/2}$. The agreement with an average error in the order of 10^{-3} is comparable with previous works, e.g. [7].

5.2.1 Four rows mill

In the decaying Taylor-Green flow, the time derivative term in Navier-Stokes momentum equation is balanced by the viscous term. We now wish to compensate for the decay by introducing a force. In a steady state situation: $\partial_t \mathbf{u} = 0$ and the force must balance the viscous term. The force is therefore chosen as



(a) Velocity field **(b)** $|\mathbf{u}|$

Figure 5.2: Visualised velocity field (a) and magnitude of the velocities (b) for a decaying Taylor-Green vortex. The velocity field is taken at $t = t_{1/2}$ and the values in (b) varies from $-u_0/2$ (blue) to $u_0/2$ (red).

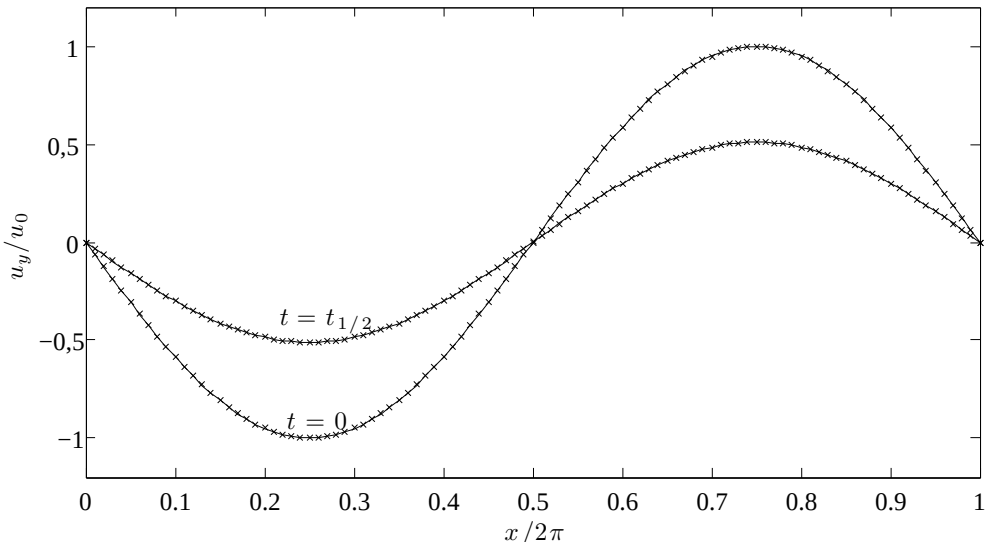


Figure 5.3: 1D section of the decaying Taylor-Green flow. The y component of the velocity is plotted at $y = \pi$. Computed solutions ($\times$) are compared with analytical (solid) at two different times $t = 0$ and $t = t_{1/2}$. A domain of 100×100 nodes were used.

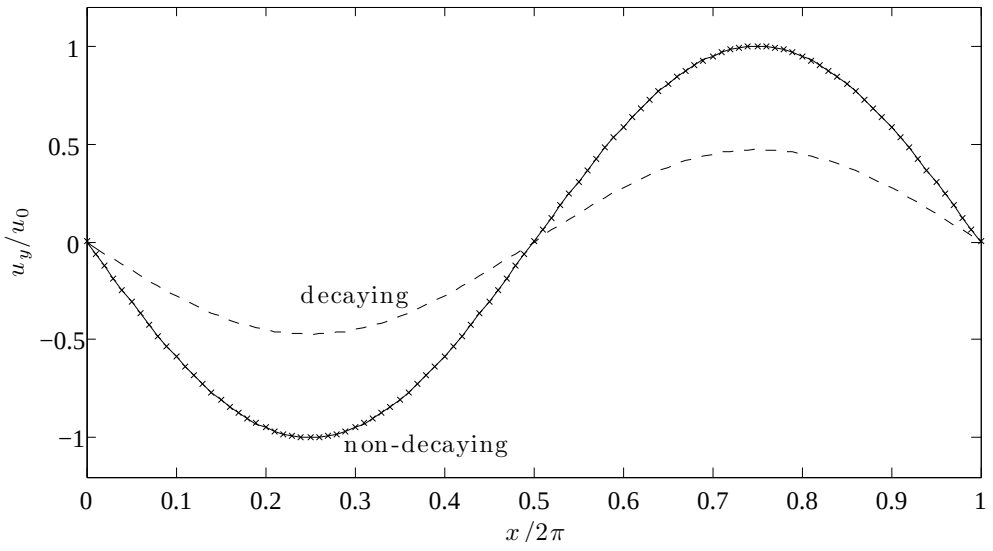


Figure 5.4: 1D section of the steady four rows mills flow at $t = t_{1/2}$. An external force have been included to compensate the decay (dashed). The y component is plotted for $y = \pi$. We observe no decay as the computed solution ($\times$) and the initial velocity (solid) coincide.

$$\begin{aligned} F_x(\mathbf{x}, t) &= \nu(a^2 + b^2)u_x(\mathbf{x}, t) \\ F_y(\mathbf{x}, t) &= \nu(a^2 + b^2)u_y(\mathbf{x}, t) \end{aligned} \tag{5.5}$$

where u_x and u_y are the velocities from eq. (5.4). Using the same parameters, initialisation and domain as in the decaying case, no decay is observed. In fig. 5.4, a section of the y component of the velocity is shown at $t = t_{1/2}$.

5.3 Helmholtz equation

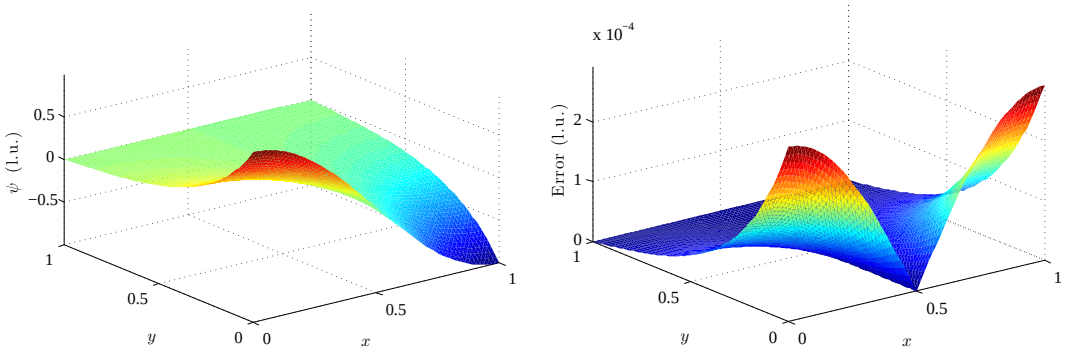
The LB formulation for solving Poisson's equation as well as the implementation was tested by solving Helmholtz equation with a certain set of boundary conditions allowing for finding an analytical solution. The homogeneous Helmholtz equation reads:

$$\nabla\psi = \lambda^2\psi \tag{5.6}$$

where λ is a real parameter. The equation was solved for $\lambda = 2$ on the domain $(x, y) \in [0, 1] \times [0, 1]$ with the following Dirichlet boundary conditions:

$$\begin{aligned} &\psi(0, y) = -\psi(1, y) = \tfrac{\sinh\sqrt{\lambda^2+\pi^2}(1-y)}{\sinh\sqrt{\lambda^2+\pi^2}}, \\ &\psi(x, 0) = \cos\pi x, \;\; \psi(x, 1) = 0. \end{aligned} \tag{5.7}$$

The analytical solution to eq. (5.6) with the given boundary conditions is [26]:



(a) Computed solution **(b)** Error

Figure 5.5: To the left, the computed solution of the Helmholtz equation, eq. (5.6). To the right, the error.

$$\psi(x, y) = \cos \pi x \frac{\sinh \sqrt{\lambda^2 + \pi^2}(1 - y)}{\sinh \sqrt{\lambda^2 + \pi^2}}. \tag{5.8}$$

A grid of 65×65 nodes was used when computing the LBM solution. The computational domain was rescaled to the desired one by setting $\delta_x = 1/(n_x - 1) = 1/64$ and $\delta_t = \delta_x^2$. An other possibility would have been to rescale the parameter λ and have $\delta_x = \delta_t = 1$.

