MASTERARBEIT / MASTER THESIS

Implementierung und Anwendung eines adjungierten Formoptimierungsverfahrens in einem Lattice-Boltzmann-Löser zur Strömungssimulation

Implementation and application of an adjoint shape optimization algorithm in a Lattice-Boltzmann solver for flow simulation

Kevin Volkmer
Naturwissenschaftlich-Technische Fakultät
Department Maschinenbau
Lehrstuhl für Strömungsmechanik
Prof. Dr.-Ing. H. Foysi

Betreuender Hochschullehrer:	Prof. Dr.-Ing., Dipl.-Phys. H. Foysi
1. Prüfer:	Prof. Dr.-Ing., Dipl.-Phys. H. Foysi
2. Prüfer:	M.Sc. Armin Ghajar Jazi

Dezember 2013

Erklärung

Hiermit erkläre ich, die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet zu haben.

Siegen, den 17. Dezember 2013

Name

Abstract

In this work an adjoint shape optimization algorithm for the Lattice-Boltzmann method is introduced. Instead of a standard, no-slip boundary treatment, an alternative approach of geometry mapping is used. The geometry is mapped through the use of porous media. This strategy is promising, because in this way, the geometry can be continuously altered from fluid to solid. Furthermore, a porosity scaling is applied to achieve a stronger convergence to binary (0-1) solutions. This model is then validated by comparison to results received with the standard no-slip boundary treatment. Afterwards the optimization gradient is derived through the use of adjoints in combination with the Lagrangian functional. This gradient is then validated viacomparison with the gradient obtained by finite differences. Thereafter it is implemented in an optimization algorithm, which is then utilized in a few cases of application.

Keywords: Lattice-Boltzmann method, porous media, adjoint optimization, shape optimization

Zusammenfassung

In dieser Arbeit wird ein adjungiertes Formoptimierungsverfahren für die Lattice-Boltzmann Methode eingeführt. Anstelle von standard, nicht gleitenden Randbedingungen, wird ein alternativer Ansatz zur Abbildung von Geometrien verwendet. Die Geometrie wird durch das Nutzen von porösen Medien abgebildet. Dieses Verfahren ist vielversprechend, da auf diesem Wege die Geometrie kontinuierlich von Fluid zu Festkörper verändert werden kann. Weiterhin wird ein Saklierungsverfahren für die Porösitäten angewendet, um eine stärkere Konvergenz hin zu binären (0-1) Lösungen zu erreichen. Als nächstes wird dieses Modell validiert durch den Vergleich mit Resultaten, die mit der standard, nicht gleitenden Randbedingung erhalten wurden. Danach wird der Optimierungsgradient hergeleitet mit Hilfe von Adjungierten in Kombination mit dem Lagrange Funktional. Dieser Gradient wird dann validiert durch Vergleich mit Gradienten, die durch finite Differenzen berechnet wurden. Hiernach wird dieser in einen Optimierungsalgorithmus implementiert, welcher dann in wenigen Anwendungsfällen eingesetzt wird.

Schlagwörter: Lattice-Boltzmann Methode, poröse Medien, adjungierte Optimierung, Formoptimierung

Contents

List of symbols

Symbol	Description
$\mathbf{v}$	molecular velocity vector
$\mathbf{x}$	position vector
f	distribution function
t	time
Ω	collision operator
Π	propagation operator
ω	collision frequency
$\mathbf{c}$	lattice velocities vector
ρ	density
$\mathbf{u}$	velocity vector
u, v	velocities in x, y direction respectively
$\mathbf{w}$	lattice weightings
D	diameter
ν	kinematic viscosity
δ_x, δ_t	discrete time or length magnitude
l_0	length of domain
c_s	speed of sound
N	domain length in number of (smallest) cell lengths
p	pressure or porosity, according to context
c_D	drag coefficient
c_L	lift coefficient
F	force or performance functional, according to context
BGK	Bhatnagar, Gross and Krook
LBM	lattice Boltzmann method
Re	Reynolds number
Ma	Mach number
y	coordinate in vertical direction
μ	dynamic viscosity
μ_e	effective viscosity
$dimles$	dimensionless
$\mathbf{K}$	permeability tensor
κ	exponent for porosity scaling

Symbol	**Description**
L	Lagrangian functional
λ	Lagrange multiplier
a	adjoint variable
s	design variable
g	constraint function
n	number of cells in whole domain
$n\_o\_distributions$	number of distributions
$\mathbf{M}$	combination of collision and propagation operator
$\mathbf{R}$	residuum vector
$\boldsymbol{\xi}$	search direction vector
ϵ	perturbation
Indexes	
$\bigcirc_1$, $\bigcirc_2$, $\bigcirc_3$	coordinate directions
$\bigcirc_i$	index for coordinate direction or running index
$\bigcirc^{\mathrm{eq}}$	at equilibrium state
$\bigcirc_\alpha, \bigcirc_\beta$	lattice direction (0 to 8 in D2Q9 model)
$\bigcirc'$	signs a new or different value or operator
$\bigcirc_{LB}$	lattice Boltzmann
$\bar{\bigcirc}$	mean magnitude
$\bigcirc^*$	adjoint
$\bigcirc_D$	drag
$\bigcirc_L$	lift

List of Figures

1. Introduction

1.1. Motivation and background

Since the behavior of fluids has great impact on technical and social situation (e.g. aircrafts, weather), there is a vast desire to predict fluid motion. Due to the increasing calculation power of modern computers, methods to numerically simulate these flows have penetrated science and industry. These methods enable us to predict fluid flow behavior and flow parameters for different problems. Hence, the influence of flow problems on the environment and technical systems, as well as the control of the flow can be determined.
One method to simulate fluid flow is the lattice Boltzmann method. This method is promising, since it has strong benefits for massive parallel calculations in comparison to other simulation methods.

