Faculty Applied Sciences

Simulating the flow through a specific stacking pattern, using the lattice Bolzmann method

J.W. Versendaal, May 2014

BACHELOR OF SCIENCE THESIS

Jochem Versendaal

May 21, 2014

Supervisor

Dr. ir. M. Rohde

Committee member

Dr. ir. D. Lathouwers

Delft University of Technology

Faculty of Applied Sciences

Department of Radiation Science and Technology

Section of Nuclear Energy and Radiation Applications

Abstract

Using computational fluid dynamics complex fluid flows can be modeled, e.g. the coolant flow through randomly stacked pebbles which is present in pebble bed reactors. In this thesis a particular stacking pattern is analyzed using the D2Q9 lattice Boltzmann method created in Matlab. Prior to the analysis the Matlab script was benchmarked using the Poiseuille flow and the flow around the cross-section of a cylinder, after which the accuracy of the used method was determined.

The benchmarks showed that the used lattice Boltzmann code can be considered to be correct. The accuracy was determined using the data obtained from Matlab and a created excel program. It showed that the used lattice Boltzmann method was of the first order, which correspondents with literature. The analysis of the stacking pattern showed how fluid flows through the chosen stacking pattern. All these results are shown and elaborated in the chapter Results.

Table of contents

1 Introduction

1.1 The pebble bed reactor

A pebble bed reactor is a relatively new nuclear reactor. It is a high temperature reactor and belongs to the nuclear reactors in the generation IV initiative. The reactor contains spherical fuel elements named pebbles (hence the name pebble bed reactor) which pass in a descending stream, due to the gravity and coolant flow, through a core cavity bounded by reflector material. The reactor core contains thousands of these pebbles through with a gas flows as coolant. Control rods, which are neutron absorbers, are inserted in the vertical column to function as an extra absorber (Lohnert, 1976).

1.1.1 The development of the pebble bed reactor

The first suggestion for this type of reactor came in 1947 from Prof. Dr. F. Daniels, who also created the name "pebble bed reactor" as stated in (Daniels, 2013). However, in 1996 a patent was requested for the design of this new type of nuclear reactor. In the patent there was a detailed description of the geometries of the pebble bed reactor. Also the requestors gave a description on what to use as fuel and what kind of coolant should be used. On the second of November in 1999, Gerwin and Scherer, got their patent on the reactor design (Gerwin & Scherer, 1999).

1.1.2 The design

The pebble bed nuclear reactor comprises a V-shaped container. At the bottom of the container, in the center, there is an outlet where the used pebbles can be disposed of. At the inlet, positioned at the top of the container, new and unused pebbles are added into the container to keep the pebble flow in steady state and generating a constant power output. Authorities consider that the used pebbles, which are still radioactive, are stable enough to store directly in geological storage. This means that the used pebbles can be collected in a storage device at the pebble outlet and then they can be directly transported to the storage location.

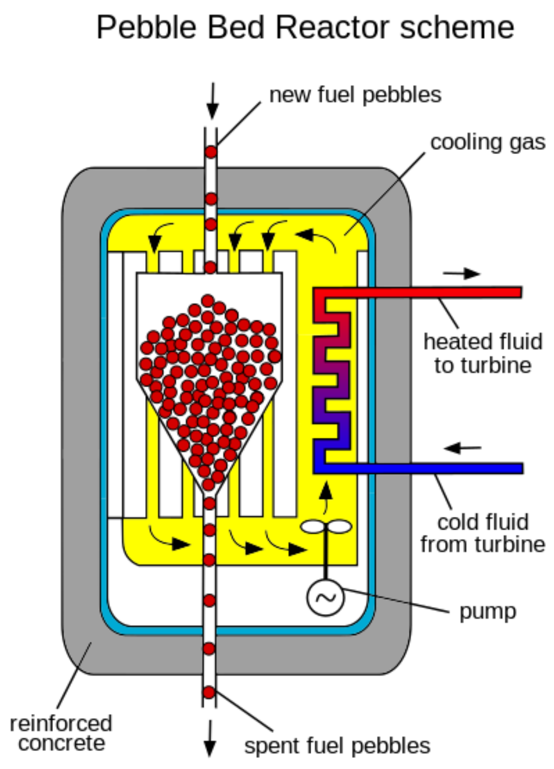

Figure 1: Schematic rending of the design of a pebble bed reactor. The new, unused pebbles are inserted at the top and the used pebbles are disposed of at the bottom. Through the core runs a secondary loop in which water is heated up to generate power (Pebble Bed Reactor scheme)

1.1.3 The fuel elements

Pebble bed reactors are fueled by spherical shaped pebbles which contain the fissile material. The pebbles are usually made of graphite and contain thousands of TRISO fuel particles. The pebbles are made of graphite because graphite is an element which has a moderating effect and, not unimportant, graphite can withstand very high temperatures. The TRISO, Tristructural-isotropic, fuel particles consist of a fissile material, such as Uranium-235, surrounded by a coated ceramic layer. The ceramic layer functions as containment for the fission product. Usually the ceramic layer is made of silicon carbide, also known as carborundum. The ceramic layer is made of grains of silicon carbide which are bonded together to form the ceramic material. Silicon carbide is not only used to fabricate the pebbles, but also for other applications requiring high endurance (Ballensiefen & Wolf, 1975).

1.1.4 The coolant

The pebble bed reactor is a gas cooled reactor, meaning that the coolant is a gas, having the advantage that it can be heated to high temperatures. Ideally, the coolant has a high thermal capacity, low viscosity, is non-toxic and does not react chemically with the fuel elements. Gasses are used because their viscosity is much lower than the viscosity of liquids and the gasses are cheap. It is also desirable that the gasses do not react chemically with the fuel elements, because this will give much problems considering the radioactive waste. A disadvantage of using gas as the coolant is that the thermal capacity and the density of gases is lower than the thermal capacity and density of liquids or salt, as one can find in a molten salt reactor. This results in a lower cooling capability due to the low density of the coolant.
Common gasses that are used as a coolant are helium, nitrogen and carbon dioxide. All these gasses have low viscosity, are cheap and do not undergo chemical reactions with the fuel elements, making them suitable as coolant for gas cooled nuclear reactors (Nicholls & Ion, 2003).

1.1.5 Advantages and disadvantages

Advantages of the pebble bed reactor are, first of all, that the pebble bed reactor is a high temperature reactor. The high temperature allows a turbine to extract more mechanical energy from the same amount of thermal energy. This is the reason why the power system uses less fuel per kWh, making it more efficient. The very high temperature allows the thermal efficiency to be higher than that of conventional nuclear reactors, operating at lower temperatures.
Another advantage is that, when the temperature increases too much, the rapid motion of the atoms in the fuel causes Doppler broadening. The Uranium-238, which is largely present in the reactor, is more likely to absorb fast neutrons at high temperatures, resulting in a decrease of the available low energy neutrons reducing the chance of fission. The Doppler broadening effect creates a negative feedback because when the temperature increases the number of available neutrons for fission decreases, resulting in a decrease in the power of the reactor. This build-in negative feedback makes the pebble bed reactor passively safe (Nicholls & Ion, 2003).
Yet another advantage of the reactor is that when the flow stops, the graphite ensures that the reactor core does not melt. This is because the graphite can withstand very high temperatures. The last major advantage is that the storage of the nuclear waste does not need to be treated before disposal, making the disposal cheaper and faster in comparison with other nuclear reactors. This last advantage comes with its own disadvantage. Although the radioactivity of the waste remains the same, the volume of the radioactive waste is much larger than other nuclear waste because the fissile product is contained in the pebbles.
The last disadvantage of the pebble bed reactor is that the coolant flow through the pebbles is not equally distributed. This results into a non-equally cooled reactor core, resulting in the occurrence of so called “hotspots”. These hotspots only occur on a micro-scale making it hard to simulate due to the high accuracy needed (Lee, Park, & Kim, 2007). Around these hotspots the coolant cannot flow properly and that specific point starts to heat up. The hotspots can damage the reactor when the temperature rises too much.

1.2 Computational Fluid Dynamics

In order to simulate the flow of a fluid through a specified region, computational fluid dynamics is the tool to use. Conservation of mass, conservation of energy and the second law of Newton are the fundamental principles forming the physical background of any fluid flow. These principles can be expressed most generally using partial differential equations. These partial differential equations need to be solved using numerical methods because the equations are too complex to solve analytically. Computational fluid dynamics provides a solution regarding the process of solving the partial differential equations numerically (Wendt, 2009).

1.2.1 The development of CFD

The Navier-Stokes equation, which will be discussed in the chapter Theory, forms the fundamental basis of close to all problems regarding fluid flow. The Navier-Stokes equation defines any single-phase fluid flow. The Navier-Stokes equation can be simplified using smart and legitimate assumptions. Removing terms describing viscous actions or actions regarding the turbulence of the flow are examples of assumptions that can be made to simplify the Navier-Stokes equation. When the right assumptions have been made, the Navier-Stokes equation can be linearized. Historically, this linearized model was the first model that was developed to solve the Navier-Stokes equation. This method is one-dimensional and the first two-dimensional methods, using conformal transformations, were developed in the 1930s (Thomson, 1973). During these years the computer became powerful enough to start modeling fluid flow. The first time that a computer was used to model the fluid flow was at Los Alamos National Labs. Francis H. Harlow was the leader of the group which performed the calculations on the computer. Harlow is widely considered as one of the pioneers of CFD, due to the fact that his group developed a variety of numerical methods to model two-dimensional fluid flow (Chung, 2010).
The first three-dimensional model was the Panel method and was published in 1967 by J. Hess and A.M.O. Smith (Hess, 1967). This method was based on the discretization of the surface of the three-dimensional object in panels, hence the name Panel method.

1.2.2 Complexity of turbulence and CFD

The complexity of computational fluid dynamics is related to the complexity of the Navier-Stokes equation. The Navier-Stokes equation is a nonlinear partial differential equation. This nonlinearity makes the problem very difficult to solve. In most cases an analytical solution is impossible, in which case supercomputers are used to give a solution for the problem.
However, when dealing with laminar flow, the complexity is reduced significantly. This will simplify the Navier-Stokes equation so it can be computed much faster than when turbulence has to be considered.
Looking closer at the turbulence of the flow one can find that the turbulence depends on the Reynolds number. The Reynolds number is a dimensionless parameter that relates inertial forces to viscous forces and is represented by (Mudde & van den Akker, 2008)

$$Re = \frac{\rho v L}{\mu} \tag{1}$$

In equation (1), ρ is the density of the fluid (SI units kg/m^3), v is the mean velocity of the fluid relative to the object (SI units m/s), L is the length of the channel (SI units m) and μ is the dynamic viscosity of the fluid (SI units kg/(m·s).
Laminar flow is characterized by smooth flows without lateral mixing. Laminar flows occur when the

Reynolds number is low, which can be realized when the velocities are low or when the viscosity is large, which both are affected by the geometry size. A fluid flow in which an object is moving is laminar when the Reynolds number is smaller than the critical Reynolds number. The critical Reynolds number gives a region from which the flow changes from laminar to turbulent.
Whenever the Reynolds number surpasses the critical Reynolds number, vortices start to appear and the flow becomes turbulent. Turbulence is the time dependent, chaotic behavior seen in almost all flows. In contrary to laminar flows turbulent flows are not smooth and the speed of the fluid at a specific point is continuously undergoing changes in both magnitude and direction, due to the occurrence of vortices. These effects make the solutions to the equations which are needed to model the fluid flow much more complex due to the fact that the assumptions are no longer valid which made the Navier-Stokes equation more easily solvable and CFD models are necessary to make a representative model of the fluid flow.

1.2.3 The lattice Boltzmann method

In order to simulate fluid flows through specified boundaries, the Navier-Stokes equation has to be solved. The lattice Boltzmann method is used to simulate the flow of a fluid using collision models (Benzi & Succi, 1992). Simulating the collision and streaming processes is the main part of the lattice Boltzmann method. It originated from the lattice gas automata, which is a simplified molecular dynamics model with a discretization of the space, time and velocities. Using the Stosszahl Ansatz will transform the LGA intro the lattice Boltzmann method, used in this thesis. Several models are made in order to solve problems of different dimension and different accuracy. The higher the accuracy and dimension the more computational time and memory is needed to solve the problem. A more elaborative explanation about the lattice Boltzmann method will be given in the chapter Theory.