The boundary conditions in eq. (5.7) was implemented using the He/Zou approach. A bounce back approach with some momentum addition would also have been possible but was not chosen due to that the actual boundary location is then not at the node location, but half a node-node distance into the computational domain. Also the bounce-back implementation is previously tested.

In fig. 5.5a, the obtained solution is presented together with the absolute error in fig. 5.5b. The agreement is satisfying, with an error which magnitude is about the same as in previous works [26]. The error takes on its maximum at the boundary of the domain, implying that the fulfilment of the boundary conditions is not complete.

5.4 Advection-Diffusion

Before the implementation of the Nernst-Planck part of the model is tested, a special case is considered, i.e. when the electrical potential in the domain is constant. This makes the flux term including the electrical potential in eq. (2.8) vanish and we have to solve only for pure advection and diffusion.

Introducing characteristic scales for the concentration (c_0), advective velocity (u_0) and length (l_0) respectively, gives the non-dimensional advection-diffusion equation:

$$\frac{\partial \mathrm{c}}{\partial t} + \mathbf{u} \cdot \nabla \mathrm{c} - \frac{D}{u_0 l_0} \nabla^2 \mathrm{c} = 0. \tag{5.9}$$

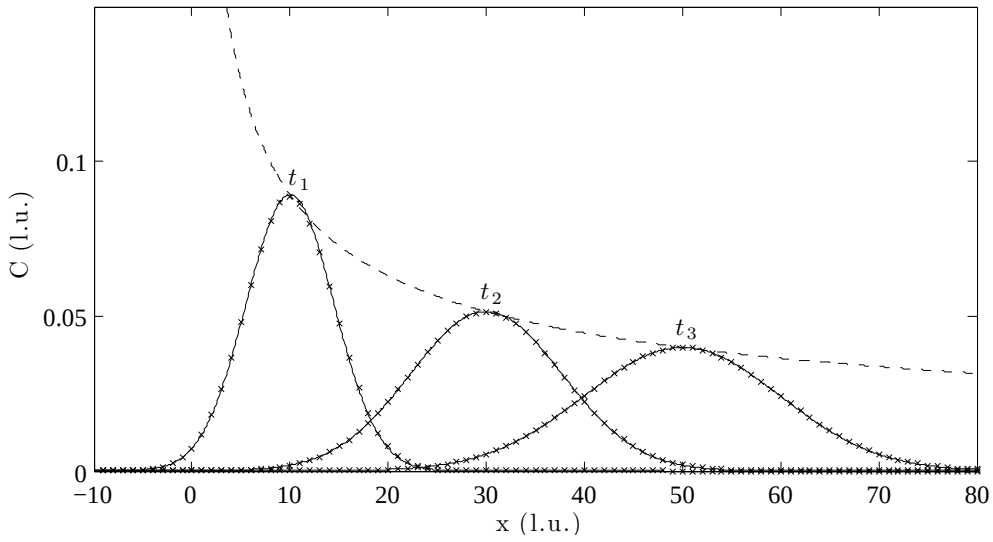


Figure 5.6: Computed solutions (×) of the advection-diffusion equation for a point mass evolving in time and space. Three different times ($t_n = 100n$) are compared to analytical solutions (solid). The Amplitude of the solutions as function of time has also been plotted (dashed). The advecting velocity, $u_0 = 0.1$ was used together with a Peclet number, Pe = 10. All units are in lattice units.

All variables in (5.9) are non-dimensional. The quantity $\mathrm{Pe} = u_0 l_0/D$ is often referred to as the Péclet number. It determines the relation between contributions to the dynamics from advection and diffusion respectively. For $\mathrm{Pe} \gg 1$ the dynamics is dominated by advection and for $\mathrm{Pe} \ll 1$ by diffusion.

The LB model described in section 3.7 was tested by studying the evolution in time and space of a point mass in one dimension. The analytical solution of eq. (5.9) in one dimension with initial conditions $\mathrm{c}(x, t = 0) = \delta(x)$ on an infinite domain is:

$$\mathrm{c}(x,t) = \sqrt{\frac{\mathrm{Pe}}{4\pi t}} \exp\left(-\frac{(x-ut)^2 \mathrm{Pe}}{4t}\right). \tag{5.10}$$

In the numerical computation, the parameters Pe = 10 and $|\mathbf{u}| = 0.1$ were used. The domain consists of 200 lattice nodes and three snapshots in time at $t = 100, 200, 300$ l.u. were compared to the analytical solution. The result is presented in fig. 5.6.

5.5 Nernst-Planck, a special case

This benchmark aims to test the advection due to present electrical fields in the advection diffusion solver. It is done by solving for the ion concentration in a system that fulfils the assumptions for the Poisson-Boltzmann distribution. Beside a system in steady state,

the assumptions are a simple geometry that allows for the one dimensional integration and a zero advective velocity, see section 2.4.2.

A 1:1 ion solution will be considered and therefore two solvers for the Nernst-Planck equation are needed. One for positive and one for negative ions. These will be coupled to a solver for Poisson's equation which will update the potential each time step according to the present ion concentration. In the steady state, the ion concentrations for positive and negative ions respectively are compared to the exponential expression in the Poisson-Boltzmann situation, eq. (2.13). The potential in this expression is obtained by solving the Poisson-Boltzmann equation, eq. (2.15).

Two solvers using the method in section, 3.7 are set up with the following set of parameters:

$z = \pm 1$

$D = 10^{-8}\ m^2 s^{-1}$

$T = 293\ K$

$\epsilon_r = 80$

The Nernst-Planck equation is put on non-dimensional form by introducing the following characteristic quantities:

$c_0 = 10^{-4}\ mol/m^3$

$\ell_0 = 2{\cdot}10^{-5}/ny\ m$

$V_0 = -50\ mV$

$u_0 = 0.1\ m/s.$

These parameters gives a Peclet number of Pe = 2 which gives a relaxation parameter $\omega_{NP} = 0.5$.

The geometry of the system simulated is an infinite channel with straight walls. A grid of 3×100 nodes is used for the computation, i.e. 100 nodes across the channel. At the walls, the no flux boundary condition, eq. (2.9), is implied through the mirror reflection approach described in section, 3.11.2. For the Poisson solver, a constant surface charge density of $\sigma = -0.17\ \mu C/m^2$ is applied to the walls, eq. (2.3). This is realised by a modified mirror reflection, see section 3.11.2.

The charge concentration at the middle of the channel (denoted by c^∞ in eq. (2.13)) in the Poisson-Boltzmann distribution are set to the values obtained from the Nernst-Planck solver.

In fig. 5.7, the resulting charge distributions are presented.