In every field in which parameters can be influenced to change an outcome, there is a great demand for adjusting these parameters to achieve the optimal output. In this way the cost to benefit ratio can be minimized, making these methods interesting for industry, science and even for everyday decisions of individuals.
Heuristic optimization methods were most probably applied since humans became able to influence their environment. For fluid flow problems, heuristic optimization methods are limited due to complexity of the flow behavior. Hence, analytical and algorithmic optimization methods are applied by using the aforementioned calculation power of modern computers.
Cases of application range from optimizing the geometry of artificial objects, such as bypasses for blood flow in medicine, to minimizing the aeroacoustic noise of aircrafts.

1.2. Aim of work

In the current thesis an adjoint shape optimization algorithm shall be implemented and applied into an existing lattice Boltzmann solver. In order to do this, a method of Pingen et al. [10] should be reproduced, improved and tested for shape optimization. Porous media shall replace standard no-slip boundary conditions to map geometry for simulation. This is promising to be a suitable, continuous method for an optimization algorithm. The main part of the work is to determine the optimization gradient through adjoint equations. This gradient is then validated with comparison to finite differences and finally applied in an optimization algorithm. A few shape optimizations shall then be obtained to test the algorithm.

1.3. Our current state

Spaid et al. [12] presented a successful way to model flow in porous media, via the lattice Boltzmann method. They intended to simulate the flow of molding applications. Porous media is considered through a change in the calculation of the equilibrium distribution. This modification lowers the magnitude of momentum at porous sites. Pingen et al. [10] have later shown that an adjoint optimization algorithm can be successfully implemented in a lattice Boltzmann solver to optimize 2D design problems. In order to achieve a continuous optimization problem, they used porous media to map geometry into the numerical method. In order to achieve better convergence to binary (0-1) solutions, they introduced a porosity scaling and showed its benefit. They applied the presented algorithm for a few topology optimizations to prove the feasibility.
In a further work Makhija et al. [5] applied the aforementioned optimization method even for multi-component flows. In this work they successfully optimized channels for mixers.

1.4. Outline of the project

The current report is divided into six chapters. Chapter 2 gives a short introduction of flow simulation using the lattice Boltzmann method. In the next chapter, Chapter 3, the main simulation setups are described with dimensions, boundary conditions etc.. Chapter 4 explains how porous media is simulated in general and why it is applied in this work. Moreover the implementation, adaption and validation is described therein. Chapter 5 is about optimization and adjoint optimization and the concrete implementation. Furthermore, both a validation and a few cases of applications are shown. Additionally, a minor improvement is presented. At last a short closure and outlook is given in Chapter 6.

2. The Lattice-Boltzmann method

In order to simulate fluid flows, there are different methods and approaches which can be followed. All of them have to fulfill the Navier-Stokes equations which describe the motion of fluid flows. In common methods, macro dynamic magnitudes are directly calculated through discretization of these equations (top-down method).

The Lattice-Boltzmann method is a particularly distinct numerical method, which is based on micro dynamic behavior of fluid flows. Macroscopic values are then calculated from the resulting flow state (the so called bottom-up method). Therefore, the first attempts in this micro dynamic direction tried to reconstruct real fluid particle collisions with energy and mass conservation collision laws. In order to achieve this, discrete particles were used. This method is called the Lattice-Gas Cellular Automata (LGCA) and was the predecessor of today's Lattice-Boltzmann method. In the modern form, instead of discrete particles, continuous distribution functions are used. Gas kinetic theory then helps to describe the behavior of ensembles of particles through statistical physics. In statistical physics, the Boltzmann equation can describe the probability $f = f(\mathbf{v}, \mathbf{x}, t)$, in which a particle with molecular velocity $\mathbf{v}$ at time t is located at $\mathbf{x}$. The Boltzmann equation without body forces is [13]:

$$\frac{\partial f}{\partial t} + \mathbf{v}\frac{\partial f}{\partial \mathbf{x}} = \Omega(f) \tag{2.1}$$

with the collision operator $\Omega(f)$ on the right hand side. The simplest collision operator, namely the BGK-collision approximation, was developed by Bhatnagar, Gross and Krook and is as follows [13]:

$$\Omega(f) = \omega(f^{eq} - f) \tag{2.2}$$

where ω is the collision frequency which depends, among other things, on the viscosity. f^{eq} is the equilibrium distribution given by the Maxwell distribution function. This distribution function, describes the state corresponding to thermodynamic equilibrium. The collision term relaxes the non-equilibrium distribution function f towards f^{eq}, hence, giving rise to the second name for the BGK model, the "single time relaxation" model. This model is used in the current thesis.