2 Theory

2.1 The Navier-Stokes equation

The main part of computational fluid dynamics is the process of solving the Navier-Stokes equation. The Navier-Stokes equation is named after Claude-Louis Navier and George Gabriel Stokes and describes amongst others the motion of fluids (i.e. gasses, liquids and plasmas). The solution of the Navier-Stokes equation gives as solution a velocity field or flow field, a description of the velocity of the fluid at a specific point in space at a certain time.
For an incompressible medium, meaning that the volume does not change when the pressure changes, the Navier-Stokes equation is given by (Mudde & van den Akker, 2008)

$$\frac{\partial(\rho\vec{v})}{\partial t} + \nabla\cdot(\rho\vec{v}\vec{v}) = \nabla\cdot(\mu\nabla\vec{v}) + \sum F_{ext} \qquad (2)$$

In equation (2), ρ is the density of the fluid (SI units kg/m^3), v is the velocity of a specific point (SI units m/s), μ is the dynamic viscosity of the fluid (SI units kg/(m·s) and F_{ext} are the external body forces working on the fluid (SI units N). $\frac{\partial}{\partial t}$ is the partial time derivative and ∇ is the Nabla operator and is defined as $\nabla = \left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z}\right)$.

2.1.1 The derivation of the Navier-Stokes equation

To derive the Navier-Stokes equation from fluid dynamics first of all a control volume is needed. The control volume is chosen to be a cube with lengths dx, dy and dz in the x, y and z direction respectively. The cubic control volume is shown in figure 2.

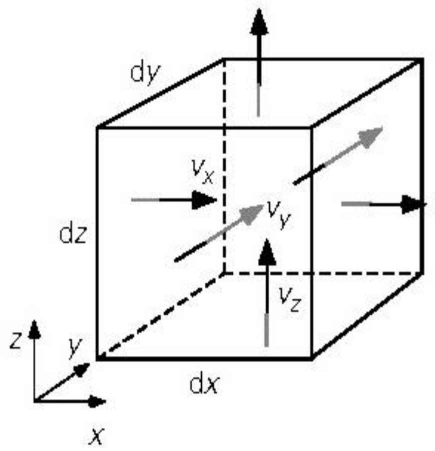

Figure 2: The control volume which is used to derive the Navier-Stokes equation using fluid dynamics (Mudde & van den Akker, 2008)

To derive the Navier-Stokes equation a microbalance has to be made for the mass and momentum. Shear stress, pressure and gravitation forces are part of the z-momentum balance. The gravitation force is an external body force working on the fluid inside the control volume and the shear stress and pressure are surface forces. All the derivations start off with the elaboration of the most general form of the balance, which is given by

$$\frac{d}{dt} = in - out + production \qquad (3)$$

2.1.1.1 Microbalance for the mass flow

To solve the balance for the mass flow one has to solve the non-steady state mass balance. The time dependent term is

$$\frac{dM}{dt} = \frac{d(\rho V)}{dt} = \frac{d(\rho \cdot dxdydz)}{dt} \tag{4}$$

When the assumption has been made that the volume is not time dependent, equation (4) can be simplified into

$$\frac{dM}{dt} = \frac{d\rho}{dt} dxdydz \tag{5}$$

To simplify equation (4) even more another assumption can be made, namely the assumption that the production of mass is zero inside the control volume. This results in the simplified balance

$$\frac{d\rho}{dt} dxdydz = in - out \tag{6}$$

To determine the in- and outflow of mass, one can look separately in the three directions. Firstly the net mass flow in the x direction will be calculated and from the obtained result the other directions will be derived. The mass flow in the x direction that enters from the left plane standing in the (0 1 0) direction, will be evaluated

$$in = [\rho v_x]_{x,y,z} \cdot dydz \tag{7}$$

Looking at the mass flow in the x direction that exits from the right plane standing in the (0 1 0) direction, one can derive an expression for the mass outflow

$$out = [\rho v_x]_{x+dx,y,z} \cdot dydz \tag{8}$$

Finding the net mass flow in the x direction is done by subtracting equation (7) from equation (8), resulting in an expression for the net mass flow in the x direction

$$net\ mass\ flow = \left([\rho v_x]_{x,y,z} - [\rho v_x]_{x+dx,y,z} \right) \cdot dydz \tag{9}$$

The derivation of an equation for the y and z direction will be analog to the derivation in the x direction. Once one has done this, one can find that the equations for all three directions will be

$$net\ mass\ flow\ x\ direction = -\frac{d}{dx}[\rho v_x] \cdot dxdydz$$

$$net\ mass\ flow\ y\ direction = -\frac{d}{dy}[\rho v_y] \cdot dxdydz$$

$$net\ mass\ flow\ z\ direction = -\frac{d}{dz}[\rho v_z] \cdot dxdydz \tag{10}$$

These equations can be inserted in equation (6) resulting in the three dimensional mass balance

$$\frac{d\rho}{dt} dxdydz = -\frac{\partial}{\partial x}[\rho v_x] \cdot dxdydz - \frac{\partial}{\partial y}[\rho v_y] \cdot dxdydz - \frac{\partial}{\partial z}[\rho v_z] \cdot dxdydz \tag{11}$$

Simplifying equation (11) can be done by dividing the equation by the volume of the control volume, which is $dxdydz$, resulting in

$$\frac{d\rho}{dt} = -\frac{\partial}{\partial x}[\rho v_x] - \frac{\partial}{\partial y}[\rho v_y] - \frac{\partial}{\partial z}[\rho v_z] \qquad (12)$$

When one looks closely at this equation, one can find this equation rather familiar. When changing the derivatives into partial derivatives and writing the long formula in a compact form using the Nabla operator $\nabla = \left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z}\right)$, one can find that the equation becomes

$$\frac{\partial\rho}{\partial t} = -\nabla \cdot (\rho\vec{v}) \qquad (13)$$

Equation (13) is an important formula known as the continuity equation. The continuity equation describes the transport of mass.

2.1.1.2 *Microbalance for the z-momentum flow*

Starting with the most general form for a momentum balance, for momentum in the z direction

$$\frac{d}{dt} = in - out + \sum F_z \qquad (14)$$

Rewriting the time dependent term results in

$$\frac{d\vec{p}_z}{dt} = \frac{d(\rho v_z)}{dt} dxdydz \qquad (15)$$

Now that the left-hand term is known, the right-hand term has to be evaluated. As done with the microbalance for the mass flow, one has to determine the net z-momentum flow in the control volume. To evaluate this the x direction will be evaluated first and the results will be used to determine the z-momentum flow in the other directions.
Looking at the z-momentum that enters from the left plane standing in the (0 1 0) direction, one can derive an expression for the momentum inflow

$$in = [v_x \rho v_z]_{x,y,z} \cdot dydz \qquad (16)$$

Analog to the mass outflow in the x direction, one can derive an expression for the z-momentum outflow in the x direction exiting from the right plane standing in the (0 1 0) direction

$$out = [v_x \rho v_z]_{x+dx,y,z} \cdot dydz \qquad (17)$$

Distracting equation (17) from equation (16) will result in the net z-momentum flow in the x direction. The result will be used to derive expressions for the z-momentum flow in all directions

$$net\ z - momentum\ flow\ x\ direction = \frac{d}{dx}[v_x \rho v_z] \cdot dxdydz$$

$$net\ z - momentum\ flow\ y\ direction = \frac{d}{dy}[v_y \rho v_z] \cdot dxdydz$$

$$net\ z - momentum\ flow\ z\ direction = \frac{d}{dz}[v_z \rho v_z] \cdot dxdydz \qquad (18)$$

2.1.1.2.1 The shear stress

The general formula for the shear stress, for a Newtonian fluid, has to be used in order to calculate the shear stress on the left plane, standing in the (0 1 0) direction. For a Newtonian fluid the shear stress is given by

$$\tau_{xz} = -\mu \frac{dv_z}{dx} \tag{19}$$

Using equation (19) for the general shear stress, an equation for the shear stress on the left plane standing in the (0 1 0) direction can be derived

$$\tau_{xz}(x, y, z) = \left[-\mu \frac{dv_z}{dx}\right]_{x,y,z} \cdot dxdz \tag{20}$$

In the same way the shear stress on the right plane standing in the (0 1 0) direction can be derived

$$\tau_{xz}(x + dx, y, z) = -\left[-\mu \frac{dv_z}{dx}\right]_{x+dx,y,z} \cdot dxdz \tag{21}$$

To obtain the net shear stress in the x direction, equation (21) has to be subtracted from equation (20). The equations for the shear stress in the other directions are obtained in the similar way with the only difference being the direction.

$$net\ shear\ stress\ in\ the\ x\ direction = \frac{d}{dx}\left[\mu \frac{dv_z}{dx}\right] \cdot dxdydz$$

$$net\ shear\ stress\ in\ the\ y\ direction = \frac{d}{dy}\left[\mu \frac{dv_z}{dy}\right] \cdot dxdydz$$

$$net\ shear\ stress\ in\ the\ z\ direction = \frac{d}{dz}\left[\mu \frac{dv_z}{dz}\right] \cdot dxdydz \tag{22}$$

2.1.1.2.2 The pressure force

Another part of the $\sum F_z$ term is the pressure force. The force due to pressure is given by the general formula

$$F_P = P \cdot A \tag{23}$$

In equation (23), *P* is the pressure (SI units Pa) and *A* is the area on which the pressure applies (SI units m^2). Using this general formula, the microbalance for the pressure force applying only in the z-direction can be derived. Analog to the other microbalances one can derive an equation for the pressure force on the bottom plane standing in the (0 0 1) direction, resulting in

$$F_P(x, y, z) = [P]_{x,y,z} \cdot dxdy \tag{24}$$

Analog to equation (24), the derivation of the pressure force applying to the top plane standing in the (0 0 1) direction can be derived

$$F_P(x, y, z + dz) = -[P]_{x,y,z+dz} \cdot dxdy \tag{25}$$

Using equation (24) and (25) to find the net pressure force in the z direction results in

$$net\ pressure\ force\ in\ the\ z\ direction = -\frac{dP}{dz} \cdot dxdydz \tag{26}$$

2.1.1.2.3 The gravitation force

The microbalance for the gravitation is rather simple once the assumption has been made that the gravity only applies in the z direction. The gravitation term, which again is part of the $\sum F_z$ term in equation (14), is given by

$$F_g = \rho g_z \cdot dxdydz \tag{27}$$

2.1.1.3 Adding the components

Now that all the components have been derived, equation (14) can be evaluated further by using the results of equations (18), (22), (26) and (27), resulting in

$$\frac{\partial(\rho v_z)}{\partial t} = -\frac{\partial}{\partial x}[v_x \rho v_z] - \frac{\partial}{\partial y}[v_y \rho v_z] - \frac{\partial}{\partial z}[v_z \rho v_z] + \frac{\partial}{\partial x}\left[\mu \frac{\partial v_z}{\partial x}\right] + \frac{\partial}{\partial y}\left[\mu \frac{\partial v_z}{\partial y}\right] + \frac{\partial}{\partial z}\left[\mu \frac{\partial v_z}{\partial z}\right] - \frac{dP}{dz} + \rho g_z \tag{28}$$

When equation (28) expands to a set of equations including x-momentum, y-momentum and z-momentum as well as the x, y and z components of the pressure force and gravitation, the following set of equations arise

$$\frac{\partial(\rho v_x)}{\partial t} = -\frac{\partial}{\partial x}[v_x \rho v_x] - \frac{\partial}{\partial y}[v_y \rho v_x] - \frac{\partial}{\partial z}[v_z \rho v_x] + \frac{\partial}{\partial x}\left[\mu \frac{\partial v_x}{\partial x}\right] + \frac{\partial}{\partial y}\left[\mu \frac{\partial v_x}{\partial y}\right] + \frac{\partial}{\partial z}\left[\mu \frac{\partial v_x}{\partial z}\right] - \frac{dP}{dx} + \rho g_x$$

$$\frac{\partial(\rho v_y)}{\partial t} = -\frac{\partial}{\partial x}[v_x \rho v_y] - \frac{\partial}{\partial y}[v_y \rho v_y] - \frac{\partial}{\partial z}[v_z \rho v_y] + \frac{\partial}{\partial x}\left[\mu \frac{\partial v_y}{\partial x}\right] + \frac{\partial}{\partial y}\left[\mu \frac{\partial v_y}{\partial y}\right] + \frac{\partial}{\partial z}\left[\mu \frac{\partial v_y}{\partial z}\right] - \frac{dP}{dy} + \rho g_y$$

$$\frac{\partial(\rho v_z)}{\partial t} = -\frac{\partial}{\partial x}[v_x \rho v_z] - \frac{\partial}{\partial y}[v_y \rho v_z] - \frac{\partial}{\partial z}[v_z \rho v_z] + \frac{\partial}{\partial x}\left[\mu \frac{\partial v_z}{\partial x}\right] + \frac{\partial}{\partial y}\left[\mu \frac{\partial v_z}{\partial y}\right] + \frac{\partial}{\partial z}\left[\mu \frac{\partial v_z}{\partial z}\right] - \frac{dP}{dz} + \rho g_z \tag{29}$$

Looking closer at these three equations, one can see similarities. These three equations can be combined using vector notations and the Nabla operator, resulting in only one vector-equation

$$\frac{\partial(\rho\vec{v})}{\partial t} = -\nabla\cdot(\rho\vec{v}\vec{v}) + \nabla\cdot(\mu\nabla\vec{v}) - \nabla\vec{P} + \rho\vec{g} \tag{30}$$

This equation is widely known as the Navier-Stokes equation (Mudde & van den Akker, 2008). However, in this thesis this equation is not directly used. The simulations have been made using the lattice Bolzmann method which will be explained in the next chapter.