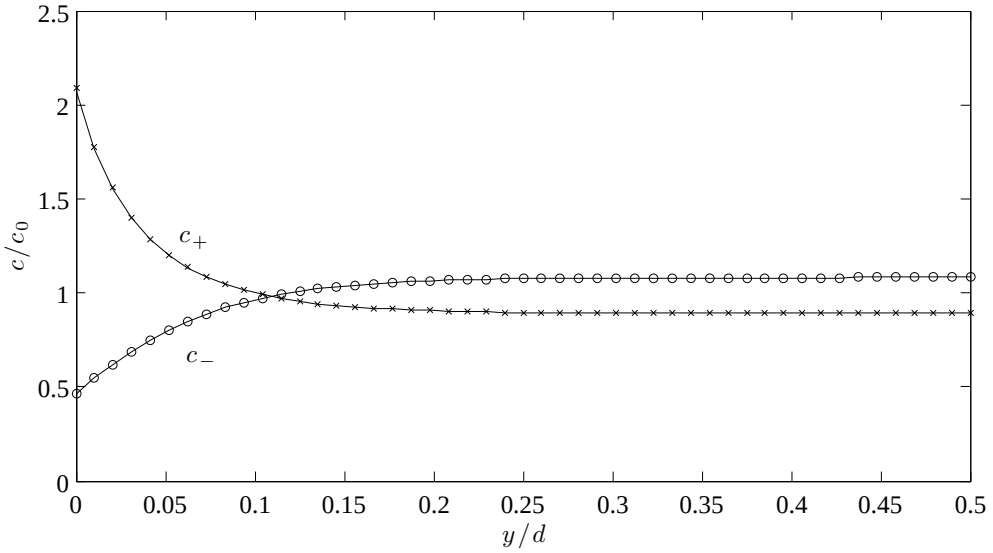


Figure 5.7: Computed charge concentrations for positive (c_+) and negative (c_-) ions respectively in contact with a negatively charged wall. The concentrations are compared with the Poisson-Boltzmann distribution (solid).

6

Modelling of electrokinetic flow

In this chapter, results of modelling of electrokinetic flow using the previously described lattice-Boltzmann method are presented. The choice of systems is done with focus on those where the Poisson-Boltzmann approach is not applicable. First simple 1D systems in channels are considered, then more complicated systems in 2D. All results presented here are for systems in steady state. The liquid used in the modelling is a KCl solution defined by the parameters in tab. 6.1.

6.1 Charge concentration and potential in 1D system

To get a feeling for the systems dealt with here, a system where no advection is present is first be considered. The geometry consists of a 2D channel where both walls are negatively charged. As earlier discussed in chapter 2, positive ions will be attracted to the walls and negative will be repelled.

Table 6.1: Physical parameters of the KCl ion solution that is modelled in this chapter. Parameters are from [1].

Relative permittivity, ϵ_r	80
Mean ionic conentration, c_0	10^{-4} mol/m^3
Conductivity, σ_c	1.5 mS/m
Temperature, T	293 K
kinematic viscosity, ν	1.0 μm^2/s
Diffusion coefficient, $D_+ = D_-$	10^{-10} m^2/s

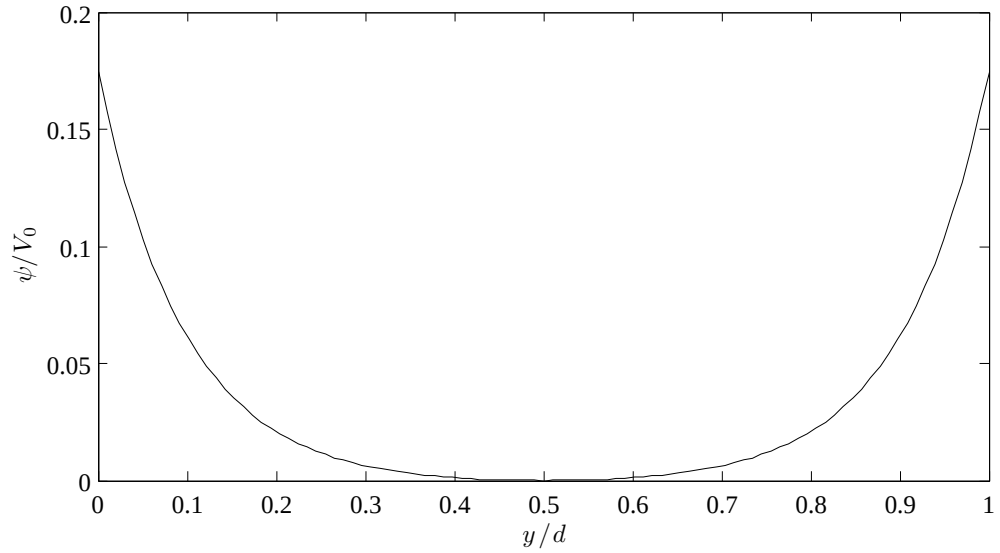


Figure 6.1: Computed electric potential across a channel of width $d = 10\mu$m. The solution in the channel is a KCl solution defined by parameters in table 6.1. The channel walls are negatively charged. Note that V_0 is negative.

A channel of width 10μm and with walls with a surface charge of 3.5μC/m is considered. The computed electric potential in the steady state for this system is presented in fig. 6.1. With the particular choice of width and surface charge, we see that the double layers extend a substantial length into the channel. The Debye length is in this case $\kappa^{-1} = 1\mu$m and $d\kappa = 10$. It is the charcteristic EDL length that is compared with when stating that a channel is wide or narrow. If $d\kappa \gg 1$, the channel is said to be wide otherwise narrow. In this section mainly narrow channels are studied.

Also the concentrations of positive and negative ions corresponding to the potential are visualised in fig. 6.2. The surplus of positive ions in the vicinity of the walls together with the surplus of negative ions at the centre of the channel are clearly shown.

6.1.1 Nernst-Planck vs. Poisson-Boltzmann

In the Poisson-Boltzmann model, section 2.4.2, the parameter c_i^∞ that determines the value of the concentration far from the EDL is chosen as the bulk concentration of the fluid. The problem with narrow channels is that there is no bulk and as we see in fig. 6.2, it would not be an accurate choice. Also if c_i^∞ would be set to the bulk concentration for a narrow channel the total number of ions would not be preserved. For a 1:1 solution and with negatively charged channel walls, the number of positive ions would not equal the number of negative ions.

Further assumptions made in the Poisson-Boltzmann model is that of a simple geometry, the thickness of the EDL must be small to the curvature of the boundary. Also

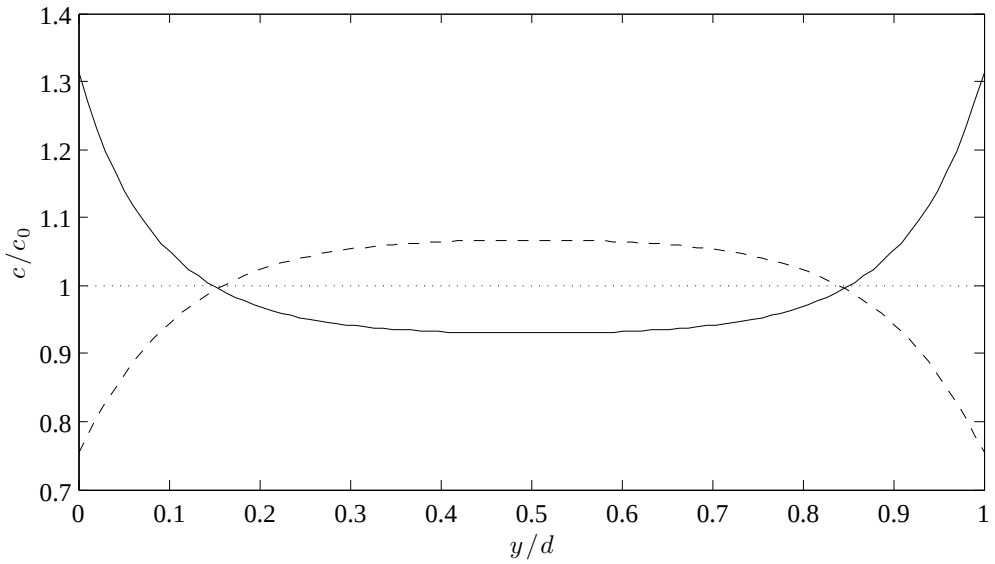


Figure 6.2: Computed positive (solid) and negative (dashed) charge distribution across a channel of width $d = 10\mu$m. The solution in the channel is a KCl solution defined by parameters in table 6.1. The channel walls are negatively charged.

no advection is assumed.

