Graz University of Technology
Institute of Process and Particle Engineering

process & particle engineering

Local Grid Refinement for Fluid Simulations Using the Lattice Boltzmann Method Applied on a Stirred Tank Bioreactor

Laura Schneider

Graz, January 2021

Laura Schneider, B.Sc.

Local Grid Refinement for Fluid Simulations Using the Lattice Boltzmann Method Applied on a Stirred Tank Bioreactor

MASTER'S THESIS

to achieve the university degree of

Diplom-Ingenieurin

Master's degree program:

Chemical and Process Engineering

submitted to

Graz University of Technology

Supervisors

Univ.-Prof. Dipl.-Ing. Dr. techn. Johannes Khinast

Dipl.-Ing. Dr.-techn. Christian Witz

Dipl.-Ing. Philipp Eibl, BSc

Institute of Process and Particle Engineering

Graz, August 2020

AFFIDAVIT

I declare that I have authored this thesis independently, that I have not used other than the declared sources / resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used. The text document uploaded to TUGRAZonline is identical to the present master's thesis.

_____

Date; Signature

Acknowledgement

I would like to express my gratitude to Christian Witz and Philipp Eibl for the excellent guidance, the helpful advices and the various intense discussions. Thank you, for integrating me into the team and for the pleasant cooperation. I am really looking forward to fruitfully working with you together, in future.

Further, I would like to thank Prof. Johannes Khinast for the opportunity to realize this work and to delve into the universe of lattice Boltzmann simulations.

Auch möchte mich an diesem Punkt besonders bei meiner Mama und Luca bedanken, die mich immer vorbehaltslos unterstütz haben und mir mit Rat und Tat beiseite gestanden haben. Danke! Ebenso gilt mein Dank der restlichen Familie – Wolfgang, Paul, Jonathan, Emilia, Aaron und Michael, sowie all meinen Großeltern – für die zahlreichen Erholungspausen die ich mit euch genießen konnte, für Kost, Logis und die finanziellen Zuschüsse. Ohne euch wäre der Weg der letzten Jahre wahrscheinlich nur halb so leicht und bestimmt nur halb so schön gewesen.

Abstract

Since the homogenously spaced grid of the lattice Boltzmann method can be a major drawback in fluid simulations, the topic of this work was to implement and validate the functionality of local grid refinement in an existing three-dimensional lattice Boltzmann code running on graphics processing units (GPUs). A multi-grid domain, where a fine grid is added in presence of a complete coarse grid was chosen. The additional grid is refined with a factor of two. A pre-collision grid coupling algorithm is used to the reconstruct the unknown particle distribution functions at the overlapping interfaces between the grids. For this, a second order accurate compact interpolation of the velocity field is applied. A detailed and illustrated description of the grid coupling and interpolation procedure is given. Further, the implementation is described and a simple method for guaranteeing the correct sequencing of grid coupling and fluid propagation in case of multiple refinement steps is introduced. The new feature is applied to laminar flow in a cylindrical tube and the results are compared to the analytical solution of the Navier-Stokes equations. Further, the agitator of stirred tank bioreactors is refined to observe the algorithms behavior in turbulent flows. The obtained results satisfyingly reproduce the velocity in laminar as well as in turbulent regimes and an accurate pressure drop can be obtained for Hagen-Poiseuille flow. However, for stirred tanks an increase in total mass over time is observed which raises the need for further investigations.

Kurzfassung

Da das gleichmäßig geteilte Gitter der Lattice-Boltzmann-Methode ein Nachteil bei Strömungssimulationen sein kann, war das Thema dieser Arbeit, die Möglichkeit der lokalen Gitterverfeinerung in einen bestehenden, dreidimensionalen, auf Grafikprozessoren rechnenden Lattice-Boltzmann-Code zu implementieren und zu validieren. Es wurde ein *multi-grid* Ansatz gewählt, in der ein doppelt so feines Gitter zusätzlich zu einem vollständigen, lückenlosen groben Gitter verwendet wird. Ein Algorithmus zur Kopplung der Gitter vor dem *Collision-Step* wird verwendet, um die unbekannten Partikel-Verteilungsfunktionen an den sich überlappenden Grenzflächen der Gitter zu rekonstruieren. Dazu wird eine lokale Methode zur Interpolation des Geschwindigkeitsfeldes angewendet. Das Verfahren zur Gitterkopplung und Interpolation wird dargelegt. Ebenso wird die praktische Umsetzung beschrieben und eine einfache Methode zur Gewährleistung der korrekten Abfolge von Gitterkopplung und Fluidberechnung bei Verwendung von mehreren Verfeinerungsstufen vorgestellt. Die neue Funktion wird bei der Berechnung einer laminaren Strömung in einem zylindrischen Rohr angewendet und die Ergebnisse werden mit der analytischen Lösung der Navier-Stokes-Gleichungen verglichen. Zudem wird der Rührer eines Bioreaktoren höher aufgelöst, um das Verhalten der Algorithmen in turbulenten Strömungen zu beobachten. Die gewonnenen Ergebnisse zeigen, dass die Geschwindigkeit, sowohl im laminaren als auch im turbulenten Regime, zufriedenstellend reproduziert wird und ebenso, ein ausreichend genauer Druckabfall für die Hagen-Poiseuille Strömung ermittelt werden kann. Im Fall von gerührten Kesseln wird jedoch eine Zunahme der Gesamtmasse über die Zeit beobachtet, was eine tiefgehende Untersuchung des Algorithmus notwendig macht.

Table of Contents

1 Introduction

The global market of biopharmaceutical medicines has evolved enormously over the past three decades. As a survey [1] among 222 biopharmaceutical decision makers across 22 countries illustrated, the annual revenue increased from $4.4 billion in 1990 to $275 billion in 2018. In comparison to the overall market of pharmaceuticals, biopharmaceuticals – also named biologics – held more than 25% of market share in 2018. This reveals the importance of biopharmaceutical sector and its persistent growth. [2]

Furthermore, biosimilars and biogenerics which are the biopharmaceutical version of generics, recently began to conquer the market leading to harsh competition among the biopharmaceutical players. The presence of biosimilars opens the market to emerging companies and developing regions which suffer from know-how deficit. As a consequence, efficiency and productivity in manufacturing are issues with high degree of importance for many. [2]

A powerful method to gain insight into the production process and, therefore, be able to increase efficiency and productivity is to simulate underlying physical and biological mechanisms. The Institute of Process and Particle Engineering of Graz University of Technology has developed a tool which is able to simulate the fluid dynamics inside a stirred tank reactor. Furthermore, temperature, oxygen, carbon dioxide and substrate distribution and transport as well as bubble and species movement can be computed. In addition, several biomodels are implemented. Using this approach, valuable information can be generated regarding scale-up, gas flow rates, shear rates, dead zones, oxygen mass transfer coefficient and many more.

The challenging task of calculating the fluid dynamics is done with the lattice Boltzmann method which will be outlined in section 2.1. This method is attracting attention in the field of computational fluid dynamics since many years, due its rather simple implementation and its parallelizability. As a consequence, computation can be performed using general purpose graphics processing units (GPGPU) which are favorable due to their high performance-to-price ratio.

This thesis gives an overview of the necessary fundamental theories in chapter 2. It contains a short introduction into the kinetic theory, a summary of the lattice Boltzmann method and the used boundary conditions and ends with an outline of the characteristics of GPU computing. Chapter 3 is focusing on the theory of local grid refinement. Different methods of mesh

arrangements are explained as well as the need for lattice conversion between grids with different lattice spacing. Further, the grid coupling algorithm is illustrated in detail. In chapter 4 the LBM code is presented and the required functions to implement grid refinement are described. The method is validated by the comparison of simulation results with reference simulations and the analytical solution of the Navier-Stokes equation for the Hagen-Poiseuille flow. The outcome is presented and discussed in chapter 5. In chapter 6 the thesis and its key findings are summarized.

1.1 Motivation

One major drawback of the original formulation of the lattice Boltzmann method compared to conventional computational fluid dynamic methods is the limitation to homogeneous quadratic (in 2 dimensions) or cubic grids (in 3 dimensions). In order to obtain a solution with satisfactory accuracy, it is necessary to choose a sufficiently fine grid. A finer grid in this context refers to a shorter distance between adjacent grid nodes than in a coarser grid. However, the finer the grid, the higher are the computational resources and time needed. Furthermore, in irregular problems only certain regions of the flow field are crucial and need a high resolution. Therefore, applying a fine grid to the whole system is unfavorable.

In the case of bio reactors, the simulation must provide all required internals like stirrer, heat exchanger, sensors, gas spargers, baffles and further with all flow relevant details. For instance, in large biogas digesters or fermentation tanks, to save head space, small side entering stirrers are mounted. As the tank-to-impeller ratio is quite high in these cases, a satisfying resolution of the agitator requires an immensely high number of computational cells to display the whole tank which makes the computational costs unnecessarily high.

This generates the need for a local grid refinement in lattice Boltzmann simulations. As a result, it is then possible to refine the grid in a user defined region, for instance around the stirrer, and therefore, be able to execute the simulation using less memory and computational time compared to the case of refining the whole domain.

2 Theoretical Background

In the calculation of fluid dynamics, distinctions can be made between different length and time scales. Each of these scales corresponds to an individual approach with its own fundamental equations for the calculation of fluid dynamics. On the microscopic scale, it is based on the fact, that a fluid is composed of a huge number of small particles which interact with each other. The prevailing measures are the size of an atom or molecule and the duration of the collision between two or more particles. In this case, Newton's dynamics apply and particle-based simulations like molecular dynamics are done. On the opposite, there is the macroscopic approach where the scale of gradients of fluid properties is decisive and relevant time scales are given by diffusion or advection. Following this approach, the fluid continuum is described by the Navier-Stokes equations (NSE) and conventional computational fluid dynamics (CFD) methods are applied, where macroscopic variables like velocity and pressure are calculated in various points in the domain. Between the microscopic and the macroscopic scale one can determine a so-called mesoscopic scale which is neither considering each particle on its own nor treating the fluid as a continuum. Rather, it describes the distribution of particles in a fluid. The mean free path of the particles defines the length scale and, consequently, the time between two successive collision events is a measure for time scale. Kinetic theory is utilized at mesoscopic scale and frames the fundament of lattice Boltzmann method (LBM). [3, pp. 11-13], [4, pp. 2-3]

The prediction of fluid dynamics with molecular dynamics simulations is limited to systems with manageable number of particles and conventional CFD methods, although very powerful, are complex and sensitive to instabilities. On the contrary, LBM, as it is only tracking distribution functions instead of single particles, can cope with much bigger systems while being easy to implement. It reproduces mass conservative flows in complex boundaries as well as multiphase or multi component flow. Furthermore, a wide range of boundary conditions for all kind of problems have been developed. [3, pp. 53-56]

2.1 Lattice Boltzmann Method

2.1.1 Kinetic Theory

A fundamental term in kinetic theory is the particle distribution function $f(x, \xi, t)$, also named particle population in some references. It represents the density of mass, more precisely density of fluid particles, at position x and time t with microscopic velocity ξ.

$$f\left[kg \cdot \frac{1}{m^3} \cdot \frac{1}{\left(\frac{m}{s}\right)^3}\right] \quad (2\text{-}1)$$

As a consequence of this, the macroscopic velocity $u_\alpha(x,t)$ and density $\rho(x,t)$ are obtained by the zeroth and first moment of the particle distribution function. [3, pp. 16-17]

$$\Pi_0 = \int f(x,\xi,t) d^3\xi = \rho(x,t) \quad (2\text{-}2)$$

$$\Pi_\alpha = \int \xi_\alpha f(x,\xi,t) d^3\xi = \rho(x,t) u_\alpha(x,t) \quad (2\text{-}3)$$

The index α characterizes one element of a vector and therefore, one spatial direction. Another important relation is the behavior of the particle distribution function over time.

$$\frac{df}{dt} = \left(\frac{\partial f}{\partial t}\right)\frac{dt}{dt} + \left(\frac{\partial f}{\partial x_\alpha}\right)\frac{dx_\alpha}{dt} + \left(\frac{\partial f}{\partial \xi_\alpha}\right)\frac{d\xi_\alpha}{dt} \quad (2\text{-}4)$$

The derivative of x over time can be expressed as velocity $\frac{dx_\alpha}{dt} = \xi_\alpha$ and from Newton's second law it is known that acceleration is equal to external forces divided by mass or body force density divided by density $\frac{d\xi_\alpha}{dt} = \frac{F_\alpha}{\rho}$.

In addition, the total differential $\frac{df}{dt}$ is expressed as Ω which is known as the collision operator. Inserting them into equation 2-4, the Boltzmann equation is finally obtained.

$$\Omega(f) = \frac{\partial f}{\partial t} + \xi_\alpha \frac{\partial f}{\partial x_\alpha} + \frac{F_\alpha}{\rho}\frac{\partial f}{\partial \xi_\alpha} \quad (2\text{-}5)$$

It can be seen that equation 2-5 resembles an advection equation. The collision operator on the left is considered as source term, while the first two terms on the right are the advection terms. The third term represents the external forces acting on the fluid, which will be considered to be zero. [3, p. 21]

The question is how to obtain this source term. Since particle collision effects are very complex to reproduce, there is the need for an approximation which conserves in mass and momentum, represents the influence of particle collisions sufficiently but still is not overly complex to implement. In literature, multiple approaches for approximating $\Omega(f)$ exist but the most widely used is the one introduced by Bhatnagar, Gross and Krook (BGK), which satisfies all requirements listed. [4, pp. 17-18], [5]

$$\Omega(f) = -\frac{1}{\tau}\left(\underbrace{f - f^{eq}}_{f^{neq}}\right) \tag{2-6}$$

In the BGK collision operator τ is meant to be the relaxation time or relaxation factor, f^{eq} is the equilibrium distribution function and f^{neq} is the non-equilibrium distribution function which will be explained in depth later on. If equation 2-6 is combined with equation 2-5, an approximation of the Boltzmann equation is found.

$$\frac{\partial f}{\partial t} + \xi_\alpha \frac{\partial f}{\partial x_\alpha} = -\frac{1}{\tau}(f - f^{eq}) \tag{2-7}$$

If a force-free, homogeneous system in steady state is considered, the advection terms of equation 2-7 vanish and the particle distribution function becomes the equilibrium distribution function. Therefore, whenever a system is left alone for long enough, it will reach equilibrium. [3, p. 64]

This equilibrium state is described by the Maxwell-Boltzmann distribution and is dependent on the temperature T, the ideal gas constant R_g and the relative velocity υ which is the difference of the microscopic ξ and macroscopic u velocity. [3, pp. 18-20]

$$\upsilon(x,t) = \xi(x,t) - u(x,t) \tag{2-8}$$

$$f^{eq}(x,|\upsilon|,t) = \rho\left(\frac{1}{2\pi R_g T}\right)^{\frac{3}{2}} e^{\frac{-|\upsilon|^2}{2R_g T}} \tag{2-9}$$

Viggen [6, pp. 59-61] showed, that by taking the zeroth and the first moment of the Boltzmann equation 2-5, the continuity equation and the Cauchy momentum equation can be obtained. Some useful correlations which emerge from these derivations will be outlined in the following paragraphs.

The zeroth moment is received by integrating the Boltzmann equation over velocity space.

$$\int \Omega(f) d^3\xi = \frac{\partial}{\partial t}\int f d^3\xi + \frac{\partial}{\partial x_\alpha}\int f\,\xi_\alpha d^3\xi + \frac{F_\alpha}{\rho}\int \frac{\partial f}{\partial \xi_\alpha} d^3\xi \tag{2-10}$$

Since the collision is required to conserve mass and momentum, the zeroth and first moment of the collision operator needs to be zero.

$$\int \Omega(f) d^3\xi = 0 \qquad (2\text{-}11)$$

$$\int \xi_\alpha \Omega(f) d^3\xi = 0 \qquad (2\text{-}12)$$

From equations 2-2 and 2-3, the zeroth and first moment of the distribution function are known. The zeroth moment of the force term as derived by Viggen [6, p. 60] equals zero. As a consequence, equation 2-10 can be converted in the continuity equation 2-13.

$$\frac{\partial \rho}{\partial t} + \frac{\partial(\rho u_\alpha)}{\partial x_\alpha} = 0 \qquad (2\text{-}13)$$

The first moment of the Boltzmann equation is computed by multiplication with the microscopic velocity ξ_α and subsequent integration over velocity space.

$$\int \xi_\alpha \Omega(f) d^3\xi = \frac{\partial}{\partial t}\int \xi_\alpha f \, d^3\xi + \frac{\partial}{\partial x_\beta}\int \xi_\alpha \xi_\beta f \, d^3\xi + \frac{F_\beta}{\rho}\int \xi_\alpha \frac{\partial f}{\partial \xi_\beta} d^3\xi \qquad (2\text{-}14)$$

The collision term can be set to zero using equation 2-12 and the first term on the right is known from equation 2-3. The last two expressions can be evaluated as demonstrated by Viggen [6, p. 60] who determined the first moment of the force term by multidimensional integration by parts.

$$\int \xi_\alpha \frac{\partial f}{\partial \xi_\beta} d^3\xi = -\rho \delta_{\alpha\beta} \qquad (2\text{-}15)$$

The second term on the right of equation 2-14, contains the second moment of the distribution function $\Pi_{\alpha\beta}$, which can be rewritten using the definition of the relative velocity from equation 2-8.

$$\int \xi_\alpha \xi_\beta f \, d^3\xi = \int (u_\alpha + v_\alpha)\big(u_\beta + v_\beta\big) f \, d^3\xi = \int \big(u_\alpha u_\beta + u_\alpha v_\beta + u_\beta v_\alpha + v_\alpha v_\beta\big) f \, d^3\xi \qquad (2\text{-}16)$$

Since the integral of the relative velocity multiplied by the particle distribution function must be zero to hold equation 2-3, the second moment of the distribution is expressed as in equation 2-18.

$$\int v_\alpha f \, d^3\xi = 0 \qquad (2\text{-}17)$$

$$\Pi_{\alpha\beta} = \int \xi_\alpha \xi_\beta f \, d^3\xi = \rho u_\alpha u_\beta + \int v_\alpha v_\beta f \, d^3\xi \qquad (2\text{-}18)$$

$\Pi_{\alpha\beta}$ which is also known as the momentum flux tensor, is composed by the macroscopic flow of momentum $\rho u_\alpha u_\beta$ and the diffusion of momentum or total stress tensor $\sigma_{\alpha\beta}$.

$$\sigma_{\alpha\beta} = -\int v_\alpha v_\beta f \, d^3\xi \qquad (2\text{-}19)$$

This total stress is summarizing the viscous stresses $\sigma'_{\alpha\beta}$ and the pressure p:

$$\sigma_{\alpha\beta} = \sigma'_{\alpha\beta} - p\delta_{\alpha\beta} \qquad (2\text{-}20)$$

Considering the previous correlations, from the first moment of the Boltzmann equation, the Cauchy momentum equation can be obtained. [6, pp. 59-61]

$$\frac{\partial(\rho u_\alpha)}{\partial t} + \frac{\partial(\rho u_\alpha u_\beta)}{\partial x_\beta} = \frac{\partial \sigma_{\alpha\beta}}{\partial x_\beta} + F_\alpha \qquad (2\text{-}21)$$

2.1.2 Discretization in Space, Time and Velocity-Space

In chapter 2.1.1 all formulae have been treated from a continuous point of view. In LBM the domain is divided into a uniform cubic grid and the equations need to be solved at each grid node. The focus now will be switched to a discretized approach.

The LBM can be applied in one, two or three dimensions. Dependent on that, the lattice arrangement and number of discrete particle distribution functions varies. Although, there exists a great diversity of lattice arrangements, some are more widely employed. For the one-dimensional case, displayed in Fig. 2-1, the D1Q3 scheme is normally used. D*d*Q*q* is a common naming convention for lattice arrangements in LBM, where *d* gives dimensions and *q* the number of distribution functions used. For the one-dimensional (1D) case, looking at the node in the middle, particles are able to stream to the right (1), to the left (2) or rest (0). Therefore, three distribution functions are needed to describe the behavior of the fluid. The movement of the particles, with a discrete microscopic velocity c_i, takes Δt time for a distance Δx between two neighboring nodes. [3, pp. 86-87]

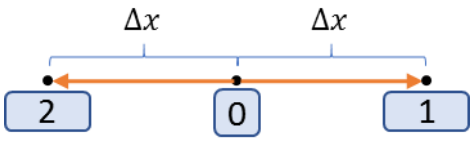


Fig. 2-1: The lattice arrangement for the D1Q3 scheme

The D2Q9 scheme is presented in Fig. 2-2 which is most popular in two-dimension (2D). This work deals with the D3Q19 lattice, illustrated in Fig. 2-3. In this case, 18 values give the distribution functions of particles which propagate from the initial node to their neighbors and one describes the resting state.

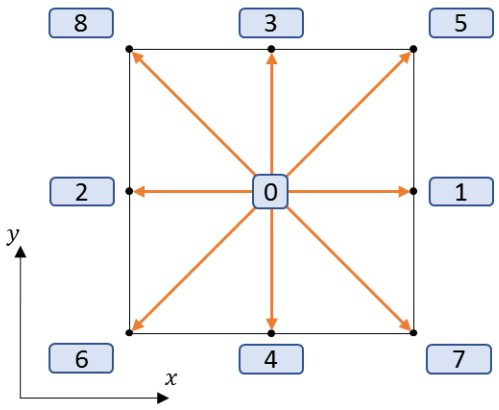


Fig. 2-2: Lattice arrangement for the D2Q9 scheme

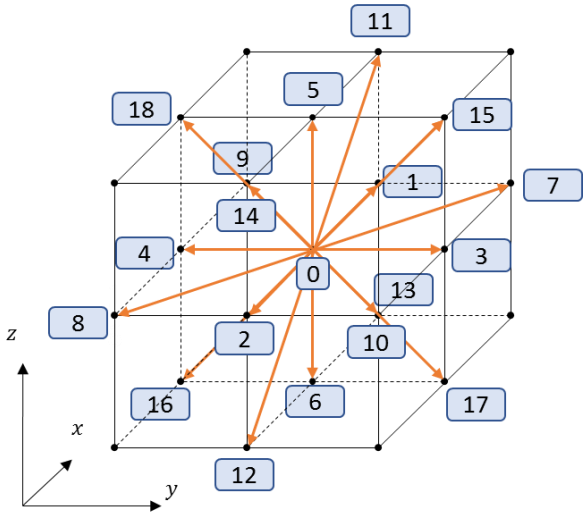


Fig. 2-3: Lattice arrangement for the D3Q19 scheme

Microscopic velocities c_i are listed in Tab. 2-1 and are also called lattice vectors because they describe exactly the length and direction of the vectors shown in Fig. 2-3. Furthermore, in Tab. 2-1 the weighting factors w_i for each distribution function can be found. These weights originate from the discretization of the equilibrium distribution function, which will not be covered in this work and can be found in the book of Krüger et al. [3, pp. 74-83].

Tab. 2-1: The D3Q19 velocity set [3, p. 89]

i	0	1	2	3	4	5	6
c_i	$\begin{pmatrix}0\\0\\0\end{pmatrix}$	$\begin{pmatrix}1\\0\\0\end{pmatrix}$	$\begin{pmatrix}-1\\0\\0\end{pmatrix}$	$\begin{pmatrix}0\\1\\0\end{pmatrix}$	$\begin{pmatrix}0\\-1\\0\end{pmatrix}$	$\begin{pmatrix}0\\0\\1\end{pmatrix}$	$\begin{pmatrix}0\\1\\-1\end{pmatrix}$
w_i	$\frac{1}{3}$	$\frac{1}{18}$	$\frac{1}{18}$	$\frac{1}{18}$	$\frac{1}{18}$	$\frac{1}{18}$	$\frac{1}{18}$

i	7	8	9	10	11	12	13	14	15	16	17	18
c_i	$\begin{pmatrix}1\\1\\0\end{pmatrix}$	$\begin{pmatrix}-1\\-1\\0\end{pmatrix}$	$\begin{pmatrix}1\\-1\\0\end{pmatrix}$	$\begin{pmatrix}-1\\1\\0\end{pmatrix}$	$\begin{pmatrix}1\\0\\1\end{pmatrix}$	$\begin{pmatrix}-1\\0\\-1\end{pmatrix}$	$\begin{pmatrix}1\\0\\-1\end{pmatrix}$	$\begin{pmatrix}-1\\0\\1\end{pmatrix}$	$\begin{pmatrix}0\\1\\1\end{pmatrix}$	$\begin{pmatrix}0\\-1\\-1\end{pmatrix}$	$\begin{pmatrix}0\\1\\-1\end{pmatrix}$	$\begin{pmatrix}0\\-1\\1\end{pmatrix}$
w_i	$\frac{1}{36}$	$\frac{1}{36}$	$\frac{1}{36}$	$\frac{1}{36}$	$\frac{1}{36}$	$\frac{1}{36}$	$\frac{1}{36}$	$\frac{1}{36}$	$\frac{1}{36}$	$\frac{1}{36}$	$\frac{1}{36}$	$\frac{1}{36}$

Together with the velocity set given in Tab. 2-1, all necessary parameters are present to calculate the discrete equilibrium distribution function f_i^{eq} which is given in its standard form in equation 2-22.

$$f_i^{eq}(x,t) = w_i\rho \cdot \left(1 + \frac{u \cdot c_i}{c_s^2} + \frac{(u \cdot c_i)^2}{2c_s^4} - \frac{u \cdot u}{2c_s^2}\right) \quad (2\text{-}22)$$

c_s is the speed of sound which will be explained in section 2.1.3. The implemented equilibrium function in equation 2-26 and subsequently, all correlated formulas in this work differ from the standard form of the equation in two points. First, the incompressible form of the LBM introduced by He and Luo [7] is utilized, which is given in 2-23.

$$f_i^{eq}(x,t) = w_i\rho + w_i\rho_0 \cdot \left(\frac{u \cdot c_i}{c_s^2} + \frac{(u \cdot c_i)^2}{2c_s^4} - \frac{u \cdot u}{2c_s^2}\right) \quad (2\text{-}23)$$

The constant macroscopic fluid density ρ_0, is given by an input variable. The second deviation from the standard formulation is done to decrease round off errors. By subtracting the term $w_i\rho_0$ from the distribution functions f_i and the equilibrium distribution functions f_i^{eq}, their values get closer to the value of the non-equilibrium distribution function f_i^{neq}. To distinguish between the original distribution function and the distribution function from Valderhaug [8, p. 31], h_i is introduced as in equation 2-24.

$$h_i = f_i - w_i\rho_0; \quad h_i^{eq} = f_i^{eq} - w_i\rho_0; \quad h_i^{neq} = f_i^{neq}; \quad (2\text{-}24)$$

From equation 2-24, the equilibrium distribution function can be derived as in equation 2-26. For that reason, the local deviation of the density $\Delta\rho$, given in equation 2-25, is introduced.

$$\Delta\rho = \rho - \rho_0 \quad (2\text{-}25)$$

$$h_i^{eq}(x,t) = w_i\Delta\rho + w_i\rho_0 \cdot \left(\frac{u \cdot c_i}{c_s^2} + \frac{(u \cdot c_i)^2}{2c_s^4} - \frac{u \cdot u}{2c_s^2}\right) \quad (2\text{-}26)$$

The last equation which needs to be discretized is the lattice Boltzmann equation. Since it is approximated with the BGK collision operator, it is known as lattice Bhatnagar, Gross and Krook (LBGK) equation. [3, p. 65]

$$f_i(x + c_i\Delta t, t + \Delta t) = f_i(x,t) - \frac{\Delta t}{\tau}\left(f_i(x,t) - f_i^{eq}(x,t)\right) \quad (2\text{-}27)$$

By utilizing the distribution functions from Valderhaug [8, p. 31], the LBGK equation can be rewritten as in equation 2-28.

$$h_i(x + c_i\Delta t, t + \Delta t) = h_i(x,t) - \frac{\Delta t}{\tau}\left(h_i(x,t) - h_i^{eq}(x,t)\right) \quad (2\text{-}28)$$

2.1.3 Dimensionless Quantities and Lattice Units

From a scientific perspective it is usually favorable to deal with dimensionless quantities to be able to compare and categorize different systems. The Reynolds number Re is one of the most important dimensionless quantities in fluid dynamics. It is giving the flow regime by laying out the proportion of inertial to viscous forces.

$$Re = \frac{u \cdot l_c}{\nu} \quad (2\text{-}29)$$

Due to the fact that the LBM is only valid for incompressible fluids, the Mach number Ma plays a crucial role. A fluid can be seen as incompressible with $Ma \leq 0.1$.

$$Ma = \frac{u}{c_s} \quad (2\text{-}30)$$

Likewise, the Knudsen number Kn, which is the ratio between the mean free path l_{mfp} of the molecules of the fluid and the characteristic length l_c, needs to be much smaller than unity to guarantee the validity of the Navier-Stokes equations. [3, pp. 13-14]

$$Kn = \frac{l_{mfp}}{l_c} \quad (2\text{-}31)$$

A further common practice is to convert all physical quantities of a system to non-dimensional lattice units and afterwards, reconvert the results into physical units. The nondimensionalization is done by dividing the physical units by factors of the same units. [3, p. 266] As an example, a length in physical units x is divided by the conversion factor of length Δx to obtain the length in lattice units $\tilde{x}$. In the following sections quantities in lattice unis will always be given with a tilde like $\tilde{a}$, all quantities written without tilde are given in physical units.

$$\tilde{x} = \frac{x}{\Delta x} \quad (2\text{-}32)$$

There are various possibilities to define these so-called correction factors of length Δx, time Δt and mass Δm. In this work, the conversion factor of length is obtained by the ratio of the length of the simulation domain in x-direction l_x and the number of nodes in x direction N_x.

$$\Delta x = \frac{l_x}{N_x} \quad (2\text{-}33)$$

There exist two prevailing scaling methods for correlating time and space discretization. The convective, in some works also known as acoustic, scaling relates the time proportional to the space $\Delta t \sim \Delta x$ and the diffusive scaling states the time proportional to the space squared $\Delta t \sim \Delta x^2$ [9]. As it can be seen in equation 2-34, in this work the convective scaling is used.

As Krüger et al. [3, p. 278] determined a limit of the maximum velocity in lattice units $\tilde{u}_{max} < 0.1$ to obtain sufficient accuracy, the input velocity in lattice units $\tilde{u}_{in}$ is fixed to 0.05 and used in the calculation of the temporal conversion factor.

$$\Delta t = \frac{\tilde{u}_{in}}{u_{in}} \cdot \Delta x \qquad (2\text{-}34)$$

Likewise, the density in lattice units $\tilde{\rho}_0$ is set to a constant value. Since Δm is not influencing the other conversion factors, the choice of $\tilde{\rho}_0$ is not limited by any rules for accuracy, efficiency or stability. Krüger et al. [3, p. 280] recommend to choose unity. The mass conversion factor is computed as in equation 2-35.

$$\Delta m = \frac{\rho_0}{\tilde{\rho}_0} \cdot \Delta x^3 \qquad (2\text{-}35)$$

The speed of sound in lattice units is a constant value, depending on the used velocity set. For the D2Q9 and the D3Q19 it is given as in 2-36.

$$\tilde{c_s} = \sqrt{\frac{1}{3}} \qquad (2\text{-}36)$$

2.1.4 The Lattice Boltzmann Algorithm

The lattice Boltzmann algorithm is supremely simple for force-free, single phase, single component systems without thermal transport. The more the complexity of the system to compute increases, the higher the complexity of the algorithm gets. For that reason, in this work only the basic LBM algorithm is applied. The very principle consists of sequenced collision and streaming events at every node of the domain as shown in Fig. 2-4 which follow the principles of kinetic theory and, therefore, are able to reproduce the required velocity field. Although the implementation is done using the D3Q19 scheme, the figures in this and the subsequent chapter are based on the two-dimensional D2Q9, as the underlying mechanisms are the same.

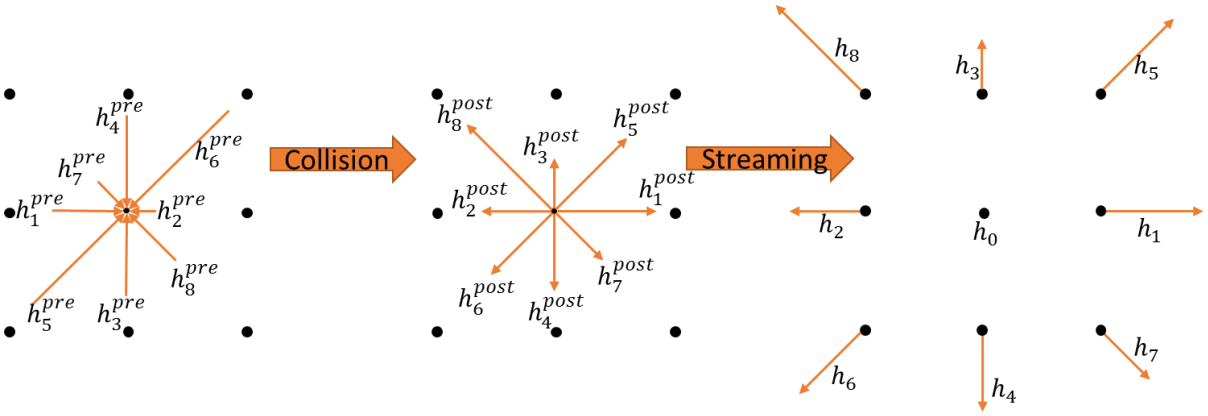


Fig. 2-4: Illustration of particle displacement following the "collide and stream" concept

At first, the macroscopic density ρ and the macroscopic velocity u_α are calculated from the zeroth and first moment of the distribution functions h_i. In the standard formulation of the LBM, the total density ρ would appear in equation 2-37 and equation 2-38. However, due to the adaption of the LBM mentioned in section 2.1.2, the macroscopic values are obtained using the constant fluid density ρ_0 and its local deviation $\Delta\rho$.

$$\Delta\rho(x,t) = \sum_i h_i(x,t) \qquad (2\text{-}37)$$

$$\rho_0(x,t)u_\alpha(x,t) = \sum_i c_{i\alpha}h_i(x,t) \qquad (2\text{-}38)$$

$$\rho = \Delta\rho + \rho_0 \qquad (2\text{-}39)$$

All equilibrium distribution functions h_i^{eq} are calculated as in equation 2-26, then the collision step is executed. The post collision distribution functions $h_i^{post-coll}$ are obtained by applying the LBGK as it is presented in equation 2-40.

$$h_i^{post-coll}(x,t) = h_i^{pre-coll}(x,t) - \frac{\Delta t}{\tau}\left(h_i^{pre-coll}(x,t) - h_i^{eq}(x,t)\right) \qquad (2\text{-}40)$$

In the subsequent streaming step, the post-collision distribution functions $h_i^{post-coll}$ are propagated to their neighboring nodes according to their velocity vectors c_i.

$$h_i(x + c_i\Delta t, t + \Delta t) = h_i^{post-coll}(x,t) \qquad (2\text{-}41)$$

The next step is to apply boundary conditions to the system, in order to constrain the domain and define certain properties at these constraints. Depending on the kind of boundary condition, this is either done in a separate procedure or concurrent with the streaming. Several possible boundary conditions are explained in section 2.1.5.

Finally, the internal time t is propagated by Δt and the procedure is repeated. [3, p. 67]

2.1.5 Boundary Conditions

For the lattice Boltzmann method, a variety of boundary conditions has been developed in order to tackle all kind of challenges. The following section will explain those which were used in the simulation for the validation of this project.

The simulation domain is always of rectangular shape, no matter whether a cylindrical or cuboidal reactor is simulated. To define the geometry, during the initialization procedure boundary conditions are assigned to certain nodes. For instance, all nodes which lie outside a certain radius are marked as solid to obtain a cylindric reactor wall and, therefore, do not need to be considered in the fluid calculation.

2.1.5.1 Mid-Grid Bounce Back Boundary Condition

For all solid and resting walls, the mid-grid bounce back scheme is used. As its name already reveals, the actual boundary is placed exactly between a node marked as liquid and a node marked as solid. During the streaming step, particles collide with the wall and bounce back with inversed velocity vectors as it can be seen in Fig. 2-5. This ensures a “no-slip boundary condition with zero velocity at the wall” [10, p. 31] which signifies that the tangential velocity at the boundary equals zero. In terms of distribution functions this means that the values of all post collision distribution functions $h_i^{post-coll}$ of boundary nodes in the liquid x_b which would end up at a solid node after streaming are assigned to their subtend distribution functions $h_{\bar{\imath}}(x_b, t + \Delta t)$.

$$h_{\bar{\imath}}(x_b, t + \Delta t) = h_i^{post-coll}(x_b, t) \qquad (2\text{-}42)$$

In equation 2-42 the index $\bar{\imath}$ stands for the subtend particle distribution of i. $c_{\bar{\imath}} = -c_i$ [10, p. 31]

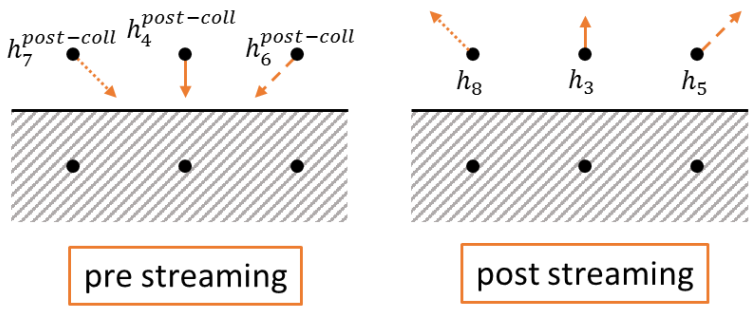


Fig. 2-5: Schematic representation of the mid-grid bounce back

2.1.5.2 Free Slip Boundary Condition

For the boundary between liquid and gas at the head of the stirred tank a free slip boundary condition is applied. This implies that the velocity, orthogonal to the surface is zero and at the same time no restrictions on the tangential velocity are made. During advection, particles move from their node of origin to the surface where they are reflected specularly and end up at their next neighboring node as it can be seen in Fig. 2-6. The bounce is considered purely elastic and therefore, the magnitude of velocity is not changed.

$$h_j\left(x_b + c_{j,t}\Delta t, t + \Delta t\right) = h_i^{post-coll}(x_b, t) \qquad (2\text{-}43)$$

However, in the resulting velocity vector c_j the orthogonal velocity component is reversed from c_i, as it is done in 2-44 and the tangential components are the same. [3, p. 207]

$$c_{j,n} = -c_{i,n} \qquad (2\text{-}44)$$

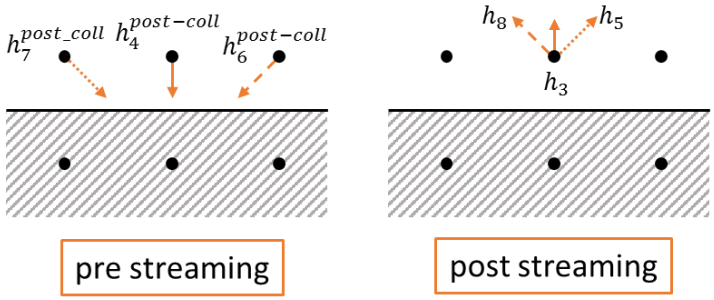


Fig. 2-6: Schematic representation of the free slip boundary condition

2.1.5.3 Moving Wall Boundary Condition

In the case of moving components such as stirrer blades, momentum is transferred to the fluid. Lallemand and Luo [11] proposed a quite complex interpolation scheme for a moving wall at an arbitrary position. They furthermore showed that treating the nodes associated to the stirrer blades such as those of the fluid leads to satisfying results either while being easy to implement. This means that streaming and collision events are performed at a constant velocity of predefined value u_w.

In practice, the equilibrium distribution functions h_i^{eq} of the stirrer nodes are computed by using the wall velocity u_w and the local pre-collision density ρ_b. Thereupon, the velocity distributions are relaxed to this equilibrium functions during collision.

$$h_i^{eq}(x_b, t) = h_i^{eq}(\rho_b, u_w) \qquad (2\text{-}45)$$

2.1.5.4 Modified Bounce Back Velocity Boundary

To be able to simulate pipe flow, boundary flow conditions based on the concept of bounce back, as described in section 2.1.5.1, were chosen, since they are very simple but lead to sufficient results. To establish a Dirichlet condition for a wall velocity u_w, the second term on the right of equation 2-46 is subtracted from the reflected velocity distribution. For the density at the wall ρ_w, the local fluid density ρ_b at the boundary node x_b is chosen. Again, the boundary is set to be $\frac{\Delta x}{2}$ away from the boundary node. [3, p. 200]

$$h_{\bar{\imath}}(x_b, t + \Delta t) = h_i^{post-coll}(x, t) - 2w_i\rho_w \frac{c_i u_w}{c_s^2} \quad (2\text{-}46)$$

2.1.5.5 Anti-Bounce Back Pressure Boundary

The pressure outlet at the end of the pipe is performed by a so-called anti-bounce back. Just as in the other bounce back based boundary conditions, the surface is displaced by $\frac{\Delta x}{2}$ from the boundary node x_b. The "anti" origins from the reversed sign of the post collision distribution function $h_i^{post-coll}$. The required pressure at the outlet is imposed by ρ_w since $p = \rho \cdot c_s^2$, as it will be illustrated in section 2.1.6. However, no information about the velocity at the outlet is known. Therefore, the known velocities at the boundary and its subsequent inward node are extrapolated, as shown in equation 2-48. [3, p. 200]

$$h_{\bar{\imath}}(x_b, t + \Delta t) = -h_i^{post-coll}(x_b, t) - 2w_i\rho_0 + 2w_i\rho_w + 2w_i\rho_0 \left[1 + \frac{(c_i u_w)^2}{2c_s^4} - \frac{u_w^2}{2c_s^2}\right] \quad (2\text{-}47)$$

$$u_w = u(x_b) + \frac{1}{2}[u(x_b) - u(x_b + 1)] \quad (2\text{-}48)$$

2.1.5.6 Absorbing Layer

In the simulation of pipe flow, pressure waves are induced into the system due to the reflecting nature of the boundary flow conditions. This effect immensely increases the required time to reach a steady state. A simple counteraction is to insert an absorbing or "sponge" layer in front of the outlet which absorbs the pressure waves instead of reflecting them. The absorbing behavior is achieved by artificially increasing the viscosity to a very high value as proposed by Vergnault et al. [12]. In this approach, the relaxation factor τ is manipulated to change as a quadratic function of the normalized length of the absorbing layer d. This can be done as the relaxation factor τ is according to equation 2-76 directly influencing the viscosity of the system.

$$\tau = \frac{3\nu+0.5}{1-0.999d^2}\ for\ d \leq 1 \quad (2\text{-}49)$$

$$\tau = \frac{3\nu+0.5}{0.001}\ for\ d > 1 \quad (2\text{-}50)$$

Increasing the relaxation factor, as it is done in this simple approximation, leads to non-physical results in the absorbing layer. A method for initializing the system which can prevent the pressure waves and accelerate convergence has yet to be developed.