2.2 The lattice Boltzmann method

The lattice Boltzmann method originally stems from lattice gas automata (Chen & Doolen, 1998) which was proposed for the first time in 1976 (Hardy & de Pazzis, 1976). Using the Stosszahl Ansatz the lattice gas automata was transformed into the lattice Boltzmann method as used in this thesis. The LBM is capable of solving complex fluid flow problems in different dimensions, because the lattice Boltzmann method uses different models for different problems. The models can be one, two or three-dimensional and within each dimension several accuracies can be used.

2.2.1 Lattice gas automata

In a lattice gas automata the particles move on a grid and will mutually interact according to general rules. LGA is capable of simulating the flow of fluids (i.e. fluid dynamics). There are several models to compute the LGA. The basic LGA model uses a two-dimensional square grid on which particles propagate from node to node in a specific direction (in the basic LGA method four different directions are allowed) and can interact with other moving particles. This interaction only occurs with particles on the nodes next to it, the so called next neighboring particles. For all of the particles in the grid the general rules (i.e. conservation of mass and momentum) apply. Another exclusion is that each node can hold only one particle moving in one of the four directions (Rohde, 2009).

2.2.2 The lattice Boltzmann method

When the fluid flow has been calculated using the LGA, one can see that it is not accurate due to the statistical noise caused by the discretization of the space, time and velocities. One way to solve this is by introducing more grid nodes, resulting in a much finer grid and a more accurate velocity profile. However, when using more grid nodes the computational memory increases significantly and the computational time becomes much more as well.
The usage of Boltzmann's so called Stosszahl Ansatz, which is an assumption of molecular chaos, will be a much computationally cheaper solution (Rohde, 2009). The usage of the Stosszahl Ansatz results directly in the following equation

$$N_i(\vec{r} + \vec{v}_i \Delta t, t + \Delta t) = N_i(\vec{r}, t) + \Omega_i(N_i) \tag{31}$$

This results in a loss of detail of the dynamics of each separate particle, which saves computational time. For some cases the loss of detail can be a disadvantage. However, for fluid dynamics applications this assumption is a valid one since the scale of the flow is much bigger than the molecular scale.

The lattice Boltzmann method is a method based on the lattice gas automata. The LBM is capable of solving complex three-dimensional problems containing fluid dynamics.
The main equation when using the lattice Boltzmann method is given by equation (31) (Rohde, 2004)

$$N_i(\vec{x} + \vec{c}_i \Delta t, t + \Delta t) = N_i(\vec{x}, t) + \Omega_i\big(N_i(\vec{x}, t)\big)$$

in which the collision operator $\Omega_i\big(N_i(\vec{x}, t)\big)$ has been defined as

$$\Omega_i\big(N_i(\vec{x}, t)\big) = -\frac{1}{\tau}\big[N_i(\vec{x}, t) - N_{i,eq}(\vec{x}, t)\big] \tag{32}$$

The collision factor $\Omega_i\big(N_i(\vec{x}, t)\big)$ is required to satisfy the law of conservation of total mass and total momentum at each lattice node. Hence, it holds that the following rule is applicable for Ω_i

$$\textstyle\sum_i \Omega_i = 0 \tag{33}$$

Looking back on equation (32), τ stands for the relaxation time (SI units s) which is a measure for how fast the collisional operator relaxes from the actual to the equilibrium configuration. The viscosity is related to the relaxation time as well as the particle velocity in a certain direction according to

$$\upsilon = (\tau - 0.5){c_i}^2 \tag{34}$$

In equation XX, υ is the kinematic viscosity, τ is the relaxation time and c_i is a set of variables describing the particle velocity. The pressure and density are also related with each other. This relationship is represented by

$$\mathrm{p} = \rho {c_i}^2 \tag{35}$$

Inserting equation (32) into equation (31) gives the complete equation for the lattice Boltzmann method.

$$N_i(\vec{x} + \vec{c}_i\Delta t, t + \Delta t) = N_i(\vec{x}, t) - \frac{1}{\tau}\left[N_i(\vec{x}, t) - N_{i,eq}(\vec{x}, t)\right] \qquad (36)$$

c

c_i (i=1,…,M) is a set of variables describing the particle velocity, where M is the number of directions of the particle velocities at each node.
N_i (i=1,…,M) is a set of variables describing the particle occupation, again with M being the number of directions. The number of variables depends on the type of model one uses.

2.2.3 The DnQm models

The problems that one can encounter are of different dimension and complexity. There are several lattice Boltzmann models which can be used that are suitable for each dimension. These models are all of the form DnQm, in which n stands for the dimension and m stands for the number of speeds of the model. These speeds are the numbers of discrete velocity vectors used to create the grid. In table 1 different models are given for two and three dimensions (Mohamad, 2011).

Table 1: Four different models that can be used to solve different problems, depending on the dimension of the problem and the required accuracy

Model used	Dimension	Number of speeds
D2Q7	Two-dimensional	7
D2Q9	Two-dimensional	9
D3Q15	Three-dimensional	15
D3Q19	Three-dimensional	19

The D2Q7 and D2Q9 models are used to solve two-dimensional problems and the D3Q15 and D3Q19 models are used to solve three-dimensional problems. The higher the value after Q, the more accurate the model is. Although the D2Q7 model is important for historical reasons (the D2Q7 model is the direct descendant of the lattice gas cellular automata), the D2Q9 model is used more often due to the higher accuracy of this model.

2.2.4 The equilibrium distribution

Going on with the evaluation of the lattice Boltzmann equation, one can see that equation (36) holds the equilibrium term $N_{i,eq}(\vec{x}, t)$ which is a Taylor expanded version (up to second order) of the Maxwell-Boltzmann equilibrium distribution. The result of this derivation is the equilibrium term which will be used in equation (36) (Benzi & Succi, 1992)

$$N_{i,eq}(\vec{x}, t) = w_i\rho[1 + A \cdot (e_i \cdot \vec{c}) + B \cdot (e_i \cdot \vec{c})^2 + C \cdot \vec{c}^2] \qquad (37)$$

in which *A*, *B* and *C* are constants determined by the conservation laws, ρ is the density of the fluid (SI unit kg/m^3) and w_i are the dimensionless weights. The values for w_i depend on the model that is used. For several models table 2 gives the right weights, depending on the integer i, which represents the direction of the velocity of a grid node. This is made visual in figure 3.

Table 2: An overview of the weight terms of several DnQm methods in two and three dimensions

Model	Weights
D2Q7	$w_i = \left\{ \begin{array}{l} \frac{1}{2}\ for\ i = 0 \\ \frac{1}{12}\ for\ i = 1,..,6 \end{array} \right\}$
D2Q9	$w_i = \left\{ \begin{array}{l} \frac{4}{9}\ for\ i = 0 \\ \frac{1}{9}\ for\ i = 1,..,4 \\ \frac{1}{36}\ for\ i = 5,..,8 \end{array} \right\}$
D3Q15	$w_i = \left\{ \begin{array}{l} \frac{2}{9}\ for\ i = 0 \\ \frac{1}{9}\ for\ i = 1,..,6 \\ \frac{1}{72}\ for\ i = 7,..,14 \end{array} \right\}$
D3Q19	$w_i = \left\{ \begin{array}{l} \frac{1}{3}\ for\ i = 0 \\ \frac{1}{18}\ for\ i = 1,..,6 \\ \frac{1}{36}\ for\ i = 7,..,18 \end{array} \right\}$

2.2.5 The D2Q9 model

In this thesis the D2Q9 model is used for the computational calculations. For this reason, this model will be explained in detail. The D2Q9 model is a two-dimensional lattice Boltzmann model on a square grid. As already explained, the D2Q9 model is used with nine discrete velocity vectors. These velocity vectors can be represented by a vector $\vec{c}(i,r)$ in which i stands for the discrete velocity vector and *r* stands for the Cartesian coordinates. These discrete velocity vectors can be expressed by a set of two-dimensional vectors represented as used in equation (36).

The D2Q9 model is shown in figure 3. Each of the vectors correspond with the discrete velocity vectors. Although only eight vectors are shown, the origin of the vectors counts as a vector as well, resulting in nine vectors total.

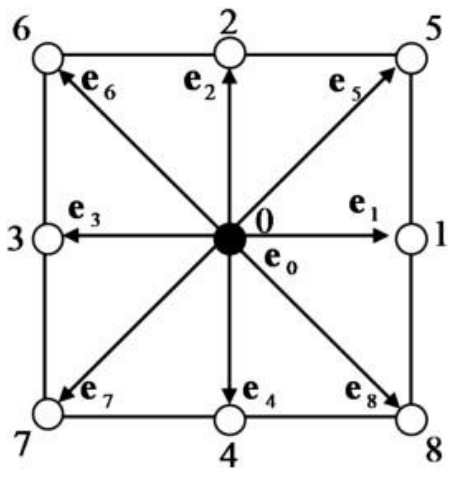

Figure 3: The nine discrete velocity vectors of the D2Q9 model

2.2.6 Different boundary techniques

All the different boundary techniques have the same boundary conditions, namely the no-slip boundary condition and the periodicity of the grid. The no-slip condition is applicable to fixed objects as well as to moving objects. The no-slip condition states that the fluid velocity on the boundaries is the same as the velocity of the object. Hence, when the object is fixed the fluid velocity on its surface is zero and when the object is moving the fluid velocity is the same as the object's velocity. In this thesis two boundary techniques will be discussed, namely the half-way bounce back and the full-way bounce back (Rohde, 2004).

The periodicity of the grid is used in order to make the finite grid infinite. This is done by stating that both ends of the grid are exactly the same. When both ends of the grid are the same the grid can be enlarged by setting two grids behind each other. This can be done for an infinite amount making the grid infinite.

2.2.6.1 The half-way bounce back method

The half-way bounce back technique requires the surface of the object to be exactly in the middle of two grid nodes. The half-way bounce back method can be explained using figure 4.

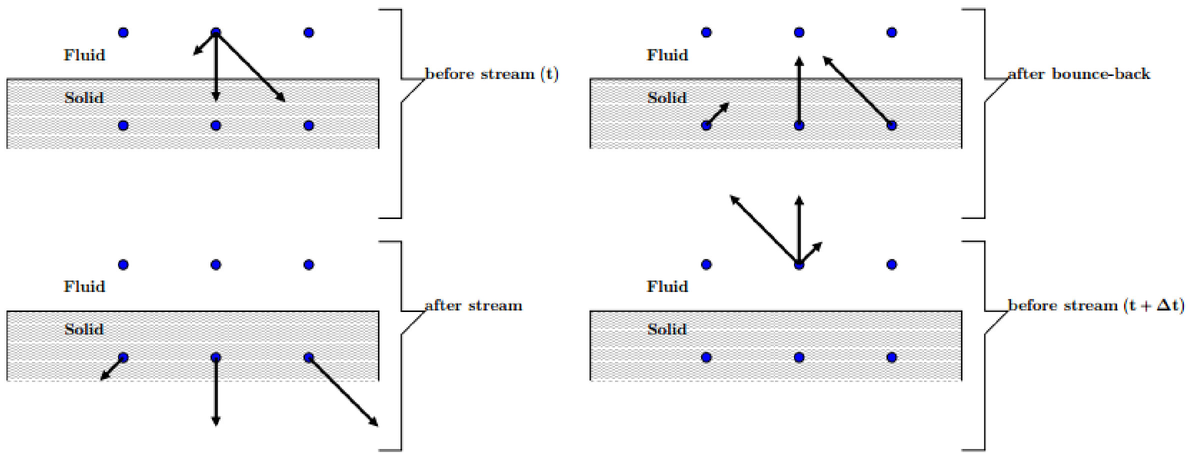

Figure 4: (Read from top-left to bottom-left to top-right to bottom-right) A simplified way to express the half-way bounce back method for the D2Q9 model (Sukop & Thorne, 2006). The arrows represent the momentum of the particle in different directions

In figure 4 the first thing that is noticeable is that the solid boundary is exactly between two grid nodes, thus the half-way bounce back method is applicable. The next thing that one can consider is that the half-way bounce back is a four-step process, as shown in figure 4. In this case the first step is showing the velocity of the fluid at each node in each of the directions, depending on the model being used. The next step is to apply the propagation step and neglecting the solid, thus the discrete velocity vectors propagate as if there was no solid in their way. The third step is to apply the bounce back method for the propagated velocity vectors to the nodes which lie within the solid. The fourth and final step is to return the bounced back velocity vectors to the nodes before the propagation step. These four consecutive steps can all be summarized as only one step. However, the usage of these four steps demonstrates the half-way bounce back method better.