6.2 Electroviscous effect

In section 2.6, the physical model behind pressure-driven electrokinetic flow is presented. The effect of interest that arise in this kind of systems is the electroviscous effect.

Velocity profiles computed in a 1D situation is presented here. We consider a 1 μm wide channel that has negatively charged walls. The wall charge is a parameter that is varied and from this the effect on the velocity profile is studied. The system is setup with a Peclet number, $\mathrm{Pe} = 1$ and a Reynolds number, $\mathrm{Re} = 10^{-4}$. To drive the flow, a pressure gradient of 1 kPa/m is set. The resulting velocity profiles are presented in fig. 6.3.

The velocity profiles obtained agrees qualitatively with a similar simulation preformed in [27]. The local minimum that arises for $\sigma_s = 20\sigma_0$ is due the high accumulation of negative ions in the middle of the channel. This is an effect that only is seen for narrow channels.

In the model proposed here, using Ohm's law to relate the ion current to the streaming potential, the force on the flow due to the electroviscous effect is opposite to the flow everywhere where there is a net charge present in the fluid. However, in most texts about the electroviscous effect, e.g. in [1], [4] and [27], this force is computed using a mean current approach. An integration of ion flux over the cross-section of the channel is

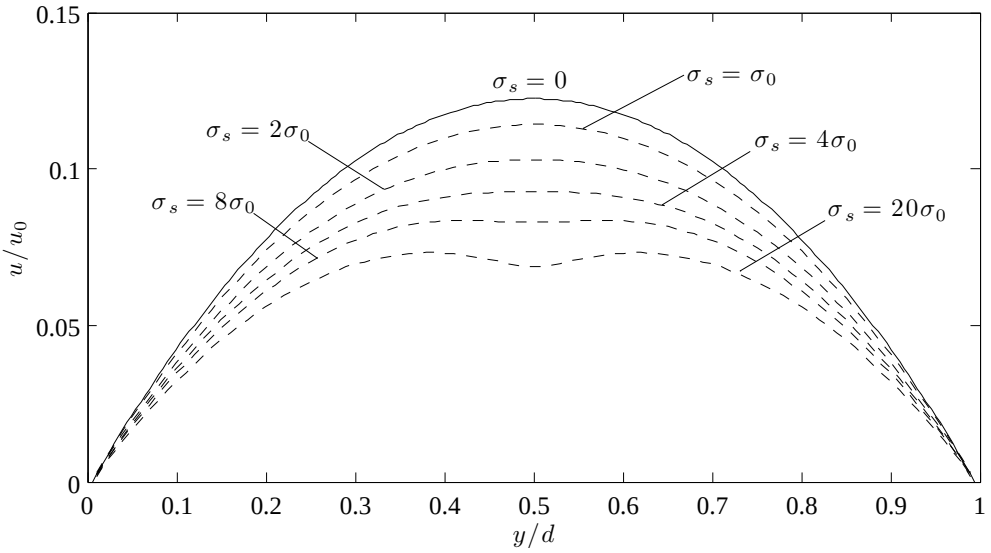


Figure 6.3: Computed velocity profiles across a 2D channel of width $d = 1\mu$m. The flow is driven by a pressure gradient and the flow is slowed down due to the electroviscouos effect, this effects dependence on the surface charge σ_s is here illustrated. The solution in the channel is a KCl solution defined by parameters in table 6.1. In this simulation, $\sigma_0 = 0.89\mu\text{C/m}^2$, $\partial_x P = 1$ kPa/m and $u_0 = 10$ mm/s.

performed and then a net current for the whole channel is obtained. From this current, a mean streaming potential is obtained for the channel and a force is calculated using the charge density. This gives that positive and negative net charged regions of the fluid will be affected with forces of opposite sign respectively. Having a mean streaming potential for the whole channel also gives contraintuitve results when considering the fact that regions in the fluid with the same net charge but different velocities is affected by the same force. Also with this approach the electroviscous effect would in principle be able to locally oppose the flow direction. Also, in a more complicated geometry, this approach would break down. In fig. 6.4, two velocity profiles from fig. 6.3 are compared with corresponding profiles computed using the mean current approach. It is seen that the force slowing down flow is smaller for the case when using a mean current. This is due to the cancellation between negative and positive ion fluxes in the integration.

6.3 Electroosmotic flow

As described in section 2.7, electroosmotic flow is driven by an electric field rather than a pressure gradient. Charge particles will be affected by a force and will drag the fluid with them. The effect is investigated in this section.

A 10 μm channel is considered with walls charged with $3.56\mu\text{C/m}^2$. The 1:1 ratio between positive and negative ions is in this section put aside for a moment. To inves-

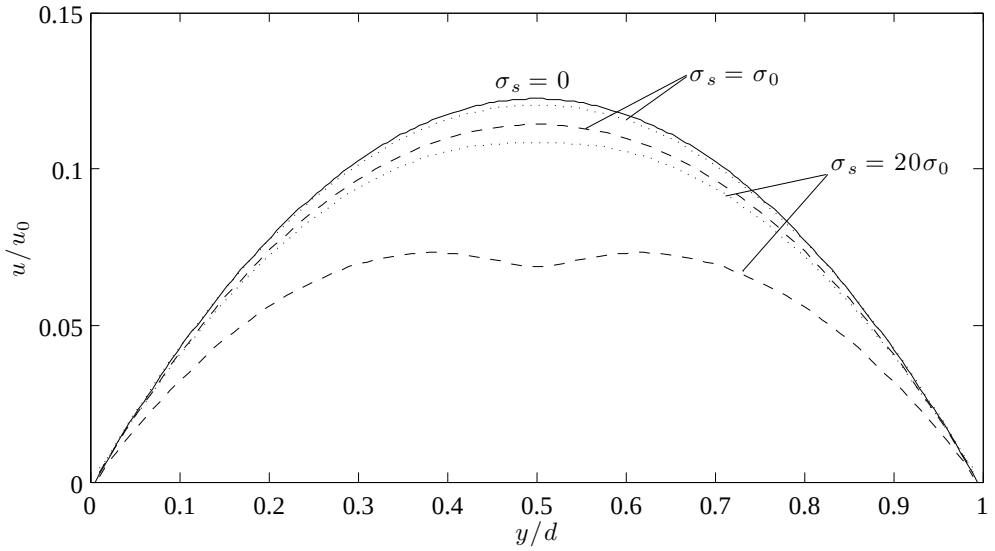


Figure 6.4: Comparison between velocity profiles computed using a mean current (dotted) and by using the actual local current (dashed) for the streaming potential. The solution in the channel is a KCl solution defined by parameters in table 6.1. In this simulation, $\sigma_0 = 0.89\mu\text{C/m}^2$, $\partial_x P = 1$ kPa/m and $u_0 = 10$ mm/s.

tigate how the electroosmotic flow behaves for different situations of the charge density, especially in the middle of the channel, the amount of negative ions are varied. The different situations are investigated, a surplus and a lack of negative ions together with a neutral solution at the middle of the channel. Thus, the following values of the mean concentration of negative ions are set: $0.7c_0$, $0.75c_0$, $0.78c_0$, $0.8c_0$ and $0.85c_0$.