2.1.6 Macroscopic Moments

As outlined in section 2.1.4, the macroscopic variables of density ρ and velocity u_α can be calculated from the first two moments $\overline{\Pi}_0$ and $\overline{\Pi}_\alpha$ of the adapted discrete velocity distribution functions h_i. Moments, which are written with a line on top are obtained from the adapted distribution functions h_i presented in the work of Valderhaug [8, p. 31], and therefore, differ from the moments obtained from the original distribution functions f_i.as it can be seen in equations 2-51 to 2-54. Furthermore, in this section the incompressible LBM introduced by He and Luo [7] is applied.

$$\Pi_0 = \sum_i f_i = \rho(x,t) \quad (2\text{-}51)$$

$$\Pi_\alpha = \sum_i c_{i\alpha} f_i = \rho_0(x,t) u_\alpha(x,t) \quad (2\text{-}52)$$

$$\overline{\Pi}_0 = \sum_i h_i = \Delta\rho(x,t) \quad (2\text{-}53)$$

$$\overline{\Pi}_\alpha = \sum_i c_{i\alpha} h_i = \rho_0(x,t) u_\alpha(x,t) \quad (2\text{-}54)$$

Furthermore, in chapter 2.1.1 it was shown that the first two moments of the collision operator Π_0^{neq} and Π_α^{neq}, the so-called non-equilibrium moments, need to be zero to conserve mass and momentum.

$$\Pi_0^{neq} = \sum_i \Omega_i = \sum_i \left(f_i - f_i^{eq}\right) = \sum_i f_i^{neq} = 0 \quad (2\text{-}55)$$

$$\Pi_\alpha^{neq} = \sum_i c_{i\alpha}\Omega_i = \sum_i c_{i\alpha}\left(f_i - f_i^{eq}\right) = \sum_i c_{i\alpha} f_i^{neq} = 0 \quad (2\text{-}56)$$

From equation 2-24, it is known that the non-equilibrium distribution functions f_i^{neq} are equal to the adapted non-equilibrium distribution functions h_i^{neq}. Therefore, equations 2-55 and 2-56 can be rewritten to obtain equations 2-57 and 2-58.

$$\overline{\Pi}_0^{neq} = \sum_i \Omega_i = \sum_i \left(h_i - h_i^{eq}\right) = \sum_i h_i^{neq} = 0 = \Pi_0^{neq} \quad (2\text{-}57)$$

$$\overline{\Pi}_\alpha^{neq} = \sum_i c_{i\alpha}\Omega_i = \sum_i c_{i\alpha}\left(h_i - h_i^{eq}\right) = \sum_i c_{i\alpha} h_i^{neq} = 0 = \Pi_\alpha^{neq} \quad (2\text{-}58)$$

This bears the consequence, that the macroscopic moments Π_0, Π_α $\overline{\Pi}_0$ and $\overline{\Pi}_\alpha$ are equal to their equilibrium moments.

$$\Pi_0 = \Pi_0^{eq} = \sum_i f_i = \sum_i f_i^{eq} \quad (2\text{-}59)$$

$$\Pi_\alpha = \Pi_\alpha^{eq} = \sum_i c_{i\alpha} f_i = \sum_i c_{i\alpha} f_i^{eq} \quad (2\text{-}60)$$

$$\overline{\Pi}_0 = \overline{\Pi}_0^{eq} = \sum_i h_i = \sum_i h_i^{eq} \quad (2\text{-}61)$$

$$\overline{\Pi}_\alpha = \overline{\Pi}_\alpha^{eq} = \sum_i c_{i\alpha} h_i = \sum_i c_{i\alpha} h_i^{eq} \quad (2\text{-}62)$$

However, obtaining the second order moment $\Pi_{\alpha\beta}$, which appeared in equation 2-18 as the momentum flux tensor and the adapted second order moment $\overline{\Pi}_{\alpha\beta}$ is not as straight forward as it was for Π_0 and Π_α.

The equilibrium part of the second order moment $\Pi_{\alpha\beta}^{eq}$, can be obtained by solving equation 2-63 with the use of the original equilibrium distribution function f_i^{eq} from equation 2-22 and the isotropy conditions of the velocity sets illustrated in the appendix with equations 7-1 to 7-6. [3, p. 93]

$$\Pi_{\alpha\beta}^{eq} = \sum_i c_{i\alpha} c_{i\beta} f_i^{eq} \quad (2\text{-}63)$$

This leads to an explicit form of the second order equilibrium momentum of:

$$\Pi_{\alpha\beta}^{eq} = \rho_0 u_\alpha u_\beta + \rho c_s^2 \delta_{\alpha\beta} \quad (2\text{-}64)$$

The second order moment of the adapted equilibrium distribution function $\overline{\Pi}_{\alpha\beta}^{eq}$ can be found in its explicit form, given in equation 2-66, by solving equation 2-65 with the adapted equilibrium distribution function h_i^{eq} from equation 2-26 and the isotropy conditions of the velocity sets.

$$\overline{\Pi}_{\alpha\beta}^{eq} = \sum_i c_{i\alpha} c_{i\beta} h_i^{eq} \quad (2\text{-}65)$$

$$\overline{\Pi}_{\alpha\beta}^{eq} = \rho_0 u_\alpha u_\beta + \Delta\rho c_s^2 \delta_{\alpha\beta} \quad (2\text{-}66)$$

To receive the non-equilibrium part of the second order moment $\Pi_{\alpha\beta}^{neq}$, Krüger et al. [3, pp. 106-112] did the Chapman-Enskog analysis. The Chapman-Enskog analysis is used to evidence that the lattice Boltzmann equation solves the Navier-Stokes equation. It is based on a perturbation expansion of f_i around f_i^{eq} by an expansion parameter ϵ of order of the Knudsen number Kn.

$$f_i = f_i^{eq} + \epsilon f_i^{(1)} + \epsilon^2 f_i^{(2)} + \ldots \quad (2\text{-}67)$$

$$\epsilon^n = \mathcal{O}(Kn^n) \quad (2\text{-}68)$$

In summary, during the analysis the lattice Boltzmann equation is first Taylor expanded, then the time derivatives are expanded and afterwards the expanded particle distributions from equation 2-67 are inserted. By subsequent separation of the terms of the expanded equation by orders of Kn and generation of zeroth to second order moments of these separated formulations, the mass and momentum equations can be assembled. After reversing the expansion of the time derivatives, the viscous stress tensor $\sigma'_{\alpha\beta}$ is identified with equation 2-69.

$$\sigma'_{\alpha\beta} = -\left(1 - \frac{\Delta t}{2\tau}\right)\Pi_{\alpha\beta}^{(1)} = -\left(1 - \frac{\Delta t}{2\tau}\right)\Pi_{\alpha\beta}^{neq} = -\left(1 - \frac{\Delta t}{2\tau}\right)\bar{\Pi}_{\alpha\beta}^{neq} \quad (2\text{-}69)$$

The explicit formulation of $\Pi_{\alpha\beta}^{neq}$ was shown to be as in 2-70, where the second term represents an error term is neglected if $u^2 \ll c_s^2$.

$$\Pi_{\alpha\beta}^{(1)} = -\rho_0 c_s^2 \tau\left(\partial_\beta^{(1)} u_\alpha + \partial_\alpha^{(1)} u_\beta\right) + \tau\partial_\gamma^{(1)}(\rho_0 u_\alpha u_\beta u_\gamma) \quad (2\text{-}70)$$

As Krüger et al. [3] stated, only the two lowest orders of Kn are needed to represent the Navier-Stokes equation, $f^{(1)} = f^{neq}$.

$$f_i^{neq} = f_i - f_i^{eq} = h_i - h_i^{eq} = h_i^{neq} \quad (2\text{-}71)$$

$$\Pi_{\alpha\beta}^{neq} = -\rho_0 c_s^2 \tau\left(\frac{\partial u_\alpha}{\partial x_\beta} + \frac{\partial u_\beta}{\partial x_\alpha}\right) = \sum_i c_{i\alpha} c_{i\beta} f_i^{neq} \quad (2\text{-}72)$$

Utilizing the adapted distribution functions h_i, the explicit form of their second order non-equilibrium moments $\bar{\Pi}_{\alpha\beta}^{neq}$ can be written explicitly as in equation 2-73.

$$\bar{\Pi}_{\alpha\beta}^{neq} = -\rho_0 c_s^2 \tau\left(\frac{\partial u_\alpha}{\partial x_\beta} + \frac{\partial u_\beta}{\partial x_\alpha}\right) = \sum_i c_{i\alpha} c_{i\beta} h_i^{neq} = \Pi_{\alpha\beta}^{neq} \quad (2\text{-}73)$$

From equation 2-72, also the strain rate tensor $S_{\alpha\beta}$ given in equation 2-74 can be correlated to the non-equilibrium moments of second order like it is shown in equation 2-75. [13]

$$S_{\alpha\beta} = \frac{1}{2}\left(\frac{\partial u_\alpha}{\partial x_\beta} + \frac{\partial u_\beta}{\partial x_\alpha}\right) \quad (2\text{-}74)$$

$$S_{\alpha\beta} = -\frac{1}{2\rho_0 c_s^2 \tau} \Pi_{\alpha\beta}^{neq} \quad (2\text{-}75)$$

Furthermore, two very important correlations are obtained in the Chapman-Enskog analysis during the derivation of the Navier-Stokes equation from the lattice Boltzmann equation. Firstly, the kinematic viscosity ν is related to the relaxation parameter τ via equation 2-76.

$$\nu = c_s^2 \left(\tau - \frac{\Delta t}{2}\right) \quad (2\text{-}76)$$

Secondly, it is shown that the equation of state for the lattice Boltzmann equation is given by the proportional relation of pressure p and density ρ. [3, pp. 106-112]

$$p = c_s^2 \rho \quad (2\text{-}77)$$

2.2 GPU Based Computing via CUDA

The immense potential of GPUs, originally designed to unburden the CPU from the enormous workload which arise of the increasing demands on 2D and 3D graphics, can be exploit for applications with similar properties. In graphical applications the workload is large, and therefore, “GPUs must deliver an enormous amount of compute performance to satisfy the demand of complex real-time applications.” [14]. Further, GPUs are prioritizing concurrent throughput, are trimmed for parallelism and as they are using a common global memory for all cores, the need for synchronizing calculation results is removed. To summarize, GPUs were developed to deal with a great amount of data points and do simple, similar and independent operations on them in parallel. Fortunately, the GPU’s driver software was further developed to enable programmers to make use of this properties besides rendering tasks. [14]

In 2006 the GPU producer NVIDIA released the Compute Unified Device Architecture (CUDA) to enable user programmable general- purpose GPU computing. Further, CUDA C/C++, a C/C++ like programming language with useful extensions exactly tailored for GPU programming, and a suitable compiler were released. Since then, programmers are able to exploit the advantages of GPU hardware without being able to use graphics API. [14], [15, pp. 6-7]

Since GPUs are not applicable for all kind of tasks, heterogeneous system of one or more CPUs and one or more GPUs are used in order to benefit from the advantages of both. Therefore, all parts of a CUDA C code which should process a great amount of data by performing simple

calculations in parallel are run on GPUs and functions which are intended to run subsequently are executed on the CPU. [15, pp. 13-19]

What causes the main difference between CPU and GPU is the underlying philosophy and the proximate layout. A CPU is assembled of (few) cores with complex control units and elevated demands in data caching to facilitate a wide range of applications with concurrent fast response times. This high rate of specialization together with limited spatial and power resources leads to the small number of cores possible. In contrast, GPUs are assembled from several streaming multiprocessors, each possible to process over hundreds of simple tasks in parallel. As a consequence, spatial expenses on data caching and control are cut short. [16] [17]

A sequence of operations is called a thread. Several of these threads on a GPU are aggregated to a block of threads. These blocks are independent of each other and can be executed from any available multiprocessor, in any order. As a consequence, the execution time is scaled by the number of cores available. Within these blocks, the threads are executed in parallel. Further, the threads can be placed inside a block in one to three dimensions leading to one-, two or three-dimensional blocks like shown in Fig. 2-7. In addition, block itself can be arranged in one to three dimensions forming a grid of blocks. [17]

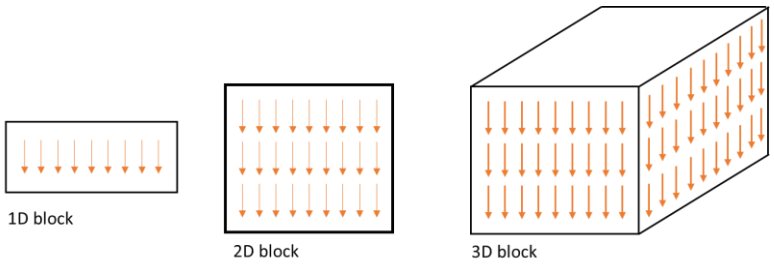


Fig. 2-7: Possible arrangement of threads, illustrated as arrows, in a block of threads

The main extensions that come with CUDA C/C++ are on the one hand kernels, which are functions executed in parallel threads on the GPU and on the other hand extra specifiers, data types and variables. In CUDA the CPU is called host and the GPU is the device. To let the compiler know whether a function is intended to be executed on the host, on the device or on both the specifiers *__host__*, *__device__* and *__global__* are used in front of the function declaration. Further, built-in vector types are available with one to four elements. For instance, a floating-point vector of three elements therefore, can be represented using the *float3* variable type. This vector types can be defined using the function *make_"type"* like

```
float3 vector=make_float3(x,y,z);
```

In addition, some built-in variables regarding the thread structure are available. *blockIdx* and *threadIdx* for instance are both of type *unit3* and give the index of the block inside the grid and the index of the thread inside the block accordingly. [17]

However, the most drastically difference to common C/C++ applications are the aforementioned kernels. They are specified to be global as they are called from the host and executed on the device like in the following example.

```
__global__ void kernel(float var1, float var2, float var3){
      uint  x = threadIdx.x,
            y = blockIdx.x,
            z = blockIdx.y;

      var3[x,y,z]=var1[x,y,z]+var2[x,y,z];
}
```

Further, in the function call it need to be specified how many times the function needs to be executed in parallel. This is done with angle brackets as shown in the following example.

```
kernel<<blocks,threads>>(var1, var2, var3);
```

The variable *blocks* specifies how many blocks are needed in every spatial direction and the variable *threads* is defining how many threads per block are required in every spatial direction. For the example given in Fig. 2-8, *blocks* would be equal to $\binom{3}{2}$ and *threads* is $\binom{9}{3}$ which leads to 162 executions of the kernel in parallel. [15, pp. 23-61]

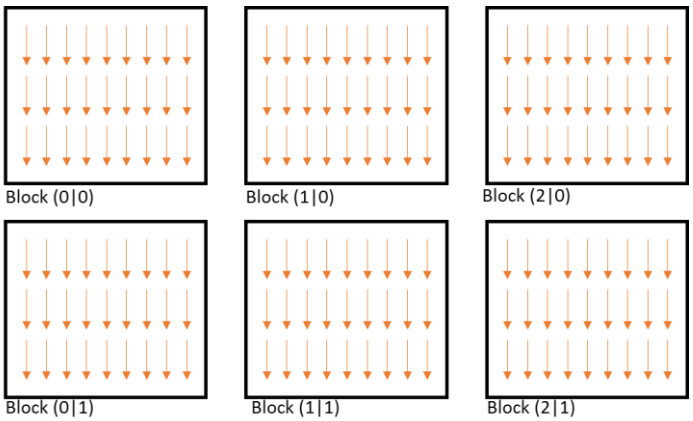


Fig. 2-8: A 2D grid of 2D blocks

From that, the advantage of this approach compared to traditional “CPU only” methods is evident. Especially, if large data sets need to be processed in simple arithmetic steps, like it is done in LBM, heterogeneous applications perform significant better in terms of performance per Euro as well as performance per watt. [15, pp. 1-8]

3 Local Grid Refinement

Although the LBM in its original formulation was meant to be used only in uniform grids, there are certain methods with which a change of grid resolution over the domain is permitted. [18] Overall, there are two different approaches. The first is to have two decoupled grids, a uniform one for the LBM and an arbitrary one refined for the physical space. Here the missing distribution functions on the unstructured grid nodes are obtained by interpolation. The second approach is the one chosen in this work and is based on patches of uniform but refined grid, embedded locally and coupled via certain boundary conditions with the main grid. [19] How the embedding and coupling is performed exactly, is demonstrated in this chapter.

Before starting the chapter, two points need to be mentioned. Firstly, for convenience mainly the two-dimensional case will be used as a reference and for illustration purposes. Secondly, in this work a central node approach was chosen as shown in Fig. 3-1 on the left, which means that the domain is divided in several computational cells with a node positioned exactly in the middle of each. In contrast to that, the central cell approach is placing the nodes on every corner of a cell.

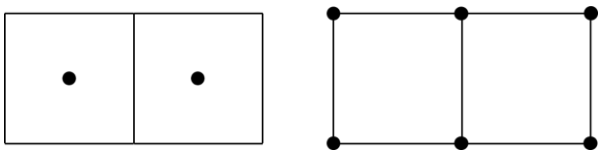


Fig. 3-1: Schematic illustration of the cell and node placement in the central node (l.) and central cell (r.) approach

3.1 Meshing

For the composition of the refined mesh, it can be differentiated between a multi-domain grid shown in Fig. 3-2 and a multi-grid domain illustrated in Fig. 3-3. The multi-domain grid is an assembly of grid patches with different resolution. Like in a puzzle, the parts fit in each other. Regions, which are covered with a finer grid are removed from the coarser domain as it can be seen in Fig. 3-2. This approach is very economical from memory perspective and as Lagrava Sandoval [20, p. 51] stated, leads to better CPU performance.

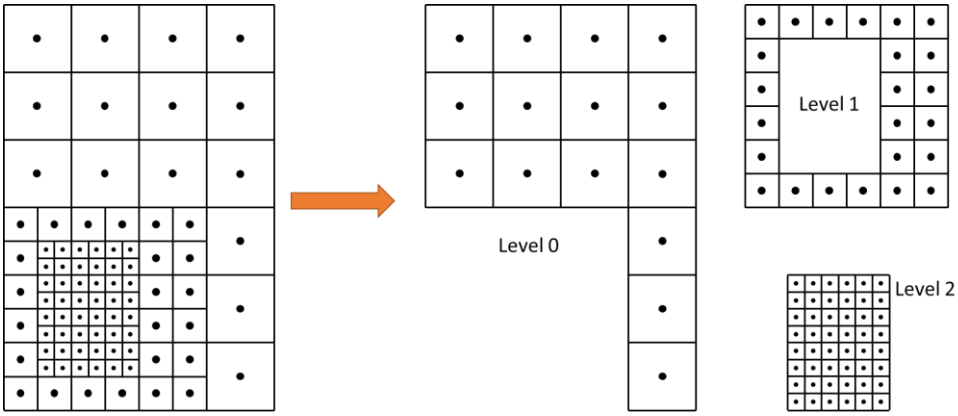


Fig. 3-2: The multi-domain grid as composition of different shaped domains

The multi grid domain in 2D can be seen as grid layers of different resolution piled upon each other and in 3D as nested boxes. This means, that regions represented by fine grids are as well represented on all coarser levels, as displayed in Fig. 3-3. This enables the opportunity to choose between one- or two-way coupling of the grids so that further algorithms like heat transport must not necessarily be integrated in grid refinement but still can work on the lowest grid level. Furthermore, the generation of the domain and grid coupling is significantly easier compared to the multi-domain approach. However, the main reason why the multi-grid domain was chosen in this work, is its suitability for parallelization due to the regular, rectangular shape of the grids. [20, pp. 51-52]

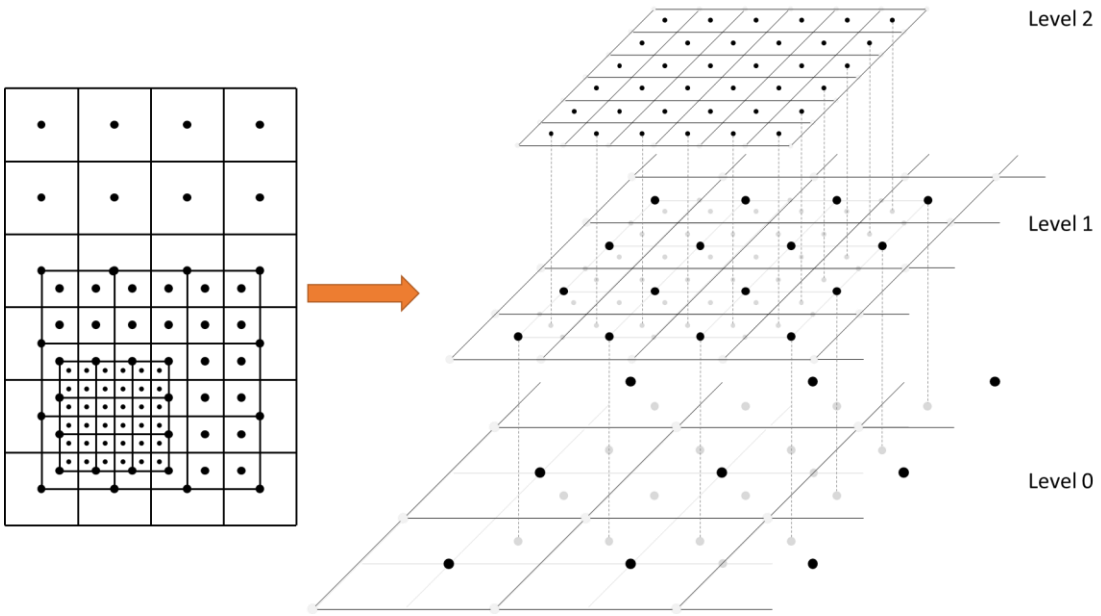


Fig. 3-3: The multi-grid domain as composition of several grid layers

It can be seen in Fig. 3-2 and Fig. 3-3, that the position of the fine cells with respect to the coarse cells is not the same. During research, the three common arrangements, illustrated in Fig. 3-4, were found. The coinciding cell arrangement is applied in the work of Chen, Filippova et al. [19] and Eitel-Amor et al. [21]. In this approach the edges of the fine grid coincide with the edges of the coarse cells. The shifted cell arrangement is used by Schönherr et al. [18] and Geier et al. [22]. As its name reveals the edges of the fine grid are shifted for half of a coarse cell from the cell edge. The coinciding node arrangement is utilized by Lagrava Sandoval [20] and Yu et al. [23] and places the nodes of the coarse grid to coincide with the nodes of the fine grid.

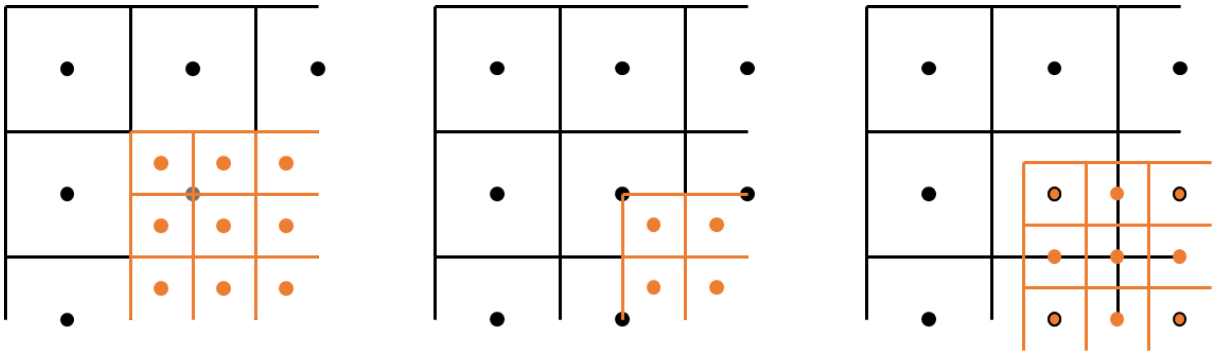


Fig. 3-4: Different grid arrangements: the coinciding cell arrangement (l.), the shifted cell arrangement (m.) and the coinciding node arrangement (r.)

The shifted cell arrangement was chosen to be implemented for two reasons, which will increase in importance when it comes to grid coupling. Firstly, it ensures that the fine grid is always surrounded by at least one row of coarse cells and secondly, four coarse nodes always enclose four explicit relatable fine nodes and therefore, form a local interpolation cell ensuring that interpolation can be done everywhere in the same manner. However, it must be kept in mind that even if the whole domain would be refined, the fine grid is, in each direction, always 1 coarse cell or 2 fine cells smaller than the coarse grid.

3.2 Rescaling of Quantities

In the following sections all quantities belonging to the coarse grid are referenced with index c and all those belonging to the fine grid will use the subscribed f.

In section 2.1.3 the convective scaling was introduced. This correlation is also applied for the refinement of a coarse grid with a fine grid.

$$\frac{\Delta x_c}{\Delta t_c} = \frac{\Delta x_f}{\Delta t_f} = const. \quad (3\text{-}1)$$

Introducing the refinement factor n we get the following relations:

$$\Delta x_c = n \cdot \Delta x_f; \qquad \Delta t_c = n \cdot \Delta t_f \quad (3\text{-}2)$$

$$\frac{\Delta t_c}{\Delta t_f} = \frac{\Delta x_c}{\Delta x_f} = n \quad (3\text{-}3)$$

From equation 2-35 the scaling of the conversion factor for mass Δm is obtained.

$$\frac{\Delta m}{\Delta x^3} = const. \rightarrow \frac{\Delta m_c}{\Delta x_c^3} = \frac{\Delta m_f}{\Delta x_f^3} \quad (3\text{-}4)$$

$$\frac{\Delta m_c}{\Delta m_f} = n^3 \quad (3\text{-}5)$$

In the course of this work the refinement factor n was chosen to be 2. As a consequence, the spatial and temporal division of the computational domain are refined by the factor of 2 in one refinement step. Therefore, in two dimensions one coarse cell gets divided in four fine cells. For three dimensions Fig. 3-5 illustrates the refinement of a coarse cell in eight fine cells.

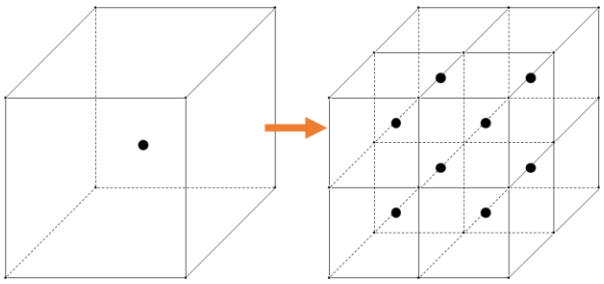


Fig. 3-5: Refinement in space

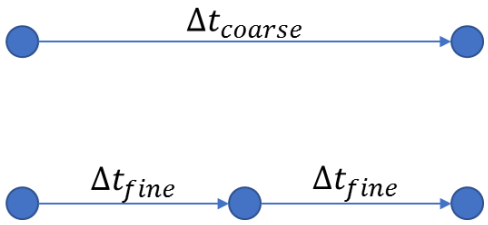


Fig. 3-6: Refinement in time

In Fig. 3-6 it is shown that also the time step is divided according to equation 3-2. As a consequence, to be synchronous in time, for one stream and collide cycle on the coarse grid, two cycles have to be fulfilled on the fine grid.

Due to the changes of spatial and temporal spacing among the grids, the quantities in lattice units $\tilde{x}$ need to be adapted in order to represent the physical quantities x correctly. However, the magnitude of the physical quantities in physical units does not change, no matter what lattice spacing is used. [24]

$$u = u_c = u_f \quad (3\text{-}6)$$

$$\rho = \rho_c = \rho_f \quad (3\text{-}7)$$

$$\nu = \nu_c = \nu_f \tag{3-8}$$

$$S_{\alpha\beta} = S_{\alpha\beta,c} = S_{\alpha\beta,f} \tag{3-9}$$

It can be shown, that the macroscopic velocity in lattice units $\tilde{u}$ is constant over the grids for convective scaling. This applies also for the microscopic velocities $\tilde{c_i}$ and the speed of sound $\tilde{c_s}$. As a consequence, the Mach number, given in equation 2-30 is preserved constant on all refinement stages. [18]

$$u = \tilde{u}_c \frac{\Delta x_c}{\Delta t_c} = \tilde{u}_f \frac{\Delta x_f}{\Delta t_f} \tag{3-10}$$

$$\tilde{u}_c \frac{2 \cdot \Delta x_f}{2 \cdot \Delta t_f} = \tilde{u}_f \frac{\Delta x_f}{\Delta t_f} \tag{3-11}$$

$$\tilde{u}_c = \tilde{u}_f \tag{3-12}$$

By obeying the equations 3-3 and 3-5, the density $\tilde{\rho}$ proves to be continuous over the grids.

$$\rho = \tilde{\rho}_f \frac{\Delta m_f}{\Delta x_f^3} = \tilde{\rho}_c \frac{\Delta m_c}{\Delta x_c^3} \rightarrow \tilde{\rho}_f \frac{\Delta m_f}{\Delta x_f^3} = \tilde{\rho}_c \frac{2^3 \cdot \Delta m_f}{2^3 \cdot \Delta x_f^3} \tag{3-13}$$

$$\tilde{\rho}_c = \tilde{\rho}_f \tag{3-14}$$

In the same manner the scaling of the viscosity $\tilde{\nu}$ and the strain rate tensor $\tilde{S}_{\alpha\beta}$ can be derived like shown in equations 3-15 to 3-18.

$$\nu = \tilde{\nu}_c \frac{\Delta x_c^2}{\Delta t_c} = \tilde{\nu}_f \frac{\Delta x_f^2}{\Delta t_f} \rightarrow \tilde{\nu}_c \frac{4\Delta x_f^2}{2\Delta t_f} = \tilde{\nu}_f \frac{\Delta x_f^2}{\Delta t_f} \tag{3-15}$$

$$2\tilde{\nu}_c = \tilde{\nu}_f \tag{3-16}$$

$$\tilde{S}_{\alpha\beta} = \tilde{S}_{\alpha\beta,c} \frac{1}{\Delta t_c} = \tilde{S}_{\alpha\beta,f} \frac{1}{\Delta t_f} \rightarrow \tilde{S}_{\alpha\beta,c} \frac{1}{2\Delta t_f} = \tilde{S}_{\alpha\beta,f} \frac{1}{\Delta t_f} \tag{3-17}$$

$$\tilde{S}_{\alpha\beta,c} = 2\tilde{S}_{\alpha\beta,f} \tag{3-18}$$

The equilibrium distribution function h_i^{eq} given in equation 2-26 is not dependent on grid resolution either since it is purely dependent on the continuous density and velocities. Since, the density and velocities are continuous over the grids in lattice units as well, equation 3-19 can be derived.

$$\tilde{h}_{i,c}^{eq} = \tilde{h}_{i,f}^{eq} \tag{3-19}$$

However, there are two more quantities, the relaxation time τ and the non-equilibrium distribution function h_i^{neq}, which play a crucial role in the LBM as they represent the collision process. First, the transformation of the relaxation factor τ can be derived from its relation to the viscosity ν.

$$\nu = c_s^2\left(\tau_c - \frac{\Delta t_c}{2}\right) = c_s^2\left(\tau_f - \frac{\Delta t_f}{2}\right) \quad (3\text{-}20)$$

$$\tau_c - \frac{2\Delta t_f}{2} = \tau_f - \frac{\Delta t_f}{2} \quad (3\text{-}21)$$

$$\tau_f = \tau_c - \frac{\Delta t_c}{4}; \quad \tau_c = \tau_f + \frac{\Delta t_f}{2} \quad (3\text{-}22)$$

From section 2.1.3 it is known that after multiplying a quantity in lattice units with the corresponding conversion factors, its value in physical units is obtained. Therefore, the relaxation time in physical units τ can be expressed with equation 3-23. Inserting this in 3-22, also correlations for the relaxation times of different grids in lattice units can be made.

$$\tau = \tilde{\tau} \cdot \Delta t \quad (3\text{-}23)$$

$$\tilde{\tau}_f \cdot \Delta t_f = \tilde{\tau}_c \cdot \Delta t_c - \frac{\Delta t_c}{4} \quad (3\text{-}24)$$

$$\tilde{\tau}_f = 2\,\tilde{\tau}_c - \frac{1}{2} \quad (3\text{-}25)$$

$$\tilde{\tau}_c \cdot \Delta t_c = \tilde{\tau}_f \cdot \Delta t_f + \frac{\Delta t_f}{2} \quad (3\text{-}26)$$

$$\tilde{\tau}_c = \frac{\tilde{\tau}_f}{2} + \frac{1}{4} \quad (3\text{-}27)$$

Secondly, the non-equilibrium distribution function h_i^{neq} needs to be examined further to obtain its behavior at different scales. Dupuis and Chopard [24] derived in their work, that the non-equilibrium function in its explicit form and in physical units h_i^{neq} needs to be rescaled with Δt and $\tilde{\tau}$. An elaborated derivation of this can be found in appendix 7.2.

$$\frac{1}{\tilde{\tau}_c \Delta t_c} h_{i,c}^{neq} = \frac{1}{\tilde{\tau}_f \Delta t_f} h_{i,f}^{neq} \quad (3\text{-}28)$$

$$h_{i,c}^{neq} = \frac{2\tilde{\tau}_c}{\tilde{\tau}_f} h_{i,f}^{neq}; \quad h_{i,f}^{neq} = \frac{\tilde{\tau}_f}{2\tilde{\tau}_c} h_{i,c}^{neq} \quad (3\text{-}29)$$

Further, in appendix 7.2 it is found that the non-equilibrium distribution functions in lattice units need to be scaled in the same manner.

$$\tilde{h}_{i,c}^{neq} = \frac{2\tilde{\tau}_c}{\tilde{\tau}_f} \tilde{h}_{i,f}^{neq}; \quad \tilde{h}_{i,f}^{neq} = \frac{\tilde{\tau}_f}{2\tilde{\tau}_c} \tilde{h}_{i,c}^{neq} \quad (3\text{-}30)$$

3.3 Grid Coupling

In the beginning of this chapter, it was described how grids of different resolution are assembled. What now still is missing, is how the grids communicate with each other and how the overall algorithm can be envisioned.

First, some terms which will be used in the following sections need to be defined and explained.

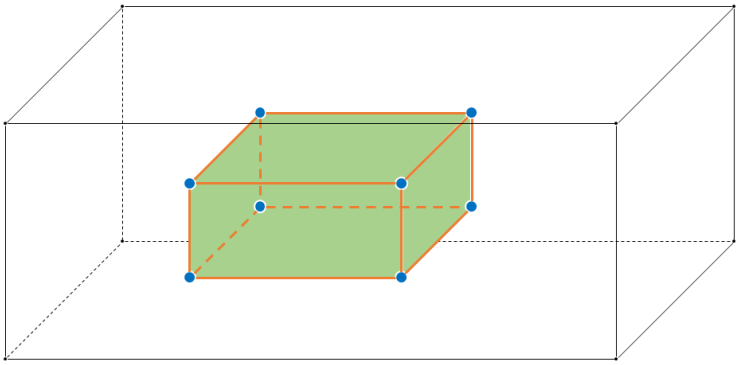


Fig. 3-7: The boundary between the grids is assembled from corners, edges and faces

As already mentioned in section 3.1 and shown in Fig. 3-7 the grids in 3D can be seen as nested boxes. Therefore, the boundary between the grids can be seen as an assembly of corners, edges and faces which are illustrated in Fig. 3-7 with blue dots, orange lines and green surfaces respectively.

Further, as it displayed in Fig. 3-8 and Fig. 3-9 in a 2D approach, each layer of nodes in the boundary region is referred to as a shell of nodes.

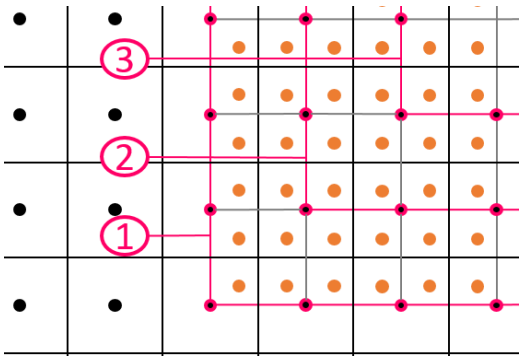


Fig. 3-8: 2D view of first, second and third shell of nodes of the coarse grid

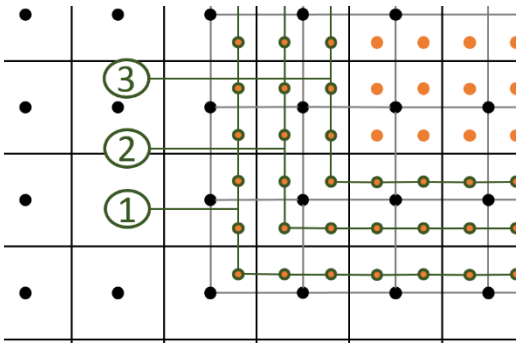


Fig. 3-9: 2D view of first, second and third shell of nodes of the fine grid

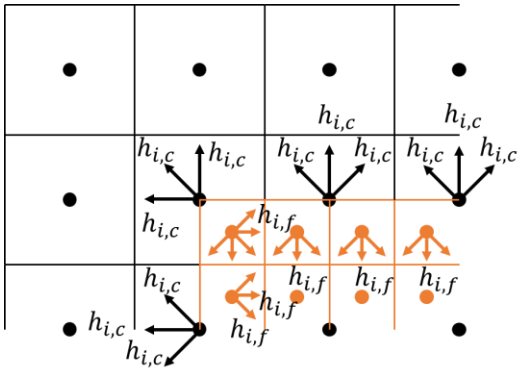


Fig. 3-10: Unknown distribution functions at the boundary between the grids

Every grid itself is seen as an individual simulation domain, solving the LB equation for every node in a parallel manner. Like illustrated in Fig. 3-10 at the boundary of the fine grid, the in-streaming distribution functions $h_{i,f}$, colored in orange, are unknown and due to the two-way coupling also the distribution functions streaming from the fine into the coarse region $h_{i,c}$, illustrated in black, are missing. To obtain the missing distribution functions of the fine grid, the velocity field of the coarse grid is interpolated as it is described in section 3.3.1.1 and the values for the density ρ and the non-equilibrium distribution functions h_i^{neq} are interpolated according to the method presented in section 3.3.1.2. The equilibrium distribution functions h_i^{eq} can be calculated with the interpolated velocity and density values as shown in equation 2-26. The missing distribution functions of the fine grid $h_{i,f}$ are then reconstructed using the interpolated non-equilibrium distribution functions, the computed equilibrium functions and knowing equation 2-71. However, as derived in section 3.2, the non-equilibrium distribution functions need to be rescaled according to equation 3-30 previously. The same procedure is done for obtaining the missing distribution functions of the coarse grid $h_{i,c}$. The whole procedure is described in greater detail in section 3.3.2.

Further, it needs to be mentioned that the equations given in sections 3.3.1 to 3.3.2 are valid for lattice units as well as physical units. Only the conversion of the relaxation time and the rescaling of the non-equilibrium distributions, as it was described in section 3.2 needs to be adapted according to the unit system.

3.3.1 Interpolation

The velocity field needs at least a cubic interpolation scheme to preserve its second order accuracy. Otherwise, the second order derivatives of the NSE become zero and “thus the correct representation of viscosity is impossible” according to Chen et al. [22]. Yu et al. [23] therefore, used a cubic spline function for interpolation. Lagrava Sandoval [20] presented a third order polynomial to interpolate between three and four points. These methods both are unfavorable for GPU implementation as they are non-local and, therefore, not only need direct neighbor information but also information of the neighbor to the neighbor. Furthermore, both methods use a string of nodes instead of nodes in cell formation for interpolation.