2.2.6.2 The full-way bounce back method

For the full-way bounce back method to be applicable, the solid must be precisely on the grid nodes. When a discrete velocity vector lies on the surface of the solid, the direction in which it was going is inverted as shown in figure 5.

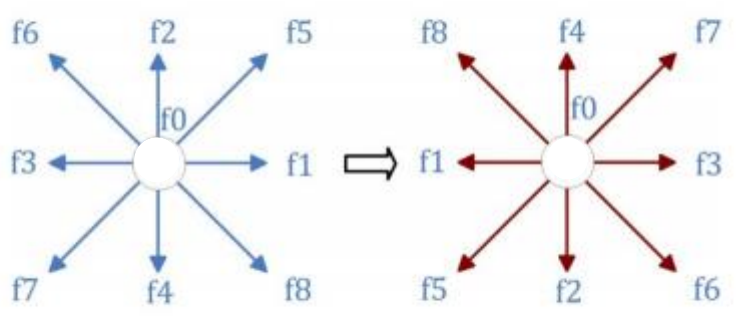

Figure 5: An overview of the reversed discrete velocity vectors when using the full-way bounce back method. The left D2Q9 scheme is before the bounce back has been applied, the right scheme is after the bounce back step has been applied. The white circle in the middle represents a grid node

As one can see in figure 5 the velocity directions swap with its associated opposite. This happens when a discrete velocity vector points into a solid. When the propagation step is applied all the directions encountering a boundary are being flipped. The white circle in figure 5 represents a node which is surrounded by boundaries. The left picture represents the propagation step before the collision step has been applied. All the vectors lie within a boundary resulting in a reversing of the vectors in all directions. The right picture represents the vectors after the collision step has been applied.

3 The computational program

In this research the lattice Boltzmann method has been used to simulate the fluid flow through a specific stacking pattern of boundaries. This is done by creating a program to execute the lattice Boltzmann equation in Matlab. In this chapter an overview will be given of the setup of the created program (i.e. a computational flow chart) as well as a more detailed explanation of the most essential Matlab functions used in this program. The entire code will be given in the appendix.

3.1 The computational flow chart

The computational program consists of four major parts which will be explained further. The first step starts with the creation of the grid which is done by defining the dimensions of the grid, i.e. *lx* and *ly*, being the lengths of the square grid. Next, some essential parameters are set a priory of the main program. These are the relaxation time, viscosity, the density and the weight factors, as given in table 2. Initializing the LB-matrix is the last part of this step and is done by creating the LB-matrix using the internal Matlab function repmat (the LB-matrix is a matrix having all the lattice Boltzmann functions in it). Repmat(A,[M N P ...]) tiles the array A to produce a multidimensional array composed of copies of A. After the LB-array has been created it has to be initialized which can be done in several ways. The one used in this thesis is by stating that the LB-array is equal to the equilibrium terms.

The second part of the code is to generate the boundaries. This is done by creating a Boolean variable in which a 1 means that it is a boundary and a 0 means that is not a boundary. An advantage of this is that it enables the opportunity to create any boundary one desires.

The third step is to execute the LBM in a while loop to create a velocity profile. The first part of the while loop includes the creation of the nine LB-functions which are initialized in step 1.Thereafter the macroscopic parameters are calculated, being the x- and y-velocity and the density. Those are needed to determine the nine equilibrium terms. The last part of the while loop is to enlarge the time by one unit so the while loop runs again until the maximum steps are used.

The fourth and final step is to create a vector field which represents the velocity field. This is done by using the internal Matlab function quiver. Quiver(X,Y,U,V) plots velocity vectors as arrows with components (u,v) at the points (x,y).

All these steps are shown in short in algorithm 1.

Algorithm 1: Short overview of the four steps

```
Step 1
    - Create the grid
    - Give the parameters
    - Initialization of the LB-matrix
Step 2
    - Create the solid boundaries
Step 3
    - Start main while loop
    - Determine the nine LB-functions
    - Determine macroscopic parametes
    - Determine the nine equilibrium terms
    - End main while loop
Step 4
    - Plot vector field
```

Algorithm 1 gives only a short overview of the different steps. Algorithm 2 gives a more in depth view of how the Matlab code looks like.

Algorithm 2: The computational flow chart of the LBM

```
Step 1
lx = ly = 100  % Create 2D grid
tau = 0.6       % Define relaxation time
rho = 1         % Define density
w1 = 4/9    % }
w2 = 1/9    % } Create the weight functions
w3 = 1/36   % }
f = repmat(1, [lx ly 9])    % Create LB-matrix
f = feq                     % Initialize the LB-matrix
Step 2
boundary = zeros(lx,ly)  % Create empty boundary matrix
for x = ..; y = ..;
boundary(x,y) = 1          % Set certain grid nodes to a solid boundary
end end
Step 3
while t < tmax     % Perform the simulation in time
f(:,:,q) = f(…)       % Create the nine LB-functions
rho = sum(f,3)    % Determine density
ux = sum(f(:,:,[+x direction]),3) – sum(f(:,:,[-x direction]),3)
uy = sum(f(:,:,[+y direction]),3) – sum(f(:,:,[-y direction]),3)
uz = sum(f(:,:,[+z direction]),3) – sum(f(:,:,[-z direction]),3)
feq(:,:,q) = w(q)*rho*(1+3($e_i \cdot \vec{c}$)+9/2($e_i \cdot \vec{c}$)$^2$ -3/2(ux$^2$+uy$^2$+uz$^2$))
Apply the needed boundary conditions here
time = time+1  % next time step
end while       % end time loop
Step 4
quiver(ux(1:lx,:),uy(1:ly,:))    % Create vector plot
```

An advantage that can be noticed in algorithm 2 is that the grid dimensions can easily be changed. The larger the grid the more accurate the simulation becomes. However, when one has many grid nodes the computational memory and time become larger than a normal computer can handle. Another advantage is that the solid boundaries are set by hand. This means that one can put a solid boundary anywhere in the grid where one desires. This enables the option of analyzing structures as complex as one can make them.

4 Benchmarks

When a computational program has been written one has to be sure that it gives physical answers. This is done by benchmarking the program by letting the program execute problems which can be evaluated analytically. In this case two benchmarks have been done. The first is the Poiseuille flow and the second is the flow around the cross-section of a cylinder using Stokes' stream function.

4.1 Poiseuille flow

The first benchmark that has been executed is the Poiseuille flow. This is a laminar flow through two solid boundaries. First the analytical solution will be given, after which the results will be compared with the results of the Matlab program.

4.1.1 The analytical solution

To obtain the velocity profile of a Poiseuille flow the general Navier-Stokes equation for an incompressible medium should be used, being equation (30)

$$\rho\frac{\partial\vec{u}}{\partial t}+\rho\vec{u}\cdot\nabla\vec{u}=-\nabla\vec{p}+\mu\nabla^2\vec{u}+\vec{F}_{ext} \qquad (30)$$

First of all when dealing with laminar flow, the Navier-Stokes equation can be simplified into

$$\nabla\vec{p}=\mu\nabla^2\vec{u}+\vec{F}_{ext} \qquad (38)$$

In this particular case there is no external force, meaning that the $\vec{F}_{ext}$ term is zero. Also due to the fact that it is a symmetric problem, equation (38) can be simplified even further into

$$\frac{\partial\vec{p}}{\partial x}=\mu\frac{\partial^2\vec{u}}{\partial y^2} \qquad (39)$$

Solving this equation for $\vec{u}$ is an easy differential equation and the result is

$$\vec{u}=\frac{1}{2\mu}\frac{\partial\vec{p}}{\partial x}y^2+C_1y+C_2 \qquad (40)$$

Boundary conditions are needed to solve C_1 and C_2
Boundary condition 1: $\vec{u}(0)=0$ gives $C_2=0$

Boundary condition 2: $\vec{u}(D)=0$ gives $C_1=-\frac{1}{2\mu}\frac{\partial\vec{p}}{\partial x}D$

Inserting the previously found C_1 and C_2 in the equation for $\vec{u}$ and simplifying the result gives

$$\vec{u}=\frac{-1}{2\mu}\frac{\partial\vec{p}}{\partial x}(Dy-y^2) \qquad (41)$$

In this equation y is the distance from the bottom plate to a point between the plates, and D is the distance between the bottom and the top plate.
Looking at the equation one can see that the maximum velocity occurs precisely in the middle of the two plates at $y=\frac{D}{2}$. This is done by solving $\frac{\partial\vec{u}}{\partial y}=0$. At the solid boundaries the velocity is zero. This is in line with the no-slip boundary condition.

4.1.1.1 Results of the analytical Poiseuille flow

The results of the analytically determined Poiseuille flow are represented in figure 6 which has been made using a Matlab code and can be found in the appendix. The velocity in the upward direction is defined as positive making the velocity in figure 6 positive as well. In practice this means that the pressure gradient is positive.

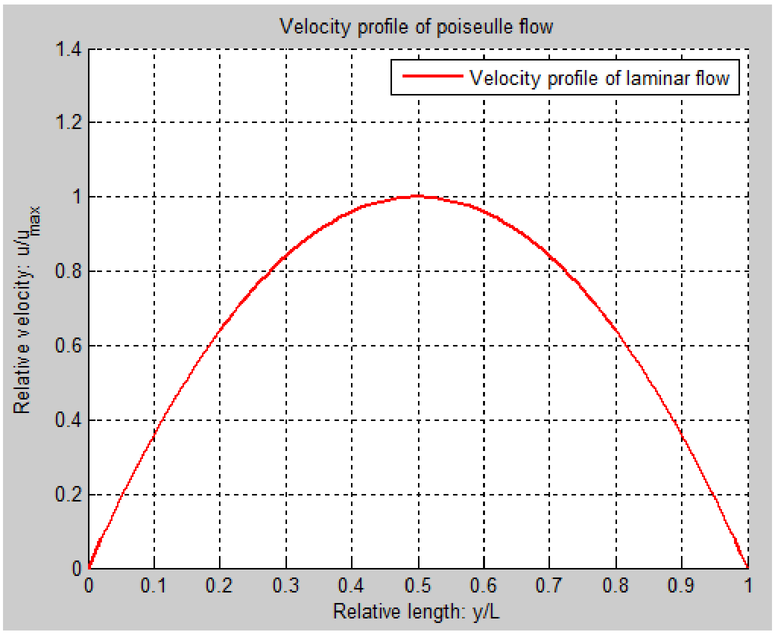

Figure 6: Visualization of the analytical solution of the Poiseuille flow

In the Matlab program the inflow and outflow boundaries have been made periodic. This has been done by stating that what flows equals that what flows out, making the boundaries periodic.

4.1.2 The computationally determined solution

For the Matlab script to give the Poiseuille flow the boundaries have to be set properly. Since the D2Q9 model is used the boundaries are set at both ends of the horizontal axis. The script exports the obtained data to an excel sheet where the values are saved. The data from excel are imported in Matlab where further calculations are executed.

4.1.3 Comparing the two results

In a secluded Matlab program the exported values of the Poiseuille flow, determined with the lattice Boltzmann method, are compared with the analytical solution. The analytical solution is plotted as a graph and the determined values are plotted in the plot as x-marks. As visible in figure 7 the analytical solution is plotted as a blue line and the red x-marks are the determined values.

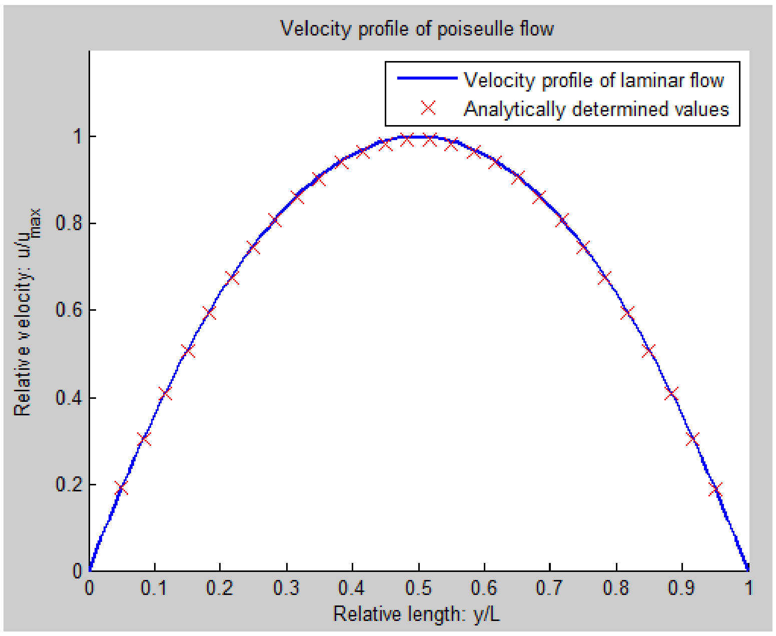

Figure 7: The analytical solution together with the determined values of the Poiseuille flow using the half-way bounce back method

Figure 7 only gives information of the chosen points, more points means that a more accurate comparison can be made. In this case 28 points are chosen to be analyzed. What can be noticed directly is that the x-marks do not begin and end at the relative length of zero and 1 respectively. This is because the Matlab script uses the half-way bounce back method (as explained before) and this method gives only values between grid nodes.