A constant electric field of 10^5 V/m along the channel are set and the same Peclet and Reynolds number as before are used. The obtained ion concentrations and the velocity profiles are presented in fig. 6.5 and fig. 6.6 respectively.

In the case with a neutral middle of the channel the traditional "plug profile" of electroosmotic flow for wide channels are reproduced. In this case the force from the electric field only affects the fluid near the walls where a net charge is present, due to viscous forces, a constant velocity profile is then obtained in the middle of the channel. For a positive net charge in the middle of the channel and thereby everywhere in the channel, the velocity profiles are not of the "plug" shape but more parabolic. For a negatively net charged middle of the channel and with the parameter of choice, the viscous effect only compensates for the opposing effect from the electric field when there is a small negative net charge. We see that in this case, with the chosen parameters, for a 1:1 solution the flow would be completely opposite of the electric field which is an apparent difference to the wide channel case.

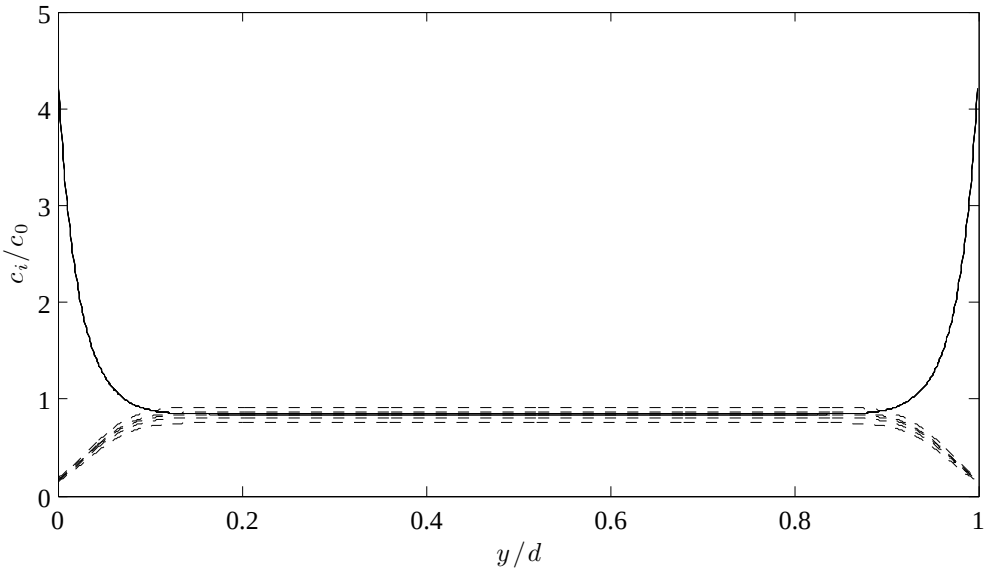


Figure 6.5: Computed charge distributions for positive (solid) and negative (dashed) ions. The mean concentration of positive ions is c_0 while that of negative ions are varied between the values $0.7c_0$, $0.75c_0$, $0.78c_0$, $0.8c_0$ and $0.85c_0$. The geometry is a channel of width 10μm.

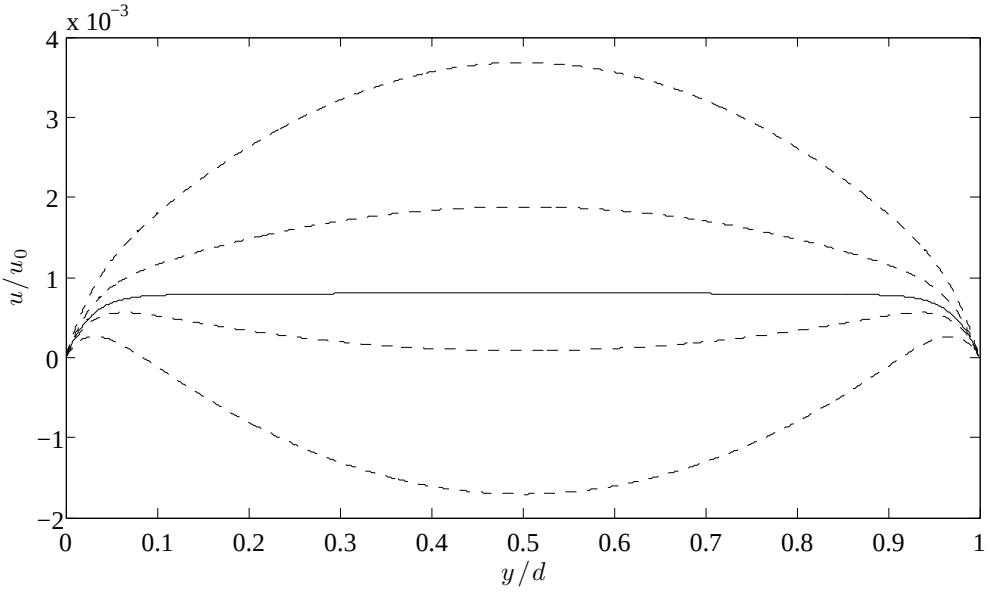
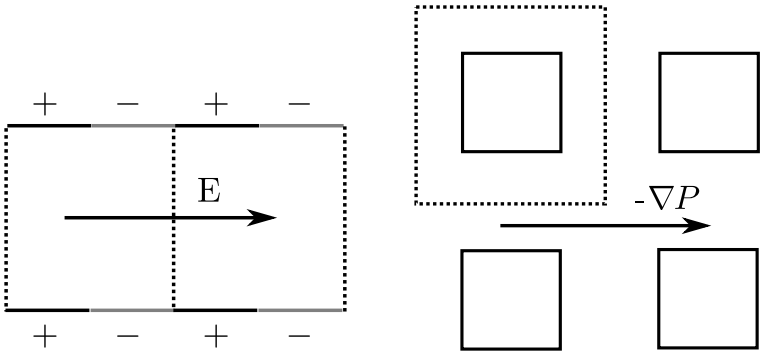


Figure 6.6: Computed velocity profiles for electroosmotic flow, in a 10μm wide channel. The different profiles correspond to different ratios between positive and negative ions, see fig. 6.5. The solid profile is the "plug flow" that corresponds to the Poisson-Boltzmann case, where the middle of the channel is net neutrally charged.



(a) Heterogeneous wall charge **(b)** Array of charged squares

Figure 6.7: Sketch of setups for the two physical 2D systems that are modelled. In (a), electroosmotic flow through a channel with heterogenously charged walls. In (b), pressure driven flow in an array of squares.

6.4 Flow in a channel with heterogeneously charged walls

Now a channel with walls charged with a varying charge is considered. The walls are charged piecewise constant with every second piece positive and negative respectively, see fig. 6.7a. The length of the charged sections are chosen as $d/4$ where $d = 10\mu$m is the width of the channel. The flow is electroosmotic and driven by an external field of 50 kV/m.

The resulting velocity profile together with the charge distribution of positive ions in the steady state is presented in fig. 6.8.