As it can be seen on the example of a 2D boundary corner in Fig. 3-11, for the interpolation in cell formation on the left, always four coarse nodes are used to obtain values for four fine nodes. As a consequence of the string interpolation scheme, corners and edges need special treatment because there not the same number of neighboring nodes are available like for nodes located on the face of a boundary. In Fig. 3-11 on the right, the string of nodes interpolation shows that for obtaining values for the fine node marked with number one, four nodes are used and for obtaining values for the nodes number two and three only three nodes are used. [18]

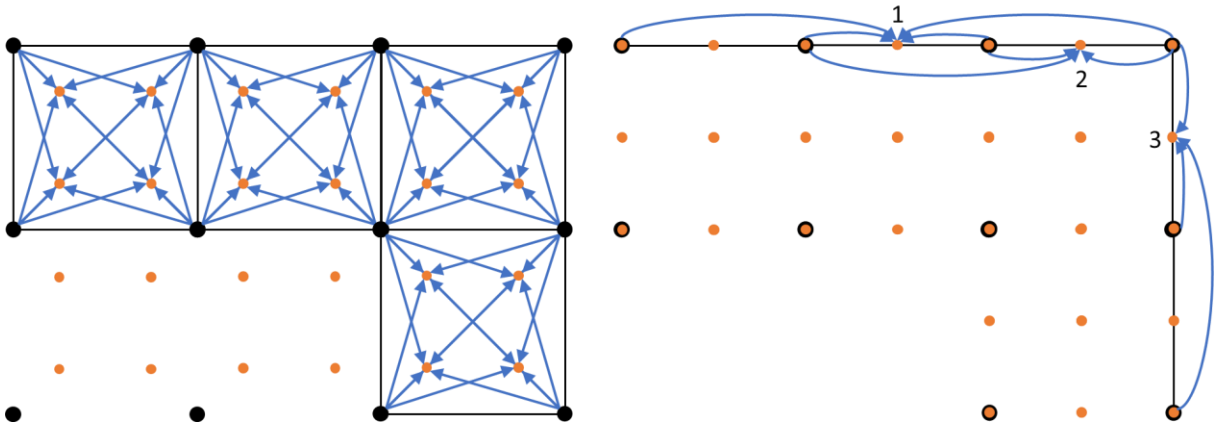
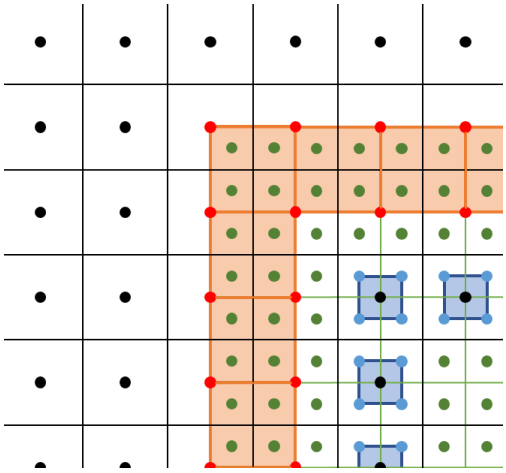


Fig. 3-11: View of a 2D interpolation schemes using four nodes in cell formation (l.) and the three possible arrangements of using a string of nodes (r.)

The second order accurate compact interpolation scheme used in this work was presented by Geier et al. [22] for 2D domains and extended into 3D by Qi et al. [13]. It is symmetric which leads to an equal treatment of all nodes and, therefore, avoids decisions and exceptions which are expensive in terms of computational resources. The compact interpolation requires an

overlap of the fine and the coarse grid by three coarse and four fine nodes in one spatial direction, as it can be seen in Fig. 3-12. In contrast to this, the non-local interpolation schemes presented above, require just an overlap of two coarse and three fine nodes. However, the used approach also is seen advantageous as no further temporal interpolation is needed and from a memory perspective, saving the values of the additional nodes is cheaper than saving the results from time interpolation. [18] In the case of a multi-grid domain, where the fine grid is completely overlapping the coarse grid, the size of the overlap region is not decisive. This only would be a topic of further considerations if the multi-grid domain is used, and therefore, a larger overlap would result in higher memory usage.



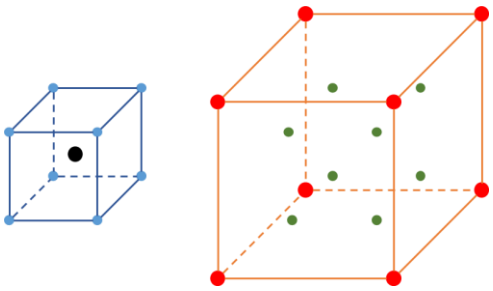


Fig. 3-13: 3D view of a fine-to-coarse (l.) and a coarse-to-fine (r.) interpolation cell

Fig. 3-12: 2D view of the overlapping grids with the fine-to-coarse interpolation cells in blue and the coarse-to-fine interpolation cells in orange

In Fig. 3-12 and Fig. 3-13 the interpolation cells are illustrated. An interpolation cell is assembled of eight nodes with known values of the provider grid which are used to interpolate unknown values at one or eight nodes of the receiver grid. The nodes of the receiver grid which obtain values from the provider grid are called ghost nodes since they are located in the overlapping region and are not necessarily needed to represent the flow field. The coarse-to-fine interpolation cells are illustrated in orange and use values at the red colored coarse nodes to compute values at the interior green ghost nodes of the fine grid. Vice versa are the fine-to-coarse interpolation cells, here colored in blue, utilized to calculate the value at the enclosed ghost node of the coarse grid from the information known on the surrounding light blue nodes of the fine grid. As it is shown in Fig. 3-12, the coarse-to-fine interpolation cells are assembled from nodes which constitute the first shell of coarse nodes before the fine grid starts and from the next shell of coarse nodes which is already overlapping with the fine grid. The enclosed

fine ghost nodes form the first and second shell of the fine grid. The fine-to-coarse interpolation cells are assembled from fine nodes belonging to the fourth and fifth shell of the fine grid and the coarse ghost nodes in the middle of every cell is generating the second overlapping shell of the coarse grid.

The interpolation cells are seen as a cube with a side length of one and one node located in the origin. Therefore, for a fine-to-coarse interpolation cell the ghost node is located at $\left(\frac{1}{2}\middle|\frac{1}{2}\middle|\frac{1}{2}\right)$. In case of a coarse-to-fine interpolation cell the ghost nodes have the coordinates $\left(\frac{1}{4}\middle|\frac{1}{4}\middle|\frac{1}{4}\right)$, $\left(\frac{3}{4}\middle|\frac{1}{4}\middle|\frac{1}{4}\right)$, $\left(\frac{1}{4}\middle|\frac{3}{4}\middle|\frac{1}{4}\right)$, $\left(\frac{1}{4}\middle|\frac{1}{4}\middle|\frac{3}{4}\right)$, $\left(\frac{3}{4}\middle|\frac{3}{4}\middle|\frac{1}{4}\right)$, $\left(\frac{1}{4}\middle|\frac{3}{4}\middle|\frac{3}{4}\right)$, $\left(\frac{3}{4}\middle|\frac{1}{4}\middle|\frac{3}{4}\right)$ and $\left(\frac{3}{4}\middle|\frac{3}{4}\middle|\frac{3}{4}\right)$.

3.3.1.1 Second Order Interpolation of the Velocity Field

The velocity field is interpolated using a second order polynomial for each spatial direction, equations 3-31, 3-32 and 3-33, as proposed by Qi et al. [13]. By applying this ansatz, thirty coefficients need to be obtained. One way to solve this is to take the velocity values of ten nodes, leading to a non-local and complex interpolation scheme. Therefore, the first and second order spatial derivatives of the polynomial u_α are included. They are presented in equations 3-34 and 3-35.

$$u_x(x,y,z) = a_0 + a_x x + a_y y + a_z z + a_{xx}x^2 + a_{yy}y^2 + a_{zz}z^2 + a_{xy}xy + a_{yz}yz + a_{xz}xz \quad (3\text{-}31)$$

$$u_y(x,y,z) = b_0 + b_x x + b_y y + b_z z + b_{xx}x^2 + b_{yy}y^2 + b_{zz}z^2 + b_{xy}xy + b_{yz}yz + b_{xz}xz \quad (3\text{-}32)$$

$$u_z(x,y,z) = c_0 + c_x x + c_y y + c_z z + c_{xx}x^2 + c_{yy}y^2 + c_{zz}z^2 + c_{xy}xy + c_{yz}yz + c_{xz}xz \quad (3\text{-}33)$$

$$\begin{aligned}
\frac{\partial u_x}{\partial x} &= a_x + 2a_{xx}x + a_{xy}y + a_{xz}z & \frac{\partial u_y}{\partial x} &= b_x + 2b_{xx}x + b_{xy}y + b_{xz}z \\
\frac{\partial u_x}{\partial y} &= a_y + 2a_{yy}y + a_{xy}x + a_{yz}z & \frac{\partial u_y}{\partial y} &= b_y + 2b_{yy}y + b_{xy}x + b_{yz}z \\
\frac{\partial u_x}{\partial z} &= a_z + 2a_{zz}z + a_{yz}y + a_{xz}x & \frac{\partial u_y}{\partial z} &= b_z + 2b_{zz}z + b_{yz}y + b_{xz}x
\end{aligned} \quad (3\text{-}34)$$

$$\begin{aligned}
\frac{\partial u_z}{\partial x} &= c_x + 2c_{xx}x + c_{xy}y + c_{xz}z \\
\frac{\partial u_z}{\partial y} &= c_y + 2c_{yy}y + c_{xy}x + c_{yz}z \\
\frac{\partial u_z}{\partial z} &= c_z + 2c_{zz}z + c_{yz}y + c_{xz}x
\end{aligned}$$

$$\begin{aligned}
\frac{\partial^2 u_x}{\partial x^2} &= 2a_{xx} & \frac{\partial^2 u_x}{\partial x \partial y} &= a_{xy} & \frac{\partial^2 u_x}{\partial x \partial z} &= a_{xz} \\
\frac{\partial^2 u_x}{\partial y^2} &= 2a_{yy} & \frac{\partial^2 u_x}{\partial y \partial z} &= a_{yz} & \frac{\partial^2 u_x}{\partial z^2} &= 2a_{zz} \\
\frac{\partial^2 u_y}{\partial x^2} &= 2b_{xx} & \frac{\partial^2 u_y}{\partial x \partial y} &= b_{xy} & \frac{\partial^2 u_y}{\partial x \partial z} &= b_{xz} \\
\frac{\partial^2 u_y}{\partial y^2} &= 2b_{yy} & \frac{\partial^2 u_y}{\partial y \partial z} &= b_{yz} & \frac{\partial^2 u_y}{\partial z^2} &= 2b_{zz} \\
\frac{\partial^2 u_z}{\partial x^2} &= 2c_{xx} & \frac{\partial^2 u_z}{\partial x \partial y} &= c_{xy} & \frac{\partial^2 u_z}{\partial x \partial z} &= c_{xz} \\
\frac{\partial^2 u_z}{\partial y^2} &= 2c_{yy} & \frac{\partial^2 u_z}{\partial y \partial z} &= c_{yz} & \frac{\partial^2 u_z}{\partial z^2} &= 2c_{zz}
\end{aligned} \qquad (3\text{-}35)$$

In the previous section it was described, that interpolation cells of eight nodes are used, leading to twenty-four equations considering the three known velocities u_x, u_y and u_z at each node. From section 2.1.6 it is known that the non-equilibrium moments of second order $\bar{\Pi}_{\alpha\beta}^{neq}$ are associated with the first derivatives of the velocity $\partial_\beta u_\alpha$ and $\partial_\alpha u_\beta$.

$$\bar{\Pi}_{\alpha\beta}^{neq} = \sum_i c_{i\alpha} c_{i\beta} h_i^{neq} = -\rho_0 c_s^2 \tau \left(\frac{\partial u_\alpha}{\partial x_\beta} + \frac{\partial u_\beta}{\partial x_\alpha} \right) = \Pi_{\alpha\beta}^{neq} \qquad (3\text{-}36)$$

Using the six known non-equilibrium moments $\Pi_{\alpha\beta}^{neq}$at the eight nodes leads to forty-eight new equations which are considerably too many to solve the system of equations.

However, the three spatial derivatives ∂_x, ∂_y and ∂_z of these six moments are correlating with the second order derivatives of the velocity field and, therefore, provide eighteen further equations.

$$\frac{\partial \Pi_{\alpha\beta}^{neq}}{\partial \gamma} = -c_s^2 \rho_0 \tau \left(\frac{\partial^2 u_\alpha}{\partial \beta \partial \gamma} + \frac{\partial^2 u_\beta}{\partial \alpha \partial \gamma} \right) \qquad (3\text{-}37)$$

Using them together with the three known velocities u_x, u_y and u_z at four instead of all nodes of the interpolation cell, exactly thirty equations are provided to solve for the thirty coefficients.

Considering a cube with a side length of one located with one node in the origin, two symmetrical interpolation stencils are possible to generate like it is shown in Fig. 3-14. Qi et al. [13] used the “Cube A” in their work. Geier [25] proposed to do the interpolation for both cubes and merge the results afterwards. This approach is adopted in this work.

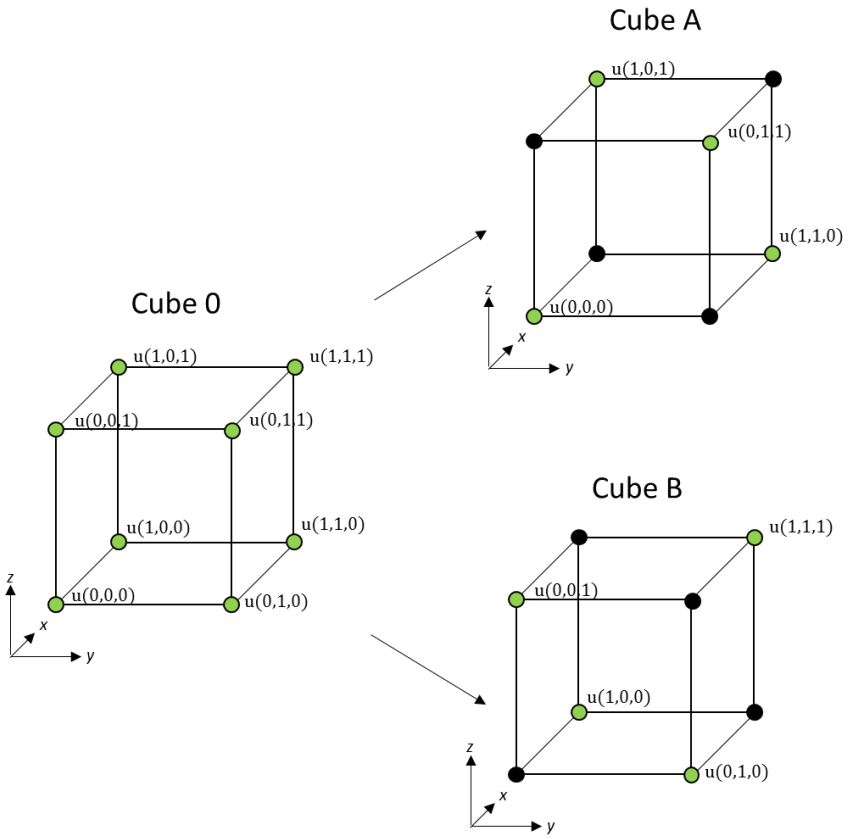


Fig. 3-14: Decomposition of a cube into the two possible interpolation stencils for four nodes

The known velocities at the four nodes lead to twelve equations per stencil.

Cube A:

$$I.\ u_x(0,0,0) = a_0 \quad (3\text{-}38)$$

$$II.\ u_y(0,0,0) = b_0 \quad (3\text{-}39)$$

$$III.\ u_z(0,0,0) = c_0 \quad (3\text{-}40)$$

$$IV.\ u_x(1,1,0) = a_0 + a_x + a_y + a_{xx} + a_{yy} + a_{xy} \quad (3\text{-}41)$$

$$V.\ u_y(1,1,0) = b_0 + b_x + b_y + b_{xx} + b_{yy} + b_{xy} \quad (3\text{-}42)$$

$$VI.\ u_z(1,1,0) = c_0 + c_x + c_y + c_{xx} + c_{yy} + c_{xy} \quad (3\text{-}43)$$

$$VII.\ u_x(1,0,1) = a_0 + a_x + a_z + a_{xx} + a_{zz} + a_{xz} \quad (3\text{-}44)$$

$$VIII.\ u_y(1,0,1) = b_0 + b_x + b_y + b_{xx} + b_{zz} + b_{xz} \quad (3\text{-}45)$$

$$IX.\ u_z(1,0,1) = c_0 + c_x + c_y + c_{xx} + c_{zz} + c_{xz} \quad (3\text{-}46)$$

$$X.\ u_x(0,1,1) = a_0 + a_y + a_z + a_{yy} + a_{zz} + a_{yz} \quad (3\text{-}47)$$

$$XI.\ u_y(0,1,1) = b_0 + b_y + b_z + b_{yy} + b_{zz} + b_{yz} \quad (3\text{-}48)$$

$$XII.\ u_z(0,1,1) = c_0 + c_y + c_z + c_{yy} + c_{zz} + c_{yz} \quad (3\text{-}49)$$

Cube B:

$$I.\ u_x(0,0,1) = a_0 + a_z + a_{zz} \quad (3\text{-}50)$$

$$II.\ u_y(0,0,1) = b_0 + b_z + b_{zz} \quad (3\text{-}51)$$

$$III.\ u_z(0,0,1) = c_0 + c_z + c_{zz} \quad (3\text{-}52)$$

$$IV.\ u_x(0,1,0) = a_0 + a_y + a_{yy} \quad (3\text{-}53)$$

$$V.\ u_y(0,1,0) = b_0 + b_y + b_{yy} \quad (3\text{-}54)$$

$$VI.\ u_z(0,1,0) = c_0 + c_y + c_{yy} \quad (3\text{-}55)$$

$$VII.\ u_x(1,0,0) = a_0 + a_x + a_{xx} \quad (3\text{-}56)$$

$$VIII.\ u_y(1,0,0) = b_0 + b_x + b_{xx} \quad (3\text{-}57)$$

$$IX.\ u_z(1,0,0) = c_0 + c_x + c_{xx} \quad (3\text{-}58)$$

$$X.\ u_x(1,1,1) = a_0 + a_x + a_y + a_z + a_{xx} + a_{yy} + a_{zz} + a_{xy} + a_{yz} + a_{xz} \quad (3\text{-}59)$$

$$XI.\ u_y(1,1,1) = b_0 + b_x + b_y + b_z + b_{xx} + b_{yy} + b_{zz} + b_{xy} + b_{yz} + b_{xz} \quad (3\text{-}60)$$

$$XII.\ u_z(1,1,1) = c_0 + c_x + c_y + c_z + c_{xx} + c_{yy} + c_{zz} + c_{xy} + c_{yz} + c_{xz} \quad (3\text{-}61)$$

The missing eighteen equations are obtained from the derivatives of the non-equilibrium moments of second order.

$$XIII.\ \frac{\partial \Pi_{xy}^{neq}}{\partial x} = -c_s^2 \rho_0 \tau \left(\frac{\partial^2 u_x}{\partial x \partial y} + \frac{\partial^2 u_y}{\partial x^2}\right) = -c_s^2 \rho_{0\tau}\left(a_{xy} + 2b_{xx}\right) \quad (3\text{-}62)$$

$$XIV.\ \frac{\partial \Pi_{xy}^{neq}}{\partial y} = -c_s^2 \rho_0 \tau \left(\frac{\partial^2 u_x}{\partial y^2} + \frac{\partial^2 u_y}{\partial x \partial y}\right) = -c_s^2 \rho_0 \tau\left(2a_{yy} + b_{xy}\right) \quad (3\text{-}63)$$

$$XV.\ \frac{\partial \Pi_{xy}^{neq}}{\partial z} = -c_s^2 \rho_0 \tau \left(\frac{\partial^2 u_x}{\partial y \partial z} + \frac{\partial^2 u_y}{\partial x \partial z}\right) = -c_s^2 \rho_0 \tau\left(a_{yz} + b_{xz}\right) \quad (3\text{-}64)$$

$$XVI.\ \frac{\partial \Pi_{yz}^{neq}}{\partial x} = -c_s^2 \rho_0 \tau \left(\frac{\partial^2 u_y}{\partial x \partial z} + \frac{\partial^2 u_z}{\partial x \partial y}\right) = -c_s^2 \rho_0 \tau\left(b_{xz} + c_{xy}\right) \quad (3\text{-}65)$$

$$XVII.\ \frac{\partial \Pi_{yz}^{neq}}{\partial y} = -c_s^2 \rho_0 \tau \left(\frac{\partial^2 u_y}{\partial y \partial z} + \frac{\partial^2 u_z}{\partial y^2}\right) = -c_s^2 \rho_0 \tau\left(b_{yz} + 2c_{yy}\right) \quad (3\text{-}66)$$

$$XVIII.\ \frac{\partial \Pi_{yz}^{neq}}{\partial z} = -c_s^2\rho_0\tau\left(\frac{\partial^2 u_y}{\partial z^2} + \frac{\partial^2 u_z}{\partial y \partial z}\right) = -c_s^2\rho_0\tau\left(2b_{zz} + c_{yz}\right) \quad (3\text{-}67)$$

$$XIX.\ \frac{\partial \Pi_{xz}^{neq}}{\partial x} = -c_s^2\rho_0\tau\left(\frac{\partial^2 u_x}{\partial x \partial z} + \frac{\partial^2 u_z}{\partial x^2}\right) = -c_s^2\rho_0\tau(a_{xz} + 2c_{xx}) \quad (3\text{-}68)$$

$$XX.\ \frac{\partial \Pi_{xz}^{neq}}{\partial y} = -c_s^2\rho_0\tau\left(\frac{\partial^2 u_x}{\partial y \partial z} + \frac{\partial^2 u_z}{\partial x \partial y}\right) = -c_s^2\rho_0\tau\left(a_{yz} + c_{xy}\right) \quad (3\text{-}69)$$

$$XXI.\ \frac{\partial \Pi_{xz}^{neq}}{\partial z} = -c_s^2\rho_0\tau\left(\frac{\partial^2 u_x}{\partial z^2} + \frac{\partial^2 u_z}{\partial x \partial z}\right) = -c_s^2\rho_0\tau(2a_{zz} + c_{xz}) \quad (3\text{-}70)$$

$$XXII.\ \frac{\partial \Pi_{xx}^{neq}}{\partial x} = -2c_s^2\rho_0\tau\left(\frac{\partial^2 u_x}{\partial x^2}\right) = -2c_s^2\rho_0\tau(2a_{xx}) \quad (3\text{-}71)$$

$$XXIII.\ \frac{\partial \Pi_{xx}^{neq}}{\partial y} = -2c_s^2\rho_0\tau\left(\frac{\partial^2 u_x}{\partial x \partial y}\right) = -2c_s^2\rho_0\tau\left(a_{xy}\right) \quad (3\text{-}72)$$

$$XXIV.\ \frac{\partial \Pi_{xx}^{neq}}{\partial z} = -2c_s^2\rho_0\tau\left(\frac{\partial^2 u_x}{\partial x \partial z}\right) = -2c_s^2\rho_0\tau(a_{xz}) \quad (3\text{-}73)$$

$$XXV.\ \frac{\partial \Pi_{yy}^{neq}}{\partial x} = -2c_s^2\rho_0\tau\left(\frac{\partial^2 u_y}{\partial x \partial y}\right) = -2c_s^2\rho_0\tau\left(b_{xy}\right) \quad (3\text{-}74)$$

$$XXVI.\ \frac{\partial \Pi_{yy}^{neq}}{\partial y} = -2c_s^2\rho_0\tau\left(\frac{\partial^2 u_y}{\partial y^2}\right) = -2c_s^2\rho_0\tau\left(2b_{yy}\right) \quad (3\text{-}75)$$

$$XXVII.\ \frac{\partial \Pi_{yy}^{neq}}{\partial z} = -2c_s^2\rho_0\tau\left(\frac{\partial^2 u_y}{\partial y \partial z}\right) = -2c_s^2\rho_0\tau\left(b_{yz}\right) \quad (3\text{-}76)$$

$$XXVIII.\ \frac{\partial \Pi_{zz}^{neq}}{\partial x} = -2c_s^2\rho_0\tau\left(\frac{\partial^2 u_z}{\partial x \partial z}\right) = -2c_s^2\rho_0\tau(c_{xz}) \quad (3\text{-}77)$$

$$XXIX.\ \frac{\partial \Pi_{zz}^{neq}}{\partial y} = -2c_s^2\rho_0\tau\left(\frac{\partial^2 u_z}{\partial y \partial z}\right) = -2c_s^2\rho_0\tau\left(c_{yz}\right) \quad (3\text{-}78)$$

$$XXX.\ \frac{\partial \Pi_{zz}^{neq}}{\partial z} = -2c_s^2\rho_0\tau\left(\frac{\partial^2 u_z}{\partial z^2}\right) = -2c_s^2\rho_0\tau(2c_{zz}) \quad (3\text{-}79)$$

The derivatives of the non-equilibrium moments of second order $\partial_\gamma \Pi_{\alpha\beta}^{neq}$ are obtained by applying the method of finite differences on the non-equilibrium moments $\Pi_{\alpha\beta}^{neq}$. Depending on which nodes are used in interpolation, the finite difference stencil is different as it is shown in equations 3-80 to 3-85.

Cube A:

$$\frac{\partial \Pi_{\alpha\beta}^{neq}}{\partial x} = \frac{1}{2}\left(\Pi_{\alpha\beta}^{neq}(1,1,0) + \Pi_{\alpha\beta}^{neq}(1,0,1) - \Pi_{\alpha\beta}^{neq}(0,1,1) - \Pi_{\alpha\beta}^{neq}(0,0,0)\right) \quad (3\text{-}80)$$

$$\frac{\partial \Pi_{\alpha\beta}^{neq}}{\partial y} = \frac{1}{2}\left(\Pi_{\alpha\beta}^{neq}(1,1,0) - \Pi_{\alpha\beta}^{neq}(1,0,1) + \Pi_{\alpha\beta}^{neq}(0,1,1) - \Pi_{\alpha\beta}^{neq}(0,0,0)\right) \quad (3\text{-}81)$$

$$\frac{\partial \Pi_{\alpha\beta}^{neq}}{\partial z} = \frac{1}{2}\left(-\Pi_{\alpha\beta}^{neq}(1,1,0) + \Pi_{\alpha\beta}^{neq}(1,0,1) + \Pi_{\alpha\beta}^{neq}(0,1,1) - \Pi_{\alpha\beta}^{neq}(0,0,0)\right) \quad (3\text{-}82)$$

Cube B:

$$\frac{\partial \Pi_{\alpha\beta}^{neq}}{\partial x} = \frac{1}{2}\left(\Pi_{\alpha\beta}^{neq}(1,1,1) + \Pi_{\alpha\beta}^{neq}(1,0,0) - \Pi_{\alpha\beta}^{neq}(0,1,0) - \Pi_{\alpha\beta}^{neq}(0,0,1)\right) \quad (3\text{-}83)$$

$$\frac{\partial \Pi_{\alpha\beta}^{neq}}{\partial y} = \frac{1}{2}\left(\Pi_{\alpha\beta}^{neq}(1,1,1) - \Pi_{\alpha\beta}^{neq}(1,0,0) + \Pi_{\alpha\beta}^{neq}(0,1,0) - \Pi_{\alpha\beta}^{neq}(0,0,1)\right) \quad (3\text{-}84)$$

$$\frac{\partial \Pi_{\alpha\beta}^{neq}}{\partial z} = \frac{1}{2}\left(\Pi_{\alpha\beta}^{neq}(1,1,1) - \Pi_{\alpha\beta}^{neq}(1,0,0) - \Pi_{\alpha\beta}^{neq}(0,1,0) + \Pi_{\alpha\beta}^{neq}(0,0,1)\right) \quad (3\text{-}85)$$

The coefficients of the polynomial can be obtained by solving the system of equations as it is done in the appendix in section 7.3. Inserting them in the velocity equations 3-31, 3-32 and 3-33 the velocity at the locations of the ghost nodes can be computed. This is done for both interpolation stencils and afterwards the resulting velocity values, $u_{\alpha,A}$ from "Cube A" and $u_{\alpha,B}$ form "Cube B" are averaged as equation 3-86 shows.

$$u_\alpha = \frac{1}{2}\left(u_{\alpha,A} + u_{\alpha,B}\right) \quad (3\text{-}86)$$

3.3.1.2 Trilinear Interpolation of Density and Non-Equilibrium Distributions

The density values and the non-equilibrium distribution functions at the locations of the ghost nodes are obtained by trilinear interpolation. In contrast to the velocity interpolation, this is done using all eight nodes of the interpolation cell. A trilinear interpolation can be seen as three subsequent linear interpolations. The values and naming convention applied in this concept is illustrated in Fig. 3-15 and is expressed in equation 3-87. First, all nodal values $C000$, $C100$ $C010$, $C110$, $C001$, $C101$, $C011$ and $C111$ are interpolated along the x-axis, resulting in $C00$, $C01$, $C10$ and $C11$. These quantities are further interpolated along the y-axis leading to $C0$ and $C1$. The resulting values in the end are interpolated along the z-axis leading to the required value C.

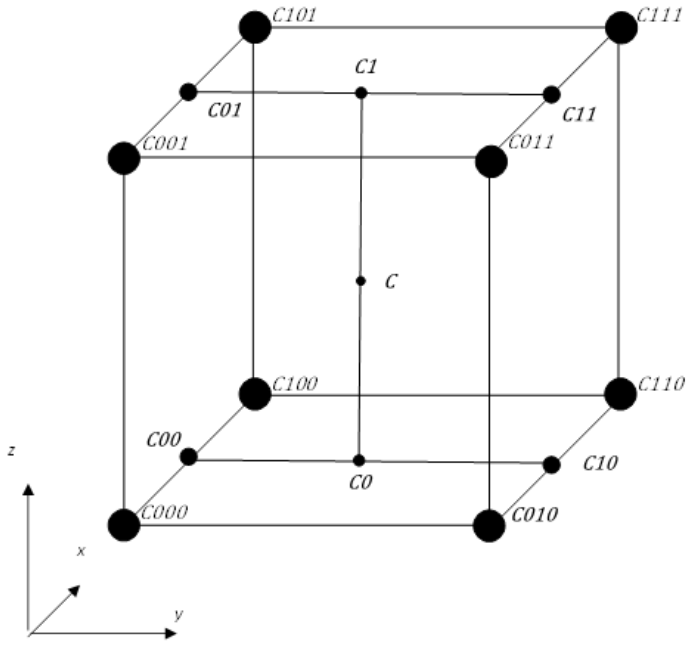


Fig. 3-15: The concept of trilinear interpolation in a cube

$$C = C000(1-x)(1-y)(1-z) + C100x(1-y)(1-z) + C010(1-x)y(1-z) + C110xy(1-z) + C001(1-x)(1-y)z + C101x(1-y)z + C011(1-x)yz + C111xyz \quad (3\text{-}87)$$

By knowing the nodal values of the interpolation cell, with equation 3-87, the density at the locations of the ghost nodes presented in paragraph 3.3.1, can be calculated. The non-equilibrium distribution functions can be equally treated.

3.3.2 Multi-Grid Algorithm

In this section the algorithm of grid coupling is described in detail and it is shown how the previously described interpolation steps are used to construct the missing distribution functions at the boundary of grids of different resolution. A pre-collision coupling is applied which means, that the distribution functions are constructed right before collision takes place. Filippova and Hänel [26] presented a post-collision grid coupling method.

The algorithm is based on five subsequent steps which are displayed in Fig. 3-16 and are described below.

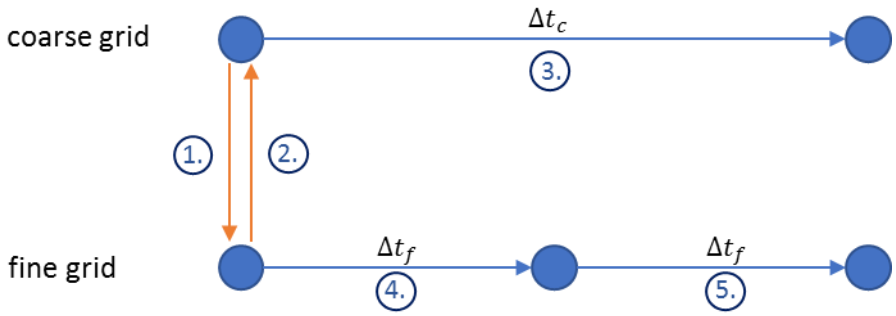


Fig. 3-16: Schematic illustration of the multi-grid algorithm for one refinement step with every arrow describing one process step

1. Coarse-to-fine coupling

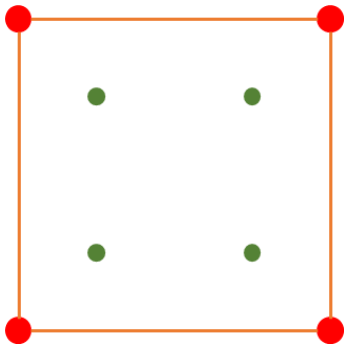


Fig. 3-17: Coarse -to-fine coupling

This step is done once for each coarse-to-fine interpolation cell which is described in detail in section 3.3.1. Each step of its workflow is illustrated on the left with a 2D sketch of a coarse-to-fine interpolation cell. Coarse nodes are colored in orange and fine ghost nodes are colored in green. The locations of interest for the particular steps are marked with blue stars. Furthermore, the variables obtained are listed inside the graphic. Variables with a superscript star are assigned to the locations of the ghost nodes and not to the nodes of the regular coarse grid.

- Calculation of nodal variables

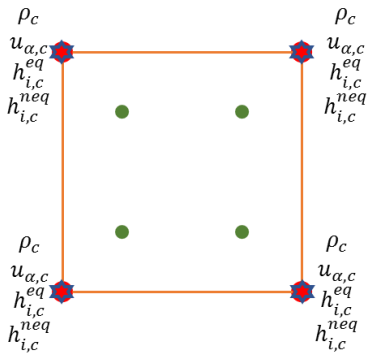


Fig. 3-18: Calculation of nodal variables

For the eight coarse nodes of the interpolation cell all distribution functions $h_{i,c}$ are known. From them, the macroscopic velocities $u_{\alpha,c}$ and the local density variations $\Delta\rho_c$ can be calculated with equations 2-53 and 2-54. Afterwards, the equilibrium distributions $h_{i,c}^{eq}$ are obtained by inserting the macroscopic quantities into equation 2-26. The coarse non-equilibrium distribution $h_{i,c}^{neq}$ is calculated with equation 2-71.

- Interpolation of the velocities

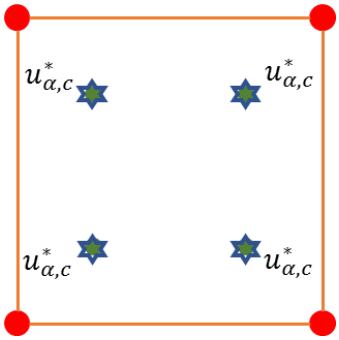


Fig. 3-19: Interpolation of the velocities

Using the coarse non-equilibrium distribution functions $h_{i,c}^{neq}$, the non-equilibrium moments of second order $\Pi_{\alpha\beta,c}^{neq}$ are calculated like it is written in equation 3-36. They are used to compute the first derivatives of the second order non-equilibrium moments $\partial_\gamma \Pi_{\alpha\beta,c}^{neq}$ as it is described in equations 3-80 to 3-85 in order to obtain all coefficients like listed in the appendix 7.3. Next, the velocities at the position of the fine ghost nodes $u_{\alpha,c}^*$ are calculated with the velocity polynomials (3-31 to 3-33).

- Interpolation of the density and the non-equilibrium distribution functions

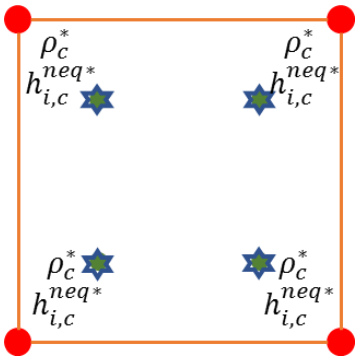


Fig. 3-20: Interpolation of the density and the non-equilibrium distribution functions

The density ρ_c and the non-equilibrium distribution functions $h_{i,c}^{neq}$ are interpolated with trilinear interpolation using equation 3-87 to obtain their values at the location of the ghost nodes ρ_c^* and $h_{i,c}^{neq*}$

- Calculation of the equilibrium distribution functions at the ghost nodes

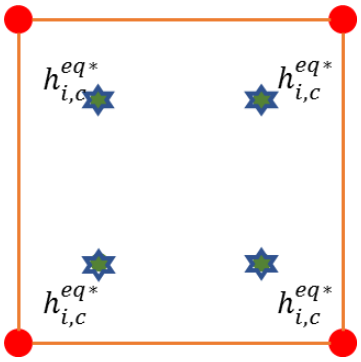


Fig. 3-21: Calculation of the equilibrium distribution functions at the ghost nodes

Knowing the velocity and density at the locations of the ghost nodes $u_{\alpha,c}^*$ and ρ_c^*, it is possible to calculate the equilibrium distribution function at the position of the ghost nodes $h_{i,c}^{eq*}$ with equation 2-26. In section 3.2 it is presented, that none of these quantities need to be scaled between the grids. Therefore, it can be said that $h_{i,c}^{eq*} = h_{i,f}^{eq*}$, $u_{\alpha,c}^* = u_{\alpha,f}^*$ and $\rho_c^* = \rho_f^*$.

- Rescaling of non-equilibrium distribution function

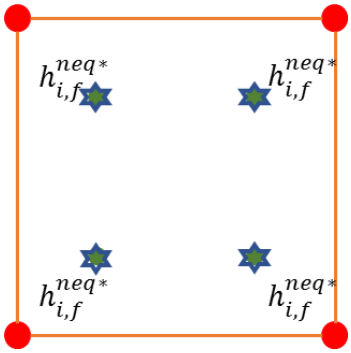


Fig. 3-22: Rescaling of non-equilibrium distribution function

In section 3.2 it is explained, why and how the non-equilibrium distribution functions $h_{i,c}^{neq*}$ have to be scaled, to represent the one from the opposite grid. Using equation 3-29 the non-equilibrium distribution functions at the location of the ghost nodes in fine lattice units $h_{i,f}^{neq*}$ are obtained.

- Calculate missing distribution functions

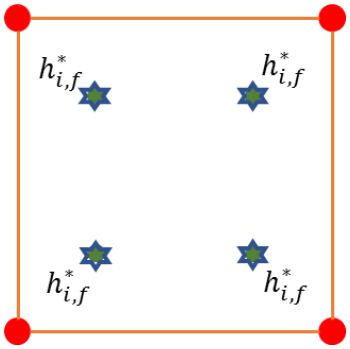


Fig. 3-23: Calculate missing distribution functions

The unknown fine distribution functions at the boundary of the fine grid $h_{i,f}^{*}$ are computed with the previous obtained equilibrium and non-equilibrium distribution functions $h_{i,f}^{eq*}$ and $h_{i,f}^{neq*}$. This is done by rearranging equation 2-71 to $h_{i,f}^{*} = h_{i,f}^{eq*} + h_{i,f}^{neq*}$. In contrast to other boundary conditions, not only the unknown distribution functions at the ghost nodes are constructed but all nineteen at each of the eight ghost nodes.

2. <u>Fine-to-coarse coupling</u>

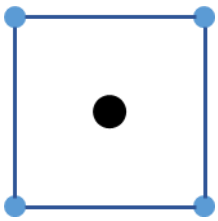


Fig. 3-24: Fine-to-coarse coupling

This step is done once for each fine-to-coarse interpolation cell which is described in detail in section 3.3.1. Each step of its workflow is illustrated on the left with a 2D sketch of a fine-to-coarse interpolation cell. Fine nodes are colored in light blue and coarse ghost nodes are colored in black. The locations of interest for the particular steps are marked with red stars. Furthermore, the variables obtained are listed inside the graphic. Variables with a superscript star are assigned to the

location of the ghost node and not to the nodes of the regular fine grid.

- Calculation of nodal variables

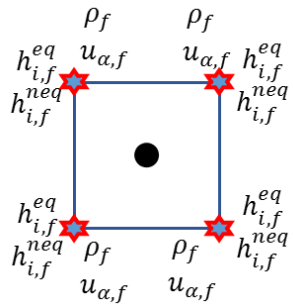


Fig. 3-25: Calculation of nodal variables

For the eight fine nodes of the interpolation cell all distribution functions $h_{i,f}$ are known. From them, the macroscopic velocities $u_{\alpha,f}$ and the local density variations $\Delta\rho_f$ can be calculated with equations 2-53 and 2-54. Afterwards, the equilibrium distributions $h_{i,f}^{eq}$ are obtained by inserting the macroscopic quantities into equation 2-26. The fine non-equilibrium distribution $h_{i,f}^{neq}$ is calculated with equation 2-71.

- Interpolation of the velocities

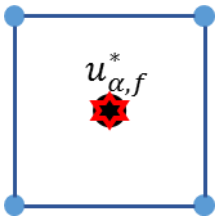


Fig. 3-26: Interpolation of $u_{\alpha,f}^{*}$

Using the fine non-equilibrium distribution functions $h_{i,f}^{neq}$, the non-equilibrium moments of second order $\Pi_{\alpha\beta,f}^{neq}$ are calculated like it is given in equation 3-36. They are used to compute the first derivatives of the second order non-equilibrium moments $\partial_\gamma \Pi_{\alpha\beta,f}^{neq}$ as it is described in equations 3-80 to 3-85 in order to obtain all coefficients like listed in appendix 7.3. Next, the velocities at the position of the coarse ghost node $u_{\alpha,f}^{*}$ are calculated with the velocity polynomials in equations 3-31 to 3-33.

- Interpolation of the density and the non-equilibrium distribution functions

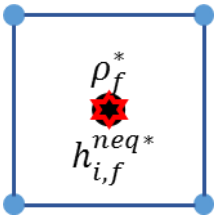


Fig. 3-27: Interpolation of ρ and $h_{i,f}^{neq}$

The density ρ_f and the non-equilibrium distribution functions $h_{i,f}^{neq}$ are interpolated with trilinear interpolation using equation 3-87 to obtain their values at the location of the ghost nodes ρ_f^{*} and $h_{i,f}^{neq*}$

- Calculation of the equilibrium distribution functions at the ghost node

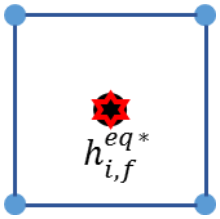


Fig. 3-28: Calculation of $h_{i,f}^{eq*}$

Knowing the velocity and density at the location of the ghost node $u_{\alpha,f}^{*}$ and ρ_{f}^{*}, it is possible to calculate the equilibrium distribution functions at the position of the ghost node $h_{i,f}^{eq*}$ with equation 2-26. In section 3.2 it is presented, that none of these quantities need to be scaled between the grids. Therefore, it can be said that $h_{i,c}^{eq*} = h_{i,f}^{eq*}$, $u_{\alpha,c}^{*} = u_{\alpha,f}^{*}$ and $\rho_{c}^{*} = \rho_{f}^{*}$.