What can be concluded from the 28 points is that their values correspondent to the analytical solution. What can be concluded from this result is that, in case of the Poiseuille flow, the Matlab script gives logical values and is representable for the reality. However, with only one benchmark the reliability of the result is still questionable, hence a second benchmark is carried out.

4.2 Flow around the cross-section of a cylinder

The second benchmark will be a more complex one to calculate analytically as well as to compute in Matlab. It will be the flow around the cross-section of a cylinder which is centered in the middle of the grid without solid walls. First the analytical solution will be given, after which the results will be compared with the results of the Matlab script.

4.2.1 The analytical solution

Whenever an infinite long cylinder, with radius a, is moving through an infinite stationary fluid with a constant velocity U, one can solve the Navier-Stokes equation to obtain the velocity profile. Using the Stokes stream function ψ, the problem of solving the Navier-Stokes equation is reduced to solving a partial differential equation for ψ. The problem is simplified even more by choosing the proper boundary conditions. This will reduce the problem to that of solving an ordinary differential equation in order to obtain ψ.

4.2.1.1 Solving the Navier-Stokes equation

The Navier-Stokes equation for an incompressible fluid is given by equation (30)

$$\rho\left(\frac{\partial\vec{u}}{\partial t}+(\vec{u}\cdot\nabla)\vec{u}\right)=-\nabla p+\mu\nabla^2\vec{u}+\vec{F}_{ext} \qquad (30)$$

in which ρ is the density of the fluid, $\vec{u}$ is the velocity of the fluid, p is the pressure, μ is the dynamic viscosity of the fluid and $\vec{F}_{ext}$ is a term which is nonzero when there are external body forces. The assumption that the Reynolds number is very low results in a reduction of the Navier-Stokes equation due to the fact that for low Reynolds numbers the term $(\vec{u} \cdot \nabla)\vec{u}$ may be disregarded. The equation is simplified further because a stationary solution has to be found. The simplified version of the Navier-Stokes equation when the Reynolds number approaches zero is

$$\nabla p = \mu \nabla^2 \vec{u} \tag{42}$$

The rule of Laplace states that the Laplace operator can be rewritten as

$$\nabla^2 \vec{u} = \nabla \times (\nabla \times \vec{u}) - \nabla(\nabla \cdot \vec{u}) \tag{43}$$

Combining equation (3) and the continuity condition which says that $\nabla \cdot \vec{u} = 0$, equation (42) can be reduced to

$$\nabla p = -\mu \nabla \times (\nabla \times \vec{u}) = -\mu \nabla \times \vec{\Omega} \tag{44}$$

in which the vector $\vec{\Omega}$ has been introduced. The vector is given by $\vec{\Omega} = \nabla \times \vec{u}$. Because the problem is symmetrical in the axis, an axisymmetric velocity field will be sought in the form of

$$\vec{u} = u_r(r,\theta)\hat{e}_r + u_\theta(r,\theta)\hat{e}_\theta \tag{45}$$

4.2.1.2 *Introducing the Stokes stream function*

The divergence $\nabla \cdot \vec{u}$ for an axisymmetric velocity field in spherical coordinates is given by

$$\nabla \cdot \vec{u} = \frac{1}{r^2}\frac{\partial}{\partial r}(r^2 u_r) + \frac{1}{r \sin\theta}\frac{\partial}{\partial \theta}(u_\theta \sin\theta) \tag{46}$$

Using the continuity equation and the introduction of the Stokes stream function *ψ(r,ϑ)* will give an expression for u_r and u_θ

$$u_r = \frac{1}{r^2 \sin\theta}\frac{\partial \psi}{\partial \theta}$$

$$u_\theta = \frac{-1}{r \sin\theta}\frac{\partial \psi}{\partial \theta} \tag{47}$$

Equation (47) shows that the velocity profile is only dependent on the stream function. In order to determine the stream function from the Navier-Stokes equation one has to calculate

$$\vec{\Omega} = \nabla \times \vec{u} = \Omega_r \hat{e}_r + \Omega_\theta \hat{e}_\theta + \Omega_\varphi \hat{e}_\varphi \tag{48}$$

Since $\vec{u}$ is axisymmetric the *r* and *ϑ* components are zero leaving only the *φ* component.

$$\begin{aligned} \Omega_\varphi &= \frac{1}{r}\left(\frac{\partial}{\partial r}(r u_\theta) - \frac{\partial u_r}{\partial \theta}\right) \\ &= \frac{1}{r}\left[\frac{\partial}{\partial r}\left(\frac{-1}{\sin\theta}\frac{\partial \psi}{\partial r}\right) - \frac{\partial}{\partial \theta}\left(\frac{1}{r^2 \sin\theta}\frac{\partial \psi}{\partial \theta}\right)\right] \\ &= \frac{-1}{\sin\theta}\left[\frac{\partial^2 \psi}{\partial r^2} + \frac{\sin\theta}{r^2}\frac{\partial}{\partial \theta}\left(\frac{1}{\sin\theta}\frac{\partial \psi}{\partial \theta}\right)\right] \\ &= \frac{-1}{\mathrm{r}\sin\theta}E^2\psi \end{aligned} \tag{49}$$

in which E^2 is a differential operator and is defined as

$$E^2 = \frac{\partial^2}{\partial r^2} + \frac{\sin\theta}{r^2}\frac{\partial}{\partial\theta}\left(\frac{1}{\sin\theta}\frac{\partial}{\partial\theta}\right) \tag{50}$$

Inserting this equation in equation (44) results in

$$\begin{aligned}\nabla p &= -\mu\nabla\times(\nabla\times\vec{u}) = -\mu\nabla\times\vec{\Omega}\\ &= \frac{-\mu}{\mathrm{r}\sin\theta}\left(\frac{\partial}{\partial\theta}\left(\frac{-1}{r}E^2\psi\right)\right)\hat{e}_r + \frac{\mu}{r}\left(\frac{\partial}{\partial r}\left(\frac{-1}{\sin\theta}E^2\psi\right)\right)\hat{e}_\theta\\ &= \frac{\mu}{\mathrm{r}^2\sin\theta}\frac{\partial}{\partial\theta}(E^2\psi)\hat{e}_r - \frac{\mu}{\mathrm{r}\sin\theta}\left(\frac{\partial}{\partial r}(E^2\psi)\right)\hat{e}_\theta\end{aligned} \tag{51}$$

Due to the fact that $\nabla p = \frac{\partial p}{\partial r}\hat{e}_r + \frac{1}{r}\frac{\partial p}{\partial\theta}\hat{e}_\theta + \frac{1}{\mathrm{r}\sin\theta}\frac{\partial p}{\partial\varphi}\hat{e}_\varphi$ one can obtain two differential equations for the pressure

$$\begin{aligned}\frac{\partial p}{\partial r} &= \frac{\mu}{\mathrm{r}^2\sin\theta}\frac{\partial}{\partial\theta}(E^2\psi)\\ \frac{\partial p}{\partial\theta} &= -\frac{\mu}{\mathrm{r}\sin\theta}\left(\frac{\partial}{\partial r}(E^2\psi)\right)\end{aligned} \tag{52}$$

When these two equations are cross-differentiated these equations become

$$\begin{aligned}\frac{\partial}{\partial\theta}\left(\frac{\partial p}{\partial r}\right) &= \frac{\mu}{r^2}\frac{\partial}{\partial\theta}\left(\frac{1}{\sin\theta}\frac{\partial}{\partial\theta}(E^2\psi)\right)\\ \frac{\partial}{\partial r}\left(\frac{\partial p}{\partial\theta}\right) &= -\frac{\mu}{\sin\theta}\frac{\partial^2}{\partial r^2}(E^2\psi)\end{aligned} \tag{53}$$

These two cross-differentiations must be equal, resulting in

$$\frac{\partial^2}{\partial r^2}(E^2\psi) + \frac{\sin\theta}{r^2}\frac{\partial}{\partial\theta}\left(\frac{1}{\sin\theta}\frac{\partial}{\partial\theta}(E^2\psi)\right) = E^2(E^2\psi) = 0 \tag{54}$$

Inserting the formula for the differential operator E^2 in equation (54) yields

$$\left[\frac{\partial^2}{\partial r^2} + \frac{\sin\theta}{r^2}\frac{\partial}{\partial\theta}\left(\frac{1}{\sin\theta}\frac{\partial}{\partial\theta}\right)\right]^2\psi = 0 \tag{55}$$

Solving this equation requires boundary conditions. For this problem two boundary conditions are imposed. The first is the no-slip condition on the surface of the cylinder resulting in $\vec{u}(a,\theta) = 0$. The second boundary condition says that the flow far from the cylinder has a constant velocity *U*.

The no-slip condition implies that

$$\begin{aligned}\left.\frac{\partial\psi}{\partial r}\right|_{r=a} &= 0\\ \left.\frac{\partial\psi}{\partial\theta}\right|_{r=a} &= 0\end{aligned} \tag{56}$$

The second boundary condition can be formulated using limits. The second boundary condition is

$$\lim_{r\to\infty} u_r = U\cos\theta$$

$$\lim_{r\to\infty} u_\theta = -U\sin\theta \tag{57}$$

This boundary condition needs to be reformulated so it will be a requirement for ψ instead of $\vec{u}$. Using equation (47) the second boundary conditions can be rewritten and one can easily find the new boundary condition for ψ

$$\lim_{r\to\infty} \psi = \frac{1}{2}Ur^2\sin^2\theta \tag{58}$$

This boundary condition results in a prediction of how the final stream function will look like. The expectation is that the stream function should have the following form

$$\psi = f(r)\sin^2\theta \tag{59}$$

To check this solution the differential operator E^2 is applied to equation (19)

$$\begin{aligned} E^2(f(r)\sin^2\theta) &= \frac{\partial^2}{\partial r^2}(f(r)\sin^2\theta) + \frac{\sin\theta}{r^2}\frac{\partial}{\partial\theta}\left(\frac{1}{\sin\theta}\frac{\partial}{\partial\theta}(f(r)\sin^2\theta)\right) \\ &= \frac{d^2f(r)}{dr^2}\sin^2\theta + \frac{\sin\theta}{r^2}\frac{\partial}{\partial\theta}\left(\frac{f(r)}{\sin\theta}2\sin\theta\cos\theta\right) \\ &= \left(\frac{d^2}{dr^2} - \frac{2}{r^2}\right)f(r)\sin^2\theta \end{aligned} \tag{60}$$

To simplify the equation a dummy function is introduced, represented by

$$g(r) = \left(\frac{d^2}{dr^2} - \frac{2}{r^2}\right)f(r) \tag{61}$$

After introducing this dummy function the differential operator E^2 is again applied to equation (60), resulting in

$$\begin{aligned} E^2\left(E^2(f(r)\sin^2\theta)\right) &= E^2(g(r)\sin^2\theta) \\ &= \left(\frac{d^2}{dr^2} - \frac{2}{r^2}\right)g(r)\sin^2\theta \\ &= \left(\frac{d^2}{dr^2} - \frac{2}{r^2}\right)^2 f(r)\sin^2\theta \end{aligned} \tag{62}$$

If the trial function given by equation (59) satisfies the differential equation for ψ, which is given by equation (54), then equation (62) must be equal to zero. This results in a requirement for $f(r)$ in the form of an ordinary differential equation

$$\left(\frac{d^2}{dr^2} - \frac{2}{r^2}\right)^2 f(r) = 0 \tag{63}$$

The form of the equation implies a solution of the type $f(r) = r^\alpha$. Inserting this into equation (63) results in

$$\left(\frac{d^2}{dr^2} - \frac{2}{r^2}\right)^2 r^\alpha = 0 \tag{64}$$

And one can easily find the solution α

$$(\alpha(\alpha-1)-2)\big((\alpha-2)(\alpha-3)-2\big)=0 \tag{65}$$

The solutions of equation (65) are $\alpha = -1, 1, 2$ and 4. Now these solutions can be implemented in the solution to the partial differential equation for ψ in the form of

$$\psi = (Ar^{-1} + Br^{1} + Cr^{2} + Dr^{4})\sin\theta \tag{66}$$

The boundary conditions will be used to find the variables *A*, *B*, *C* and *D*. The second boundary condition (uniform flow far from the cylinder) requires that Cr^2 is the dominant term as $r \rightarrow \infty$ which in turn implies $D = 0$. Using equation (58) one can find that

$$C = \frac{1}{2}U \tag{67}$$

The no-slip condition leads to two requirements of the derivatives of ψ. These are used to determine the two remaining constants, *A* and *B*. Using equation (56) results in two functions

$$\begin{aligned} -\frac{A}{a^2} + B + Ua &= 0 \\ \frac{A}{a} + Ba + \frac{1}{2}Ua^2 &= 0 \end{aligned} \tag{68}$$

These equations form a set which can be solved using linear algebra. Using this technique one can find that

$$\begin{aligned} A &= \frac{1}{4}Ua^3 \\ B &= -\frac{3}{4}Ua \end{aligned} \tag{69}$$

Now that the variables *A*, *B*, *C* and *D* are known the stream function can be obtained

$$\psi = \frac{1}{4}U(2r^2 - 3ar + a^3r^{-1})\sin^2\theta \tag{70}$$

Combining equation (47) and equation (70), the velocity profile of the fluid can be found

$$\vec{u} = U\cos\theta\left(1 - \frac{3a}{2r} + \frac{a^3}{2r^3}\right)\hat{e}_r - U\sin\theta\left(1 - \frac{3a}{4r} - \frac{a^3}{4r^3}\right)\hat{e}_\theta \tag{71}$$

4.2.1.3 *Results of the analytical flow around the cross-section of a cylinder*

To visualize the analytically calculated flow around the cross-section of a cylinder given by equation (71), a Matlab program has been written which can be found in the appendix. This program makes a surface plot of the velocity field as shown in figure 8.