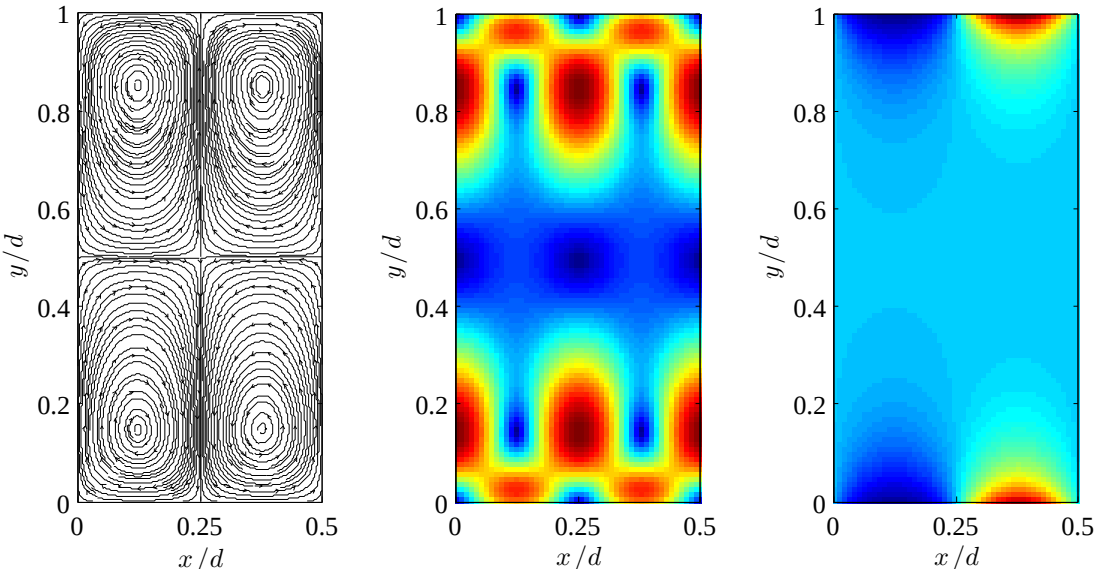
There is an accumulation of positive and negative ions in the vicinity of the negative and positive charged boundaries respectively. In the middle of the channel, the fluid is neutral.

The vortexes obtained agrees qualitatively with those computed in [28]. This kind of system is for example used in mixing of charge fluids [29]. Varying charges are imposed on the boundary of a domain and an electric field is applied, this results typically in a flow similar to the vortexes obtained here.

6.5 Flow in an array of charged squares

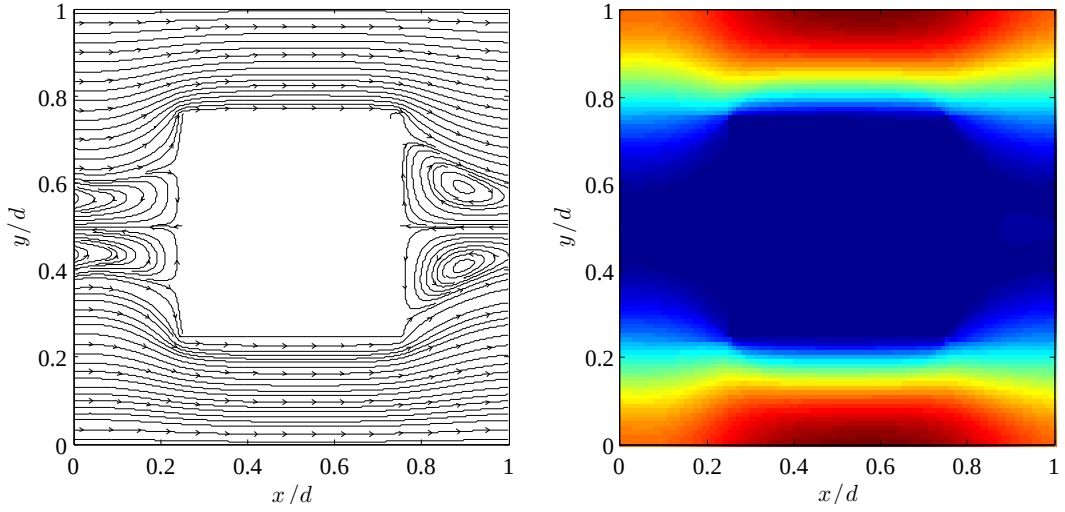
In this section, flow through a periodic structure is modelled. The structure is consists of squares placed in a periodic pattern shown in fig. 6.7b. The unit cell, used in the computation is marked by the dashed box in the figure. The dimension of the unit cell is chosen to 10μm and the permeability through the structure is investigated for different sizes of the squares. Also the effect of having charged vs. uncharged squares is studied.

In fig. 6.9 and fig. 6.10 the computed velocity fields around a square in a unit cell is visualised. The side of the square is of length $0.5d$.



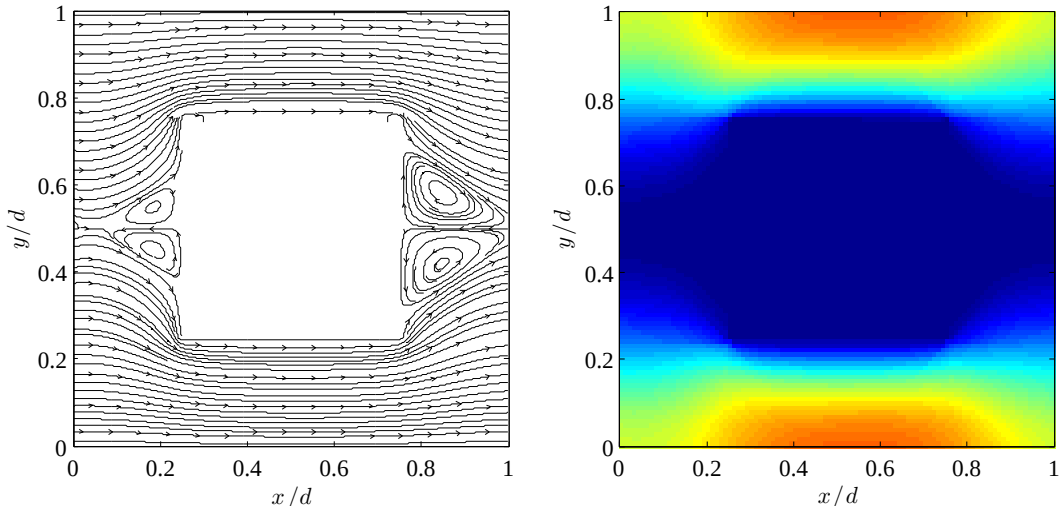
(a) Velocity field **(b)** $|\mathbf{u}|$ **(c)** c_+

Figure 6.8: Visualised velocity field (a), magnitude of the velocities (b) and charge concentration of positive ions for a flow in a 2D channel with heterogeneously charged walls. A constant electric field of 50 kV/m drives the flow. The velocity field in (b) varies from 0.02μm/s (blue) to 2.2μm/s (red). The charge concentration in (c) varies from $0.45c_0$ (blue) to $2.12c_0$ (red).



(a) Velocity field **(b)** $|\mathbf{u}|$

Figure 6.9: Visualised velocity field for flow through an array of **uncharged** squares. The magnitude of the velocity varies between 0.20 m/s (red) to 0 m/s (blue).



(a) Velocity field **(b)** $|\mathbf{u}|$

Figure 6.10: Visualised velocity field for flow through an array of **charged** squares. The magnitude of the velocity varies between 0.16 m/s (orange) to 0 m/s (blue). The surface charge is set to $\sigma_s = 1.78\mu\text{C/m}^2$. The same colour scale as is fig. 6.9 is used.

The main effect noted that the charge on the square has on the flow, is that it is slowed down. This is due to that charge inhomogeneities arise when the charged square is introduced and from this the flow is slowed down due to the electroviscous effect. In figs. 6.11 and 6.12, the velocity component in the pressure drop direction (x) is shown in a section at $x = d/2$ and $x = 0$ respectively.