- Rescaling of non-equilibrium distribution function

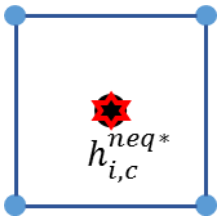


Fig. 3-29: Rescaling of $h_{i,f}^{neq*}$

In section 3.2 it is explained, why and how the non-equilibrium distribution functions $h_{i,f}^{neq*}$ have to be scaled, to represent the one from the opposite grid. Using equation 3-29 the non-equilibrium distribution functions at the location of the ghost node in coarse lattice units $h_{i,c}^{neq*}$ are obtained.

- Calculate missing distribution functions

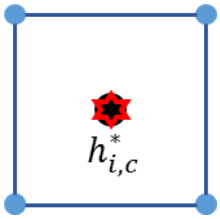


Fig. 3-30: Calculate $h_{i,c}^{*}$

The unknown coarse distribution functions which represent the fluid streaming from the fine grid into the coarse region $h_{i,c}^{*}$ are computed with the previous obtained equilibrium and non-equilibrium distribution functions $h_{i,c}^{eq*}$ and $h_{i,c}^{neq*}$. This is done by rearranging equation 2-71 to $h_{i,c}^{*} = h_{i,c}^{eq*} + h_{i,c}^{neq*}$. In contrast to other boundary conditions, not only the unknown distribution functions at the ghost node are constructed but all nineteen distribution functions.

3. “Collide and stream” on coarse grid

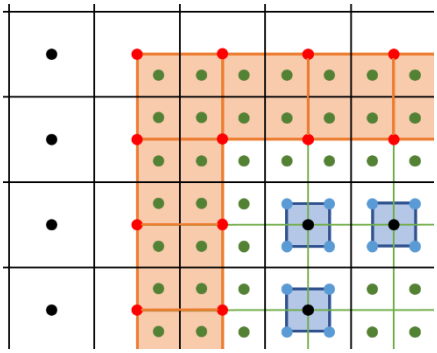


Fig. 3-31: 2D view of the overlapping grids after the grid

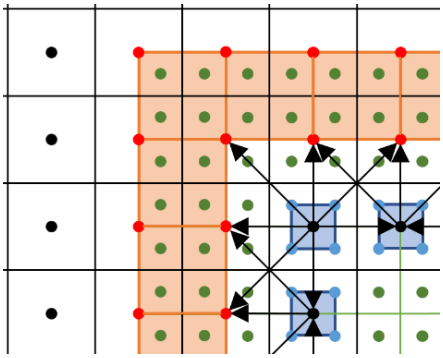


Fig. 3-32: Simplified illustration of the coarse streaming step

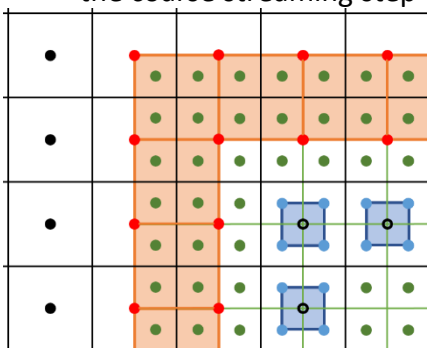


Fig. 3-33: Invalid ghost nodes on coarse grid

After the two-way grid coupling which is done in step one and two, all nodes on the fine and coarse grid are valid, meaning that the distribution function on all nodes are known. In Fig. 3-31 valid nodes are displayed as filled dots. Next, one LBM step is done on all nodes of the coarse grid. The procedure described in section 2.1.4 is executed. Fig. 3-32 shows a simplified illustration of the streaming step in the coarse grid where just some arrows for propagation are displayed to maintain conspicuousness. In the end, the simulation is at time $t + \Delta t_c$. At this point, the distribution functions at the coarse ghost nodes are invalid for two-way coupled simulations like it is shown in Fig. 3-33 with unfilled dots. During the streaming step these nodes have not received information from the fine grid and, therefore some of their distribution functions are unknown.

4. Asynchronous “collide and stream” on fine grid

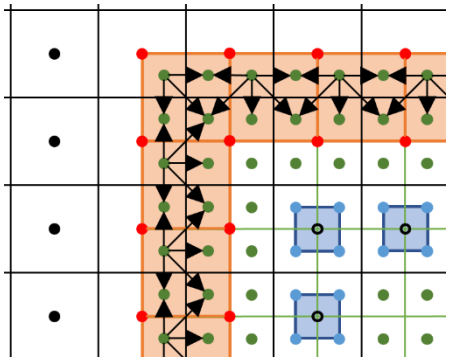


Fig. 3-34: Simplified illustration of the fine streaming step

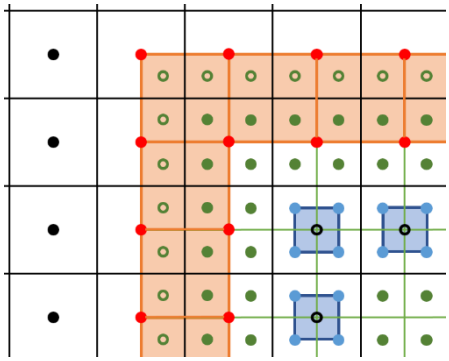


Fig. 3-35: Invalid nodes in first shell of fine grid

An LBM step according to section 2.1.4 is also performed on all nodes of the fine grid, bringing it to $t + \Delta t_f$ which is equal to $t + \frac{\Delta t_c}{2}$. It is called asynchronous step due to the fact, that no comparable results are generated on the coarse grid at this time as it can be seen in Fig. 3-16. During the streaming step, which is outlined in Fig. 3-34 distributions are propagated. However, in the end of the asynchronous time step, the distribution functions in the first shell of fine nodes are partly unknown since all valid distribution functions have streamed to their neighbors and no information from the coarse grid is obtained to update the boundary nodes.

5. Synchronous “collide and stream” on fine grid

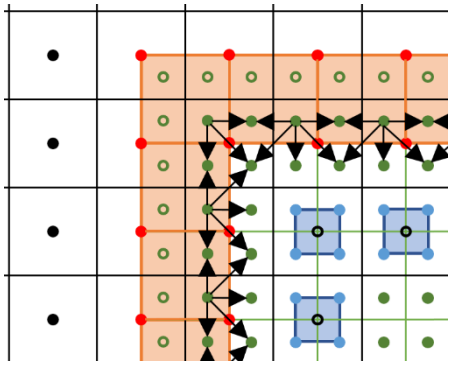


Fig. 3-36: Simplified second fine streaming step

A further LBM step according to chapter 2.1.4 is done on all nodes of the fine grid, bringing it to $t + 2\Delta t_f$ which is equal to $t + \Delta t_c$. The fine and the coarse grid have reached the same degree of progress and, therefore, are synchronous. In the end of the synchronous time step the distribution functions belonging to nodes of the first and sec-

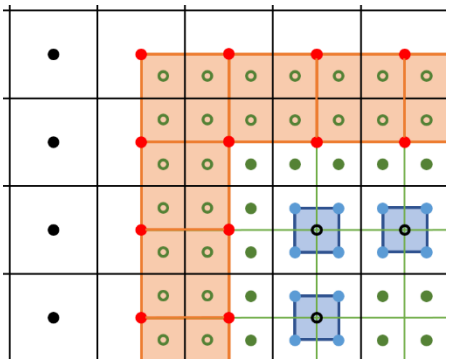


Fig. 3-37: Invalid ghost nodes on the fine grid

ond shell of fine grid, apparent from Fig. 3-37 exactly all fine ghost nodes, are invalid. This is because, invalid values from the first shell have streamed to their neighbors.

6. Starting again at 1. and repeating the procedure

This procedure is valid for convective grid scaling (see section 3.2) only. If diffusive scaling is applied, four fine time steps are needed to propagate the fluid to the same point as it is done in one coarse time step. [18]

Further, as mentioned already in section 3.3.1, no time interpolation is needed within this procedure. This is due to the fact, that the distribution functions of two shells of fine nodes are constructed in the coarse-to-fine coupling. In case the distribution functions of only one shell of nodes are updated, like it is mostly done when the coinciding nodes arrangement for meshing is used (see section 3.1), a temporal interpolation is needed. The reasons are explained in depth by Lagrava-Sandoval [20].

4 Implementation

The functionality of grid refinement was implemented in an already comprehensive simulation program. Therefore, it was required to integrate it with a minimum of impact on the existing code. Most changes in code have been done with the mindset to be able to use several refinement steps in future. Although, this is not possible in the current version, preparations have been made. Some changes and innovations will be highlighted in the following chapter.

4.1 Input

The feature of grid refinement requires few input parameters. It is possible to define a cuboid in which the grid resolution is increased. To describe its size and location, six variables x_1, x_2, y_1, y_2, z_1 and z_2 are used as illustrated in Fig. 4-1. They are describing the distances of the fine grid boundaries to the boundaries of the coarse grid in all spatial dimensions. Further, the number of required grids and, therefore refinement levels, is fixed to be two. As a consequence, the whole domain is resolved by a coarse grid and in the defined region, a grid with its resolution increased by factor two is placed.

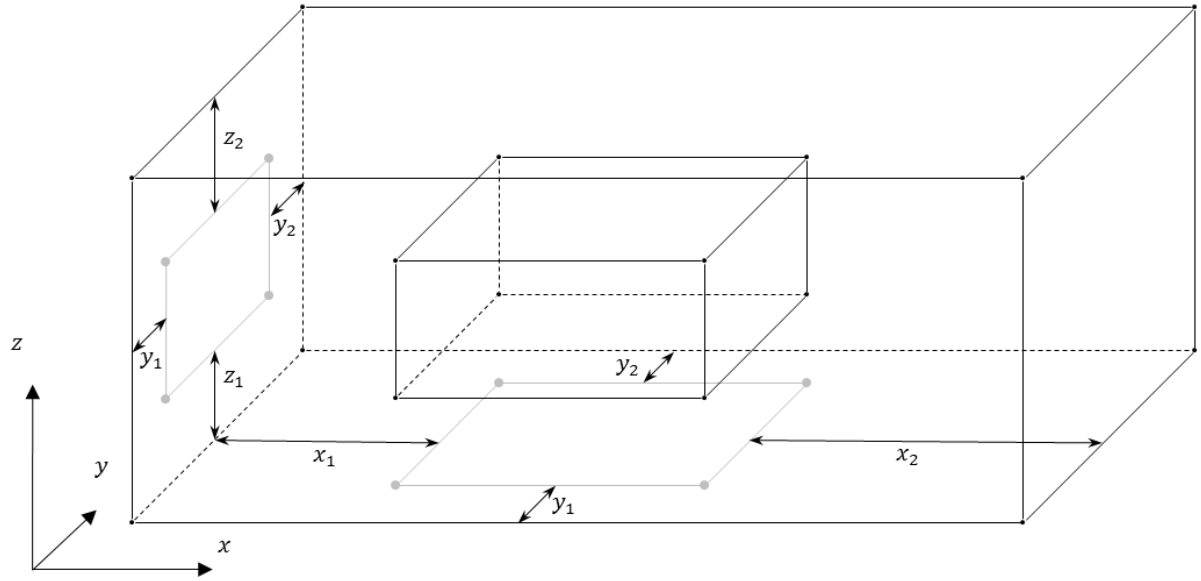


Fig. 4-1: Position of the fine grid inside the coarse grid

4.2 Local and Global Coordinates

Every grid has its own local coordinate system with the origin set to one corner of the grid. In this work, the origin is set to the bottom left front corner like it is shown in Fig. 3-14 and Fig. 4-1. In Fig. 4-2 on the left, a fine node is marked in red. In local fine coordinates it is located in $\begin{pmatrix} x_{f,loc} \\ z_{f,loc} \end{pmatrix} = \begin{pmatrix} 4 \\ 5 \end{pmatrix}$ and in local coarse coordinates it is placed in $\begin{pmatrix} x_{c,loc} \\ z_{c,loc} \end{pmatrix} = \begin{pmatrix} 4{,}25 \\ 3{,}75 \end{pmatrix}$. However, for some operations, like the definition of the solid reactor wall, it is important to know the global location of a fine node. Therefore, the concept of global coordinates is introduced in which the domain is considered to be completely covered with an imaginary fine grid. In the global fine coordinate system, like it is shown in Fig. 4-2 on the right, the fine node's location can be identified with $\begin{pmatrix} x_{f,glo} \\ z_{f,glo} \end{pmatrix} = \begin{pmatrix} 8 \\ 7 \end{pmatrix}$. Since the coarse grid is already covering the whole domain its global coordinates are equal the local coordinates $\begin{pmatrix} x_{c,glo} \\ z_{c,glo} \end{pmatrix} = \begin{pmatrix} x_{c,loc} \\ z_{c,loc} \end{pmatrix}$.

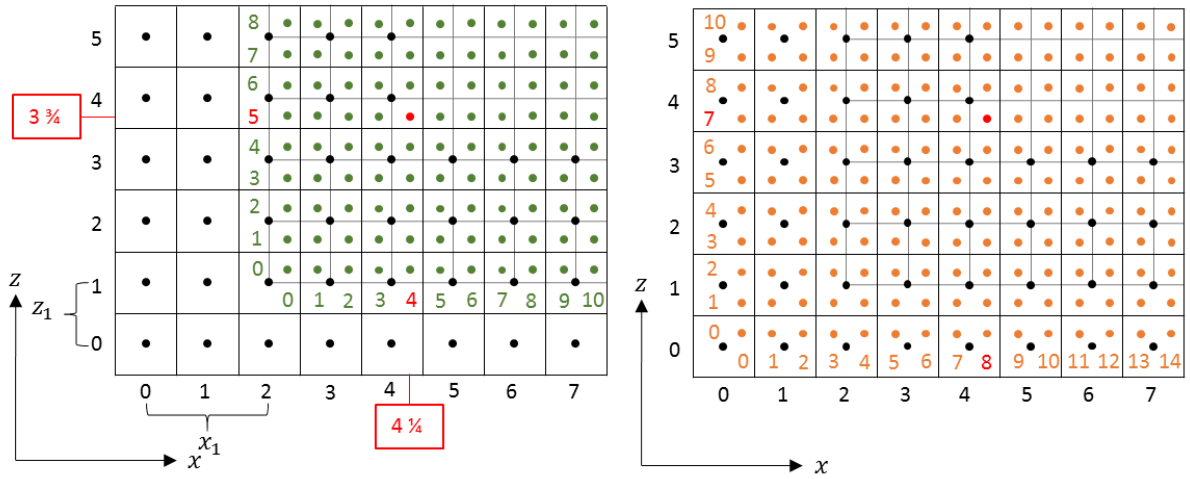


Fig. 4-2: 2D illustration of the local grid coordinates (l.) and the global coordinates (r.) with a node in the same location marked in red in both grids and the corresponding values of the coordinate systems

To obtain global coordinates the input variables x_1, y_1 and z_1, described in section 4.1, are used as they describe exactly the offset of the local coordinate system. The offset shown in Fig. 4-2 is given with $\begin{pmatrix} x_1 \\ y_1 \\ z_1 \end{pmatrix} = \begin{pmatrix} 2 \\ y_1 \\ 1 \end{pmatrix}$ and has to be multiplied by two to account for the different lattice spacing. The coordinate translation can therefore be done according to equation 4-1.

$$\begin{pmatrix} x_{f,glo} \\ y_{f,glo} \\ z_{f,glo} \end{pmatrix} = \begin{pmatrix} x_{f,loc} \\ y_{f,loc} \\ z_{f,loc} \end{pmatrix} + 2 \begin{pmatrix} x_1 \\ y_1 \\ z_1 \end{pmatrix} \qquad (4\text{-}1)$$

Further, it is required to correlate the locations of the eight nodes of an interpolation cell with the locations of its one or eight ghost nodes in order to construct the missing distribution functions at the grid boundaries described in section 3.3. To accomplish this, the grid offsets x_1, y_1 and z_1 are used as well. It is considered that the position of every interpolation cell and every cell of eight ghost nodes is defined by the coordinates of the node in the bottom left front corner which correlates to the $(0|0|0)$ node of Fig. 3-14 and will be called base node in the following sections. The other nodes can be obtained by adding the vector $r_{ijk} = \begin{pmatrix} i \\ j \\ k \end{pmatrix}$ given in Tab. 4-1, where i, j and k are either 0 or 1, to the base node coordinates accordingly.

Tab. 4-1: The vectors r_{ijk} which point to every cell node from the base node

$$r_{000} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \quad r_{100} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \quad r_{010} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \quad r_{001} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \quad r_{110} = \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} \quad r_{011} = \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix} \quad r_{101} = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} \quad r_{111} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$$

To obtain the eight fine ghost nodes from a coarse-to-fine interpolation cell base node with the coordinates $\begin{pmatrix} x_{c,loc} \\ y_{c,loc} \\ z_{c,loc} \end{pmatrix}$, the grid offsets x_1, y_1 and z_1 which are given in coarse lattice units must be subtracted and the result needs to be multiplied by two to account for the different lattice spacing. According to that, the coordinates of the base node of the fine ghost cell $\begin{pmatrix} x_{f,loc} \\ y_{f,loc} \\ z_{f,loc} \end{pmatrix}$ is obtained. The coordinates of the other ghost nodes located in that cell are calculated by adding the vector r_{ijk} as it is given in equation 4-2.

$$\begin{pmatrix} x_{f,loc} \\ y_{f,loc} \\ z_{f,loc} \end{pmatrix} = 2 \left(\begin{pmatrix} x_{c,loc} \\ y_{c,loc} \\ z_{c,loc} \end{pmatrix} - \begin{pmatrix} x_1 \\ y_1 \\ z_1 \end{pmatrix} \right) + \begin{pmatrix} i \\ j \\ k \end{pmatrix} \qquad (4\text{-}2)$$

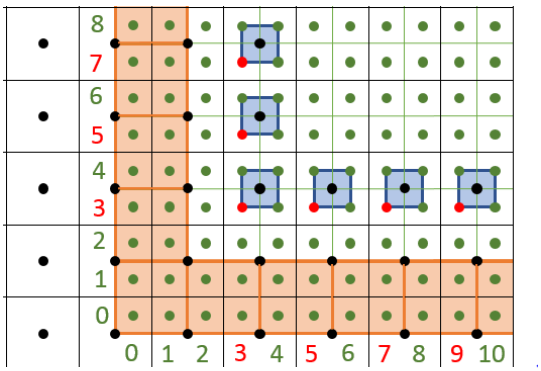


Fig. 4-3: 2D scheme of fine-to-coarse interpolation cells in blue and their base nodes in red

The coordinates of a coarse ghost node $\begin{pmatrix} x_{c,loc} \\ y_{c,loc} \\ z_{c,loc} \end{pmatrix}$ can be calculated from the coordinates of a fine-to-coarse interpolation cell base node $\begin{pmatrix} x_{f,loc} \\ y_{f,loc} \\ z_{f,loc} \end{pmatrix}$ with equation 4-3. Due to the fact, that the coordinates of the base nodes of fine-to-coarse interpolation cells always have odd integer values like it is shown in Fig. 4-3, it is ensured that the result of the division in equation 4-3 is always an integer.

$$\begin{pmatrix} x_{c,loc} \\ y_{c,loc} \\ z_{c,loc} \end{pmatrix} = \frac{\left(\begin{pmatrix} x_{f,loc} \\ y_{f,loc} \\ z_{f,loc} \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}\right)}{2} + \begin{pmatrix} x_1 \\ y_1 \\ z_1 \end{pmatrix} \quad (4\text{-}3)$$

4.3 Design of Variables

In section 3.2 it was shown, that a change of grid resolution entails a change in conversion factors and, therefore lattice units. If for instance, the reactor height in coarse lattice units is equal to 75, it would be 150 in fine lattice units according to equations 2-32 and 3-2. To cope with different lattice units, the simulation variables including all measures of length, various parameters and pointers to massive matrices with node related data are expanded by a further dimension. Therefore, as shown in Tab. 4-2 single values are converted into vectors and vectors into matrices. The parameter *max_lvl* used in Tab. 4-2 gives the number of grids used in the simulation, like it was prescribed in the input in section 4.1 and is defining the number of elements in the further dimension. As a consequence, the variable *reactorHeight* is defined as a vector with two elements, one for every grid level.

Tab. 4-2: Examples of variables used in simulation

```
float reactorHeight[max_lvl]
float3 shaftBottom[max_lvl]
uint *isSolid[max_lvl]
float3 *velocity[max_lvl]
```

4.4 Function Design

In order to understand how the feature of grid refinement can be integrated into a lattice Boltzmann code Fig. 4-4 illustrates the sequence of functions implemented.

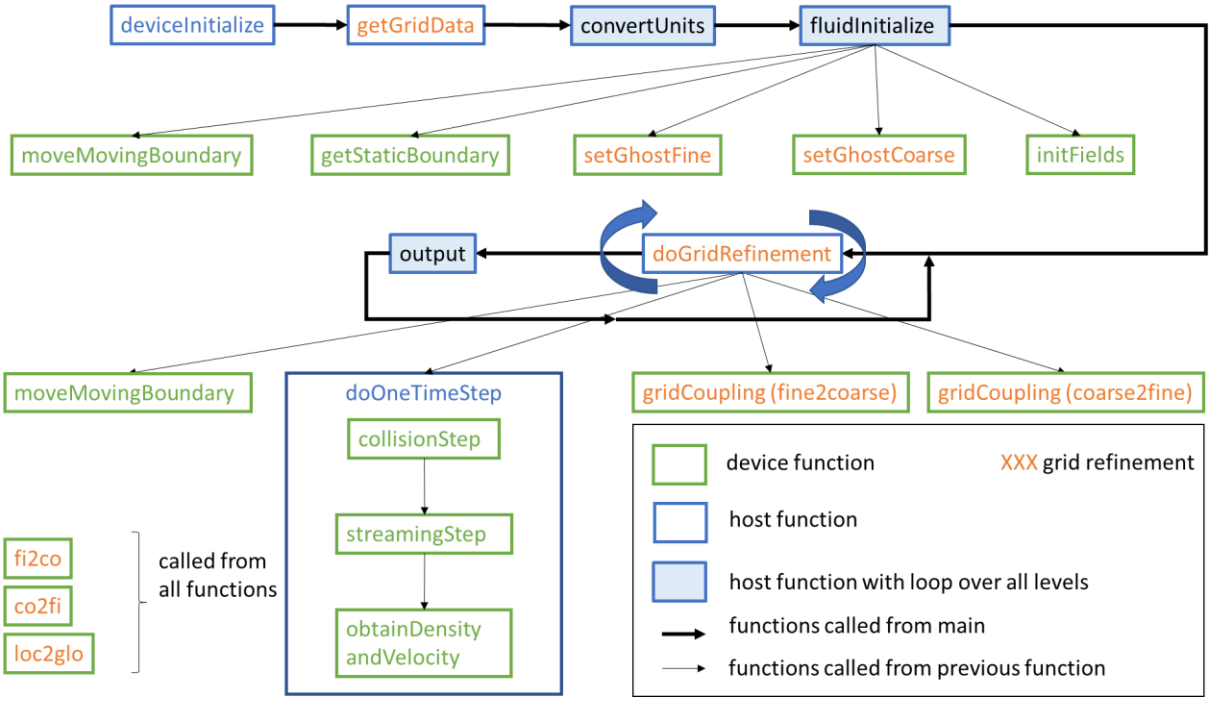


Fig. 4-4: Function diagram

The functions in Fig. 4-4 which are written in orange, have been realized to facilitate grid refinement. Omitting them leads to a simple LBM algorithm. As described in section 2.2, using the CUDA architecture it must be differentiated between device and host functions. In the sketch, they are framed green and blue respectively. All actions, which need to be done in sequence and are not applied on all nodes are implemented in host functions, like the conversion of quantities from physical units to lattice units. All computations executed in parallel on the nodes, like the computation of velocity and density from the distribution functions, are done using device functions. Each of the functions listed in Fig. 4-4 will be explained further depending on their classification in section 4.4.1 or 4.4.2. From the main file several host functions are called in a sequence to set up the fluid calculation, which is indicated with bold black arrows. The core of the simulation is formed by a looped call of the functions *doGridRefinement* and *output*. This loop is proceeded for a user defined period of time.

4.4.1 Host Functions

There are two different practices of realizing a host function together with grid refinement. In Fig. 4-4, this is indicated with functions in blue frames with white or light blue background. The one, with light blue background are implemented with a for-loop to execute the function body once for every level, like it is shown in the pseudo code example below. This is a simple way to adapt these functions to feature grid refinement.

```
for(uint lvl=0; lvl < max_lvl; lvl++){
      variable1[lvl]=variable2[lvl]/variable3[lvl]
}
```

The parameter *lvl* is used in the whole application to distinguish between the different refinement levels. If it is equal to zero, the coarsest grid is addressed. Since only one refinement step is used the fine grid is addressed with level equals to one.

Functions with a white background do not have this loop as it is not necessary or possible to calculate them for every level.

- *deviceInitialize*: The GPUs used for the calculation are reset and it is checked if they are communicating properly.
- *getGridData*: The relevant data for grid refinement like number of nodes, position of grids and the conversion factors are imported and converted to lattice units. No loop over the levels is possible because the calculations differ for every level.
- *convertUnits*: All further input data is imported and converted to lattice units; all required parameters are calculated.
- *fluidInitiaize:* Various functions are called to initialize the domain and set up the simulation boundaries.
- *doGridRefinement*: The grid coupling and fluid calculation is triggered from this function. It is a recursive function and is described in depth in section 4.5.
- *doOneTimeStep*: The functions *collisionStep*, *streamingStep* and *obtainDensityandVelocity* are called from this function as they are needed to fulfill one time step as described in section 2.1.4. *doOneTimeStep* needs to be called for each grid separately from the function *doGridRefinement.*

- *output*: The velocity and density data calculated in this time step is written to output files and saved. This function is called in user defined time intervals. It is looped over all levels and, therefore, the data for all levels is saved subsequently.

4.4.2 Device Functions

The device functions are all executed in parallel on every node. To be compatible with grid refinement, they use the variable for grid level *lvl* as input parameter to know which element of the arrays they need to address during calculations. Further, the variables *blocks* and *threads* are redefined for every grid to use the exact number of threads needed in every kernel call.

- *initFields*: The value fields for density, velocity, distribution functions, fine ghost nodes and interpolation cells, coarse ghost nodes and interpolation cells and boundary are initialized with zero values.

- *setGhostCoarse*: This function defines which nodes of the coarse grid belong to coarse-to-fine interpolation cells and which are ghost nodes. The base nodes of the interpolation cells are marked.

- *setGhostFine*: This function defines which nodes of the fine grid belong to fine-to-coarse interpolation cells and which are ghost nodes. The base nodes of the interpolation and ghost cells are marked.

- *getStaticBoundary*: The nodes belonging to the reactor wall and the stirrer shaft are set as solid.

- *moveMovingBoundary*: The nodes belonging to the stirrer blades are set as solid according to the current rotation angle of the agitator.

- *gridCoupling*: The grids are coupled according to the procedure described in section 3.3. The distinction between coarse-to-fine and fine-to-coarse is made by the function input.

- *collisionStep:* The collision of particles is done as it is described in equation 2-40.

- *streamingStep:* The distributions are streamed to their neighboring nodes according to equation 2-41 and the boundary conditions described in section 2.1.5 are applied if necessary.

- *obtainDensityandVelocity:* The macroscopic quantities are calculated according to equations 2-37, 2-38 and 2-39.

Further, to facilitate grid refinement three functions for coordinate transformation between the grids *loc2glo, co2fi* and *fi2co* are called from the device functions.

- *loc2glo*: Global coordinates are obtained from local grid coordinates like it is shown in section 4.2 and equation 4-1.

- *co2fi:* Local coordinates of the coarse grid are used to obtain matching local coordinates of the fine grid according to equation 4-2.

- *fi2co:* Local coordinates of the fine grid are used to obtain matching local coordinates of the coarse grid according to equation 4-3.

4.5 Recursive Function

In Fig. 4-4 the function *doGridRefinement* is surrounded by blue arrows. This should represent the fact, that it is a recursive function. This type of functions is calling themselves in the function body until a break condition is fulfilled. As a consequence, the correct sequence and therefore right time for the grid coupling procedure is guaranteed for more than one refinement step. In Fig. 4-5 it is illustrated with the example of three refinement steps that the complexity of sequencing increases with increasing degree of refinement. The grid coupling in step number 17 and 18 for instance can only be done if the fluid calculation on all levels has reached the time $t + \Delta t_1$ which is equal to $t + 2\Delta t_2$ and $t + 4\Delta t_3$.

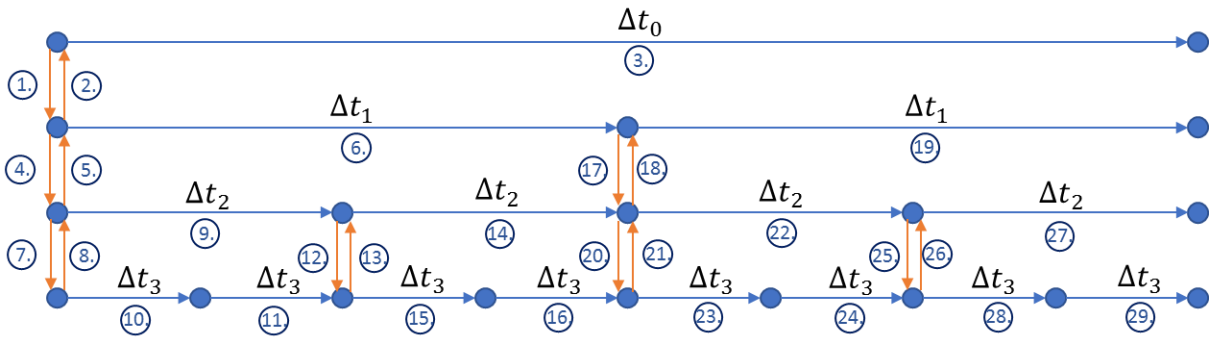


Fig. 4-5: Schematic illustration of the multi-grid algorithm for three refinement steps where every arrow is describing one process step

In the paragraph below a pseudo code example of the recursive function *doGridRefinement* can be found.

```
void doGridRefinement(lvl){

      dim3 blocks = make_uint3(localNodesY[lvl], localNodesZ[lvl], 1);
      uint threads = localNodesX[lvl];

      if(lvl < (max_lvl-1)){ //first if-condition

            dim3 blocksfine = make_uint3(localNodesY[lvl+1],
                                         localNodesZ[lvl+1], 1);
            uint threadsfine= localNodesX[lvl+1];

            gridCoupling<<<blocks,threads,0>>>(h_i[lvl],h_i[lvl+1],lvl);
            gridCoupling<<<blocksfine,threadsfine,0>>>
                                              (h_i[lvl+1],h_i[lvl],lvl+1);
      }

      doOneTimeStep(blocks, threads, lvl);
      current_teta=current_teta+ angularVelocityLU[lvl];
      moveMovingBoundary(current_teta, blocks, threads, lvl);

      if(lvl < (max_lvl-1)){//second if-condition -> break condition
            doGridRefinement(lvl+1);//first recursive call
            doGridRefinement(lvl+1); //second recursive call
      }
}
```

The function is called from the main file by *doGridRefinement(0)* meaning the input parameter *lvl* equals zero and as a consequence all values for ‘grid 0’ are taken. A 2D grid of blocks is defined with the number of local nodes in y and z direction. Each block is a mono-dimensional array of threads with the same number of elements than number of local nodes in x direction.

If the level under focus *lvl* is smaller than the maximum number of levels *max_lvl* minus one, grid coupling is executed. In the case of three refinement steps, as in the example in Fig. 4-5, four grids are used and, therefore, *max_lvl* equals four. As a result, the if-condition is true and grid coupling between ‘grid 0’ and ‘grid 1’ is done. For the fine-to-coarse coupling the blocks and threads are redefined with the number of local nodes of ‘grid 1’.

Afterwards, the functions *doOneTimeStep(0)* and *moveMovingBoundary(0)* are called.

The abortion condition of the recursive function is given by the second if-condition. As long as *lvl* is smaller than three, the function *doGridRefinement(lvl+1)* is called with its input parameter increased by one. This is the first recursive call of *doGridRefinement(1)*.

In Fig. 4-6 a function tree is illustrated for the example case of four grids. The numbers next to the function calls signify the value of parameter *lvl*. The blue circled numbers are referencing to the process steps from Fig. 4-5. The pink bold arrows signify the function control flow and, therefore, the sequence of function calls. In the function tree it can be seen, that after the first recursive call of *doGridRefinement(1)*, the whole procedure is redone with *lvl* equals one. Since the second if-condition evaluates as true, the function *doGridRefinement* is called with its input level increased by one. This is the first recursive call of *doGridRefinement(2)*.

The whole procedure is redone with *lvl* equals two. Since the second if-condition still evaluates as true, the function *doGridRefinement* is called with its input level increased by one. This is the first recursive call of *doGridRefinement(3)*.

In the case of *lvl* equals three, the first if-condition is false and no grid coupling is done as there does not exist a finer grid than ‘grid 3’. The function *doOneTimeStep(3)* and *moveMovingBoundary(3)* are called. Afterwards, the second if-condition evaluates as false and the first recursive call of *doGridRefinement(3)* is exited. By doing so, the function control flow is jumping to the second recursive call of *doGridRefinement(3).* Again, the first if-condition is false and no grid refinement is done. The function *doOneTimeStep(3)* and *moveMovingBoundary(3)* are called. Afterwards, the second if-condition evaluates as false and the second recursive call of *doGridRefinement(3)* is exited.

The function control flow is jumping to the second recursive call of *doGridRefinement(2).* The procedure continues according to the function tree until *doOneTimeStep(3)* with the process

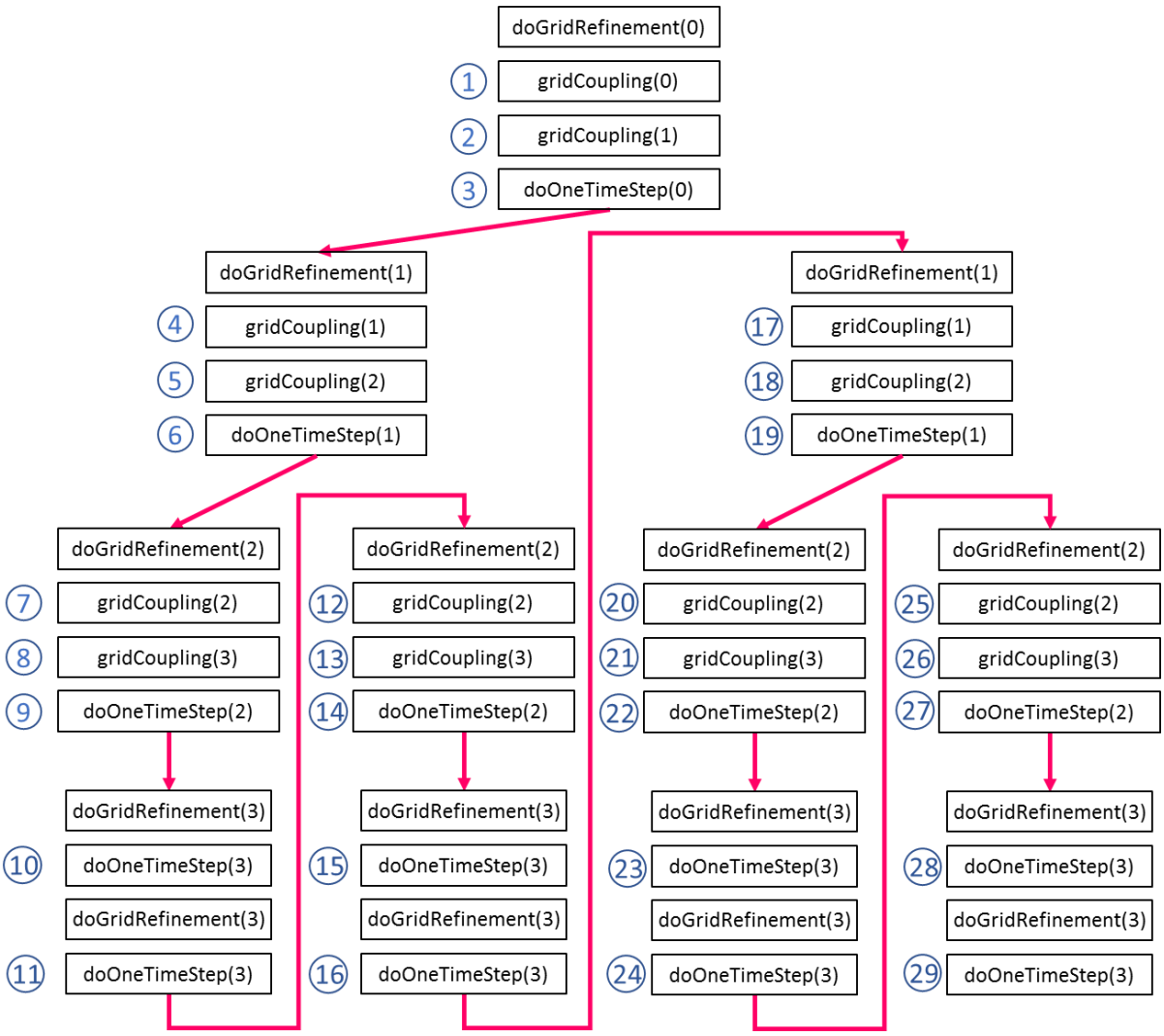


Fig. 4-6: Function tree of the recursive function for three refinement levels

step number 29 is finished. After that the function control flow is jumping to the main file and the function *output* is called.

5 Testing and Validation

In this chapter, the simulation results are compared with the analytical solution of the Navier-Stokes equation in the case of Hagen-Poiseulle flow. For the case where no analytical solution is available, as for the stirred tank reactors, the simulation results are evaluated with experimental results from literature. For the validation, two different cases are chosen. The first – Hagen-Poiseuille flow in a cylindrical tube – is examined in section 5.1. The second case is investigated in section 5.2 and is focusing on agitated tanks.

5.1 Hagen-Poiseuille Flow in a Cylindrical Tube

In this section, the simulation results for a pipe flow are presented. The geometry of the three-dimensional simulation cases is uniform and given in Tab. 5-1. To investigate the influence of grid refinement on the results of the simulation, the size and position of the refined region is varied as well as the Reynolds number. A sketch of the simulated geometry can be found in Fig. 5-1. At the inlet, which is shown on the left for all cases, a constant velocity is applied by the velocity boundary condition described in section 2.1.5.4. Just before the outlet a small absorbing layer, as illustrated in section 2.1.5.6, is placed. The pressure boundary condition from section 2.1.5.5 is used at the outlet on the right side of the tube.

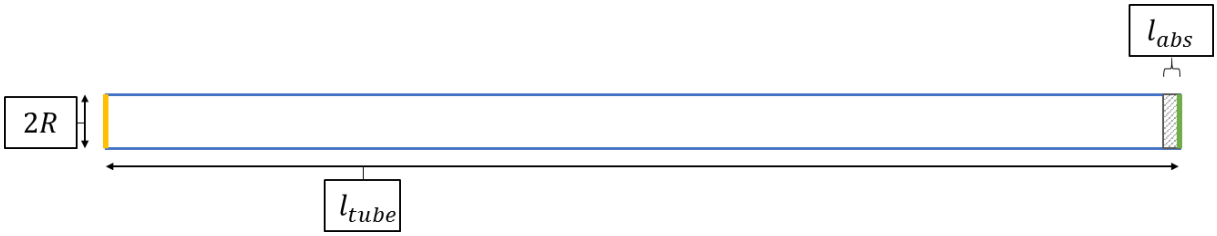


Fig. 5-1: Sketch of the simulated geometry of the first case with inlet shown in yellow and outlet in green

Tab. 5-1: Geometric data of the first case

Nodes in x direction	75	Radius of tube R [m]	0.05
Nodes in y direction	75	Length of tube l_{tube} [m]	2
Nodes in z direction	1401	Length of absorbing layer l_{abs} [m]	0.03

Fully developed, laminar and steady-state flow of an incompressible Newtonian fluid in a cylindrical tube is called Hagen-Poiseuille flow and leads to a simplification of the Navier-Stokes equations. From them, equation 5-1 can be derived, which gives the parabolic velocity profile in z-direction as a function of the dynamic viscosity μ, the pressure drop along the tube length $\frac{\Delta p}{l_{dev}}$ and the radius of the tube R. [27, pp. 115-117]

$$u_z(r) = -\frac{1}{4\mu}\frac{\Delta p}{l_{dev}} R^2 \left(1 - \frac{r^2}{R^2}\right) \quad (5\text{-}1)$$

Since the velocity is set to a constant value across the inlet, the flow profile is not developed at the entry section, and therefore, the law of Hagen-Poiseuille is not valid along the whole length of the tube. The region, in which the flow is developing, is referred to as hydrodynamic inlet length l_{hyd}.There exist different correlations for estimating this hydrodynamic inlet length. An extensive list can be found in the work of Bochardt [28, p. 10]. The method of Jirka and Lang was chosen, as it is also referenced by Brenn [29] and is given in equation 5-2.

$$\frac{l_{hyd}}{2R} \approx 0.05 * Re \quad (5\text{-}2)$$

Therefore, in the following sections the length of the tube without the hydrodynamic inlet length from equation 5-2 and further without the length of the absorbing layer is used for calculation.

$$l_{dev} = l_{tube} - l_{hyd} - l_{abs} \quad (5\text{-}3)$$

Likewise, the data from the absorbing layer was excluded in the plots of velocity and pressure over the length of the pipe. Additionally, for the total pressure drop from simulation Δp_{simu} the data from the absorbing layer as well as the hydrodynamic inlet length from equation 5-2 was excluded.