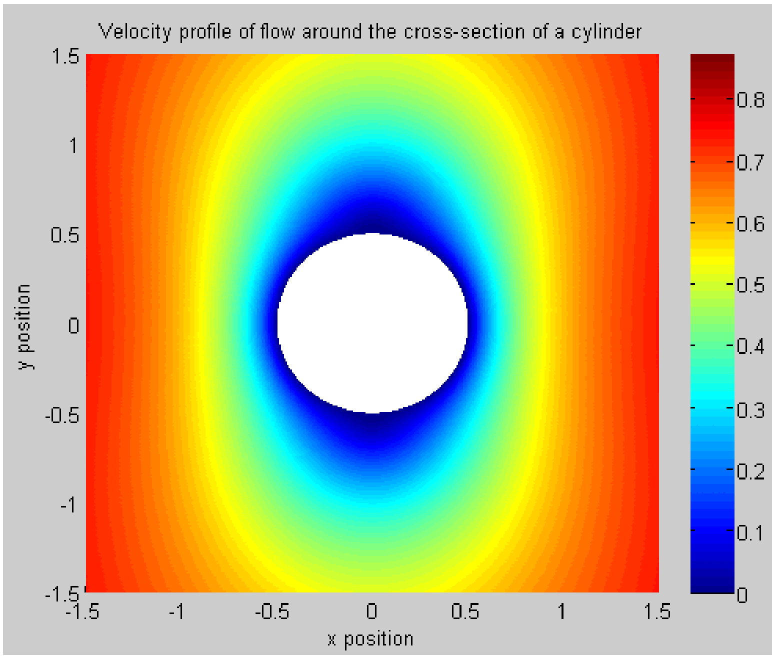

Figure 8: The surface plot of the analytically determined velocity field of the flow around the cross-section of a cylinder

4.2.2 The computationally determined solution

For the Matlab script to give the flow around the cross-section of a cylinder, the boundaries are set properly. The boundaries are built by hand in a high precision model, which represents the cross-section of a cylinder good while keeping the computational time limited.

4.2.2.1 Results of the computationally determined flow around the cross-section of a cylinder

The Matlab program has been modified in order to give the velocity field around a model of the cross-section of a cylinder. This result is shown in figure 9.

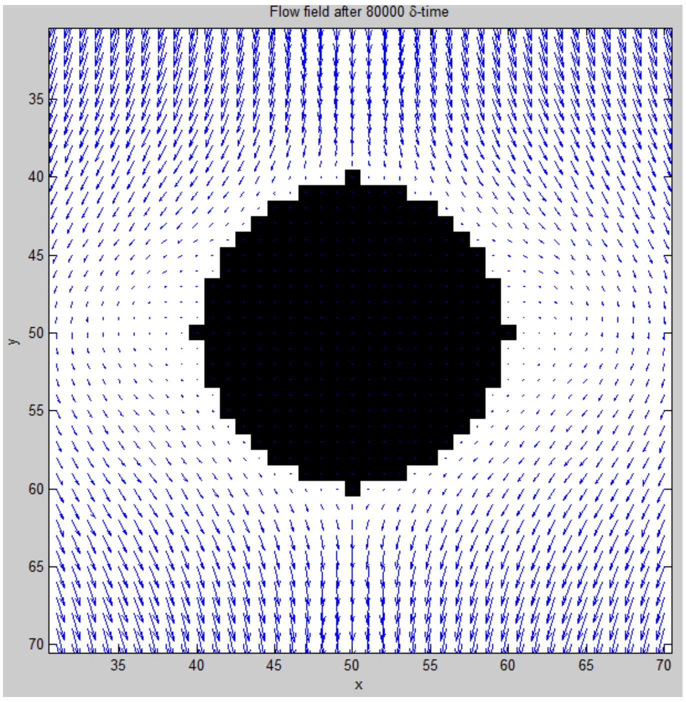

Figure 9: The computationally determined velocity field around a model of the cross-section of a cylinder

4.2.3 Comparing the two results

Comparing figure 8 and figure 9 gives an interesting result. One can see that inside the cross-sections the velocity in both figures is zero. Also the velocity field far away from the cross-section of the cylinder becomes uniform which is shown in both figures. In figure 9 however, the full axes are not shown. Because of this the flow near the sides of the figure are not uniform and could therefore be misinterpreted. The major similarity is that the fluid around both cross-sections (figure 8 has a real sphere and figure 9 has a modeled sphere) is similar, the velocity becomes slower and follows the boundaries of the cross-sections smoothly.

The only difference between both figures is that in figure 9 the cross-section is not smooth resulting in velocity vectors at places where there should not be any. However, this is not the result of a mistake in the Matlab program but it is due to lack of accuracy of the model of the cross-section. Figure 9 has already a quite high accuracy making it is safe to say that both figures should become similar when the accuracy of figure 9 increases even more. However, this is not done due to large computational time needed to perform such calculations.

Another difference is that the domain which is used is not infinitely large, which will result in different outcomes than the analytical solution.

4.2.4 Final conclusion of the benchmarks

In both benchmarks the Matlab program performed as expected. The first benchmark was the Poiseuille flow and due to all the similarities and the absence of large differences, the Matlab program for this case can be considered to be correct. The second benchmark was a more complex one, not only to solve analytically but it took also a lot more computational time (e.g. the computational time needed for a 100x100 grid was 34 hours on a regular laptop). However, because of all the similarities and the only difference caused by lack of accuracy it is again safe to say that the Matlab program gives the correct answers.

The final conclusion of the benchmarks is that in both cases the Matlab program gives physically plausible results which are the same as the analytically determined results. The Matlab program can therefore be considered to be correct.

4.3 Determining the error of the used lattice Boltzmann method

In order to get an indication of the accuracy of the used method, the error is determined. To determine the error, four resolutions have been made of a cross-section of a cylinder, each having an increase in the accuracy. For these four resolutions several points in the plot have been compared with the results of the analytically determined values of those points, after which an error has been calculated using equation (72)

$$\varepsilon = \frac{|value_{analytical} - value_{numerical}|}{value_{analytical}} \qquad (72)$$

The errors of the different models have been calculated using a program made in excel. The results of the excel program were imported in Matlab where further calculations were made. In the chapter Results the outcome of the determination of the error of the used method is shown.

5 Results

In both benchmark cases the results are plausible, physically correct and have many similarities with the analytically results. Therefore the Matlab program is considered to be correct, meaning that more complex geometries can be evaluated and the results can also be considered to be correct. The results of the determination of the error of the used method will be shown first, after which the results of the analysis of the stacking pattern are shown.

5.1 The determination of the error of the used method

The error of the used method has been determined using four resolutions of increasing accuracy (i.e. more grid nodes used to tighten the grid). The errors were determined using an excel program after which Matlab has been used to make plots of the error vs the grid accuracy. Two plots were made as shown in figures 10 and 11, one on regular axes and one on semi-log axes (i.e. the y-axis is logarithmic).

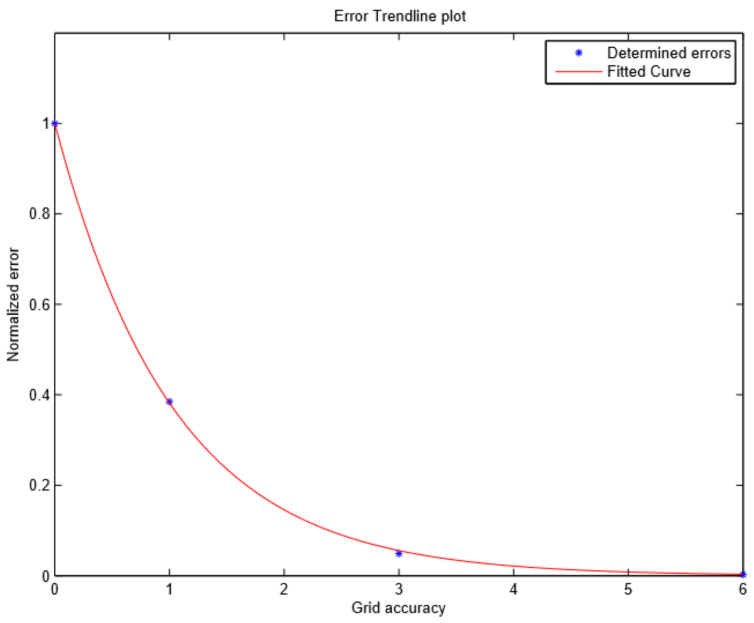

Figure 10: The plot of the normalized error vs the grid accuracy of the flow around the cross-section of a cylinder on regular axes

Figure 10 shows that when the accuracy of the grid increases the normalized error drops like an exponential function. To confirm this estimation another plot is made in Matlab using semi-log axes. This means that the y-axis has a logarithmic scale and the x-axis has a regular scale. The results of this semi-log plot are shown in figure 11.

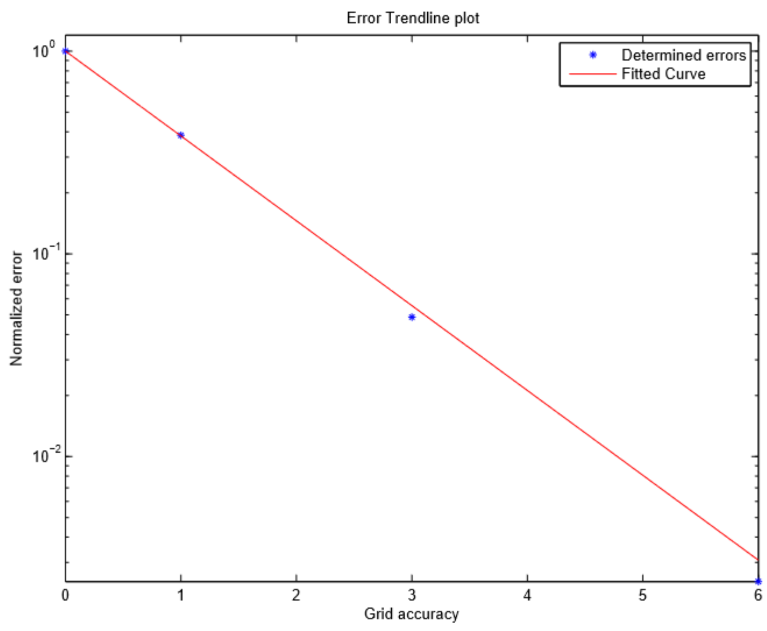

Figure 11: The plot of the normalized error vs the grid accuracy on semi-log axes

These plots are used in order to get a better understanding of the order of the error. This is done by using the "fit" function in Matlab. The function gives the parameters of the estimated function which connects the points in the best way possible (i.e. Matlab uses the least squares method). From figure 10 and 11, the function is estimated to be an exponential function of the form

$$fit = Aexp(Bx) \tag{73}$$

In which A and B are constants with A = 1 since it is normalized and B = -0.9639, which is estimated by Matlab with an accuracy of 95%. This results in the function of the error as

$$error = \exp(-0.9639x) \tag{74}$$

In equation (74), x represents the accuracy of the grid which is the inverse of the error of the grid.

From equation (74) one can see that the chosen method is of the first order. Since the half-way bounce back method is used, this result correspondents with the one mentioned in the literature.

5.2 Fluid flow through a stacking pattern

In this thesis a Matlab program has been written which is capable of creating the flow field through a certain stacking pattern. The stacking pattern which is analyzed in Matlab is shown in figure 12.