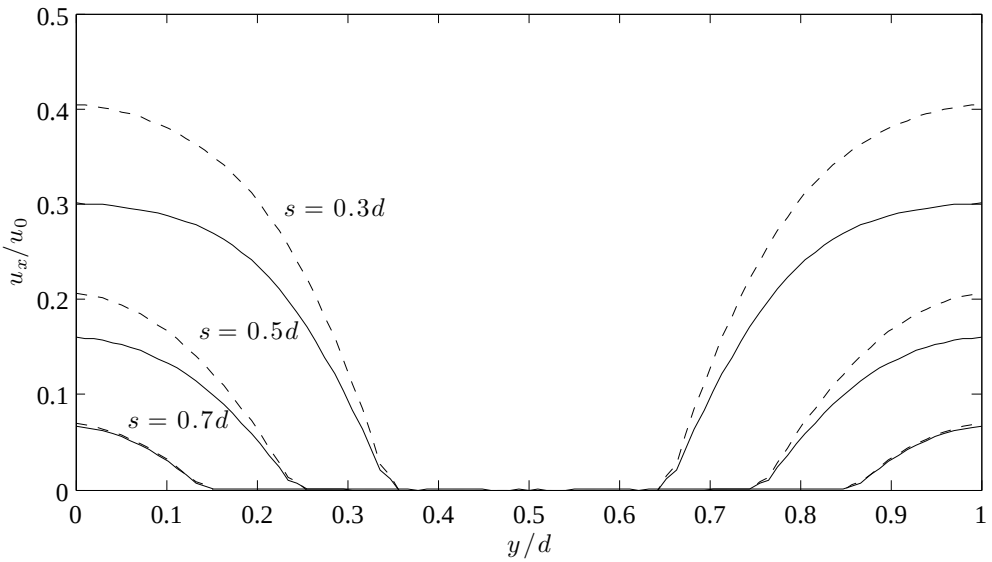


Figure 6.11: Velocity profiles across the square array at $x = d/2$ in the cell. The sides of the squares are varied between $0.3d$, $0.5d$ and $0.7d$ where $d = 10\mu$m is the length of the cell. The flow is driven by a pressure gradient and the uncharged (dashed) and charged (solid) squares are compared. $\sigma_s = 1.78\mu\text{C/m}^2$ (solid), $\partial_x P = 0.5$ kPa/m and $u_0 = 1$ mm/s.

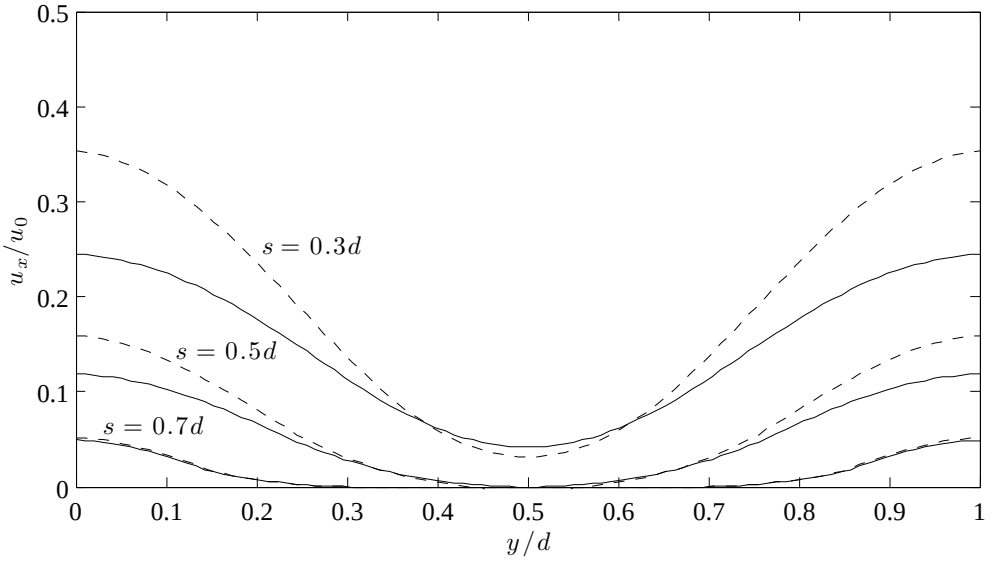


Figure 6.12: Velocity profiles across the square array at $x = 0$ in the cell. The sides of the squares are varied between $0.3d$, $0.5d$ and $0.7d$ where $d = 10\mu$m is the length of the cell. The flow is driven by a pressure gradient and the uncharged (dashed) and charged (solid) squares are compared. $\sigma_s = 1.78\mu\text{C/m}^2$ (solid), $\partial_x P = 0.5$ kPa/m and $u_0 = 1$ mm/s.

7

Conclusions

From the theoretical analysis made, the benchmarks performed and the results obtained, the main conclusion in this work is that the lattice-Boltzmann method may indeed be used to model electrokinetic systems. In [28], a coupling of the same equations as in this work is done. Also a lattice-Boltzmann approach for solving the equations is proposed. There are however several differences between what is done here and in [28]. First, different equilibrium distributions are used for the Nernst-Planck and Poisson's equations. In [28], a Navier-Stokes-like equilibrium including quadratic terms[1] are used for the Nernst-Planck equation and for the Poisson's equation different weights are used. Second, different implementations of the boundary conditions are used. Also, the no-flux boundary condition used in the Nernst-Planck equation is integrated into the fluid domain and rewritten as a Dirichlet condition. This is not very accurate, since the boundary condition only apply for the boundary and not in the interior of the domain. Further, some results are contradictory to the system modelled, e.g. it is stated that a 1:1 solution is modelled even though the computed, charge density is strictly positive. It must therefore be some source of positive or leak of negative ions present, a guess is that it is due to the incorrect boundary treatment for the ion flux at the channel walls.

The lack of scientific results on the lattice-Boltzmann method or results which sometimes seem a bit off, is a sign of the youth of the method and a rather great limitation when working with it. There are few "school books" with detailed explanations. Instead, scientific papers where all details sometimes are not written out explicitly have to be addressed.

When browsing work on modelling of electrokinetics, usually the Poisson-Boltzmann method is utilised. Sometimes even without reflection of whether it is applicable or not. From this work it is concluded that as the thickness of the double layer gets comparable to

[1] One of the "quadratic terms" is actually missing the square, this must be a typo. Otherwise the concentration is not obtained as the zeroth moment as stated. It also does not agree with the reference used for motivating the equilibrium distribution.

the dimensions of the system considered or if advection is present, the Poisson-Boltzmann approach may not be an accurate model. The underlying assumptions used in the derivation must be studied and assured to be fulfilled in the system under consideration.

In situations where the Poisson-Boltzmann approach fails, the more general method proposed in this work may be used instead. Few results of the more complicated systems modelled in this work was found and it is therefore difficult to determine a qualitative and quantitative correctness. Experimental results for this kind of system is very difficult to obtain, the detailed measurement of a velocity field in a system is not simple to determine. This is also the main reason why a computational approach is so much desired.

To get an efficient implementation of the method, topics from high-performance computing must be considered. The distribution function must be organised in memory to allow for good locality in the computation. Also unnecessary branches and data dependence should be avoided in the innermost loop. It is important to use a modern and uptodate compiler and it may be a good idea to tell the compiler to optimise the code as much as possible. The LBM algorithm is also very well suited for parallelisation. This is due to that there is no data dependence between nodes in the collision step.