Further, from the NSE it is found, that the maximum velocity u_{max}, located in the cross-sectional mid-point of the tube, is equal twice the volume equivalent – mean – velocity u_m. In equation 5-4, the volume equivalent velocity u_m, is shown to be the volume flow $\dot{Q}$ divided by the cross-sectional area of the tube A. [27, pp. 117-118]

$$u_m = \frac{\dot{Q}}{A} \quad (5\text{-}4)$$

$$u_{max} = 2u_m \quad (5\text{-}5)$$

By inserting equation 5-5 into equation 5-1 at $r = 0$, the total pressure drop occurring in the tube can be determined. [27, pp. 118-119]

$$-\Delta p = \frac{8 u_m l_{dev} \mu}{R^2} \quad (5\text{-}6)$$

The relative error of the simulated pressure drop is computed as in equation 5-7.

$$\varepsilon = \frac{|\Delta p_{simu} - \Delta p_{calc}|}{\Delta p_{calc}} \cdot 100 \quad (5\text{-}7)$$

For the case of pipe flow, the code was adapted in order to be able to compare the simulation results with the results of the Hagen-Poiseuille flow. These changes refer to the definition of the solid boundary which is intended to represent the cylindrical pipe. Typically, one major advantage of grid refinement is the higher resolution of boundaries inside the refined region. As it can be seen in the example case of Fig. 5-2 and Fig. 5-3, the shape of the cross-sectional area is approaching a circular shape with increased refinement. However, a problem emerges from the deviation of area available for the fluid motion. It is slightly different in the region of refined grid, leading to jumps in the plots of pressure and velocity over tube length. Therefore, the implementation of the fine boundary was adapted to exactly represent the boundary of the coarse grid as it can be seen in Fig. 5-4.

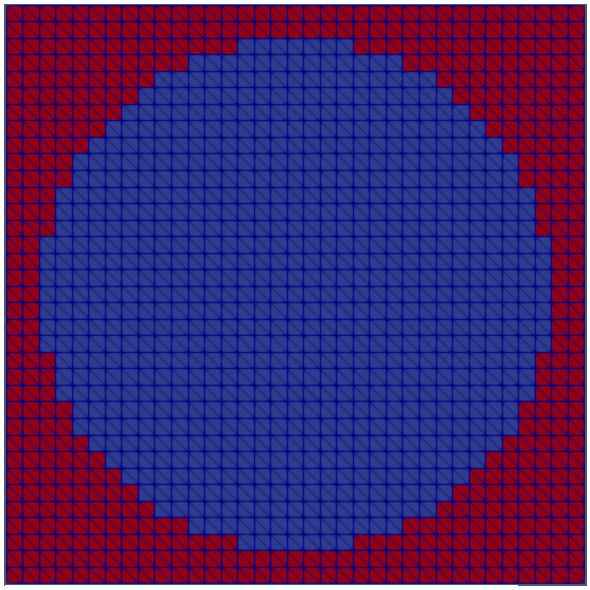


Fig. 5-2: Cross-sectional view of the solid boundary of the coarse grid for 35x35 coarse nodes

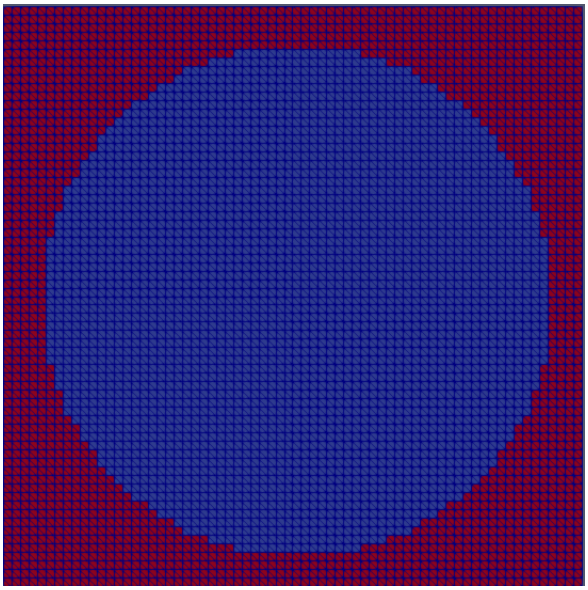


Fig. 5-3: Cross-sectional view of the solid boundary of the fine grid for 68x68 fine nodes

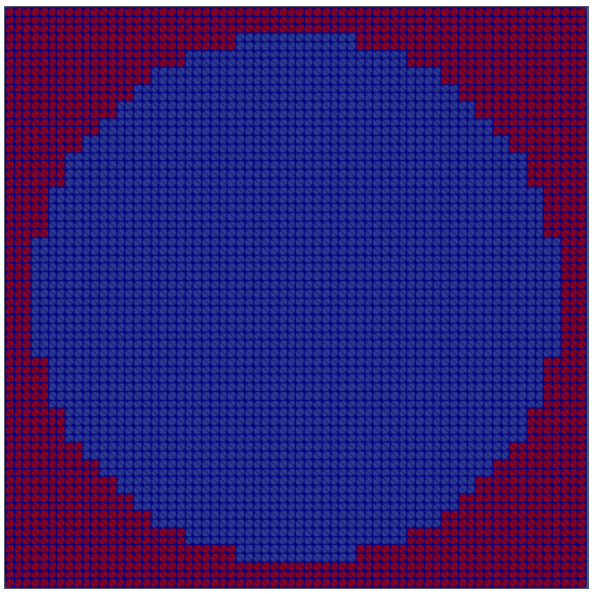


Fig. 5-4: Adapted solid boundary of the fine grid to guarantee the same cross-sectional area as in the coarse grid for 68x68 fine nodes

In the following sections, the results are processed to be comparable. Therefore, for instance the velocity at the center line of the tube u is normalized with the input velocity u_{in} and is plotted over the normalized length obtained by dividing the position on the tube l_z with the relevant region of the tube $l_{tube} - l_{abs}$. This leads to readable plots which can be matched with the expected results from the Hagen-Poiseuille equation. The longitudinal velocity and pressure fields of each simulation can be found in the appendix in section 7.5.

5.1.1 Set-up 1

Two simulations have been done in a pipe using the geometry described in Tab. 5-1 and the fine grid placed, as shown in Fig. 5-5, in the second half of the tube, where the flow profile is already fully developed. The first simulation was done with an inlet velocity of $u_{in} = u_m = 0.0001\left[\frac{m}{s}\right]$, leading to a Reynolds number of 10 and for the second simulation an inlet velocity of $u_{in} = u_m = 0.001\left[\frac{m}{s}\right]$ was used to reach a Reynolds number of 100.

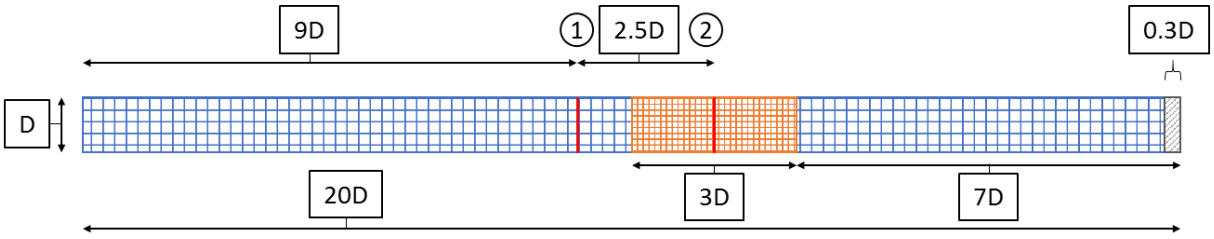


Fig. 5-5: Dimensional sketch of set-up 1 with the coarse grid in blue, the fine grid in orange, the absorption layer in grey and positions 1 and 2 marked with red lines

5.1.1.1 Simulation of a Reynolds Number of 10

For a Reynolds number of 10, the relative error between calculated and simulated pressure drop is 0.306%, as shown in Tab. 5-2. In appendix 7.4 the results from a coarse grid reference simulation can be found. Comparing the magnitude of the relative error, the grid refinement reduced the error by 0.205%.

Tab. 5-2: Comparison of pressure drop for a Reynolds number of 10 for set-up 1

$\Delta p_{calc}[Pa]$	$\Delta p_{simu}[Pa]$	$\varepsilon[\%]$
$6.053 \cdot 10^{-4}$	$6.072 \cdot 10^{-4}$	0.306

In Fig. 5-6 and Fig. 5-7 the impact of grid refinement on the resolution of the velocity field can be seen. As it is illustrated in Fig. 5-5, position 1, and therefore, Fig. 5-6 is located in the coarse region and position 2 and Fig. 5-7 is located in the fine region. Both displayed velocity fields are axially symmetrical and do not have any artifacts from grid coupling.

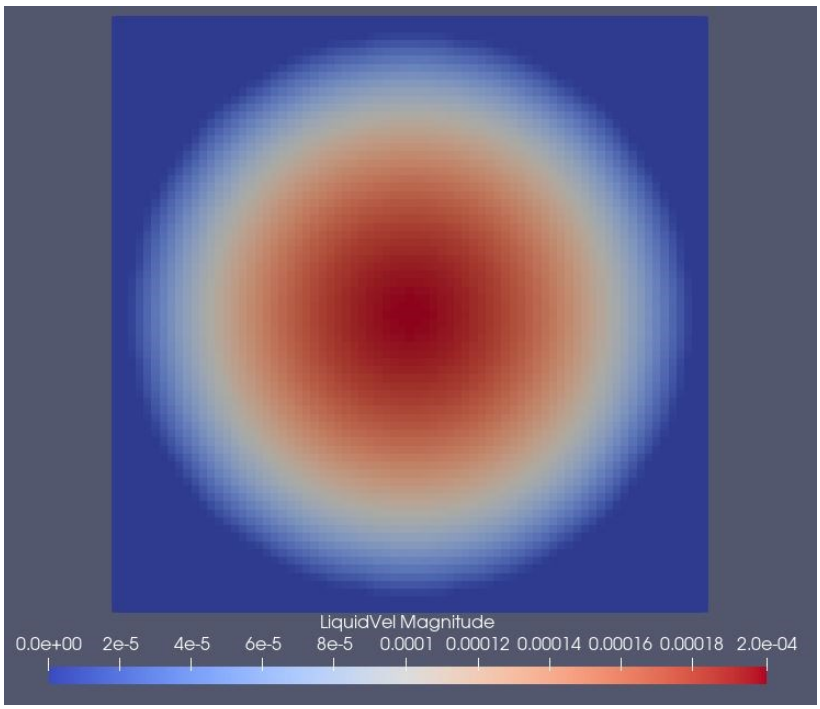


Fig. 5-6: Cross-sectional view of the velocity field at position 1

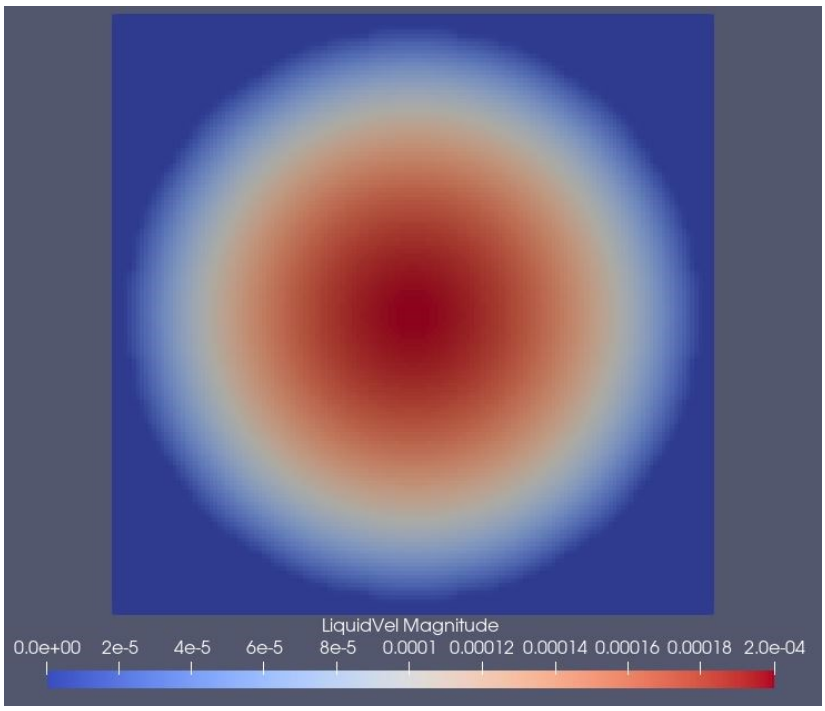


Fig. 5-7: Cross-sectional view of the velocity field at position 2

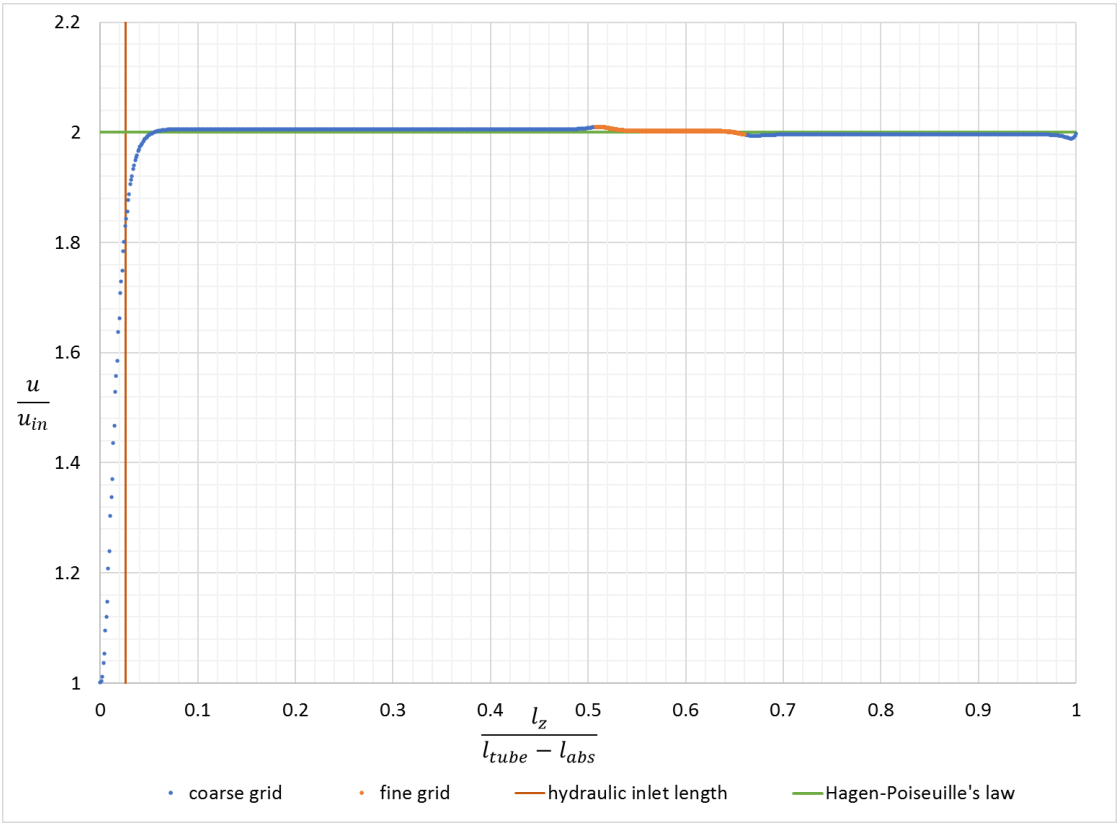


Fig. 5-8: Velocity at center line plotted over the length of the tube for set-up 1 and $Re = 10$

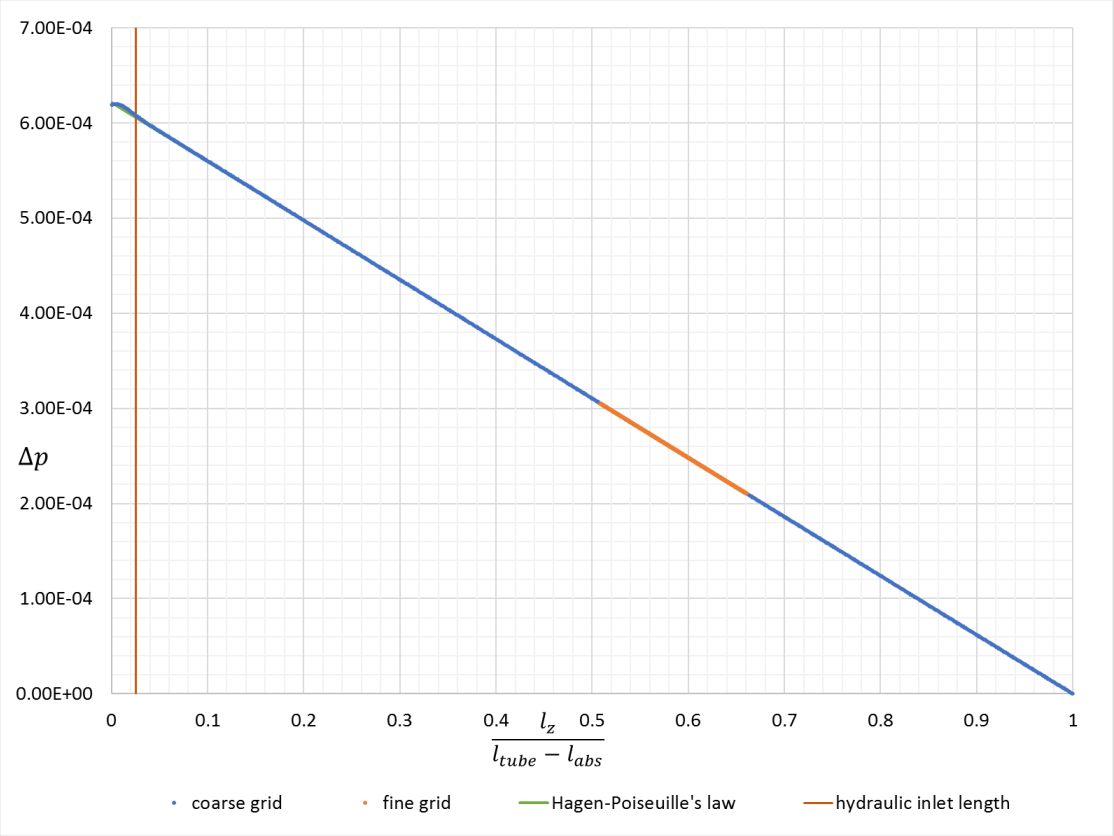


Fig. 5-9: Difference to ambient pressure at centerline plotted over the length of the tube for set-up 1 and $Re = 10$

In Fig. 5-8 and Fig. 5-9 the normalized velocity and the pressure drop is plotted over the normalized length of the tube respectively. Further, the hydrodynamic inlet length calculated from equation 5-2 is marked with a brown line. The analytical solution is shown with a green line. In the velocity plot, the maximum velocity from the simulation is higher than twice the inlet velocity. In the appendix, in Fig. 7-1 the results of the non-refined simulations show, that this is an effect of the simulation itself and not due the grid refinement. This effect is increased with decreased grid resolution. A possible explanation of this effect is, that the Hagen-Poiseuille equation in the form given in equation 5-1 is only valid for a circular cross-sectional area and the real cross-sectional area differs from the shape of a circle as it can be seen in Fig. 5-2 and Fig. 5-3

The velocity plot shows a peak of the maximum velocity after the fluid is entering the refined region and a valley after it is leaving it. In addition, the refined region seems to induce a small flow resistance, as the maximum velocity after the refined region is smaller than in front of it.

Taking a look at the hydrodynamic inlet length, it can be seen, that the simulated flow is fully developed at a pipe length of 0.09 which is larger than the calculated of 0.026. However, as the plotted pressure gradient matches the analytical pressure gradient – both plotted in Fig. 5-9 – it is sufficient to use the calculated value for the calculation of the pipe length as in equation 5-3.

5.1.1.2 Simulation for Reynolds Number of 100

For a Reynolds number of 100, the relative error between calculated and simulated pressure drop is 0.53%, as shown in Tab. 5-3. In appendix 7.4 the results from a coarse grid reference simulation can be found. Comparing the magnitude of the relative error, the grid refinement reduced the error by 0.189%.

Tab. 5-3: Comparison of pressure drop for a Reynolds number of 100 for set-up 1

$\Delta p_{calc}[Pa]$	$\Delta p_{simu}[Pa]$	$\varepsilon[\%]$
$4.613 \cdot 10^{-3}$	$4.638 \cdot 10^{-3}$	0.53

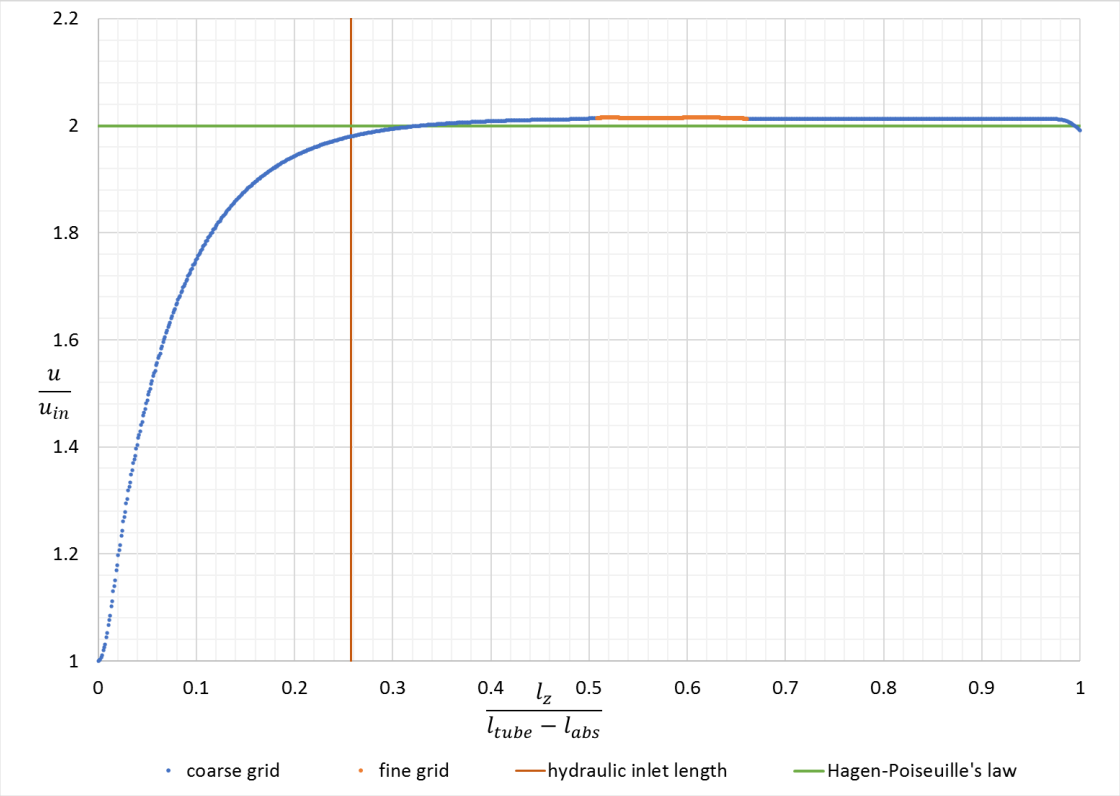


Fig. 5-10: Velocity at center line plotted over the length of the tube for set-up 1 and $Re = 100$

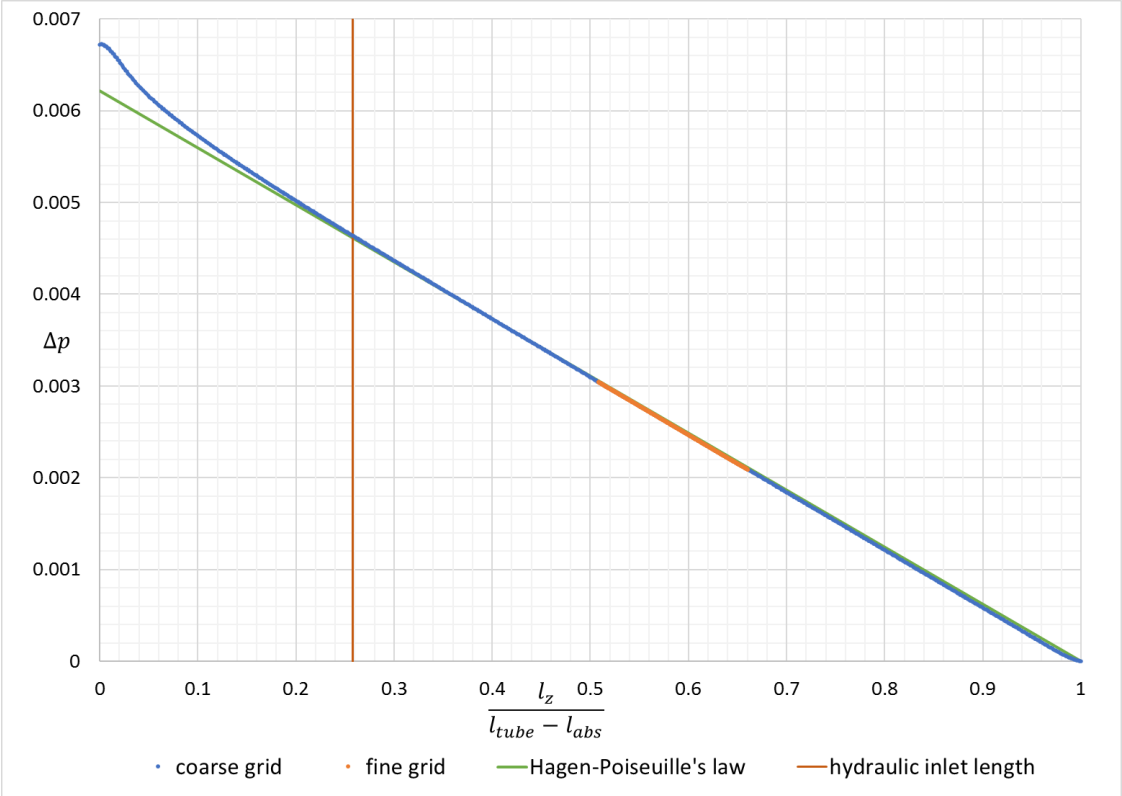


Fig. 5-11: Difference to ambient pressure at centerline plotted over the length of the tube for set-up 1 and $Re = 100$

In Fig. 5-10 and Fig. 5-11 the normalized velocity and the pressure drop is plotted over the normalized length of the tube respectively. Further, the hydrodynamic inlet length calculated from equation 5-2 is marked with a brown line. The analytical solution is shown with a green line. Due to the increase of the inlet velocity u_{in}, the hydrodynamic inlet length is increased. It is approximated with equation 5-2 to be 0.26 and from the velocity plot a value of 0.5 can be extracted. However, the pressure plot reveals that the gradient of the pressure from the simulation starts matching the gradient from the analytical solution at the calculated inlet length.

Comparing the velocity plots of the simulations for a Reynolds number of 10 and of 100, i.e. Fig. 5-8 and Fig. 5-10, it can be seen that the effects of flow resistance, as well as, the peaks and valleys of velocity at the boundaries between the grids are decreased with increased velocity. It can be observed, that the velocity is decreased in the very last section from the tube. This is due to the absorbing layer, which is placed right after this section and is influencing the fluid motion slightly.

5.1.2 Set-up 2

For the second set up the fine grid is located at the inlet of the tube, as illustrated in Fig. 5-12. The geometry is the same as given in Tab. 5-1. Like in set-up 1, two simulations have been done. The first with an inlet velocity of $u_{in} = u_m = 0.0001 \left[\frac{m}{s}\right]$, leading to a Reynolds number of 10 and the second with an inlet velocity of $u_{in} = u_m = 0.001 \left[\frac{m}{s}\right]$, to reach a Reynolds number of 100.

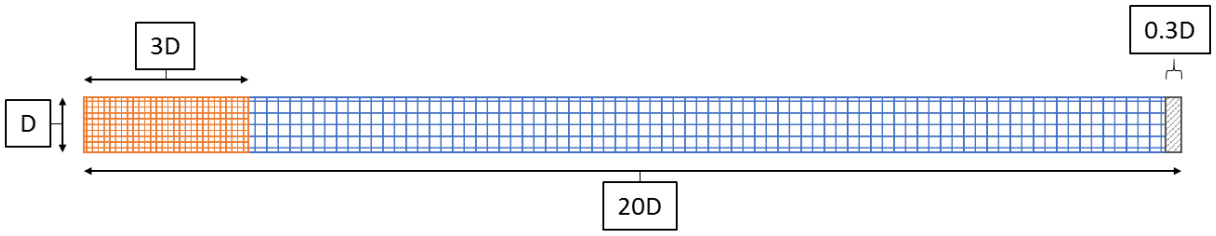


Fig. 5-12: Dimensional sketch of set-up 2 with the coarse grid in blue, the fine grid in orange and the absorption layer in grey

5.1.2.1 Simulation for Reynolds Number of 10

For a Reynolds number of 10, the relative error between calculated and simulated pressure drop is 0.669%, as shown in Tab. 5-4. In appendix 7.4 the results from a coarse grid reference simulation can be found. Comparing the magnitude of the relative error, the grid refinement reduced the error by 0.158%. Comparing this with the results of the other cases, it can be seen, that solely with this configuration the relative error is increased by refining the grid. In all other cases the simulated pressure drop Δp_{simu} is higher than the calculated pressure drop Δp_{calc}.

Tab. 5-4: Comparison of pressure drop for a Reynolds number of 10 for set-up 2

$\Delta p_{calc}[Pa]$	$\Delta p_{simu}[Pa]$	$\varepsilon[\%]$
$6.053 \cdot 10^{-4}$	$6.013 \cdot 10^{-4}$	0.669

Taking the focus on the velocity plot in Fig. 5-13 and the pressure plot in Fig. 5-14, where the normalized velocity and the difference to ambient pressure in the mid-point of the tube are plotted over the normalized length of the tube, a similar behavior is observed as with the total pressure difference. The results of the simulations done with set-up 1, as well as the results of set-up 2 with a Reynolds number of 100, which is shown in the next section, all show a maximum velocity higher than the – in green plotted – analytical solution of Hagen-Poiseuille flow. Further, from the inlet on the simulated pressure line is located below the line of the

analytical solution which is not the case in any of the other simulations. Just as in the simulation of set-up 1 with a Reynolds number of 10, the fluid velocity is decreased after leaving the refined region.

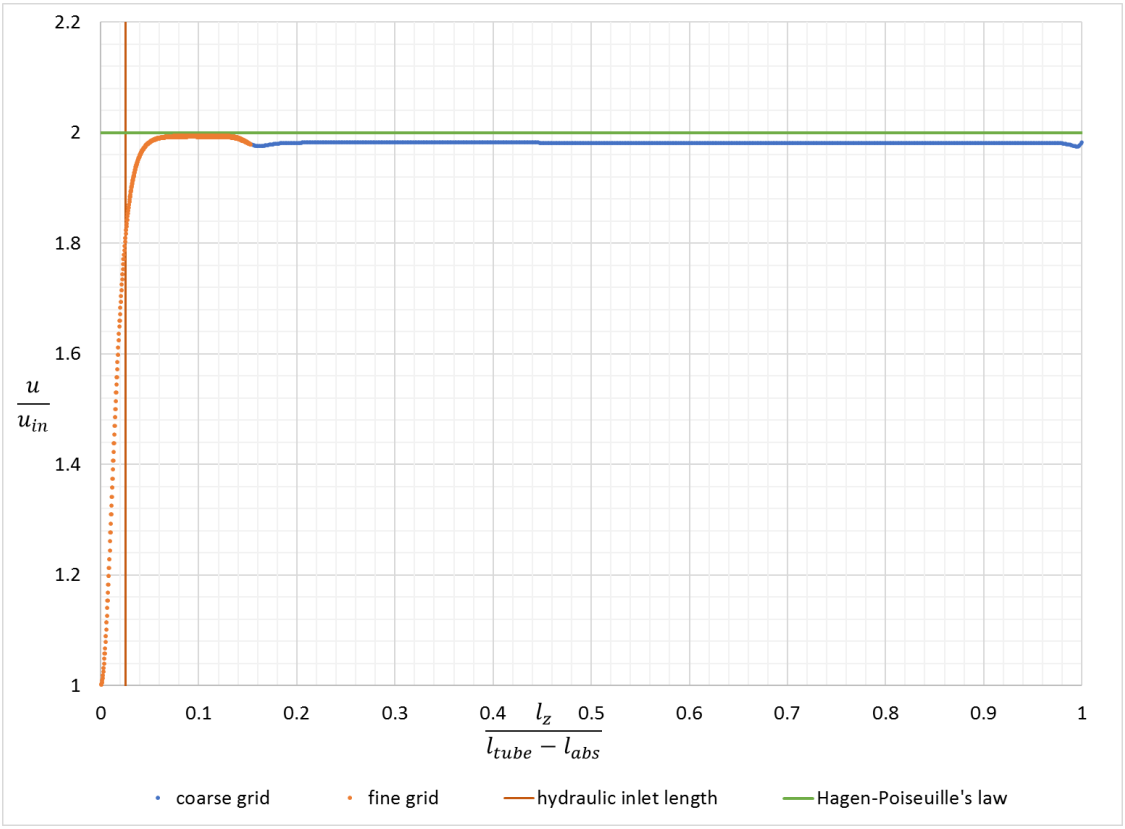


Fig. 5-13: Velocity at center line plotted over the length of the tube for set-up 2 and $Re = 10$

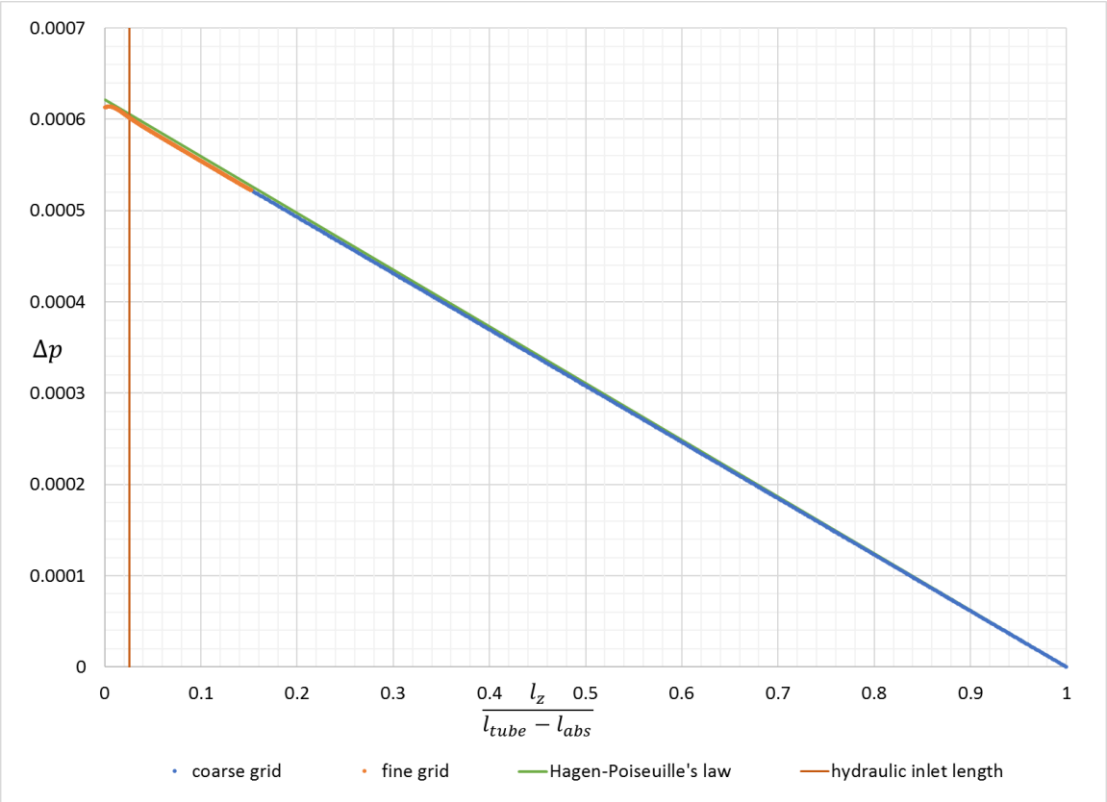


Fig. 5-14: Difference to ambient pressure at centerline plotted over the length of the tube for set-up 2 and $Re = 10$

5.1.2.2 Simulation for Reynolds Number of 100

For a Reynolds number of 100, the relative error between calculated and simulated pressure drop is 0.467%, as shown in Tab. 5-5. In appendix 7.4 the results from a coarse grid reference simulation can be found. Comparing the magnitude of the relative error, the grid refinement reduced the error by 0.252%.

Tab. 5-5: Comparison of pressure drop for a Reynolds number of 100 for set-up 2

$\Delta p_{calc}[Pa]$	$\Delta p_{simu}[Pa]$	$\varepsilon[\%]$
$4.613 \cdot 10^{-3}$	$4.635 \cdot 10^{-3}$	0.467

In Fig. 5-15 and Fig. 5-16 the normalized velocity and the pressure drop is plotted over the normalized length of the tube respectively. Further, the hydrodynamic inlet length calculated from equation 5-2 is marked with a brown line. The analytical solution is shown with a green line. As for the simulation of a Reynolds number of 100 in set-up 1 the plotted velocity curve matches the curve of the reference simulation in Fig. 7-1. Further, the effect of a velocity decrease after the fluid is leaving the refined region is not visible.

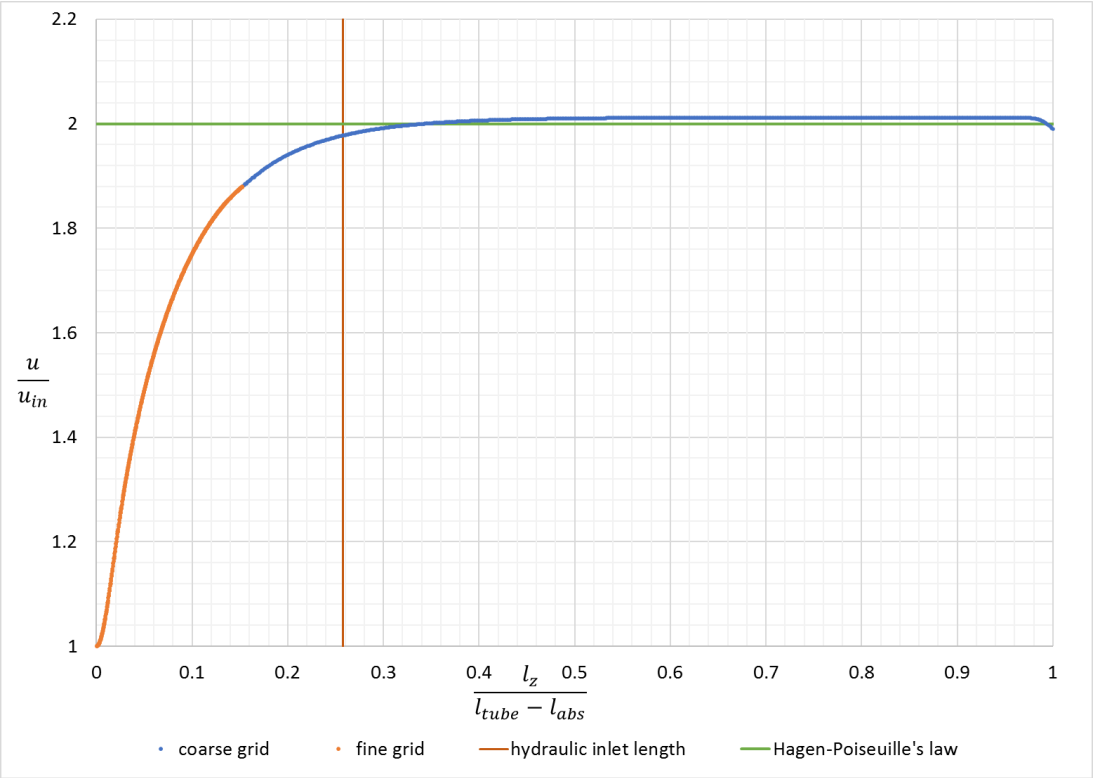


Fig. 5-15: Velocity at center line plotted over the length of the tube for set-up 2 and $Re = 100$

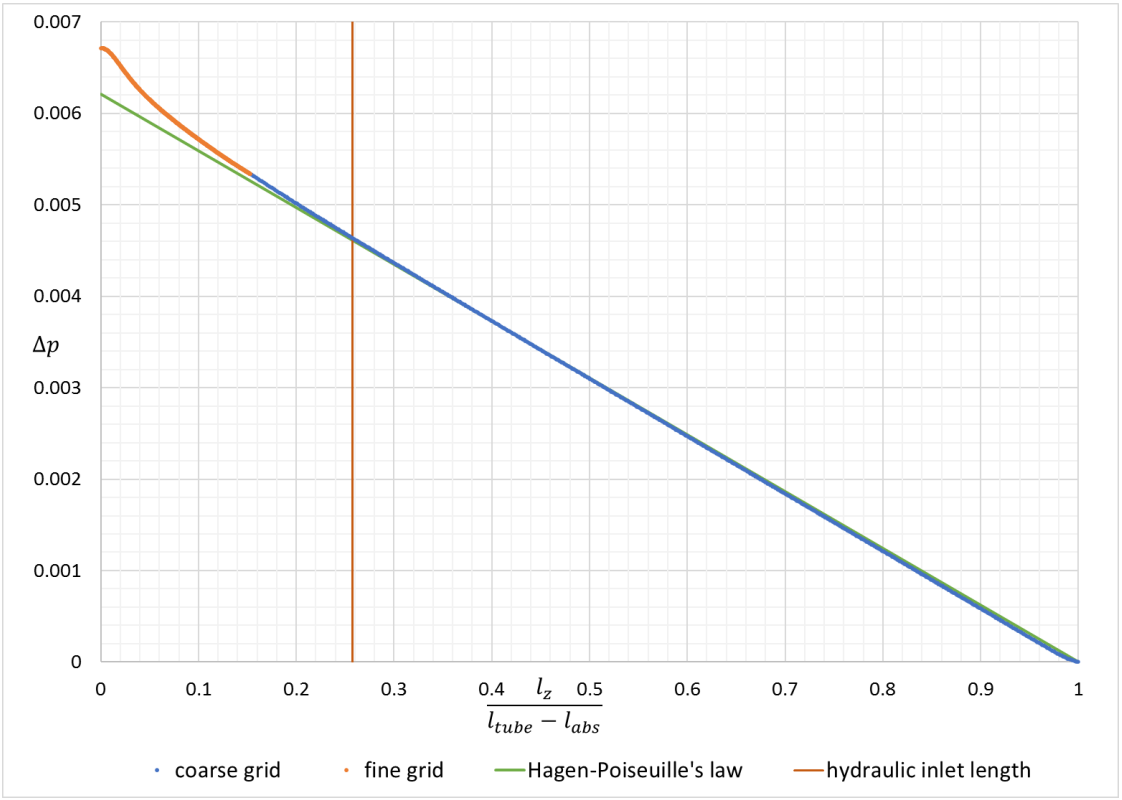


Fig. 5-16: Difference to ambient pressure at centerline plotted over the length of the tube for set-up 2 and $Re = 100$

5.1.3 Set-up 3

The aim of the third set up is to distinguish if the parabolic flow profile can be reproduced in case of partial cross-sectional refinement. Therefore, as illustrated in Fig. 5-17, the refined region was placed in the second half of the pipe to guarantee a fully developed flow inside the refined region. The geometry is the same as given in Tab. 5-1. An inlet velocity of $u_{in} = u_m = 0.0001 \left[\frac{m}{s}\right]$ was chosen to reach a Reynolds number of 10.