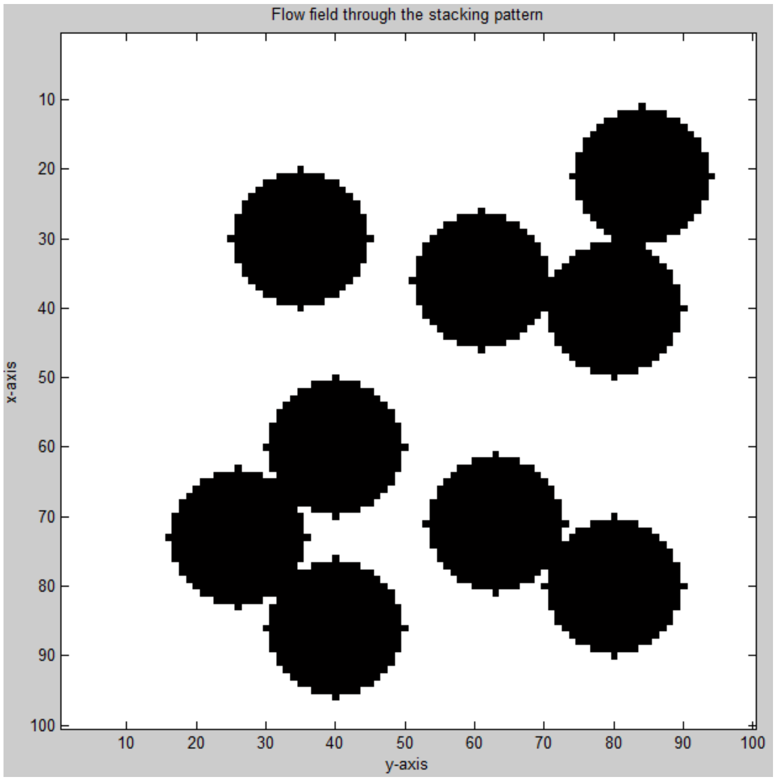

Figure 12: The stacking pattern which is analyzed in Matlab using the lattice Boltzmann method

After running the Matlab program a vector field is created which shows how the fluid moves through this specific stacking pattern. However, since the grid has a high accuracy the vector field can not be analyzed when looking at the complete picture as one can see in figure 13.

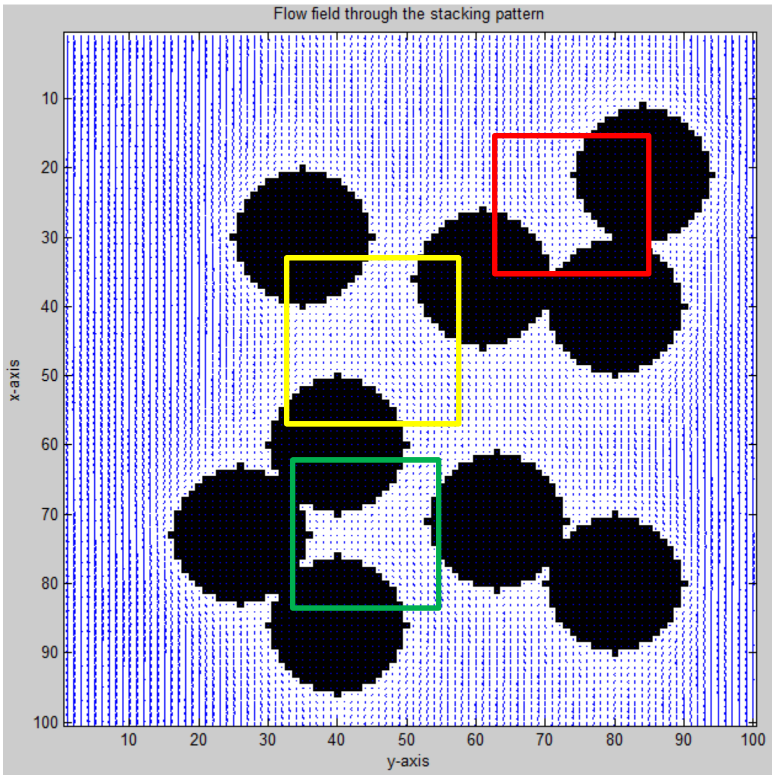

Figure 13: The complete view of the vector field created by Matlab with the marked areas which will be looked at more with more detail (i.e. red = area 1, green = area 2, yellow = area 3). Parameters used are τ = 0.6, Re < 100, D = 20 lu

To analyze the vector field, three areas are looked upon more closely. The first area (marked red in figure 13) which is analyzed is the cluster of three cross-sections of cylinders in the top-right of figure 13. The second area (marked green in figure 13) is the other cluster of three cross-sections of cylinders in the bottom-left corner. These two areas both have a cluster of three cross-sections but they are oriented in different ways. The last area (marked yellow in figure 13) which is analyzed is the pathway through the middle of the cross-sections. These areas are chosen because in these areas the most interesting things will happen.

5.2.1 Analyzing area 1

The first area which will be analyzed is the one marked red in figure 13. This area is chosen because in the cluster of the cross-sections there is a gap, where the fluid will come to a dead-end. As shown in figure 14 the incoming fluid moves around the cross-sections towards the gap. Once the fluid is in the gap the velocity drops rapidly and the fluid stands almost completely still at the end of the gap. From figure 14 one can see that there is mass accumulating in the gap, which is not physically possible.

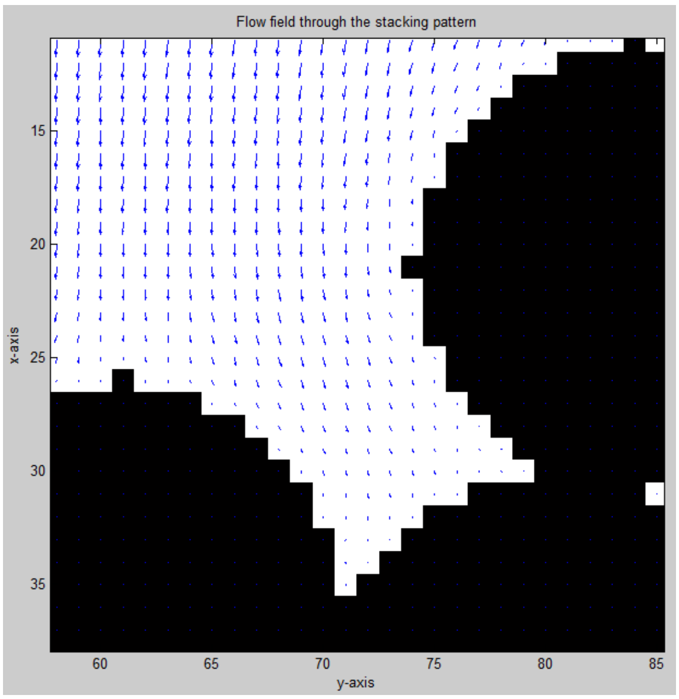

Figure 14: The vector field of the analyzed stacking pattern focused on area 1

5.2.2 Analyzing area 2

The second area which will be analyzed is a similar one to area 1 but the orientation of the gap is now to the east side instead of to the north side. Another difference is that the incoming fluid is squeezed through a narrow path as one can see in figure 13. The vector field of area 2 is shown in figure 15. What can be observed is that, due to the orientation of the gap, the fluid in the gap almost stands still. This resistance is much greater in the gap than through the pathway, therefore the fluid has no velocity in the large portion of the gap.

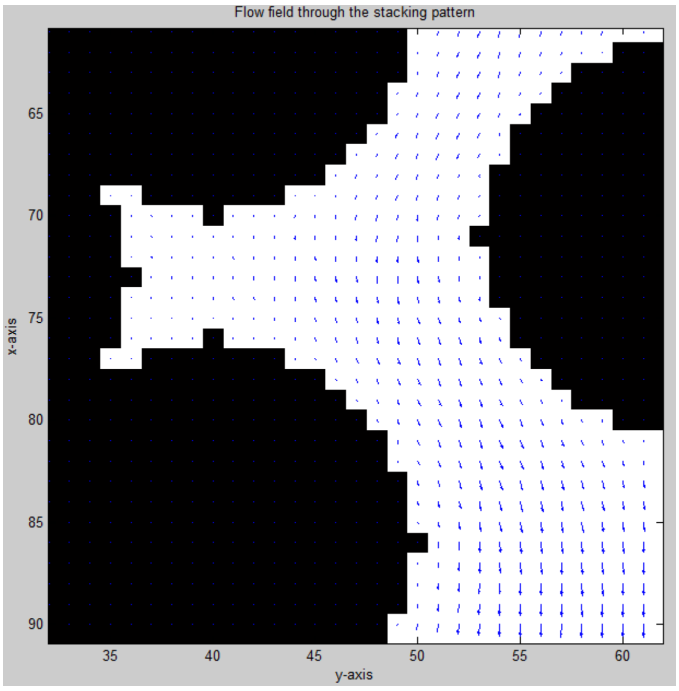

Figure 15: The vector field of the analyzed stacking pattern focused on area 2

5.2.3 Analyzing area 3

The last area which is analyzed is the one marked yellow in figure 13. It is the area between the majority of the cross-sections in which the most preferable pathway is examined. In figure 13 the most preferable pathway can be seen when looking at the density of the vectors, but a more detailed overview can be found in figure 16. From figure 16 the most preferable pathway can be noticed. It is the path passing by the right side of the bottom-left cluster. This is in accordance with the expectations, due to expected lower resistance in the right side of the bottom-left cluster in comparison with the left side of this cluster.

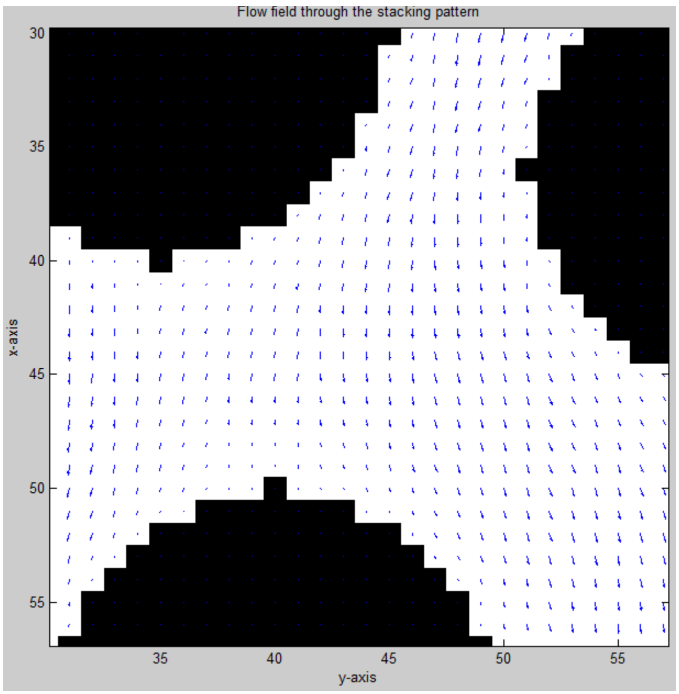

Figure 16: The vector field of the analyzed stacking pattern focused on area 3

6 Conclusion

In this chapter conclusions will be drawn for the benchmarks, the determination of the accuracy of the used lattice Boltzmann method and for the results of the vector fields for the three areas. From the comparison between the obtained results and the analytically determined values the conclusions for the benchmarks will be drawn. For the conclusion of the used lattice Boltzmann method, literature values are compared with the obtained values. The conclusions for the vector field are based upon the results and expectations due to fluid resistances.

6.1 Conclusions regarding the benchmarks

The first benchmark was the Poiseuille flow. The analytically determined values resulted in a graph, which can be seen in figure 6. Using the created Matlab script the velocities of 28 points were calculated. These values were plotted in the same graph as the analytically determined graph. What can be seen is that the 28 points lie on the analytically determined graph. From this the conclusion is drawn that the Matlab script gives the correct values for the first benchmark.

The second benchmark was the flow around a cross-section of a cylinder. The analytical derivation was more complex than the one of the Poiseuille flow but it resulted in a surface plot, which can be seen in figure 8. After the result of the analytically determined flow field was known, the Matlab script was used in order to get a high precision model of a cross-section of a cylinder, which was used to model the fluid flow around a cross-section of a cylinder. Due to the similarities between the modeled flow field and the analytically determined flow field, the Matlab script is considered to be correct regarding the second benchmark.

Since both the benchmarks gave results which were in accordance with the analytically determined results, the Matlab code is considered to be giving the correct results and more complex geometries can be analyzed.

6.2 Conclusions regarding the determined accuracy

The used Matlab code was modelling the cross-section of a cylinder. Since it is a model the accuracy of the used model was examined. The accuracy was determined using a created excel program combined with Matlab. The accuracy of the modelled cross-section was enlarged and with each enlargement the error was determined between the obtained data and the analytically determined data. Finally a plot was made of the error vs the accuracy, which can be seen in figures 10 and 11. Using Matlab's internal function fit an estimate was made of the order of the error of the used lattice Boltzmann model. The result was that the error was of the first order which is in accordance with the literature value for the half-way bounce back method.