Bibliography

[1] D. Li, Electrokinetics In Microfluidics, Interface Science and Technology Series, Academic Press, 2004.
URL http://books.google.se/books?id=1QkpluJqQegC

[2] D. Hlushkou, D. Kandhai, U. Tallarek, Coupled lattice-boltzmann and finite-difference simulation of electroosmosis in microfluidic channels, International Journal for Numerical Methods in Fluids 46 (5) (2004) 507–532.
URL http://dx.doi.org/10.1002/fld.765

[3] Q. Zou, X. He, On pressure and velocity boundary conditions for the lattice boltzmann bgk model, Physics of Fluids 9 (6) (1997) 1591–1598.
URL http://link.aip.org/link/?PHF/9/1591/1

[4] J. Wang, M. Wang, Z. Li, Lattice poisson–boltzmann simulations of electro-osmotic flows in microchannels, Journal of Colloid and Interface Science 296 (2) (2006) 729 – 736.
URL http://www.sciencedirect.com/science/article/pii/S0021979705009872

[5] S. Succi, M. Sbragaglia, S. Ubertini, Lattice boltzmann method, Scholarpedia 5 (5) (2010) 9507.

[6] D. Wolf-Gladrow, Lattice-Gas Cellular Automata and Lattice Boltzmann Models: An Introduction, no. v. 1725 in Lecture Notes in Mathematics, Springer, 2000.
URL http://books.google.se/books?id=cHcpWgxAUu8C

[7] M. Junk, A. Klar, L.-S. Luo, Asymptotic analysis of the lattice boltzmann equation, Journal of Computational Physics 210 (2) (2005) 676 – 704.
URL http://www.sciencedirect.com/science/article/pii/S0021999105002573

[8] Wolfram Alpha, Information content of the web - wolfram alpha (November 2012).
URL http://www.wolframalpha.com/input/?i=estimated+data+content+of+the+web

[9] M. Junk, Z. Yang, Assymptotic Analysis of Lattice Boltzmann Boundary Conditions, Journal of Statistical Physics 121 (2005) 3–35.

[10] T. Gebäck, A. Heintz, Neumann boundary conditions for the lattice Boltzmann advection diffusion equation, Article in press.

[11] H. Huang, M. Krafczyk, X. Lu, Forcing term in single-phase and shan-chen-type multiphase lattice boltzmann models, Phys. Rev. E 84 (2011) 046710.
URL http://link.aps.org/doi/10.1103/PhysRevE.84.046710

[12] X. Shan, H. Chen, Lattice boltzmann model for simulating flows with multiple phases and components, Phys. Rev. E 47 (1993) 1815–1819.
URL http://link.aps.org/doi/10.1103/PhysRevE.47.1815

[13] A. Caiazzo, M. Junk, M. Rheinländer, Comparison of analysis techniques for the lattice boltzmann method, Computers & Mathematics with Applications 58 (5) (2009) 883 – 897, <ce:title>Mesoscopic Methods in Engineering and Science</ce:title>.
URL http://www.sciencedirect.com/science/article/pii/S0898122109000935

[14] Intel, Microsoft .net* on intel xeon processor (November 2012).
URL http://software.intel.com/en-us/articles/microsoft-net-on-intel-xeont-processor

[15] Wikipedia, Classic risc pipeline — wikipedia, the free encyclopedia, [Online; accessed 11-December-2012] (2012).
URL http://en.wikipedia.org/w/index.php?title=Classic_RISC_pipeline&oldid=495044207

[16] Wikipedia, Cpu cache — wikipedia, the free encyclopedia, [Online; accessed 11-December-2012] (2012).
URL http://en.wikipedia.org/w/index.php?title=CPU_cache

[17] M. Håkansson, Spinodal decomposition with the lattice boltzmann method, Master's thesis, Chalmers University of Technology (2012).

[18] Open MPI, Open mpi, [Online; accessed 11-December-2012] (2012).
URL http://www.open-mpi.org/

[19] OpenMP, Openmp, [Online; accessed 11-December-2012] (2012).
URL http://openmp.org/wp/

[20] Palabos, Palabos wiki - Lattice Boltzmann in various languages, [Online; accessed 28-December-2012] (2012).
URL http://wiki.palabos.org/numerics:codes

[21] T. Georgiou, Realtime computational fluid dynamics simulations using the lattice boltzmann method, tJHSST Senior Research Project (2010).

[22] K. Mattila, J. Hyväluoma, J. Timonen, T. Rossi, Comparison of implementations of the lattice-boltzmann method, Computers & Mathematics with Applications 55 (7) (2008) 1514 – 1524, <ce:title>Mesoscopic Methods in Engineering and Science</ce:title>.
URL http://www.sciencedirect.com/science/article/pii/S0898122107006232

[23] P. Bailey, J. Myre, S. Walsh, D. Lilja, M. Saar, Accelerating lattice boltzmann fluid flow simulations using graphics processors, in: Parallel Processing, 2009. ICPP '09. International Conference on, 2009, pp. 550 –557.

[24] Free Software Foundation, Inc., GNU gprof, [Online; accessed 28-December-2012] (2012).
URL http://www.cs.utah.edu/dept/old/texinfo/as/gprof.html

[25] Valgrind Developers, Valgrind, [Online; accessed 28-December-2012] (2012).
URL http://valgrind.org

[26] Z. Chai, B. Shi, A novel lattice Boltzmann model for the Poisson equation, Applied Mathematical modeling 32 (2007) 2050–2058.

[27] C. Ren, D. Li, Electroviscous effects on pressure-driven flow of dilute electrolyte solutions in small microchannels, Journal of colloid and interface science 274 (1) (2004) 319–330.

[28] M. Wang, Q. Kang, Modeling electrokinetic flows in microchannels using coupled lattice boltzmann methods, Journal of Computational Physics 229 (3) (2010) 728–744.

[29] J. Zhang, G. He, F. Liu, Electro-osmotic flow and mixing in heterogeneous microchannels, Physical Review E 73 (5) (2006) 056305.

A

Code snippet

An example code defining Poiseuille flow driven by a constant force (pressure gradient) using the LBM library developed in this work.

```
#include <iostream>
#include "../LBM.h"

using namespace std;

int main(){
  int nx = 3, ny = 50, tMax = 1000;
  double w = 0.75;
  double c = 1.0;

  LatticeModel *lm = new Lattice2D(nx, ny);
  StreamD2Q9Periodic *sm = new StreamD2Q9Periodic();
  CollisionD2Q9BGKNSF *cm = new CollisionD2Q9BGKNSF();
  LBM *lbm = new LBM(lm, cm, sm);

  double **fx = allocate2DArray(ny, nx);
  double **fy = allocate2DArray(ny, nx);

  cm->setW(w);
  cm->setC(c);


  /* Set boundary conditions*/
  BounceBackNodes<CollisionD2Q9BGKNSF> *bbns =
          new BounceBackNodes<CollisionD2Q9BGKNSF>();
```

```
  bbns->setCollisionModel(cm);
  for(int i = 0; i < nx; i++){
    bbns->addNode(i, 0, 0);
    bbns->addNode(i, ny-1, 0);
  }
  lbm->addBoundaryNodes(bbns);

  /* Set force */
  for(int i = 0; i < nx; i++){
    for(int j = 0; j < ny; j++){
      fx[j][i] = 0.0001;
      fy[j][i] = 0.0;
    }
  }
  cm->setForce(fx, fy);

  /* Initialize solver */
  lbm->init();

  /* Main loop */
  for(int t = 0; t < tMax; t++){
    lbm->collideAndStream();
  }

  cm->dataToFile("bench_force_poi/");

  return 0;
}
```