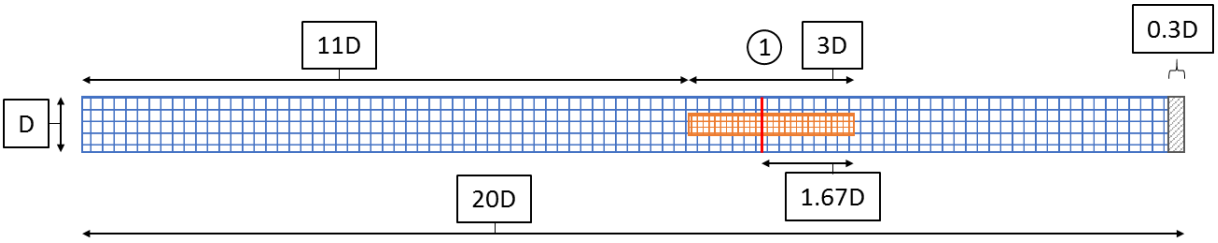


Fig. 5-17: Dimensional sketch of set-up 3 with the coarse grid in blue, the fine grid in orange, the absorption layer in grey and position 1 marked with a red line

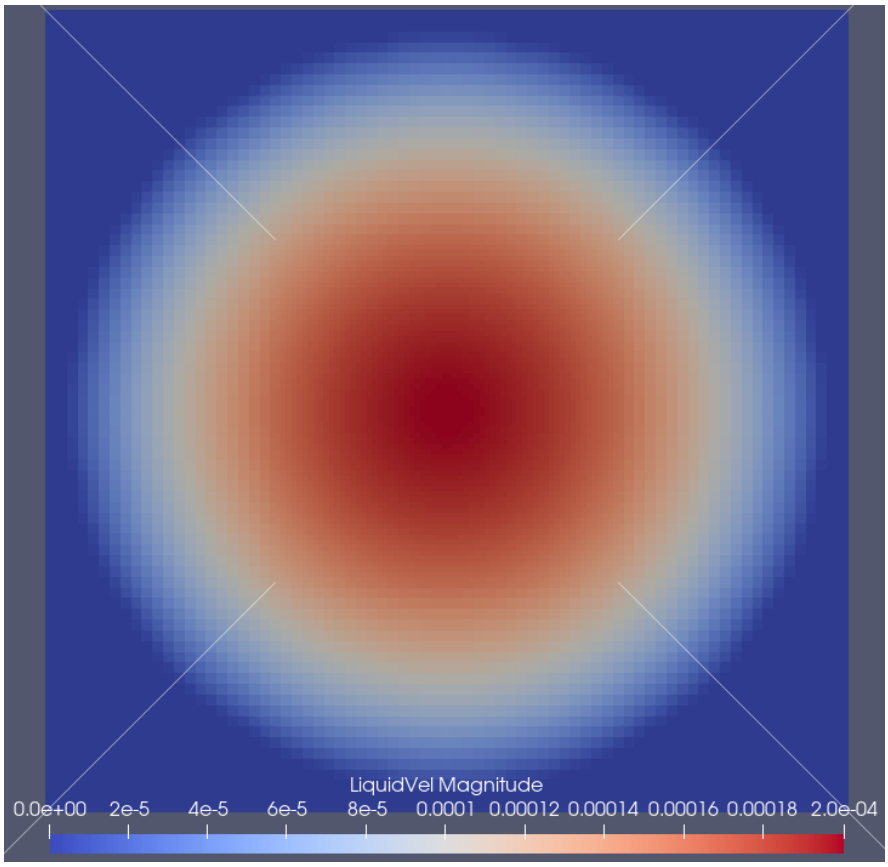


Fig. 5-18: Cross-sectional velocity field at position 1. The white lines are positioning the corners of the fine grid

In Fig. 5-18 the cross-sectional velocity field at position 1 – in Fig. 5-17 marked with a red line – is shown. The plotted white lines end each in a corner of the fine grid. It can be seen, that the velocity field is axically symmertical and that no artefacts of the grid coupling are visible.

This is further confirmed by Fig. 5-19, where the velocity profile is plotted over the diameter of the tube. The velocity values obtained from the coarse grid as well as the velocity values from the fine grid lie exactly on the curve of the analytical solution. The Hagen-Poiseuille profile was obtained, by using the exact pipe diameter from the simulation at position 1 and the prevailing velocity.

In addition, it can be seen in Fig. 5-20 that the pressure over the diameter is constant, as it is intrinsic for Hagen-Poiseuille flow. However, the magnitude of pressure obtained from the simulation at position 1 is slightly higher than it is obtained from the analytical solution at this point.

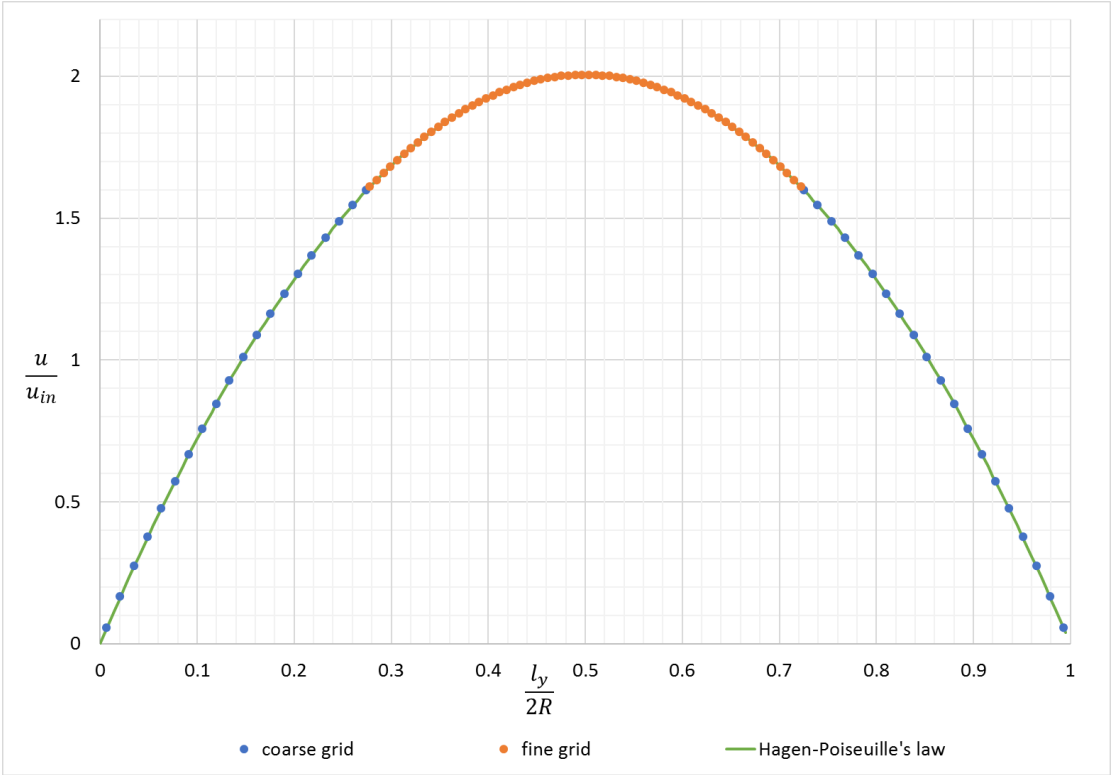


Fig. 5-19:Velocity profile over the diameter of the tube at position 1

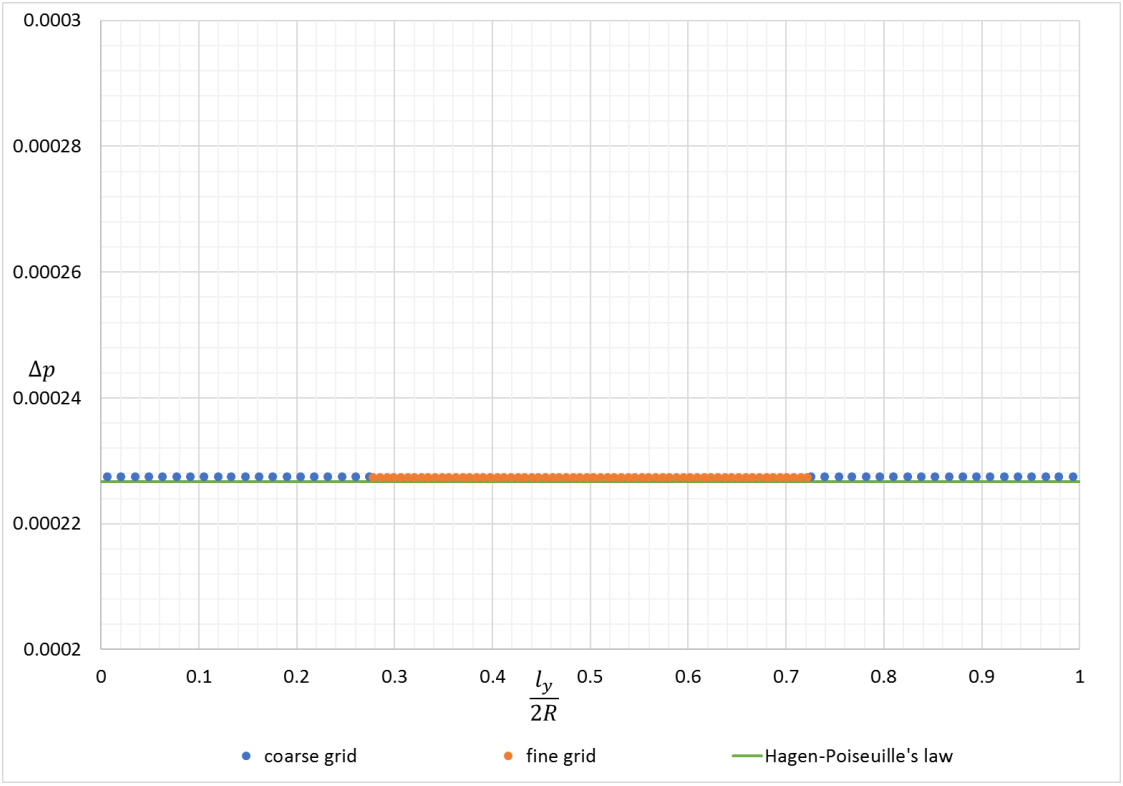


Fig. 5-20:Pressure profile over the diameter of the tube at position 1

An indeed interesting behavior can be seen, if the normalized velocity is plotted over the normalized length of the tube, as it is done in Fig. 5-21. After reaching its maximum value the velocity is constant along the length and no peaks nor valleys are observed after the fluid is entering and leaving the refined region. By comparing the velocity plot in Fig. 5-21 with the velocity plot obtained from the reference simulation without grid refinement in it is discovered that the curves match exactly.

Further, the relative error between calculated and simulated pressure drop is 0.513%, as shown in Tab. 5-6. By comparing this with magnitude of the relative error of the reference simulation a deviation of 0.002% is found. This implies eighter, that by refining the solid boundary, as it is done in set-up 1 and set-up 2, still a slight mismatch of boundaries occurs. Since, in the analytical solution of the problem (equation 5-6) the total radius of the pipe occurs with a power of two, the flow is very sensitive to variations of the cross-sectional area. Due to the law of conservation of mass after a decrease of the tube radius the mean as well as the maximum velocity is increased for incompressible fluids. [27, p. 56] Otherwise, the boundary handling of the grid coupling algorithm could be an error source. At the moment, the code does not check whether a ghost node which is receiving values from interpolation is

part of the solid boundary or not. Therefore, if the half of an interpolation cell is lying in the solid region it is possible for ghost nodes which lie in the solid region as well to receive non-zero velocity values. However, these nodes do not participate in the steaming and collision cycles.

Tab. 5-6: Comparison of pressure drop for a Reynolds number of 10 for set-up 3

$\Delta p_{calc}[Pa]$	$\Delta p_{simu}[Pa]$	$\varepsilon[\%]$
$6.053 \cdot 10^{-4}$	$6.084 \cdot 10^{-4}$	0.513

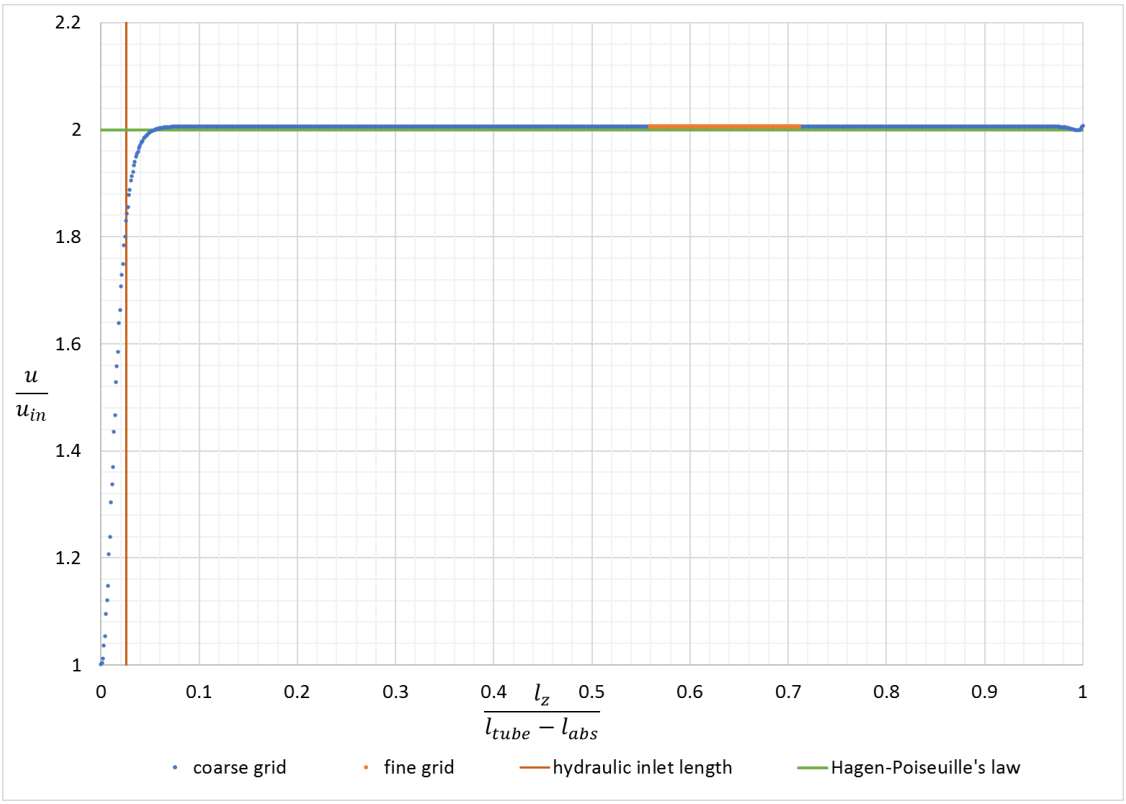


Fig. 5-21: Velocity at center line plotted over the length of the tube for set-up 3 and $Re = 10$

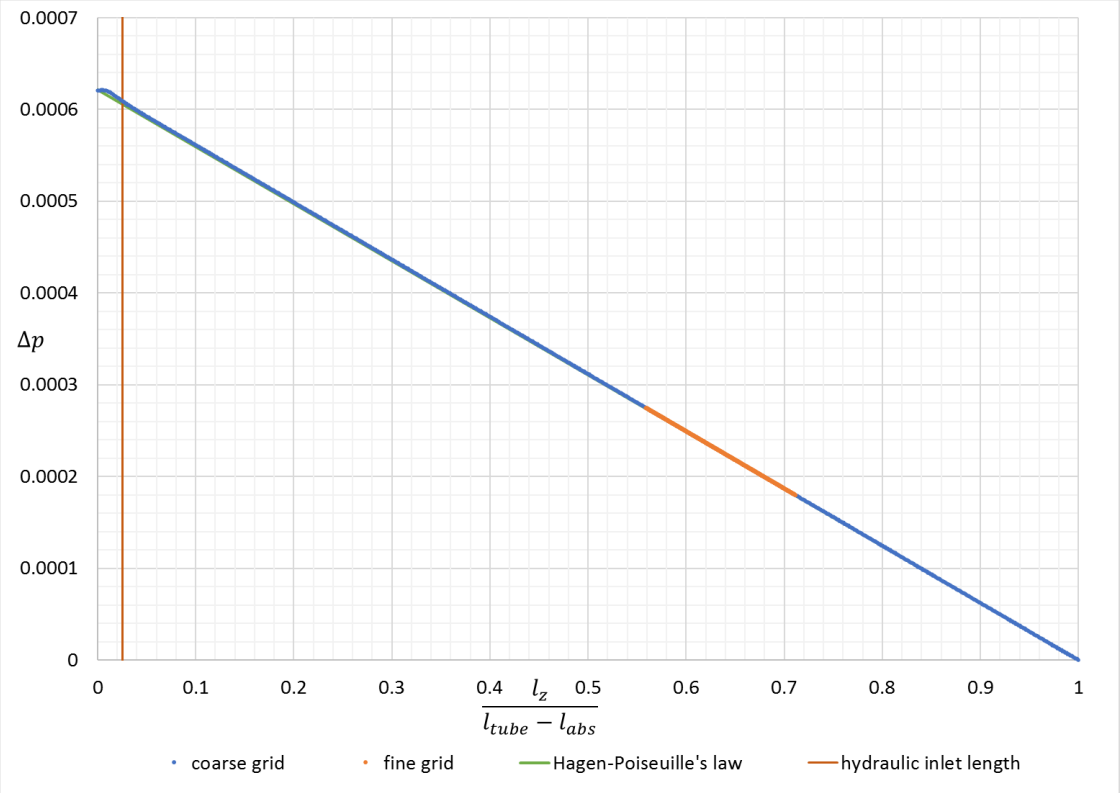


Fig. 5-22: Difference to ambient pressure at centerline plotted over the length of the tube for set-up 3 and $Re = 10$

5.2 Grid Refinement in Agitated Tanks

In this section, the simulation results for stirred tanks are presented to evaluate the behavior of grid refinement in turbulent flows. For that reason, two different set-ups have been chosen. In the first, presented in 5.2.1, a single-Rushton impeller is examined and in the second, presented in 5.2.2, a dual-Rushton impeller is investigated. The resulting flow patterns of both simulations are compared to flow patterns obtained from Particle Image Velocimetry (PIV) provided in reference literature. In this measurement technique, particles with the size of a few micrometers are dispersed in the flow and a high-speed camera takes two subsequent pictures of a laser illuminated sheet inside the transparent tank. In post-processing, the particles from the first picture are correlated with the particles of the second picture to calculate a two-dimensional velocity field. [30]

In addition, measurement results of a laser-Doppler anemometry are presented. There, two pairs of laser beams with different wave lengths are used to obtain the velocity components at a measurement point by detecting the scattered light which can be correlated to the velocity via its frequency shift. [31]

The references use T as variable for the tank diameter. This was changed in this work to D_T but is still present in the description of the axis in the diagrams taken from the references.

At the stirrer and the shaft, the moving wall boundary, described in section 2.1.5.3, is applied. For both cases, water under standard conditions is used as fluid.

5.2.1 Set-up 4

The geometry of the tank with the single-Rushton impeller is shown in Fig. 5-23 and all parameters are given in

Tab. 5-7. With the chosen size of the parameter C_1 which is the distance between the tank bottom and the stirrer – in literature often referred to as off-bottom clearance – a single loop flow pattern is observed. This means, that instead of the typical radial flow off the stirrer which is then separated at the tank wall into two loops, the stream is moving towards the bottom and forms a single loop by moving upwards along the tank wall. This behavior is stated to occur at an off-bottom clearance of $C_1 = 0.15 D_T$, where D_T is the diameter of the tank. [32] The fine grid is positioned around the impeller to better resolve its geometry and increase the number of nodes in this crucial flow region.

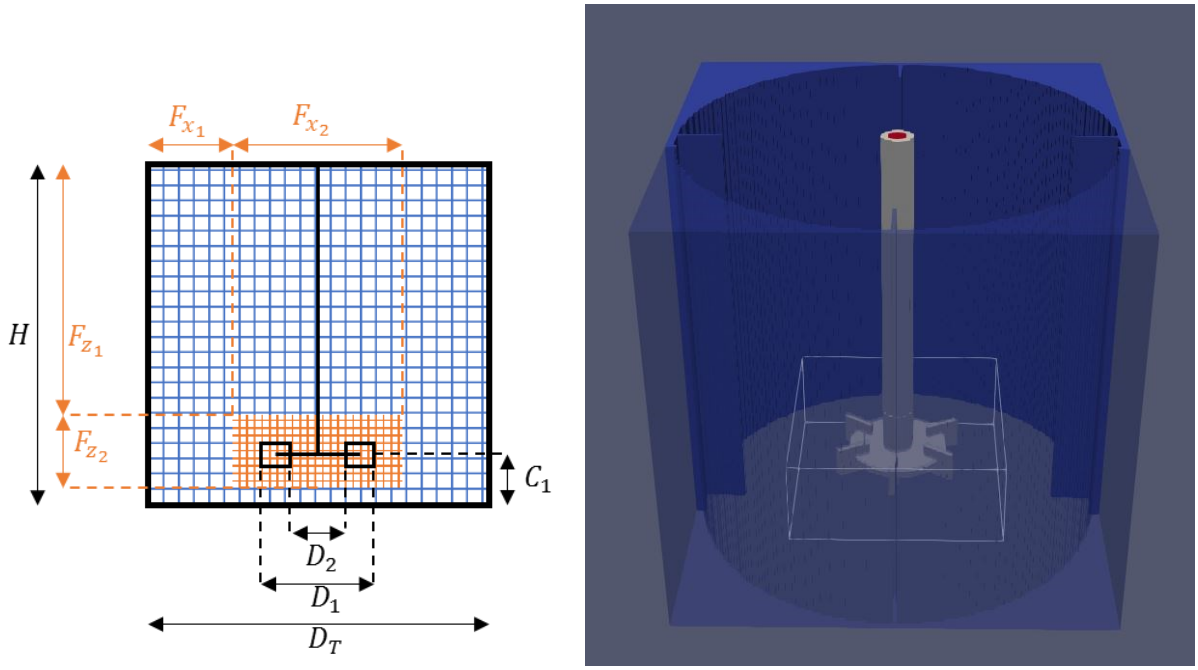


Fig. 5-23: Dimensional sketch of set-up 4 with the coarse grid in blue and the fine grid in orange on the left and 3D model of the simulation boundary on the right

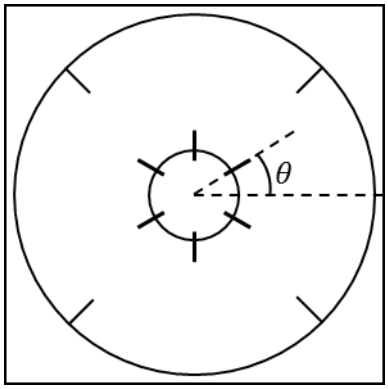


Fig. 5-24: Sketch of a tank from above showing the phase angle θ

In Fig. 5-25 the phase-averaged plot of a PIV measurement done by Li, Bao and Gao [32] is presented. It is obtained by averaging the velocity fields of over several hundred recorded flow patterns. The recordings were done exactly between two baffles for phase angles θ of 0° and further in intervals of 5° starting directly at the blade as it is shown in Fig. 5-24. The underlying contour plot in Fig. 5-25 shows the turbulent kinetic energy k, which is not calculated during the simulation. Therefore, in Fig. 5-26 the in-plane velocity vectors are plotted over the averaged liquid velocity. The reference plot shows velocity vectors which are scaled by the magnitude of the vector. In Fig. 5-26, the vectors are scaled by a constant value in order to make the secondary flows better visible. As well as in the reference a phase-averaged velocity field is shown, generated from two hundred snapshots at arbitrary phase-angles. The position of the fine grid is visible to the higher number of velocity vectors in that region. By comparing both results it can be seen, that the main flow correlates with the main flow from the measurements. At the outer blade diameter $D_1$the flow tends to move radially and is deflected towards reactor bottom shortly after. There, it is streaming parallel to the bottom until the reactor wall is reached, were it rises again to form a distinct single loop. The reference shows two secondary twirls. One in the region under the outer blade diameter and one at a radius of 0.235 of the normalized length scale. In the simulation this secondary flow appears as two vortices located at the outer blade diameter.

Besides that, in Fig. 5-27, the phase-averaged velocity of the whole vertical section of the reactor is plotted. It can be seen, that in the boundary region between the grids next to the stirrer shaft higher velocities occur in the fine grid than in the coarse grid. It seems that, as in the case of pipe flow, the velocity is artificially increased if the fine-coarse grid boundary crosses solid parts.

In Fig. 5-28 the cross-sectional view of an instantaneous velocity field is given. It shows how grid refinement increases the visibility of smaller eddies.

Tab. 5-7: Parameters for the configuration of set-up 4

Number nodes in x-direction	$[-]$	123
Number nodes in y-direction	$[-]$	123
Number nodes in z-direction	$[-]$	119
Tank diameter D_T	$[m]$	0.476
Tank height H	$[m]$	0.476
Distance of impeller from bottom C_1	$[m]$	0.071
Impeller speed	$[\frac{1}{min}]$	97
Impeller blade outer diameter D_1	$[m]$	0.158
Impeller blade inner diameter D_2	$[m]$	0.078
Number of blades	$[-]$	6
Blade height	$[m]$	0.032
Blade thickness	$[m]$	0.003
Disk diameter	$[m]$	0.118
Shaft diameter	$[m]$	0.035
Number of equally spaced baffles	$[-]$	4
Baffle width	$[m]$	0.048
Position of fine grid from top F_{z_1}	$[m]$	0.35
Height of fine grid F_{z_2}	$[m]$	0.101
Position of fine grid from domain boundary x=0 $F_{x_1} = F_{y_1}$	$[m]$	0.12
Width of fine grid F_{x_2}=F_{y_2}	$[m]$	0.263

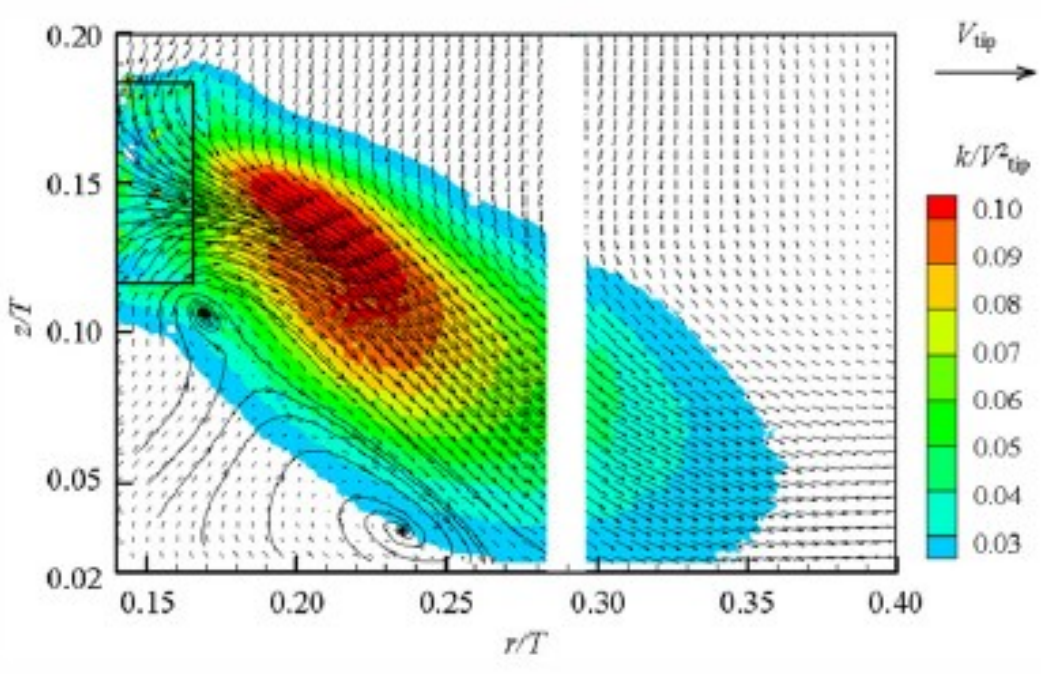


Fig. 5-25: Phase-averaged plot of the velocity from a PIV measurement done by Li et al. [32]

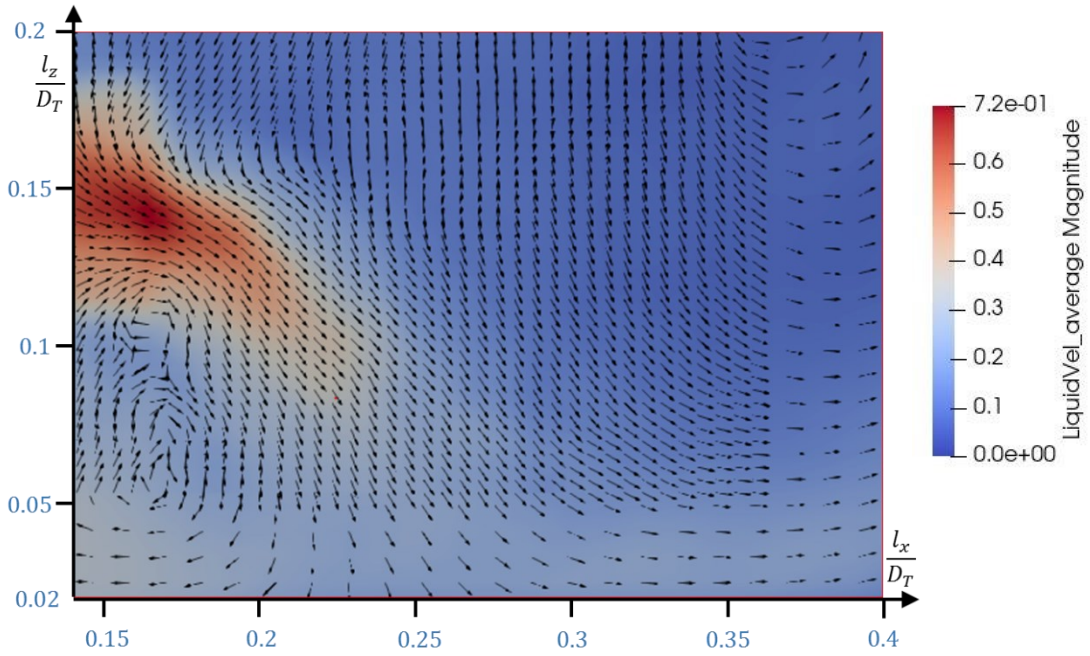


Fig. 5-26: Phase-averaged plot of the blade section of the velocity field obtained from the simulation with values in blue marking positions in units of normalized scale as used in reference in Fig. 5-25

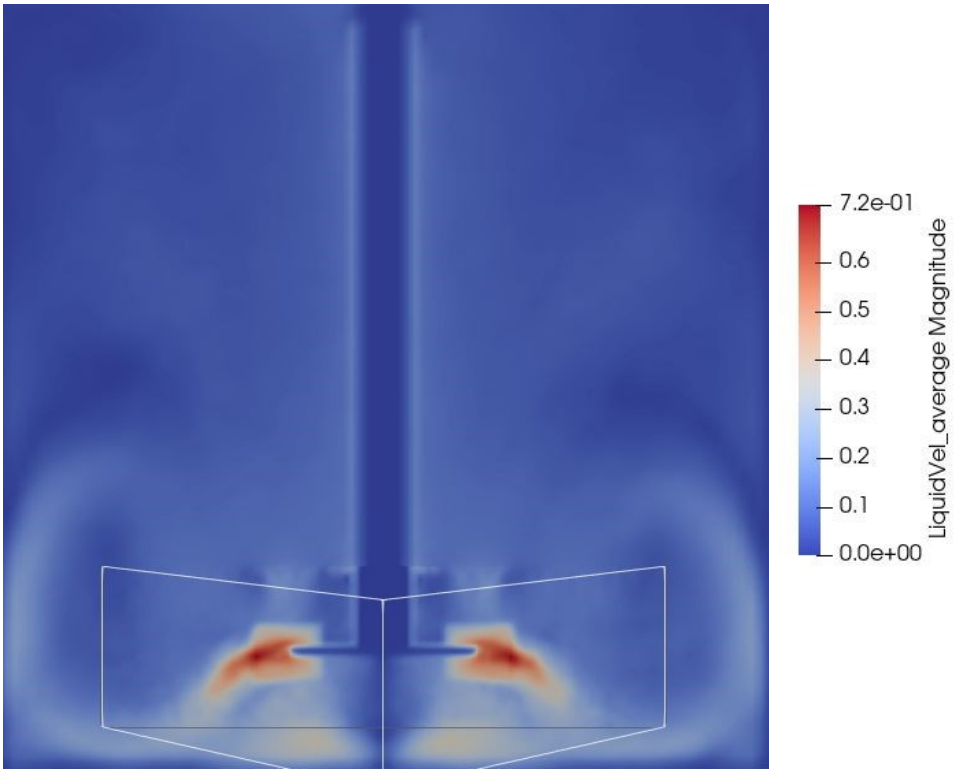


Fig. 5-27: Phase-averaged plot of the velocity field of the whole vertical section of the reactor obtained from the simulation for set-up 4

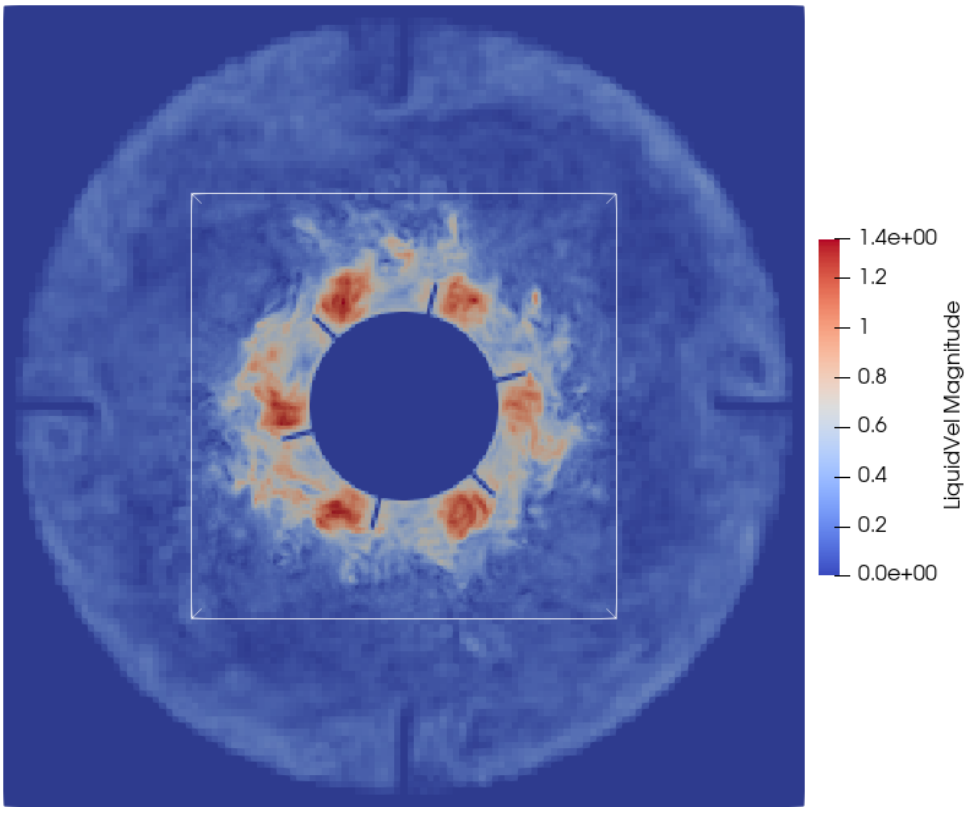


Fig. 5-28: Instantaneous velocity plot of the cross-section at the height of the stirrer

An effect which is observed in all stirred tank simulations is a nearly linear increase of mass inside the reactor. This is not a local phenomenon and could not be traced to a certain region of the vessel. It is not appearing in the unrefined reactor and not observed in pipe flow. However, the effect is decreased with an increase of blade thickness. Hence, it is assumed, that the problem correlates with the moving wall boundary condition. The grid boundary crossing stirrer shaft is not or not solely causing the problem, as this was tested.

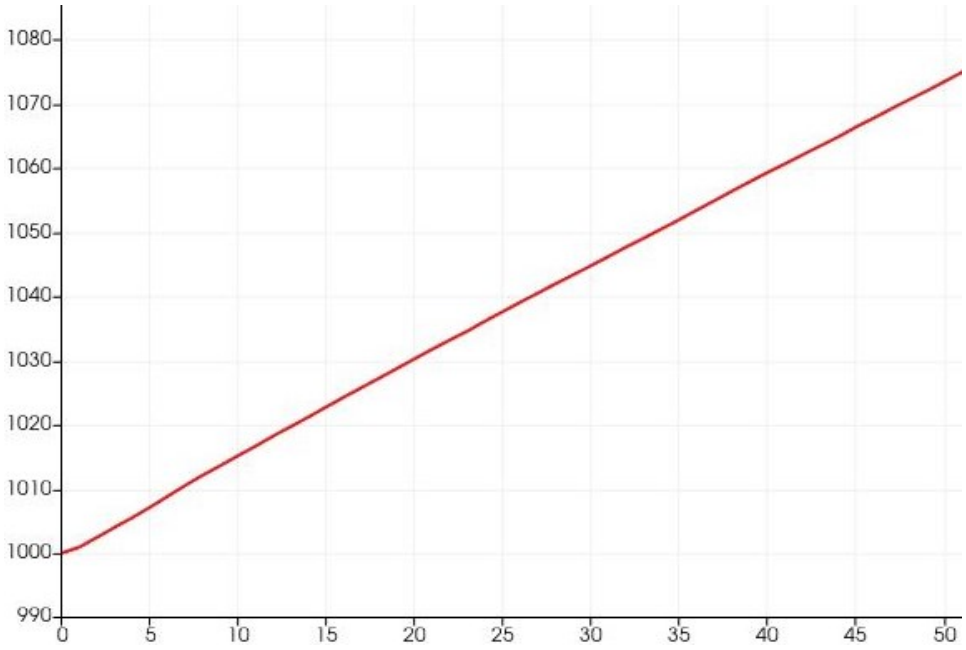


Fig. 5-29: Increase of density observed for 50 seconds of simulation for set-up 4 and 87x87x83 nodes

5.2.2 Set-up 5

The geometry of the tank with the dual-Rushton impeller is shown in Fig. 5-30 and all parameters are given in Tab. 5-8. Different positions of the impellers lead to different flow patterns in the double impeller configuration as well. If the distance separation between the impellers is large enough, each stirrer produces an independent flow pattern, as it would in a single-Rushton configuration. Rutherford et al. [31] found that to obtain a so called, parallel flow pattern, the distance of the lower stirrer from the bottom of the tank C_1 must be greater than $0.2D_T$ and the distance between the impellers C_2 must be greater than $0.385D_T$. By decreasing the off-bottom clearance C_1 of the lower stirrer to a value of $0.15D_T$ or below, its flow pattern is transformed to the single loop flow described in section 5.2.1. A merging flow occurs

for a distance of the lower stirrer to the vessel bottom C_1 of $0.17D_T$ or higher and a maximal distance between the impellers C_2 of $0.385D_T$. As the impellers are close to each other, their streams are influencing each other. Therefore, they move toward each other, meet vertically midway and form there one radially streaming stream. Where the stream reaches the reactor wall, it is separated again into two streams. One is moving upwards and forms a loop towards the upper impeller and one is moving downwards and forms a loop towards the lower impeller. [31]

The fine grid is positioned to solely refine the upper impeller to be able to compare directly the performance of grid refinement with the non-refined lower impeller.