6.3 Conclusions regarding the vector field

The main purpose of this thesis was to create a Matlab function which was capable of creating vector fields for arbitrary geometries. A certain geometry, presented in figure 12, was created and analyzed using three areas in which more detail was visible.

6.3.1 Conclusions regarding area 1

The first area resulted in a vector field towards a cluster of cross-sections with a gap positioned north, which can be seen in figure 14. The fluid flows in the gap and the fluid velocity goes to zero as the fluid comes closer to the boundaries. This is in accordance with the expectations because the fluid comes to a dead-end from which it can not escape resulting in a very small velocity in the gap.

6.3.2 Conclusions regarding area 2

The second area was similar to the first one but the orientation of the gap of this cluster was east and a different entry of the fluid. This resulted in an almost stagnant flow in the gap due to the higher fluid resistance in the gap in comparison with the pathway between the boundaries. This is also in accordance with the expectations, because the gap creates a large fluid resistance and the fluid flows with a smaller resistance through the pathway resulting in a preferable path the fluid will take, which is visible in figure 15.

6.3.3 Conclusions regarding area 3

The last area is the area between the majority of the cross-sections in which the most preferable path is examined. This path runs through the middle to the right side of the bottom-left cluster, which can be seen in figure 16. This is also in accordance with the expectations, because the resistance the fluid encounters is expected to be higher to the left of the cluster than to the left side. This is due to the small asymmetry between the placement of the three cross-sections visible in area 3. Due to this asymmetry the fluid is expected to follow the path to the right side of the bottom-left cluster. This expectation is in accordance with the obtained result from Matlab.

6.4 Final conclusion

In this thesis a Matlab program has been made which is capable of creating vector plots representing the fluid flow through a stacking pattern. From the benchmarks the computational program can be considered to give similar results in comparison with the analytically determined results. The error of the used half-way bounce back method is analyzed and the relation between the error of the used method and the grid accuracy was in accordance with the relation found in the literature. When a full stacking pattern was analyzed using the created program a vector field was created and analyzed. The results were logical but there is a mistake in the program causing mass accumulation. This mistake can be caused by the fact that the spheres are modelled with squares or that there is a small mistake in the code. However, this remains uncertain and in order to be sure what is wrong more stacking patterns have to be analyzed and further improvements have to be made in the Matlab program.

Since the mass accumulation mistake is the only mistake that can be noticed from the vector field of the stacking pattern the Matlab program is not useless, it still gives logical overall fluid fields, but in order to be completely satisfied further research is needed in order to solve the mass accumulation problem.

7 Biography

Ballensiefen, G., & Wolf, L. (1975). Fuel elements of the high temperature pebble bed reactor. *Nuclear Engineering and Design*, 93-108.

Benzi, R., & Succi, S. (1992). The lattice Boltzmann equation: theory and applications. *Physics Reports*.

Bergmans, L. (2013, 7 11). Matlab discussion. (J. Versendaal, Interviewer)

Chen, S., & Doolen, G. D. (1998). Lattica Boltzmann method for fluid flows. In S. Chen, & G. D. Doolen, *Lattica Boltzmann method for fluid flows.* Los Alamos.

Chung, T. J. (2010). Computational Fluid Dynamics. In T. J. Chung, *Computational Fluid Dynamics.* Cambridge: Cambridge University Press.

Daniels, F. (2013, 8 20). *Nuclear Power and Reasearch Reactors.* Retrieved 8 20, 2013, from Oak Ridge National Labratory: http://web.ornl.gov/info/ornlreview/v36_1_03/article_01.shtml

Gerwin, H., & Scherer, W. (1999, 08 08). *Patent No. 5978434A.* United States of America.

Hardy, J., & de Pazzis, O. (1976). Molecular dynamics of a classical lattice gas. In J. Hardy, & O. de Pazzis, *Molecular dynamics of a classical lattice gas.*

Hess, J. L. (1967). Calculation of Potential Flow About Arbitrary Bodies. In J. L. Hess, *Calculation of Potential Flow About Arbitrary Bodies.*

Hobbel, E. (2013, 07 11). Matlab discussion. (J. Verseldaal, Interviewer)

Lee, J.-J., Park, G., & Kim, K.-Y. (2007). Numerical treatment of pebble contact in the flow and heat transfer analysis of a pebble bed reactor core. *Elsivier*, 2183-2196.

Lohnert, G. (1976). *Patent No. 3.960.656.* United States of America.

Mohamad, A. (2011). *Lattice Boltzmann method.* Calgary: Springer.

Mudde, R., & van den Akker, H. (2008). Fysische transportverschijnselen. In R. Mudde, & H. van den Akker, *Fysische transportverschijnselen.* Delft: VSSD.

Nicholls, D., & Ion, S. (2003). Pebble Bed Modular Reactor The First Generation IV Reactor To Be Constructed. *World Nuclear Association Annual Symposium*.

Pebble Bed Reactor scheme. (n.d.). Retrieved 8 18, 2013, from Uploads Wikipedia: http://upload.wikimedia.org/wikipedia/commons/0/0f/Pebble_bed_reactor_scheme_%28English%29.svg

Rohde, M. (2004). *Extending the lattice Boltzmann method.* Delft.

Rohde, M. (2009). Cellular Automata. In M. Rohde, *Cellular Automata.* Delft.

Sukop, M., & Thorne, D. (2006). *Lattice boltzmann modeling : an introduction for geoscientists and engineers.* Springer.

Thomson, M. (1973). Theoretical Aerodynamics. In M. Thomson, *Theoretical Aerodynamics.* Dover Publications.

Wendt, J. F. (2009). Computational Fluid Dynamics. In J. F. Wendt, *Computational Fluid Dynamics.* Belgium: Springer.

8 Appendix

In the appendix the complete code can be found which is used for the creation of the vector field plots, this Matlab code can be found in appendix A. The Matlab code which plots the Poiseuille flow can be found in appendix B and in appendix C the Matlab code which is used for the creation of the flow around a cross-section of a cylinder can be found.

8.1 Appendix A – Matlab script for the computationally determined velocity profiles

```
%%% The D2Q9 lattice Boltzmann method %%%
clear all;
close all;
clc;

% Define parameters
lx = 100; %
ly = 100; % Grid dimensions
gridsize = lx*ly;
dg = [0:gridsize:(gridsize*9)];
tau = 0.7; % Relaxation time
omega = 1/tau;
rho = 1; % Density

% Defining the weights of the D2Q9 model
w1 = 4/9;
w2 = 1/9;
w3 = 1/36;

% Creating the f-matrix using the repmat function
f = repmat(rho/9, [lx ly 9]);

% Initializing the equilibrium term
feq = f;

% Making the boundaries
%%% D = 20
boundary = zeros(lx,ly);

% Insert boundaries here %

% Create function when the boundary is hit by a vector
hit = find(boundary); % hit gives the points where the boundary is nonzero
incoming = [hit+dg(1) hit+dg(2) hit+dg(3) hit+dg(4) hit+dg(5) hit+dg(6)...
            hit+dg(7) hit+dg(8)];
reflected = [hit+dg(5) hit+dg(6) hit+dg(7) hit+dg(8) hit+dg(1) hit+dg(2)...
             hit+dg(3) hit+dg(4)];

% Setting some parameters to start the while loop
uavg1 = 1; uavg2 = 1; duavg = (uavg2-uavg1)/uavg1; time = 0; dU = 1e-8;
T = 40000;
while (time < T & abs(duavg)) > 1e-12 | time < 100
    % Propagation step
    f(:,:,1) = f([lx 1:lx-1],:,1);              % East
    f(:,:,3) = f(:,[ly 1:ly-1],3);              % North
    f(:,:,5) = f([2:lx 1],:,5);                 % West
    f(:,:,7) = f(:,[2:ly 1],7);                 % South
    f(:,:,2) = f([lx 1:lx-1],[ly 1:ly-1],2);    % North-East
    f(:,:,4) = f([2:lx 1],[ly 1:ly-1],4);       % North-West
    f(:,:,6) = f([2:lx 1],[2:ly 1],6);          % South-West
    f(:,:,8) = f([lx 1:lx-1],[2:ly 1],8);       % South-East

    f_inc = f(incoming);
    rho = sum(f,3);
    ux = (sum(f(:,:,[1 2 8]),3) - sum(f(:,:,[4 5 6]),3))./rho;
    uy = (sum(f(:,:,[2 3 4]),3) - sum(f(:,:,[6 7 8]),3))./rho;
    uy(1,:) = uy(1,:) + dU; %Increase inlet velocity
    ux(hit) = 0; % Set x-velocity in boundaries to zero
    uy(hit) = 0; % Set y-velocity in boundaries to zero
    rho(hit) = 0;% Set density in boundaries to zero
    usq = ux.^2+uy.^2;

    % Determine the equilibrium distribution
    feq(:,:,1) = w2*rho.*(1 + 3*ux + 9/2*ux.^2 - 3*usq/2);
    feq(:,:,2) = w3*rho.*(1 + 3*(ux+uy) + 9/2*(ux+uy).^2 - 3*usq/2);
    feq(:,:,3) = w2*rho.*(1 + 3*uy + 9/2*uy.^2 - 3*usq/2);
    feq(:,:,4) = w3*rho.*(1 + 3*(-ux+uy) + 9/2*(-ux+uy).^2 - 3*usq/2);
    feq(:,:,5) = w2*rho.*(1 - 3*ux + 9/2*ux.^2 - 3*usq/2);
    feq(:,:,6) = w3*rho.*(1 + 3*(-ux-uy) + 9/2*(-ux-uy).^2 - 3*usq/2);
    feq(:,:,7) = w2*rho.*(1 - 3*uy + 9/2*uy.^2 - 3*usq/2);
    feq(:,:,8) = w3*rho.*(1 + 3*(ux-uy) + 9/2*(ux-uy).^2 - 3*usq/2);
    feq(:,:,9) = w1*rho.*(1 - 3*usq/2);
    % Determine total f function
    f = f+omega*(feq-f);
    f(reflected) = f_inc;
    uavg2 = uavg1;
    uavg1 = sum(sum(ux));
    duavg = (uavg2-uavg1)/uavg1;
    time = time + 1;
end
% Plot the velocity field
figure(1);
colormap(gray(2));     % }
image(2-boundary');    % } Plot boundaries black
hold on;
scale = 4;
quiver(ux(1:lx,:),uy(1:ly,:),scale); % Plot the vector field
xlabel('x-axis');
ylabel('y-axis');
axis square;
title(['Flow field through the stacking pattern']);
```

8.2 Appendix B – Matlab script for the analytically determined Poiseuille flow

```
%%% Velocity profile of Poiseuille flow %%%

clear all;
close all;
clc;

% Define parameters and calculate velocity profile
mu = 1;
dpdx = 1;
L = 1;
y = 0:0.01:L;
for i = 1:length(y);
    u(i) = -1/(2*mu)*dpdx*(L*y(i)-y(i).^2); % analytical formula
```

```
end                                         % for Poiseuille flow

% Plotting the velocity profile
u_max = u(50);
hold on
plot(y,u/u_max,'b-','LineWidth',2),grid
title('Velocity profile of Poiseuille flow');
ylabel('Relative velocity: u/u_m_a_x'); xlabel('Relative length: y/L');
legend('Velocity profile of Poiseuille flow');
```

8.3 Appendix C – Matlab script for the analytically determined flow around a cross-section of a cylinder

```
%%% Velocity profile for flow around a cross-section of a cylinder
%%% The formula
% V=cos(theta)*(1-3*a/(2*r)+a^3/(2*r^3))[r-dir]-sin(theta)*(1-3*a/(4*r)-
a^3/(4*r^3))[theta-dir];
% radius of the sphere = 0.5
% 0 < theta < 2*pi
% a < r < 3

clear all;
close all;
clc;

a = 0.5; % Radius of the cross-section
%%% Make a grid
theta = linspace(0,2*pi,500);
r = linspace(a,3,500);
[THETA,RR] = meshgrid(theta,r);
[A,B] = pol2cart(THETA,RR);

%%% Calculate the velocity vectors
for i = 1:length(r);
    for j = 1:length(theta);
        V1(i,j)  = cos(theta(j))*(1-3*a/(2*r(i))+a^3/(2*r(i)^3));
        V2(i,j)  = -sin(theta(j))*(1-3*a/(4*r(i))-a^3/(4*r(i)^3));
    end
end

%%% Compute the total velocity
V3 = sqrt(V1.^2+V2.^2);

%%% Plot the velocity profile
figure
hold on
surf(B,A,V3,'edgecolor','none')
xlabel('x position')
ylabel('y position')
title('Velocity profile of flow around the cross-section of a cylinder');
colorbar;
grid off
view(0,90)
ylim([-1.5 1.5])
xlim([-1.5 1.5])
```