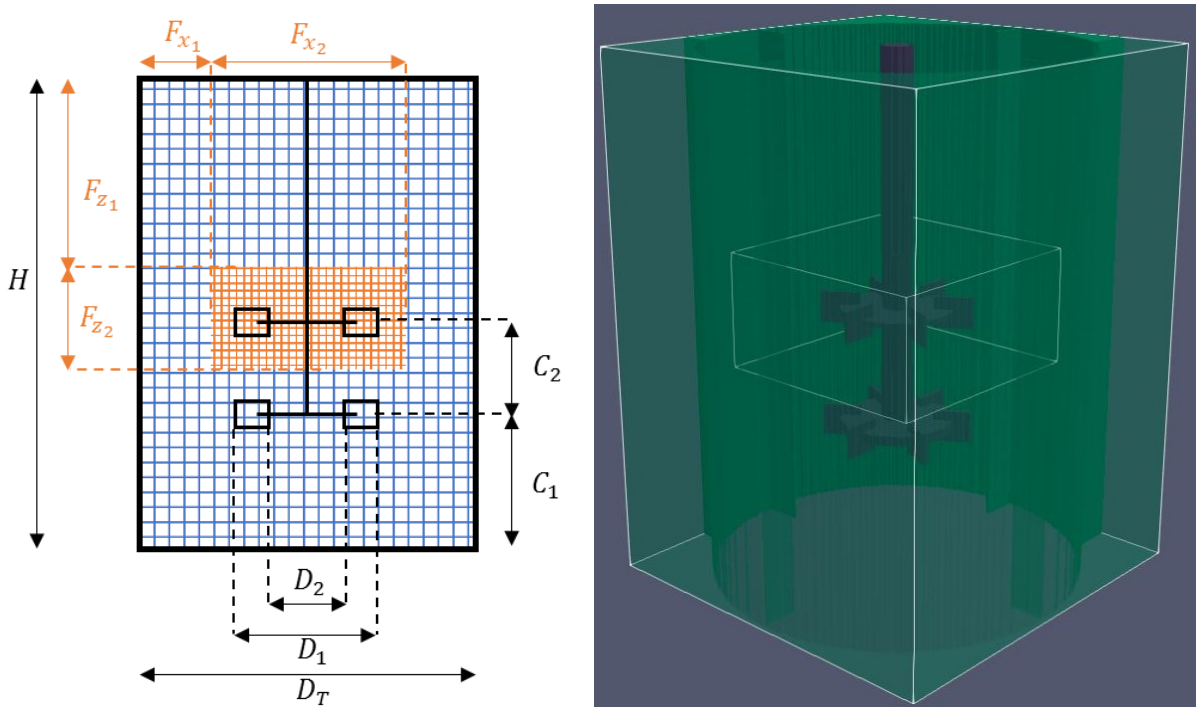


Fig. 5-30: Dimensional sketch of set-up 5 with the coarse grid in blue and the fine grid in orange on the left and 3D model of the simulation boundary on the right

The phase-averaged velocity field of the simulation is obtained as described in the previous section and a section of it is presented in Fig. 5-31. As reference, the phase-averaged velocity field from a PIV measurement obtained from Pan et al. [30] is taken and presented in Fig. 5-33. Further, the flow patterns obtained from Rutherford et al. [31] from a LDA measurement are

presented in Fig. 5-32. In their work, they used slightly different values for C_1 and C_2. However, they lie in the range to generate merging flow.

Comparing the simulation results in Fig. 5-31 with the reference results in Fig. 5-32 and Fig. 5-33 it can be seen that they match very well. Unfortunately, the resolution of Fig. 5-33 is quite low. However, the major parts of the flow are visible. The streams in Fig. 5-31, meet at half the distance between the impellers and afterwards form two loops. Futher, a secondary flow is produced at each blade which generates a small vertex between the blades. This is also shown in Fig. 5-33. In Fig. 5-32 it can be seen, that the upper main loop fills the whole region above the upper stirrer. For the simulation, as it can be seen in the appendix in Fig. 7-13, this is not the case. However, Rutherford et al. [31] used a tank which had the same height as diameter and in the simulation a height of $H = 1.4D_T$ was used. It is evident, that for a vessel of the same height the loops would fill the whole space above the stirrer.

The averaged velocity over the whole vertical cross-section of the reactor is plotted in Fig. 5-34. In the fluid region, next to where the stirrer shaft crosses the fine-coarse grid boundary, the velocity is increased, as it is the case for the singe-Rushton impeller in Fig. 5-27.

In Fig. 5-35 an instantaneous velocity plot of the cross-section at the height of the upper stirrer can be seen. Again, a considerable improvement of the resolution is observed. In the x and y-direction no artefacts of the grid coupling are visible as in Fig. 5-34. This is explained by the fact, that the fine-coarse grid boundary in x and y direction is completely located in the fluid region and is not crossing any solids.

Tab. 5-8: Parameters for the configuration of set-up 5

Number nodes in x-direction	$[-]$	123
Number nodes in y-direction	$[-]$	123
Number nodes in z-direction	$[-]$	167
Tank diameter D_T	$[m]$	0.476
Tank height H	$[m]$	0.666
Distance of lower impeller from bottom C_1	$[m]$	0.19
Distance between impellers C_2	$[m]$	0.15
Impeller speed	$[\frac{1}{min}]$	66
Impeller blade outer diameter D_1	$[m]$	0.19
Impeller blade inner diameter D_2	$[m]$	0.094
Number of blades	$[-]$	6
Blade height	$[m]$	0.038
Blade thickness	$[m]$	0.003
Disk diameter	$[m]$	0.142
Shaft diameter	$[m]$	0.035
Number of baffles equally spaced	$[-]$	4
Baffle width	$[m]$	0.048
Position of fine grid from top F_{z_1}	$[m]$	0.25
Height of fine grid F_{z_2}	$[m]$	0.146
Position of fine grid from domain boundary x=0 $F_{x_1} = F_{y_1}$	$[m]$	0.1
Width of fine grid F_{x_2}=F_{y_2}	$[m]$	0.276

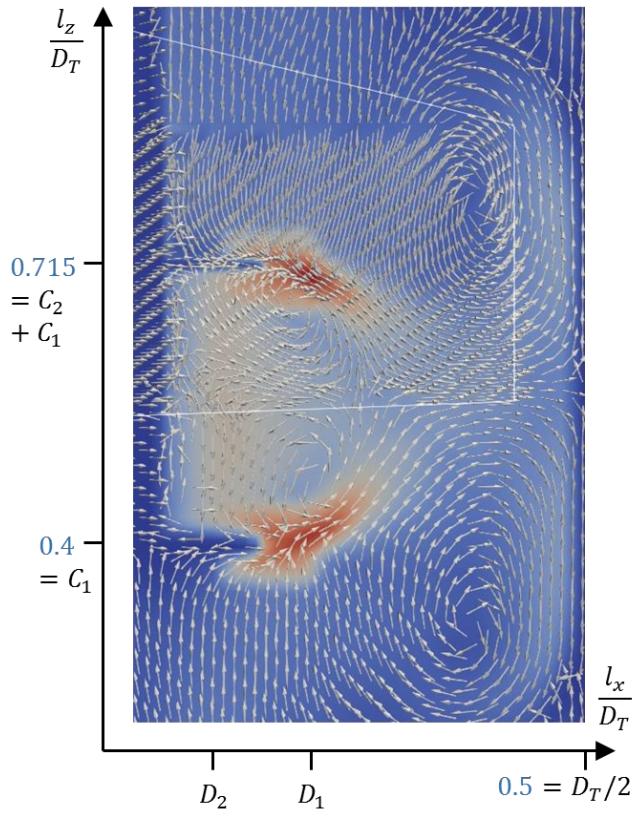


Fig. 5-31: ↑ Phase-averaged plot of the blade section of the velocity field obtained from the simulation with values in blue marking positions in units of normalized scale as used in references in Fig. 5-33 and Fig. 5-32

Fig. 5-32: → 360° phase-averaged plot of the velocity from a LDA measurement done by Rutherford et al. [31] at $\theta = 0°$ for $C_1 = 0.33D_T$ and $C_2 = 0.33D_T$

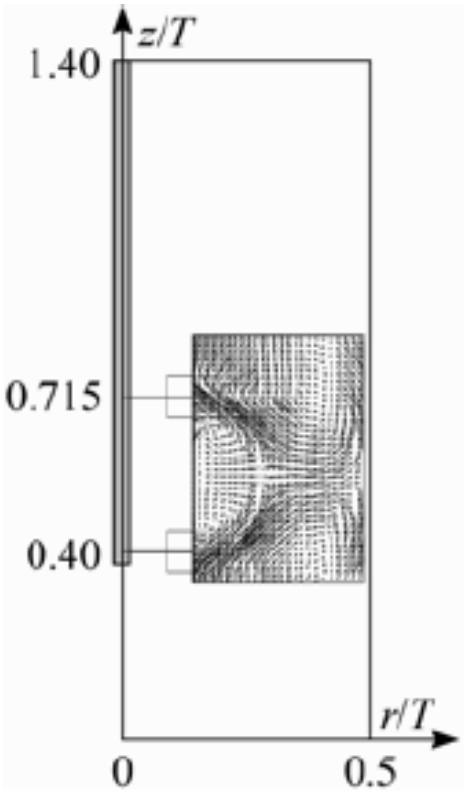
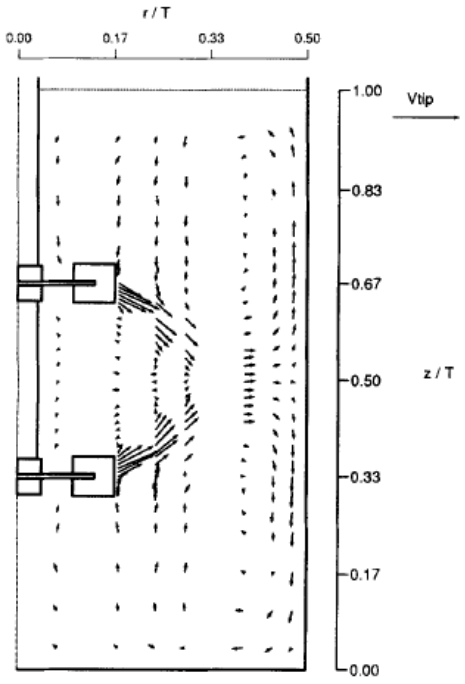


Fig. 5-33: ↑ : Phase-averaged plot of the velocity from a PIV measurement done by Pan et al. [30]



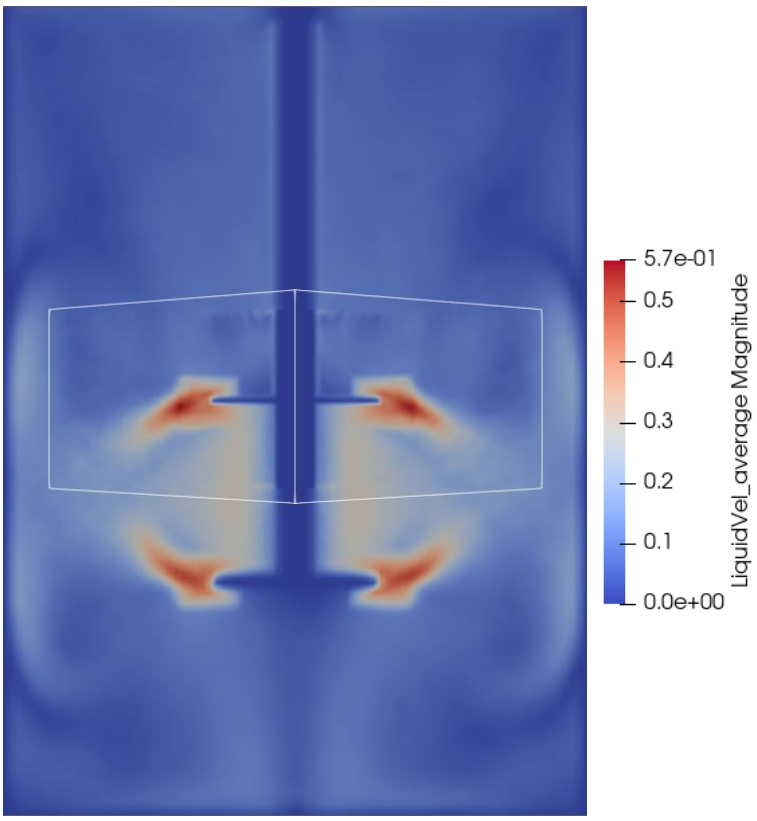


Fig. 5-34: Phase-averaged plot of the velocity field of the whole vertical section of the reactor obtained from the simulation for set-up 5

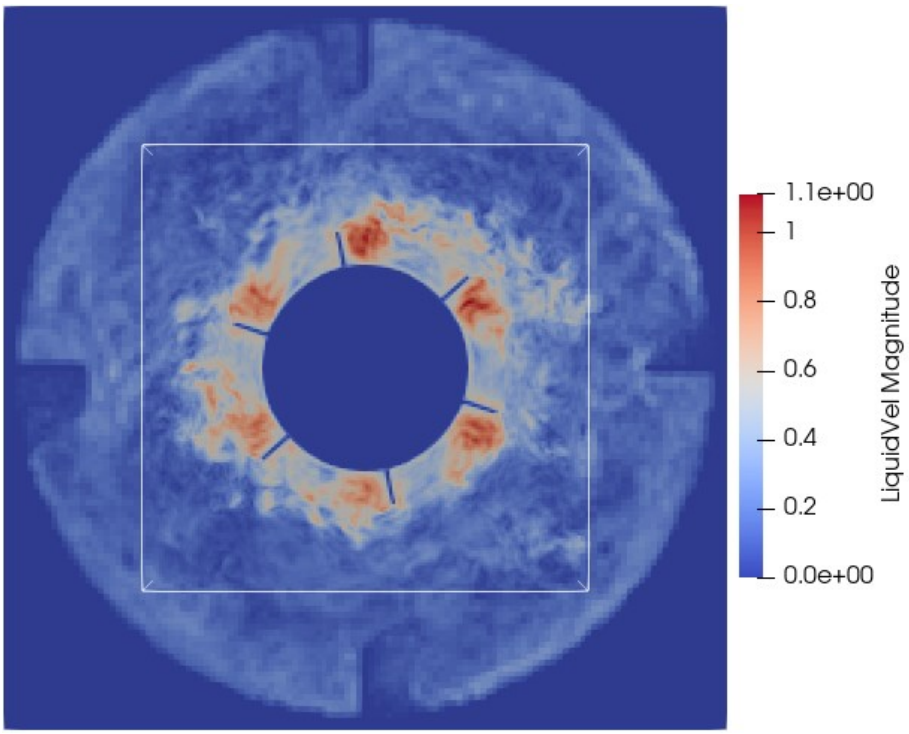


Fig. 5-35: Instantaneous velocity plot of the cross-section at the height of the upper stirrer

5.3 Conclusion

For laminar flow in a tube the grid refinement algorithm works excellent, if the fine grid is placed completely inside the fluid region. In case of a completely refined cross-section and therefore, an intersection of the solid-liquid boundary with the fine-coarse grid boundary the plotted velocity shows a peak in the section of coarse to fine boundary and a valley in the section of fine to coarse boundary. To find the origin of this behavior further testing is necessary. At the current state of development, it is assumed that the algorithm is not acting properly if solid nodes lie within the interface of the different grids. This is further observed in the case of stirred tank reactors where an increase of velocity can be seen in the region where the stirrer shaft crosses the grid boundaries. Besides that, the velocity values and vectors at the grid boundaries are coinciding.

Additionally, a rising density can be observed in stirred tank simulations. It is increasing nearly linear over time. It would be possible that the variations in the equilibrium distribution function presented in section 2.1.2, cause deviations of the algorithm which have been overlooked. As the problem is only observed in turbulent flows, the method for modeling the turbulence could be the origin of the problem. During implementation, it was noticed that the simulation is very sensitive to the scaling of the relaxation time.

If the two problems mentioned above are resolved, the grid refinement will be a helpful, accurate and reliable tool to decrease the computational costs or increase fluid resolution.

6 Summary

An existing lattice Boltzmann code was expanded by the functionality of local grid refinement. The user is able to define a region in which the resolution of the grid is increased by a factor of two. As a result, following the multi-grid approach an additional grid is generated. This has the advantage, that the coarse grid stays complete and can be used for other algorithms which are not integrated into the grid refinement function. Further, the approach is beneficial for the implementation on GPUs, as it is suitable for parallelization due to the regular, rectangular shape of the grids. A shifted cell arrangement is used to place the grids on top of each other. Every grid itself is seen as an individual simulation domain, solving the LB equation for every node in a parallel manner. At the interfaces between the grids, some particle distributions are unknown. These are obtained by a pre-collision grid coupling procedure. This means, that before the collision step is proceeded on every grid, the missing distribution functions are reconstructed from data of the adjacent grid. As the nodes of the fine and the coarse grid do not coincide, the velocity and density fields need to be interpolated. For the density a trilinear interpolation is sufficient. The velocity is interpolated by a compact second order accurate interpolation method. From the interpolated velocity and density, the equilibrium distribution function can be obtained. After trilinear interpolation of the non-equilibrium part of the distribution function and a subsequent rescaling, the missing distribution functions at the grid interface can be obtained.

The grid refinement was tested for a Hagen-Poiseuille flow with a Reynolds number of 10 and with a Reynolds number of 100. Three different placements of the fine grid were tested. To observe the behavior in turbulent flow, the grid refinement was tested in two different agitated tanks. It is observed, that if the solid-liquid boundary is intersecting the fine-coarse grid boundaries, the velocities deviate slightly from the expected values. Otherwise, the pressure drop and the reached maximum velocity match the results of the unrefined reference simulation. In the case of the agitated tanks, the mass of the simulated fluid increases. The reasons for that deviation still need to be identified. Besides that, the velocity field is reproduced in a satisfactory manner by using grid refinement.

6.1 Outlook

In the course of validating the grid refinement, as already mentioned in section 5.2.1, it was found that the algorithm in current state is not mass conservative. As this issue was observed in the stirred tank reactor cases only, it needs to be investigated if the origin lies in the modeling of turbulence, if it is provoked from the boundary condition of the stirrer blades or if it is due to the usage of an adapted, incompressible form of the lattice Boltzmann method.

One of the next steps is to facilitate more than one refinement step and investigate the possibility of a user settable refinement factor as this would lead to lower memory consumption than using several grids. Further, refinement at more than one position in the reactor could be implemented.

Also, further effort will be spent on providing grid refinement for the other functionalities of the code like the species transport or bubble movement.

7 Appendix

7.1 Isotropy Conditions for the Velocity Sets [3, p. 85]

$$\sum_i w_i = 1 \tag{7-1}$$

$$\sum_i c_{i\alpha} w_i = 0 \tag{7-2}$$

$$\sum_i c_{i\alpha} c_{i\beta} w_i = c_s^2 \delta_{\alpha\beta} \tag{7-3}$$

$$\sum_i c_{i\alpha} c_{i\beta} c_{i\gamma} w_i = 0 \tag{7-4}$$

$$\sum_i c_{i\alpha} c_{i\beta} c_{i\gamma} c_{i\delta} w_i = c_s^4 (\delta_{\alpha\beta}\delta_{\gamma\delta} + \delta_{\alpha\gamma}\delta_{\beta\delta} + \delta_{\alpha\delta}\delta_{\beta\gamma}) \tag{7-5}$$

$$\sum_i c_{i\alpha} c_{i\beta} c_{i\gamma} c_{i\delta} c_{i\varepsilon} w_i = 0 \tag{7-6}$$

7.2 Rescaling of the Non-Equilibrium Distribution Function

Krüger et al. [3, p. p. 118] derived the non-equilibrium distribution function from the Chapman-Enskog perturbation in its explicit form

$$f_i^{neq} = -\tau \underbrace{\frac{w_i \rho}{c_s^2}\left[Q_{i\alpha\beta}\partial_\beta u_\alpha - c_{i\alpha}\partial_\beta(u_\alpha u_\beta) + \frac{Q_{i\alpha\beta}}{2c_s^2} c_{i\gamma}\partial_\gamma(u_\alpha u_\beta) - \frac{Q_{i\alpha\beta}}{2c_s^2}\partial_\gamma(u_\alpha u_\beta u_\gamma)\right]}_{K} \tag{7-7}$$

with

$$Q_{i\alpha\beta} = c_{i\alpha}c_{i\beta} - c_s^2\delta_{\alpha\beta} \tag{7-8}$$

From section 3.2 it is known that the quantities u_α, $c_{i\alpha}$, c_s^2, $S_{\alpha\beta}$ and ρ are continuous over different grid resolutions in physical units. Further, also the first derivatives of velocity $\partial_\beta u_\alpha$ must be continuous in physical units. As a consequence, the term K in equation 7-7 is continuous over the grids as well. Therefore, the non-equilibrium distribution function can be rescaled in physical units like it is shown in equation 7-9.

$$\frac{f_{i,c}^{neq}}{\tau_c} = K = \frac{f_{i,f}^{neq}}{\tau_f} \tag{7-9}$$

From equation 3-23 the relation of the relaxation time in lattice units $\tilde{\tau}$ with the relaxation time in physical units τ is known. Inserting this into equation 7-9, the same formulation is obtained as given in the work from Dupuis and Chopard [24].

$$\frac{f_{i,c}^{neq}}{\tilde{\tau}_c \Delta t_c} = \frac{f_{i,f}^{neq}}{\tilde{\tau}_f \Delta t_f} \tag{7-10}$$

$$f_{i,c}^{neq} = \frac{2\tilde{\tau}_c}{\tilde{\tau}_f} f_{i,f}^{neq}; \quad f_{i,f}^{neq} = \frac{\tilde{\tau}_f}{2\tilde{\tau}_c} f_{i,c}^{neq} \tag{7-11}$$

From Valderhaug [8] it is known that $f_i^{neq} = h_i^{neq}$, and therefore, equation 7-12 can be derived.

$$h_{i,c}^{neq} = \frac{2\tilde{\tau}_c}{\tilde{\tau}_f} h_{i,f}^{neq}; \quad h_{i,f}^{neq} = \frac{\tilde{\tau}_f}{2\tilde{\tau}_c} h_{i,c}^{neq} \tag{7-12}$$

In lattice units, the equation can be derived like in the following lines.

$$\tilde{f}_i^{neq} = -\frac{\tilde{\tau} w_i \tilde{\rho}}{\tilde{c}_s^2}\left[\tilde{Q}_{i\alpha\beta}\partial_\beta \tilde{u}_\alpha - \tilde{c}_{i\alpha}\partial_\beta(\tilde{u}_\alpha\tilde{u}_\beta) + \frac{\tilde{Q}_{i\alpha\beta}}{2\tilde{c}_s^2}\tilde{c}_{i\gamma}\partial_\gamma(\tilde{u}_\alpha\tilde{u}_\beta) - \frac{\tilde{Q}_{i\alpha\beta}}{2\tilde{c}_s^2}\partial_\gamma(\tilde{u}_\alpha\tilde{u}_\beta\tilde{u}_\gamma)\right] \tag{7-13}$$

With equation 3-1 for the scaling between Δt and Δx, 3-14 for the rescaling of density and 3-22 for the relaxation factor all influences of the grid spacing are determined.

$$c_s^2 = \tilde{c}_{s,c}^2 \frac{\Delta x_c^2}{\Delta t_c^2} = \tilde{c}_{s,f}^2 \frac{\Delta x_f^2}{\Delta t_f^2} \rightarrow \tilde{c}_{s,c}^2 = \tilde{c}_{s,f}^2 = \tilde{c}_s^2 \tag{7-14}$$

$$c_{i\alpha} = \tilde{c}_{i\alpha,c} \frac{\Delta x_c}{\Delta t_c} = \tilde{c}_{i\alpha,f} \frac{\Delta x_f}{\Delta t_f} \rightarrow \tilde{c}_{i\alpha,c} = \tilde{c}_{i\alpha,f} = \tilde{c}_{i\alpha} \tag{7-15}$$

Knowing from 7-14 and 7-15 that the microscopic velocities and the speed of sound are continuous, 7-16 can be derived.

$$\tilde{Q}_{i\alpha\beta} = \tilde{c}_{i\alpha}\tilde{c}_{i\beta} - \tilde{c}_s^2\delta_{\alpha\beta} \rightarrow \tilde{Q}_{i\alpha\beta,f} = \tilde{Q}_{i\alpha\beta,c} = \tilde{Q}_{i\alpha\beta} \tag{7-16}$$

By obeying equation 3-1, the derivatives of the velocity turn out to be dependent on the temporal resolution.

$$\partial_\beta u_\alpha = (\partial_\beta \tilde{u}_\alpha)_f \frac{\Delta x_f}{\Delta t_f \Delta x_f} = (\partial_\beta \tilde{u}_\alpha)_c \frac{\Delta x_c}{\Delta t_c \Delta x_c} \rightarrow (\partial_\beta \tilde{u}_\alpha)_f \frac{1}{\Delta t_f} = (\partial_\beta \tilde{u}_\alpha)_c \frac{1}{\Delta t_c} \tag{7-17}$$

$$(\partial_\beta \tilde{u}_\alpha)_c = \frac{\Delta t_c}{\Delta t_f}(\partial_\beta \tilde{u}_\alpha)_f \rightarrow (\partial_\beta \tilde{u}_\alpha)_c = 2\,(\partial_\beta \tilde{u}_\alpha)_f \tag{7-18}$$

$$\partial_\beta(u_\alpha u_\beta) = \left(\partial_\beta(\tilde{u}_\alpha\tilde{u}_\beta)\right)_f \frac{\Delta x_f^2}{\Delta t_f^2 \Delta x_f} = \left(\partial_\beta(\tilde{u}_\alpha\tilde{u}_\beta)\right)_c \frac{\Delta x_c^2}{\Delta t_c^2 \Delta x_c} \tag{7-19}$$

$$\left(\partial_\beta(\tilde{u}_\alpha\tilde{u}_\beta)\right)_f \frac{1}{\Delta t_f} = \left(\partial_\beta(\tilde{u}_\alpha\tilde{u}_\beta)\right)_c \frac{1}{\Delta t_c} \tag{7-20}$$

$$\left(\partial_\beta(\tilde{u}_\alpha\tilde{u}_\beta)\right)_c = 2\,\left(\partial_\beta(\tilde{u}_\alpha\tilde{u}_\beta)\right)_f \tag{7-21}$$

$$\partial_\gamma(u_\alpha u_\beta u_\gamma) = \left(\partial_\gamma(\tilde{u}_\alpha\tilde{u}_\beta\tilde{u}_\gamma)\right)_f \frac{\Delta x_f^3}{\Delta t_f^3 \Delta x_f} = \left(\partial_\gamma(\tilde{u}_\alpha\tilde{u}_\beta\tilde{u}_\gamma)\right)_c \frac{\Delta x_c^3}{\Delta t_c^3 \Delta x_c} \tag{7-22}$$

$$\left(\partial_\gamma(\tilde{u}_\alpha\tilde{u}_\beta\tilde{u}_\gamma)\right)_f \frac{1}{\Delta t_f} = \left(\partial_\gamma(\tilde{u}_\alpha\tilde{u}_\beta\tilde{u}_\gamma)\right)_c \frac{1}{\Delta t_c} \tag{7-23}$$

$$\left(\partial_\gamma\left(\tilde{u}_\alpha\tilde{u}_\beta\tilde{u}_\gamma\right)\right)_c = 2\ \left(\partial_\gamma\left(\tilde{u}_\alpha\tilde{u}_\beta\tilde{u}_\gamma\right)\right)_f \tag{7-24}$$

Inserting equations 7-14, 7-15, 7-16, 7-18, 7-21 and 7-24 into the non-equilibrium distributions function in equation 7-13, the equations below are obtained.

$$\tilde{f}_{i,c}^{neq} = -2\widetilde{\tau_c}\underbrace{\frac{w_i\tilde{\rho}}{\tilde{c}_s^2}\left[\tilde{Q}_{i\alpha\beta}\partial_\beta\tilde{u}_{\alpha,f} - \tilde{c}_{i\alpha}\partial_\beta\left(\tilde{u}_\alpha\tilde{u}_\beta\right)_f + \frac{\tilde{Q}_{i\alpha\beta}}{2\tilde{c}_s^2}\tilde{c}_{i\gamma}\partial_\gamma\left(\tilde{u}_\alpha\tilde{u}_\beta\right)_f - \frac{\tilde{Q}_{i\alpha\beta}}{2\tilde{c}_s^2}\partial_\gamma\left(\tilde{u}_\alpha\tilde{u}_\beta\tilde{u}_\gamma\right)_f\right]}_{\widetilde{K}} \tag{7-25}$$

$$\tilde{f}_{i,f}^{neq} = -\widetilde{\tau_f}\underbrace{\frac{w_i\tilde{\rho}}{\tilde{c}_s^2}\left[\tilde{Q}_{i\alpha\beta}\partial_\beta\tilde{u}_{\alpha,f} - \tilde{c}_{i\alpha}\partial_\beta\left(\tilde{u}_\alpha\tilde{u}_\beta\right)_f + \frac{\tilde{Q}_{i\alpha\beta}}{2\tilde{c}_s^2}\tilde{c}_{i\gamma}\partial_\gamma\left(\tilde{u}_\alpha\tilde{u}_\beta\right)_f - \frac{\tilde{Q}_{i\alpha\beta}}{2\tilde{c}_s^2}\partial_\gamma\left(\tilde{u}_\alpha\tilde{u}_\beta\tilde{u}_\gamma\right)_f\right]}_{\widetilde{K}} \tag{7-26}$$

From that, the relationship between the non-equilibrium distribution functions in lattice units is obtained.

$$\frac{\tilde{f}_{i,c}^{neq}}{-2\widetilde{\tau_c}} = \widetilde{K} = \frac{\tilde{f}_{i,f}^{neq}}{-\widetilde{\tau_f}} \tag{7-27}$$

$$\tilde{f}_{i,c}^{neq} = \frac{2\widetilde{\tau_c}}{\widetilde{\tau_f}}\tilde{f}_{i,f}^{neq};\quad \tilde{f}_{i,f}^{neq} = \frac{\widetilde{\tau_f}}{2\widetilde{\tau_c}}\tilde{f}_{i,c}^{neq} \tag{7-28}$$

From Valderhaug [8] it is known that $f_i^{neq} = h_i^{neq}$, and therefore, equation 7-29 can be derived.

$$\tilde{h}_{i,c}^{neq} = \frac{2\widetilde{\tau_c}}{\widetilde{\tau_f}}\tilde{h}_{i,f}^{neq};\quad \tilde{h}_{i,f}^{neq} = \frac{\widetilde{\tau_f}}{2\widetilde{\tau_c}}\tilde{h}_{i,c}^{neq} \tag{7-29}$$

7.3 The Coefficients for the Velocity Interpolation

Cube A:

$$a_0 = u_x(0,0,0) \tag{7-30}$$

$$b_0 = u_y(0,0,0) \tag{7-31}$$

$$c_0 = u_z(0,0,0) \tag{7-32}$$

$$a_{xx} = -\frac{3}{4}\frac{1}{\tau\rho_0}\left(\frac{\partial\Pi_{xx}^{neq}}{\partial x}\right) \tag{7-33}$$

$$a_{xy} = -\frac{3}{2}\frac{1}{\tau\rho_0}\left(\frac{\partial\Pi_{xx}^{neq}}{\partial y}\right) \tag{7-34}$$

$$a_{xz} = -\frac{3}{2}\frac{1}{\tau\rho_0}\left(\frac{\partial\Pi_{xx}^{neq}}{\partial z}\right) \tag{7-35}$$

$$b_{xy} = -\frac{3}{2}\frac{1}{\tau\rho_0}\left(\frac{\partial \Pi_{yy}^{neq}}{\partial x}\right) \quad (7\text{-}36)$$

$$b_{yy} = -\frac{3}{4}\frac{1}{\tau\rho_0}\left(\frac{\partial \Pi_{yy}^{neq}}{\partial y}\right) \quad (7\text{-}37)$$

$$b_{yz} = -\frac{3}{2}\frac{1}{\tau\rho_0}\left(\frac{\partial \Pi_{yy}^{neq}}{\partial z}\right) \quad (7\text{-}38)$$

$$c_{xz} = -\frac{3}{2}\frac{1}{\tau\rho_0}\left(\frac{\partial \Pi_{zz}^{neq}}{\partial x}\right) \quad (7\text{-}39)$$

$$c_{yz} = -\frac{3}{2}\frac{1}{\tau\rho_0}\left(\frac{\partial \Pi_{zz}^{neq}}{\partial y}\right) \quad (7\text{-}40)$$

$$c_{zz} = -\frac{3}{4}\frac{1}{\tau\rho_0}\left(\frac{\partial \Pi_{zz}^{neq}}{\partial z}\right) \quad (7\text{-}41)$$

$$b_{xx} = -\frac{3}{2}\frac{1}{\tau\rho_0}\left(\frac{\partial \Pi_{xy}^{neq}}{\partial x}\right) + \frac{3}{4}\frac{1}{\tau\rho_0}\left(\frac{\partial \Pi_{xx}^{neq}}{\partial y}\right) \quad (7\text{-}42)$$

$$b_{zz} = -\frac{3}{2}\frac{1}{\tau\rho_0}\left(\frac{\partial \Pi_{yz}^{neq}}{\partial z}\right) + \frac{3}{4}\frac{1}{\tau\rho_0}\left(\frac{\partial \Pi_{zz}^{neq}}{\partial y}\right) \quad (7\text{-}43)$$

$$a_{yy} = -\frac{3}{2}\frac{1}{\tau\rho_0}\left(\frac{\partial \Pi_{xy}^{neq}}{\partial y}\right) + \frac{3}{4}\frac{1}{\tau\rho_0}\left(\frac{\partial \Pi_{yy}^{neq}}{\partial x}\right) \quad (7\text{-}44)$$

$$a_{zz} = -\frac{3}{2}\frac{1}{\tau\rho_0}\left(\frac{\partial \Pi_{xz}^{neq}}{\partial z}\right) + \frac{3}{4}\frac{1}{\tau\rho_0}\left(\frac{\partial \Pi_{zz}^{neq}}{\partial x}\right) \quad (7\text{-}45)$$

$$c_{xx} = -\frac{3}{2}\frac{1}{\tau\rho_0}\left(\frac{\partial \Pi_{xz}^{neq}}{\partial x}\right) + \frac{3}{4}\frac{1}{\tau\rho_0}\left(\frac{\partial \Pi_{xx}^{neq}}{\partial z}\right) \quad (7\text{-}46)$$

$$c_{yy} = -\frac{3}{2}\frac{1}{\tau\rho_0}\left(\frac{\partial \Pi_{yz}^{neq}}{\partial y}\right) + \frac{3}{4}\frac{1}{\tau\rho_0}\left(\frac{\partial \Pi_{yy}^{neq}}{\partial z}\right) \quad (7\text{-}47)$$

$$a_{yz} = -\frac{3}{2}\frac{1}{\tau\rho_0}\left(-\frac{\partial \Pi_{yz}^{neq}}{\partial x} + \frac{\partial \Pi_{xz}^{neq}}{\partial y} + \frac{\partial \Pi_{xy}^{neq}}{\partial z}\right) \quad (7\text{-}48)$$

$$b_{xz} = -\frac{3}{2}\frac{1}{\tau\rho_0}\left(\frac{\partial \Pi_{yz}^{neq}}{\partial x} - \frac{\partial \Pi_{xz}^{neq}}{\partial y} + \frac{\partial \Pi_{xy}^{neq}}{\partial z}\right) \quad (7\text{-}49)$$

$$c_{xy} = -\frac{3}{2}\frac{1}{\tau\rho_0}\left(\frac{\partial \Pi_{yz}^{neq}}{\partial x} + \frac{\partial \Pi_{xz}^{neq}}{\partial y} - \frac{\partial \Pi_{xy}^{neq}}{\partial z}\right) \quad (7\text{-}50)$$

$$a_x = \frac{1}{2}\big(u_x(1,1,0) + u_x(1,0,1) - u_x(0,1,1) - u_x(0,0,0)\big) - a_{xx} + \frac{1}{2}\big(-a_{xy} - a_{xz} + a_{yz}\big) \quad (7\text{-}51)$$

$$a_y = \frac{1}{2}\big(u_x(1,1,0) - u_x(1,0,1) + u_x(0,1,1) + u_x(0,0,0)\big) - a_{yy} + \frac{1}{2}\big(-a_{xy} + a_{xz} - a_{yz}\big) \quad (7\text{-}52)$$

$$a_z = \frac{1}{2}\big(-u_x(1,1,0) + u_x(1,0,1) + u_x(0,1,1) - u_x(0,0,0)\big) - a_{zz} + \frac{1}{2}\big(a_{xy} - a_{xz} - a_{yz}\big) \quad (7\text{-}53)$$

$$b_x = \frac{1}{2}\Big(u_y(1,1,0) + u_y(1,0,1) - u_y(0,1,1) - u_y(0,0,0)\Big) - b_{xx} + \frac{1}{2}\big(-b_{xy} - b_{xz} + b_{yz}\big) \quad (7\text{-}54)$$

$$b_y = \frac{1}{2}\Big(u_y(1,1,0) - u_y(1,0,1) + u_y(0,1,1) - u_y(0,0,0)\Big) - b_{yy} + \frac{1}{2}\big(-b_{xy} + b_{xz} - b_{yz}\big) \quad (7\text{-}55)$$

$$b_z = \frac{1}{2}\Big(-u_y(1,1,0) + u_y(1,0,1) + u_y(0,1,1) - u_y(0,0,0)\Big) - b_{zz} + \frac{1}{2}\big(b_{xy} - b_{xz} - b_{yz}\big) \quad (7\text{-}56)$$

$$c_x = \frac{1}{2}\big(u_z(1,1,0) + u_z(1,0,1) - u_z(0,1,1) - u_z(0,0,0)\big) - c_{xx} + \frac{1}{2}\big(-c_{xy} - c_{xz} + c_{yz}\big) \quad (7\text{-}57)$$

$$c_y = \frac{1}{2}\big(u_z(1,1,0) - u_z(1,0,1) + u_z(0,1,1) - u_z(0,0,0)\big) - c_{yy} + \frac{1}{2}\big(-c_{xy} + c_{xz} - c_{yz}\big) \quad (7\text{-}58)$$

$$c_z = \frac{1}{2}\big(-u_z(1,1,0) + u_z(1,0,1) + u_z(0,1,1) - u_z(0,0,0)\big) - c_{zz} + \frac{1}{2}\big(c_{xy} - c_{xz} - c_{yz}\big) \quad (7\text{-}59)$$

Cube B:

$$a_{xx} = -\frac{3}{4}\frac{1}{\tau\rho_0}\left(\frac{\partial \Pi_{xx}^{neq}}{\partial x}\right) \quad (7\text{-}60)$$

$$a_{xy} = -\frac{3}{2}\frac{1}{\tau\rho_0}\left(\frac{\partial \Pi_{xx}^{neq}}{\partial y}\right) \quad (7\text{-}61)$$

$$a_{xz} = -\frac{3}{2}\frac{1}{\tau\rho_0}\left(\frac{\partial \Pi_{xx}^{neq}}{\partial z}\right) \quad (7\text{-}62)$$

$$b_{xy} = -\frac{3}{2}\frac{1}{\tau\rho_0}\left(\frac{\partial \Pi_{yy}^{neq}}{\partial x}\right) \quad (7\text{-}63)$$

$$b_{yy} = -\frac{3}{4}\frac{1}{\tau\rho_0}\left(\frac{\partial \Pi_{yy}^{neq}}{\partial y}\right) \quad (7\text{-}64)$$

$$b_{yz} = -\frac{3}{2}\frac{1}{\tau\rho_0}\left(\frac{\partial \Pi_{yy}^{neq}}{\partial z}\right) \quad (7\text{-}65)$$

$$c_{xz} = -\frac{3}{2}\frac{1}{\tau\rho_0}\left(\frac{\partial \Pi_{zz}^{neq}}{\partial x}\right) \quad (7\text{-}66)$$

$$c_{yz} = -\frac{3}{2}\frac{1}{\tau\rho_0}\left(\frac{\partial \Pi_{zz}^{neq}}{\partial y}\right) \quad (7\text{-}67)$$

$$c_{zz} = -\frac{3}{4}\frac{1}{\tau\rho_0}\left(\frac{\partial \Pi_{zz}^{neq}}{\partial z}\right) \quad (7\text{-}68)$$

$$b_{xx} = -\frac{3}{2}\frac{1}{\tau\rho_0}\left(\frac{\partial \Pi_{xy}^{neq}}{\partial x}\right) + \frac{3}{4}\frac{1}{\tau\rho_0}\left(\frac{\partial \Pi_{xx}^{neq}}{\partial y}\right) \quad (7\text{-}69)$$

$$b_{zz} = -\frac{3}{2}\frac{1}{\tau\rho_0}\left(\frac{\partial \Pi_{yz}^{neq}}{\partial z}\right) + \frac{3}{4}\frac{1}{\tau\rho_0}\left(\frac{\partial \Pi_{zz}^{neq}}{\partial y}\right) \quad (7\text{-}70)$$

$$a_{yy} = -\frac{3}{2}\frac{1}{\tau\rho_0}\left(\frac{\partial \Pi_{xy}^{neq}}{\partial y}\right) + \frac{3}{4}\frac{1}{\tau\rho_0}\left(\frac{\partial \Pi_{yy}^{neq}}{\partial x}\right) \quad (7\text{-}71)$$

$$a_{zz} = -\frac{3}{2}\frac{1}{\tau\rho_0}\left(\frac{\partial \Pi_{xz}^{neq}}{\partial z}\right) + \frac{3}{4}\frac{1}{\tau\rho_0}\left(\frac{\partial \Pi_{zz}^{neq}}{\partial x}\right) \quad (7\text{-}72)$$

$$c_{xx} = -\frac{3}{2}\frac{1}{\tau\rho_0}\left(\frac{\partial \Pi_{xz}^{neq}}{\partial x}\right) + \frac{3}{4}\frac{1}{\tau\rho_0}\left(\frac{\partial \Pi_{xx}^{neq}}{\partial z}\right) \quad (7\text{-}73)$$

$$c_{yy} = -\frac{3}{2}\frac{1}{\tau\rho_0}\left(\frac{\partial \Pi_{yz}^{neq}}{\partial y}\right) + \frac{3}{4}\frac{1}{\tau\rho_0}\left(\frac{\partial \Pi_{yy}^{neq}}{\partial z}\right) \quad (7\text{-}74)$$

$$a_{yz} = -\frac{3}{2}\frac{1}{\tau\rho_0}\left(-\frac{\partial \Pi_{yz}^{neq}}{\partial x} + \frac{\partial \Pi_{xz}^{neq}}{\partial y} + \frac{\partial \Pi_{xy}^{neq}}{\partial z}\right) \quad (7\text{-}75)$$

$$b_{xz} = -\frac{3}{2}\frac{1}{\tau\rho_0}\left(\frac{\partial \Pi_{yz}^{neq}}{\partial x} - \frac{\partial \Pi_{xz}^{neq}}{\partial y} + \frac{\partial \Pi_{xy}^{neq}}{\partial z}\right) \quad (7\text{-}76)$$

$$c_{xy} = -\frac{3}{2}\frac{1}{\tau\rho_0}\left(\frac{\partial \Pi_{yz}^{neq}}{\partial x} + \frac{\partial \Pi_{xz}^{neq}}{\partial y} - \frac{\partial \Pi_{xy}^{neq}}{\partial z}\right) \quad (7\text{-}77)$$

$$a_0 = \frac{1}{2}\left(-u_x(1,1,1) + u_x(1,0,0) + u_x(0,1,0) + u_x(0,0,1)\right) + \frac{1}{2}\left(a_{xy} + a_{xz} + a_{yz}\right) \quad (7\text{-}78)$$

$$a_x = \frac{1}{2}\left(u_x(1,1,1) + u_x(1,0,0) - u_x(0,1,0) - u_x(0,0,1)\right) - a_{xx} - \frac{1}{2}\left(a_{xy} + a_{xz} + a_{yz}\right) \quad (7\text{-}79)$$

$$a_y = \frac{1}{2}\left(u_x(1,1,1) - u_x(1,0,0) + u_x(0,1,0) - u_x(0,0,1)\right) - a_{yy} - \frac{1}{2}\left(a_{xy} + a_{xz} + a_{yz}\right) \quad (7\text{-}80)$$

$$a_z = \frac{1}{2}\left(u_x(1,1,1) - u_x(1,0,0) - u_x(0,1,0) + u_x(0,0,1)\right) - a_{zz} - \frac{1}{2}\left(a_{xy} + a_{xz} + a_{yz}\right) \quad (7\text{-}81)$$

$$b_0 = \frac{1}{2}\left(-u_y(1,1,1) + u_y(1,0,0) + u_y(0,1,0) + u_y(0,0,1)\right) + \frac{1}{2}\left(b_{xy} + b_{xz} + b_{yz}\right) \quad (7\text{-}82)$$

$$b_x = \frac{1}{2}\left(u_y(1,1,1) + u_y(1,0,0) - u_y(0,1,0) - u_y(0,0,1)\right) - b_{xx} - \frac{1}{2}\left(b_{xy} + b_{xz} + b_{yz}\right) \quad (7\text{-}83)$$

$$b_y = \frac{1}{2}\left(u_y(1,1,1) - u_y(1,0,0) + u_y(0,1,0) - u_y(0,0,1)\right) - b_{yy} - \frac{1}{2}\left(b_{xy} + b_{xz} + b_{yz}\right) \quad (7\text{-}84)$$

$$b_z = \frac{1}{2}\left(u_y(1,1,1) - u_y(1,0,0) - u_y(0,1,0) + u_y(0,0,1)\right) - b_{zz} - \frac{1}{2}\left(b_{xy} + b_{xz} + b_{yz}\right) \quad (7\text{-}85)$$

$$c_0 = \frac{1}{2}\left(-u_z(1,1,1) + u_z(1,0,0) + u_z(0,1,0) + u_z(0,0,1)\right) + \frac{1}{2}\left(c_{xy} + c_{xz} + c_{yz}\right) \quad (7\text{-}86)$$

$$c_x = \frac{1}{2}\left(u_z(1,1,1) + u_z(1,0,0) - u_z(0,1,0) - u_z(0,0,1)\right) - c_{xx} - \frac{1}{2}\left(c_{xy} + c_{xz} + c_{yz}\right) \quad (7\text{-}87)$$

$$c_y = \frac{1}{2}\left(u_z(1,1,1) - u_z(1,0,0) + u_z(0,1,0) - u_z(0,0,1)\right) - c_{yy} - \frac{1}{2}\left(c_{xy} + c_{xz} + c_{yz}\right) \quad (7\text{-}88)$$

$$c_z = \frac{1}{2}\left(u_z(1,1,1) - u_z(1,0,0) - u_z(0,1,0) + u_z(0,0,1)\right) - c_{zz} - \frac{1}{2}\left(c_{xy} + c_{xz} + c_{yz}\right) \quad (7\text{-}89)$$

7.4 Results of Reference Simulations without Grid Refinement

Tab. 7-1: Comparison of pressure drop for a Reynolds number of 10 without grid refinement

$\Delta p_{calc}[Pa]$	$\Delta p_{simu}[Pa]$	$\varepsilon[\%]$
$6.053 \cdot 10^{-4}$	$6.084 \cdot 10^{-4}$	0.511

Tab. 7-2: Comparison of pressure drop for a Reynolds number of 100 without grid refinement

$\Delta p_{calc}[Pa]$	$\Delta p_{simu}[Pa]$	$\varepsilon[\%]$
$4.613 \cdot 10^{-3}$	$4.647 \cdot 10^{-3}$	0.719

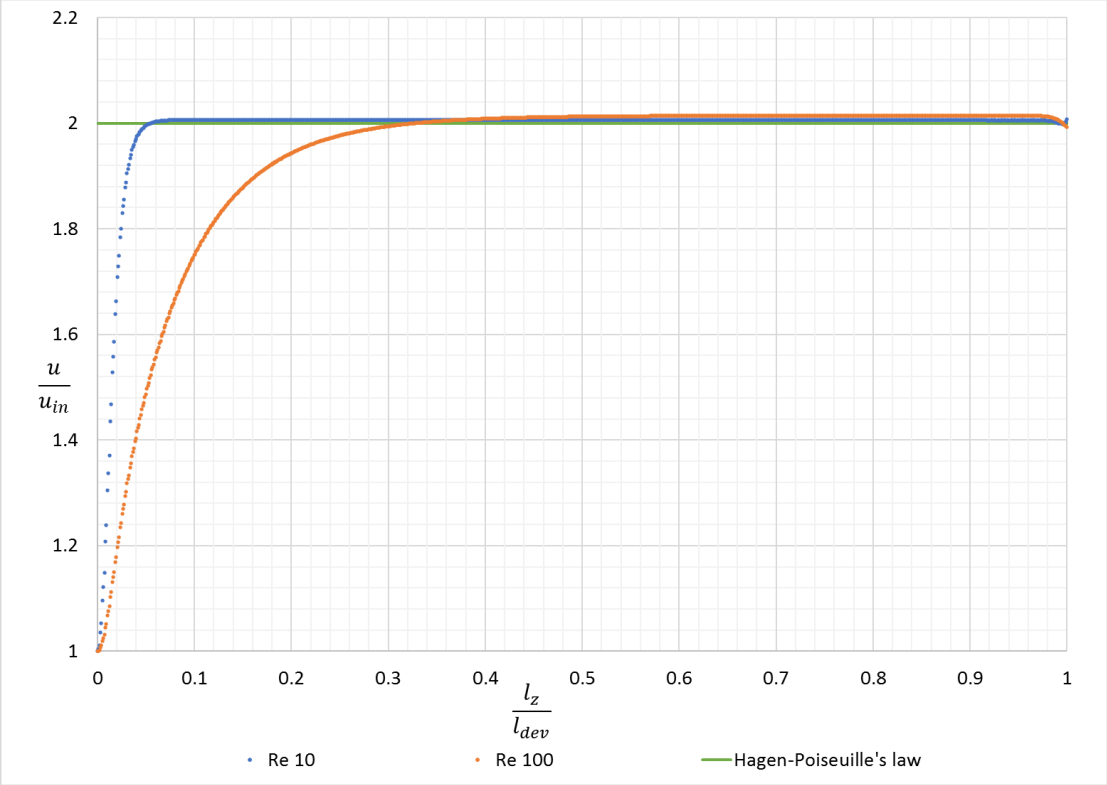


Fig. 7-1: Velocity at center line plotted over the length of the tube from simulations without grid refinement

7.5 Further Simulation Results of Poiseuille Flow in a Cylindrical Tube

7.5.1 Longitudinal Velocity and Pressure Fields of set-up 1

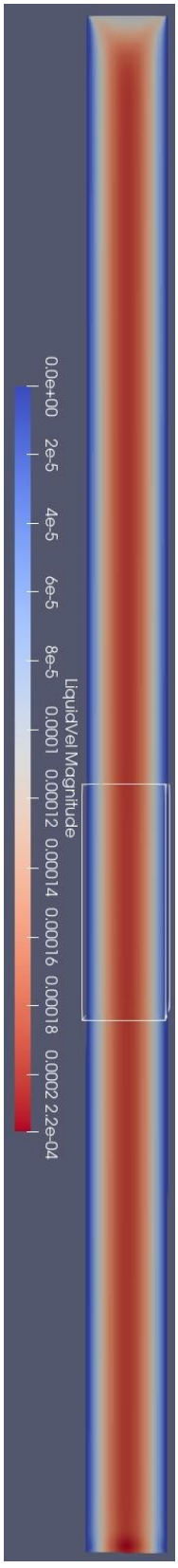


Fig. 7-2: Longitudinal velocity field for set-up 1 and a Reynolds number of 10

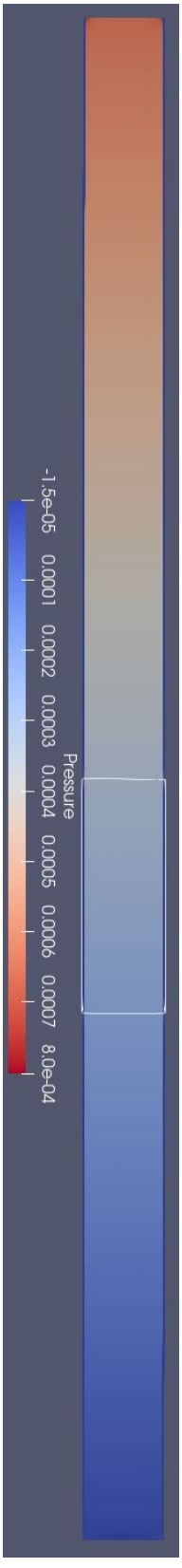


Fig. 7-3: Longitudinal pressure field for set-up 1 and a Reynolds number of 10

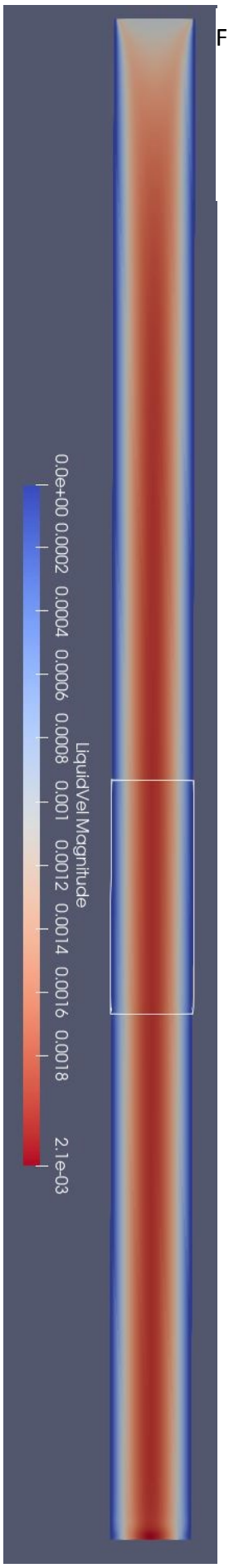


Fig. 7-4: Longitudinal velocity field for set-up 1 and a Reynolds number of 100

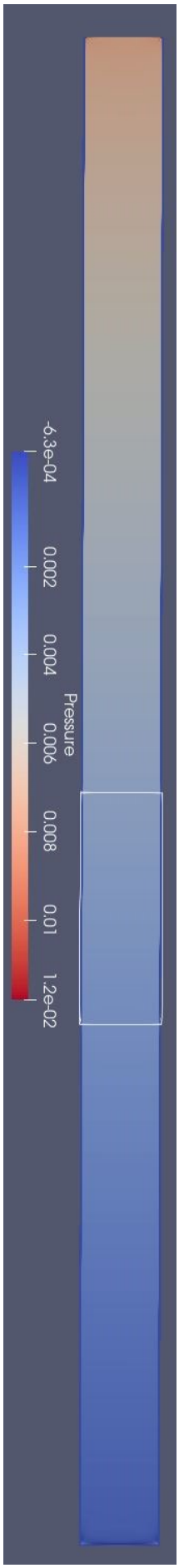


Fig. 7-5: Longitudinal pressure field for set-up 1 and a Reynolds number of 100

7.5.2 Longitudinal Velocity and Pressure Fields of Set-up 2

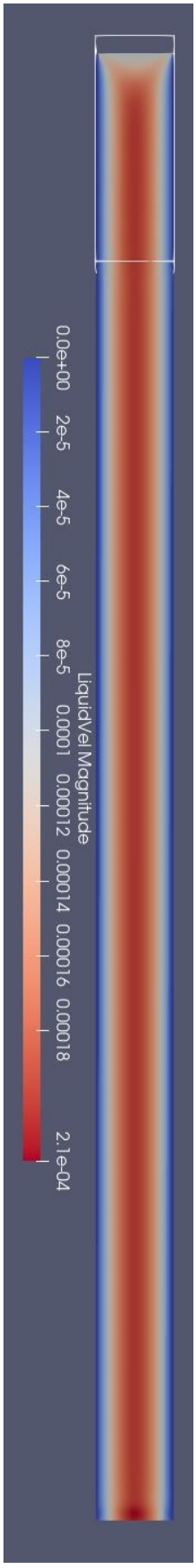


Fig. 7-6: Longitudinal velocity field for set-up 2 and a Reynolds number of 10

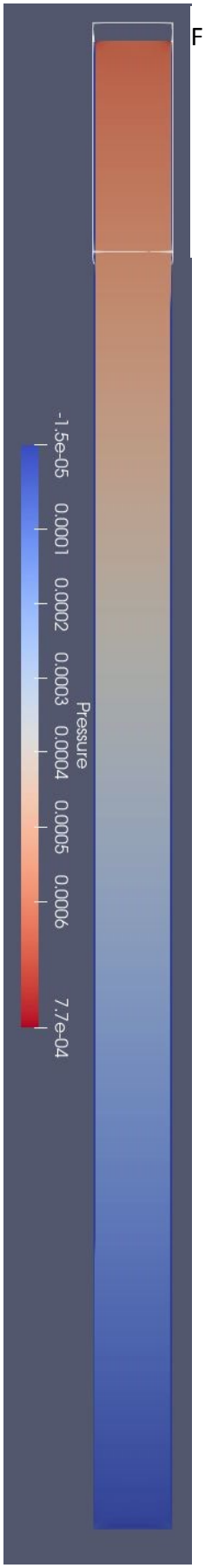


Fig. 7-7: Longitudinal pressure field for set-up 2 and a Reynolds number of 10

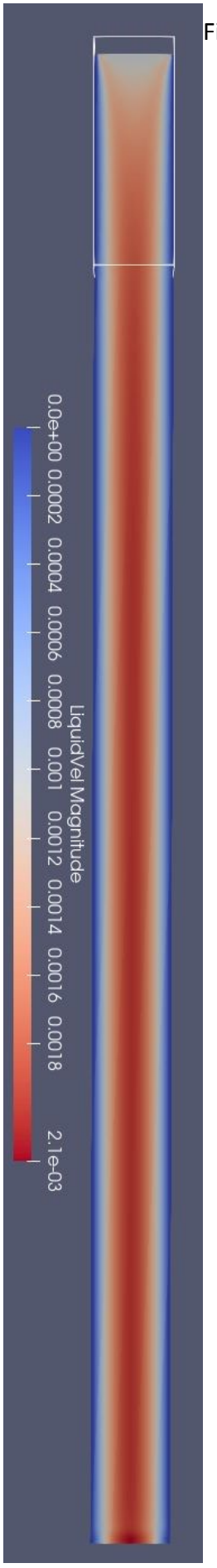


Fig. 7-8: Longitudinal velocity field for set-up 2 and a Reynolds number of 100

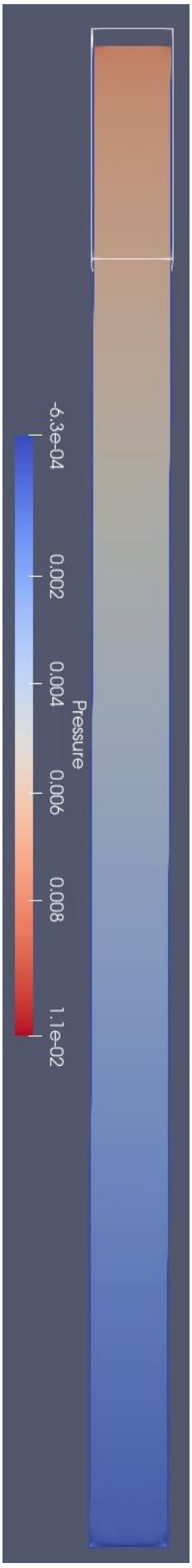


Fig. 7-9: Longitudinal pressure field for set-up 2 and a Reynolds number of 100

7.5.3 Longitudinal Velocity and Pressure Fields of Set-up 3

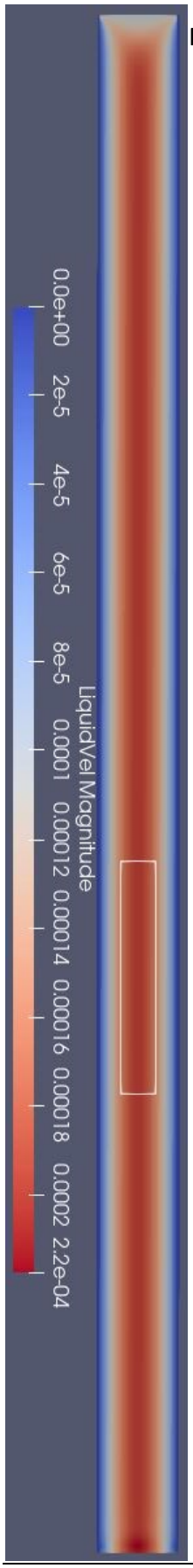
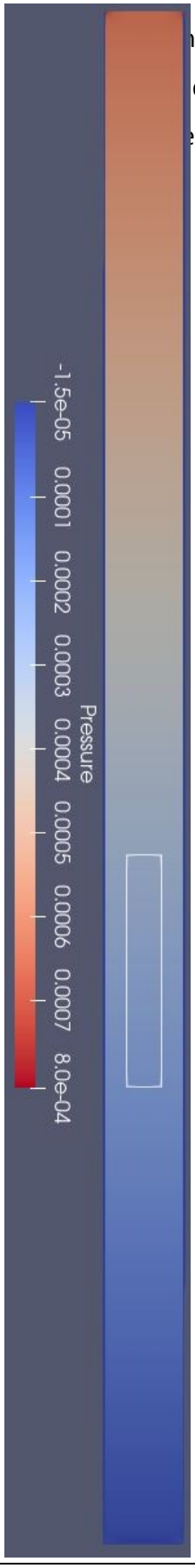


Fig. 7-10: Longitudinal velocity field for set-up 3 and a Reynolds number of 100



ngitudinal pres-
d for set-up 3
eynolds number

7.6 Further Simulation Results of Agitated Tanks

7.6.1 Set-up 4

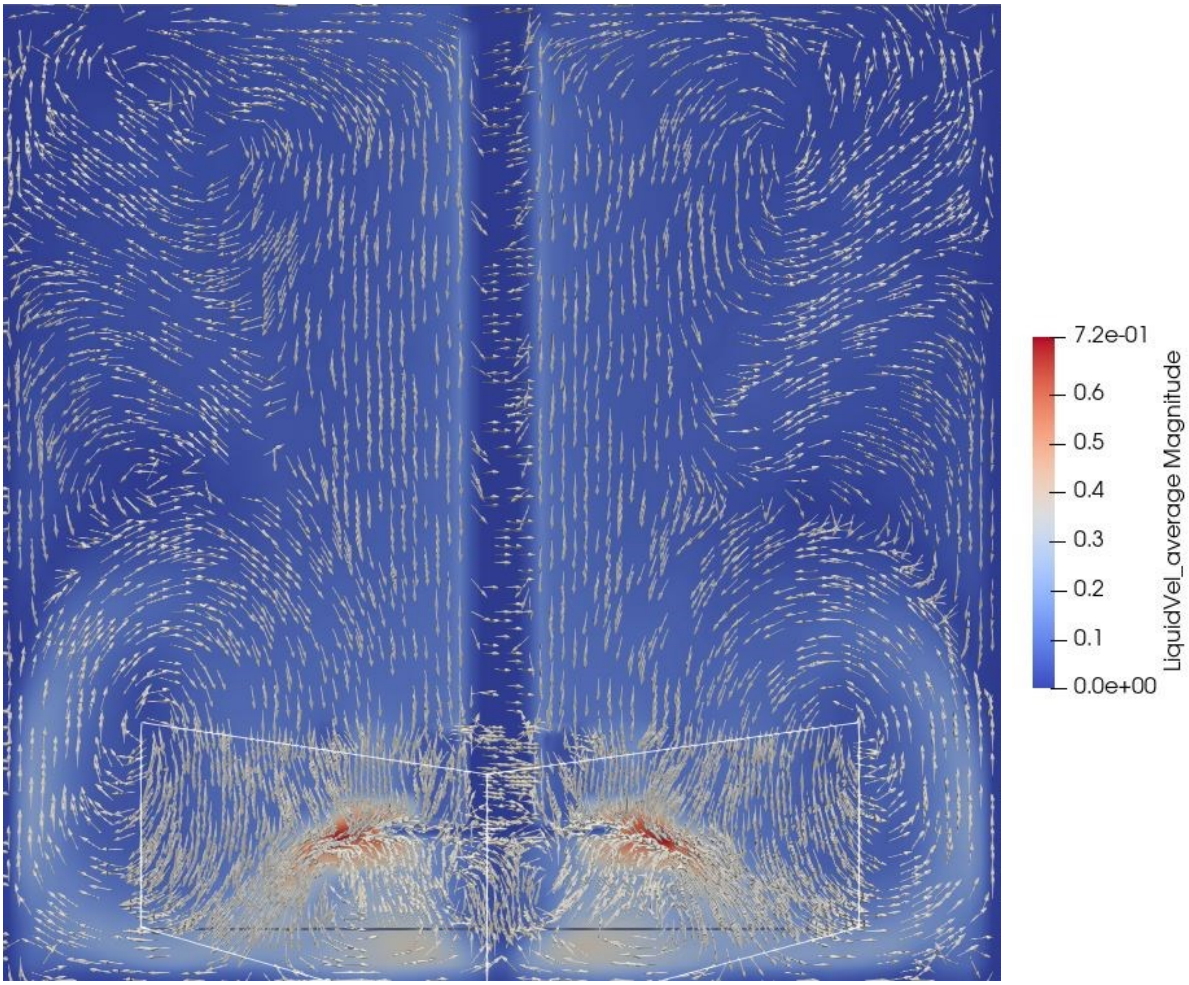


Fig. 7-12: Phase-averaged plot of the velocity field of the whole vertical section of the reactor obtained from the simulation with in-plane velocity vectors for set-up 4

7.6.2 Set-up 5

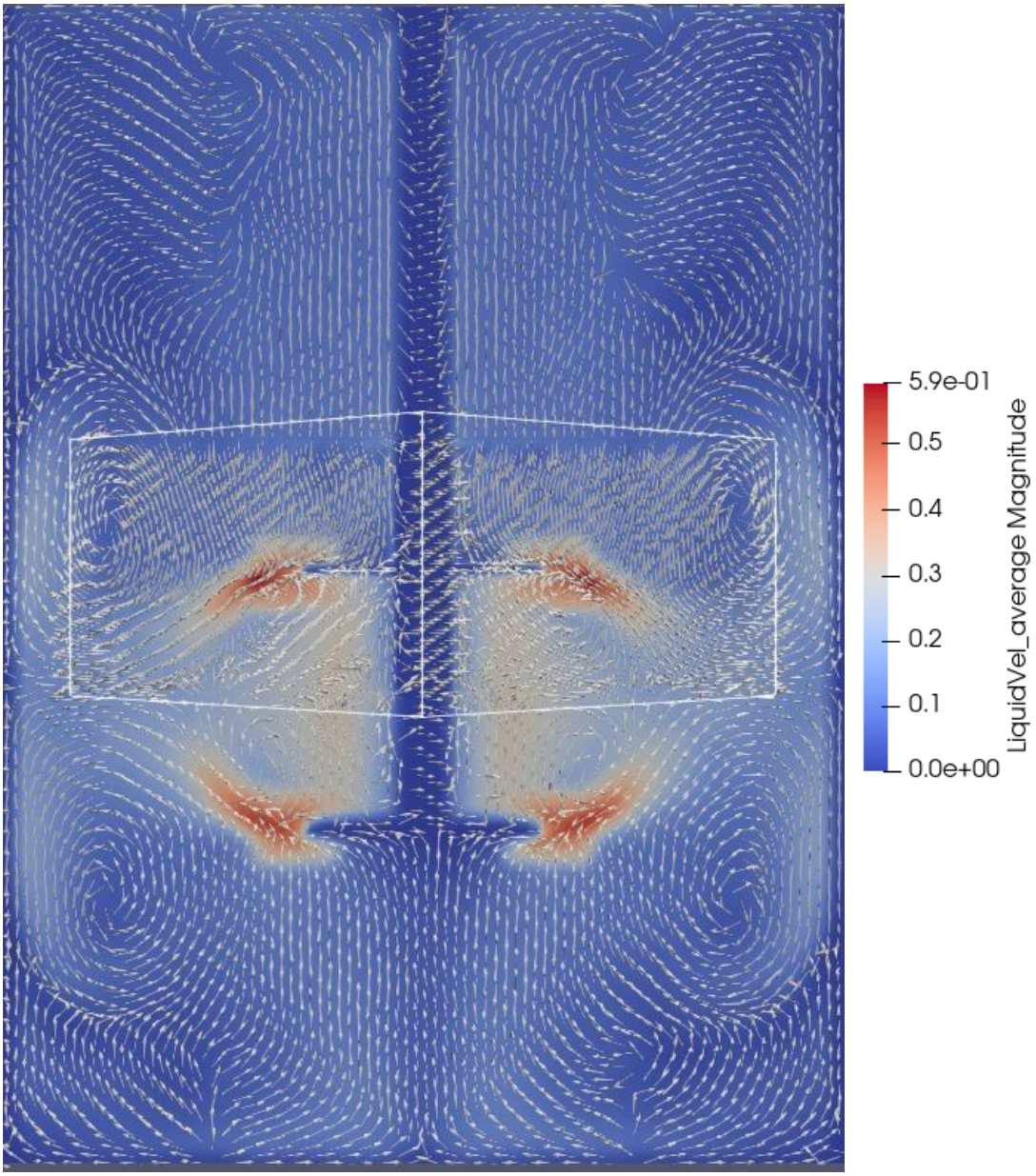


Fig. 7-13: Phase-averaged plot of the velocity field of the whole vertical section of the reactor obtained from the simulation with in-plane velocity vectors for set-up 5

8 Bibliography

[1] E. S. Langer, D. E. Gillespie, R. A. Rader und S. H. Cosper, „15th Annual Report and Survey of Biopharmaceutical Manufactoring Capacity and Production," BioPlan Associates, Inc., Rockville, 2018.

[2] R. A. Rader und E. S. Langer, „Biopharma Market: An Inside Look," Pharma Manufacturing, 2018.

[3] T. Krüger, H. Kusumaatmaja, A. Kuzmin, O. Shardt, G. Silva und E. M. Viggen, The Lattice Boltzmann Method, Switzerland: Springer, 2017.

[4] A. A. Mohamad, Lattice Boltzmann Method - Fundamentals and Engineering Applications with Computer Code, London: Springer, 2011.

[5] P. L. Bhatnagar, E. P. Gross und M. Krook, „A model for collision process in gases. I. Small amplitude processes in charged and neutral one-component systems," *Physical Review 94-3,* pp. 511-525, 1 05 1954.

[6] E. M. Viggen, The lattice Boltzmann method: Fundamentals and acoustics, Trondheim, PhD Thesis: NTNU Trondheim, 2014.

[7] X. He und L.-S. Luo, „Lattice Boltzmann model for the incompressible Navier-Stokes Equation," *Journal of Statistical Physics V. 88 Nos. 3/4,* pp. 927-944, 21 February 1997.

[8] T. K. Valderhaug, „The Lattice Boltzmann Simulation on Multi-GPU Systems," Master Thesis; Norwegian University of Science and Technology; Department of Computer and Information Science, Trondheim, 2011.

[9] M. Junk, A. Klar und L. S. Luo, „Asymptotic analysis of the lattice Boltzmann equation," *Journal of computational physics 210,* pp. 676-704, 2005.

[10] E. M. Viggen, „The lattice Boltzmann method with applications in acoustics," Master Thesis at NTNU, Trondheim, Norway, 2009.

[11] P. Lallemand und L.-S. Luo, „Lattice Boltzmann method for moving boundaries," *Journal of Computational Physics 184,* pp. 406-421, 203.

[12] E. Vergnault, O. Malaspinas und P. Sagaut, „A lattice Boltzmann method for nonlinear disturbances around an arbitrary base flow," *Journal of Computational Physics,* pp. 8070-8082, 2012.

[13] J. Qi, H. Klimach und S. Roller, „Implementation of the compact interpolation within the octree based Lattice Boltzmann solver Musubi," *Computers and MAthematics with Applications,* pp. 1131-1141, 2019.

[14] J. D. Owens, M. Houston, D. Luebke, S. Green, J. E. Stone und J. C. Phillips, „GPU Computing Graphics Processing Units-powerful, programmable, and highly parallel-are increasingly targeting general-purpose computing applications," *Proceedings of the IEEE 96/5,* pp. 879-899, May 2008.

[15] J. Sanders und E. Kandrot, CUDA by Example - An Introduction to General-Purpose GPU Programming, Boston: Pearson Education, Inc., 2011.

[16] V. W. Lee, C. Kim, J. Chhugani und M. Deisher, „Debunking the 100X GPU vs. CPU Myth: An Evaluation of Throughput Computing on CPU and GPU," *ACM SIGARCH Computer Architecture News,* pp. 451-460, 2010.

[17] NVIDIA, „CUDA C++ Programming Guide," NVIDIA, 27 October 2020. [Online]. Available: https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html#abstract. [Zugriff am 11 November 2020].

[18] M. Schönherr, K. Kucher, M. Geier, M. Stiebler, S. Freudiger und M. Krafczyk, „Multi-thread implementations of the lattice Boltzmann method on non-uniform Grids for CPUs and GPUs," *Computers and Mathematics with Applications 61,* pp. 3730-3743, 2011.

[19] H. Chen, O. Filippova, J. Hoch, K. Molvig, R. Shock, C. Teixeira und R. Zhang, „Grid refinement in lattice Boltzmann methods based on volumetric formulation," *Physica A 362,* pp. 158-167, 2006.

[20] D. W. Lagrava Sandoval, „Revisiting grid refinement algorithms for the lattice Boltzmann method," Univ. Genève. PhD Thesis, 2012.

[21] G. Eitel-Amor, M. Meinke und W. Schröder, „A lattice-Boltzmann method with hierarchically refined meshes," *Computers & Fluids 75,* pp. 127-139, 2013.

[22] M. Geier, A. Greiner und J. G. Korvink, „Bubble functions for the lattice Boltzmann method and their application to grid refinement," *The European Physical Journal Special Topics 171,* pp. 173-179, 2009.

[23] D. Yu, R. Mei, L. S. Luo und W. Shyy, „Viscous flow computations with the method of lattice Boltzmann equation," *Progress in Areospace Sciences 39,* pp. 329-367, 2003.

[24] A. Dupuis und B. Chopard, „Theory and applications of an alternative lattice Boltzmann refinement algorithm," *Physical Review E 67,* pp. 066707-1 - 066707-7, 2003.

[25] M. S. M. S. M. K. M. Geier, „Compact second-order accurate momentum interpolation for the lattice Boltzmann method in three dimensions," in *International Conference for Mesoscopic Methods in Engineering and Science*, Edmonton, Canada, 2010.

[26] O. Filippova und D. Hänel, „Grid Refinement for Lattice-BGK Models," *Journal of Computational Physics 147,* pp. 219-228, 10 August 1998.

[27] G. Brenn und W. Meile, Strömungslehre und Wärmeübertragung - Skriptum, Graz: Institut für Strömungslehre und Wärmeübertragung, Graz Technical University, 2016.

[28] S. Bochardt, *Experimente zum Bewegungsbeginn in laminarer und turbulenter Strömung,* Darmstadt: Masters-Thesis, Fachgebiet Wasserbau und Hydraulik, Technische Universität Darmstadt, 2015.

[29] G. Brenn, *Materialien zu den Vorlesungen und Übungen Strömungslehre und Wärmeübertragung,* Graz: Institut für Strömungslehre und Wärmeübertragung - Graz University of Technology, 2020.

[30] C. Pan, J. Min, X. Liu und Z. Gao, „Investigation of Fluid Flow in a Dual Rushton Impeller Stirred Tank Using Particle Image Velocimetry," *Chinese Journal of Chemical Engineering 16(5),* pp. 693-699, 2008.

[31] K. Rutherford, K. C. Lee, S. M. S. Mahmoudi und M. Yianneskis, „Hydrodynamic Characteristics of Duao Rushton Impeller Stirred Vessels," *AIChE Journal Vol. 42 2,* February 1996.

[32] Z. Li, Y. Bao und Z. Gao, „PIV experiments and large eddy simulations of single-loop flow fiels in Rushton turbine stirred tanks," *Chemical Engineering Science 66,* pp. 1219-1231, 2011.

[33] M. Stiebler, M. Krafczyk, S. Freudiger und M. Geier, „Lattice Boltzmann large eddy simulation of subcritical flows around a sphere on non-uniform grids,“ *Computers and Mathematics with Applications 61,* pp. 3475-3484, 2011.

9 List of Symbols

9.1 Latin Letters

A	$[m^2]$	Cross-sectional area
$a_0, a_x, \dots, a_{xz},$	$[-]$	Coefficients of 2nd order polynomial in x-direction
$b_0, b_x, \dots, b_{xz}$	$[-]$	Coefficients of 2nd order polynomial in y-direction
$C000, \dots, C111$	$[-]$	Nodal values in trilinear interpolation
C_1	$[m]$	Distance between stirrer and tank bottom
C_2	$[m]$	Distance between the two stirrers
$c_0, c_x, \dots, c_{xz}$	$[-]$	Coefficients of 2nd order polynomial in z-direction
c_i	$\left[\frac{m}{s}\right]$	Discrete microscopic velocity vector
$c_{\bar{\imath}}$	$\left[\frac{m}{s}\right]$	Subtend discrete microscopic velocity vector of c_i
c_s	$\left[\frac{m}{s}\right]$	Speed of sound
$c_{i\alpha}$	$\left[\frac{m}{s}\right]$	Component of the discrete microscopic velocity vector
d	$[m]$	Normalized length of absorbing layer
D_1	$[m]$	Outer diameter of the stirrer blade
D_2	$[m]$	Inner diameter of the stirrer blade
D_T	$[m]$	Tank diameter
F_α	$\left[\frac{kg}{m^2 s^2}\right]$	Component of body force density vector
F_{x_1}	$[m]$	Position of fine grid from domain boundary x=0
F_{x_2}	$[m]$	Width of the fine grid
F_{z_1}	$[m]$	Position of fine grid from tank bottom
F_{z_2}	$[m]$	Height of fine grid
f	$\left[\frac{kg \cdot s^3}{m^6}\right]$	Particle distribution function
f_i	$\left[\frac{kg}{m^3}\right]$	Discrete particle distribution function

h_i	$\left[\frac{kg}{m^3}\right]$	Adapted particle distribution function from Valderhaug [8]
$h_{\bar{\imath}}$	$\left[\frac{kg}{m^3}\right]$	Subtend particle distribution function of h_i
h_j	$\left[\frac{kg}{m^3}\right]$	Specularly reflected particle distribution function of h_i
H	$[m]$	Tank height
Kn	$[-]$	Knudsen number
l_{abs}	$[m]$	Length of absorbing layer
l_c	$[m]$	Characteristic length
l_{dev}	$[m]$	Length of fully developed flow
l_{hyd}	$[m]$	Hydrodynamic inlet length
l_{mfp}	$[m]$	Mean free path
l_{tube}	$[m]$	Length of tube
l_x	$[m]$	Length of simulation domain in x-direction
l_y	$[m]$	Length of simulation domain in y-direction
l_z	$[m]$	Length of simulation domain in z-direction
Ma	$[-]$	Mach number
Δm	$[kg]$	Conversion factor mass
N_x	$[-]$	Number of nodes in x-direction
n	$[-]$	Refinement factor
p	$[Pa]$	Pressure
Δp	$[Pa]$	Difference to ambient pressure
$\dot{Q}$	$\left[\frac{m^3}{s}\right]$	Volume flow
R	$[m]$	Radius of tube
R_g	$\left[\frac{J}{K \cdot mol}\right]$	Ideal gas constant
Re	$[-]$	Reynolds number

r	$[m]$	Coordinate of radius in polar coordinates
r_{ijk}	$[-]$	Vectors pointing to all nodes of an interpolation cell from base node
$S_{\alpha\beta}$	$\left[\frac{1}{s}\right]$	Strain rate tensor
T	$[K]$	Temperature
t	$[s]$	Time
Δt	$[s]$	Length of one time step – conversion factor time
u	$\left[\frac{m}{s}\right]$	Macroscopic velocity vector
u_α	$\left[\frac{m}{s}\right]$	Component of the macroscopic velocity vector
u_{in}	$\left[\frac{m}{s}\right]$	Input velocity vector
u_m	$\left[\frac{m}{s}\right]$	Volume equivalent velocity
u_{max}	$\left[\frac{m}{s}\right]$	Maximum velocity
u_w	$\left[\frac{m}{s}\right]$	Macroscopic velocity vector of the wall
w_i	$[-]$	Weighting factor
x	$[m]$	Spatial coordinate vector or x component of coordinate vector
Δx	$[m]$	Distance between adjacent nodes – conversion factor length
x_α	$[m]$	Component of the spatial coordinate vector
x_b	$[m]$	Spatial coordinate vector of boundary
x_1, x_2	$[-]$	Distances of the fine grid boundaries to the boundaries of the coarse grid in x-direction
y	$[m]$	y component of coordinate vector
y_1, y_2	$[-]$	Distances of the fine grid boundaries to the boundaries of the coarse grid in y-direction
z	$[m]$	z component of coordinate vector

z_1, z_2	$[-]$	Distances of the fine grid boundaries to the boundaries of the coarse grid in z-direction

9.2 Greek Letters

$\delta_{\alpha\beta}$	$[-]$	Kronecker delta
ε	$[\%]$	Relative error
θ	$[°]$	Phase angle between measurement point and blade
μ	$[Pa \cdot s] = \left[\frac{kg}{ms}\right]$	Dynamic viscosity
ν	$\left[\frac{m^2}{s}\right]$	Kinematic viscosity
ξ	$\left[\frac{m}{s}\right]$	Microscopic velocity vector
ξ_α	$\left[\frac{m}{s}\right]$	Component of the microscopic velocity vector
Π_0	$\left[\frac{kg}{m^3}\right]$	Zeroth order moment of particle distribution function
Π_α	$\left[\frac{kg}{m^2 s}\right]$	First order moment of particle distribution function
$\Pi_{\alpha\beta}$	$\left[\frac{kg}{ms^2}\right]$	Second order moment of particle distribution function – Momentum flux tensor
ρ	$\left[\frac{kg}{m^3}\right]$	Density
$\Delta\rho$	$\left[\frac{kg}{m^3}\right]$	Local deviation of density
ρ_0	$\left[\frac{kg}{m^3}\right]$	Constant fluid density
ρ_b	$\left[\frac{kg}{m^3}\right]$	Local fluid density at boundary node
ρ_w	$\left[\frac{kg}{m^3}\right]$	Density at boundary
$\sigma_{\alpha\beta}$	$\left[\frac{kg}{ms^2}\right]$	Total stress tensor
$\sigma'_{\alpha\beta}$	$\left[\frac{kg}{ms^2}\right]$	Viscous stress tensor
τ	$[s]$	Relaxation time
υ	$\left[\frac{m}{s}\right]$	Relative velocity vector

Ω	$\left[\frac{kg \cdot s^2}{m^6}\right]$	Collision operator
Ω_i	$\left[\frac{kg}{m^3}\right]$	Discrete collision operator
ϵ	$[-]$	Expansion parameter of Chapman-Enskog

9.3 Superscripts

X^{eq}	Equilibrium quantity
X^{neq}	Non-equilibrium quantity
$\tilde{X}$	Quantity in lattice units
$X^{post-coll}$	Post-collision quantity
$X^{pre-coll}$	Pre-collision quantity
$\bar{X}$	Moment of adapted distribution function h_i
$X^{(1)}, X^{(2)}$	Quantity of 1st or 2nd order perturbation
X^*	Quantity at location of the ghost nodes

9.4 Subscripts

X_c	Quantity of coarse grid
X_f	Quantity of fine grid
X_{loc}	Quantity of local coordinate system
X_{glo}	Quantity of global coordinate system
X_{calc}	Quantity obtained from an equation
X_{sim}	Quantity obtained from a simulation

10 List of Figures

11 List of Tables