Hereafter two more steps are added to reach the final form of this equation. At first we have to discretize our equation in time and space. This is done by restricting possible propagation velocities and directions to a discrete number, the so called lattice velocities, as well as restricting the time to a discrete time step. Secondly we can make the discrete Boltzmann equation non-dimensional. The result is the Lattice-Boltzmann equation:

$$f_{\alpha'}(x_i + c_{\alpha,i}\Delta t, t + \Delta t) - f_\alpha(x_i, t) = -\omega(f_\alpha - f_\alpha^{eq}) \tag{2.3}$$

We then discretize the Maxwell distribution with a Taylor series [10]:

$$f_\alpha^{eq} = w_\alpha \rho \left[1 + 3(c_{\alpha,i}u_i) + \frac{9}{2}(c_{\alpha,i}u_i)^2 - \frac{3}{2}u_i^2 \right] \tag{2.4}$$

Here w_α's are the lattice weightings, ρ is the density and the $c_{\alpha,i}$'s are the lattice velocity vectors. Unfortunately, by approximating the Maxwell distribution with the Taylor series, we are now constrained to low Mach numbers.
w_α and $c_{\alpha,i}$ depend on the chosen lattice model. For the model with nine velocities and two dimensions, the so called D2Q9-model, the lattice velocities and their numerations are seen in Figure 2.1.

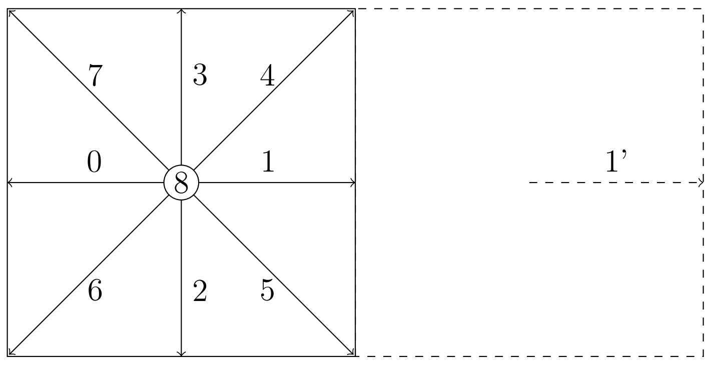


Figure 2.1: D2Q9-lattice cell with an indicated neighbor cell; distribution 1 will propagate into this neighbor cell and then will occupy the place of distribution 1'

The lattice directions link neighboring cells and are the same for each cell, even boundary cells.
Thus, in this case $c_{\alpha,i}$ is:

$$c_{\alpha,i} = \begin{pmatrix} -1 & 1 & 0 & 0 & 1 & 1 & -1 & -1 & 0 \\ 0 & 0 & -1 & 1 & 1 & -1 & -1 & 1 & 0 \end{pmatrix} \tag{2.5}$$

w_α is introduced to ensure isotropy of certain lattice tensors. Its entries depend on the lattice velocity. For the D2Q9-model they are:

$$w_\alpha = \begin{pmatrix} \frac{1}{9} & \frac{1}{9} & \frac{1}{9} & \frac{1}{9} & \frac{1}{36} & \frac{1}{36} & \frac{1}{36} & \frac{1}{36} & \frac{4}{9} \end{pmatrix} \tag{2.6}$$

The macroscopic values density ρ, velocity in flow direction (x-direction) u and the velocity vertical to the flow direction v are given by:

$$\rho = \sum_{\alpha=0}^{8} f_\alpha \tag{2.7}$$

$$\rho u = \sum_{\alpha=0}^{8} c_{\alpha,1} f_\alpha$$

$$\Leftrightarrow u = \sum_{\alpha=0}^{8} c_{\alpha,1} f_\alpha / \rho \tag{2.8}$$

$$v = \sum_{\alpha=0}^{8} c_{\alpha,2} f_\alpha / \rho \tag{2.9}$$

The density is just a sum of all distributions contained in a cell. The macroscopic velocities depend on the lattice model and are a result of the momentum contribution of each distribution.
The Lattice-Boltzmann equation can be split into two steps. The collision step, where each cell locally performs its BGK-collisions

$$\tilde{f}_\alpha = f_\alpha - \omega(f_\alpha - f_\alpha^{eq}) \tag{2.10}$$

and the propagation step, where the new distributions propagate along their lattice directions to their neighbor (if not the rest distribution):

$$f_{\alpha'}(x_i + c_{\alpha,i}\Delta t, t + \Delta t) = \tilde{f}_\alpha(x_i, t) \tag{2.11}$$

Thus, a simulation can be depicted as in Figure 2.2. The evaluation of ρ and $\mathbf{u}$ is depicted separately in this figure, to clarify the algorithm, but from now on it is seen as part of the collision step. For boundary cells this procedure varies as described in Chapter 2.1. The advantage of the LBM against other simulation methods is the great potential for parallel calculation, since only the propagation step has to access neighbor cells.

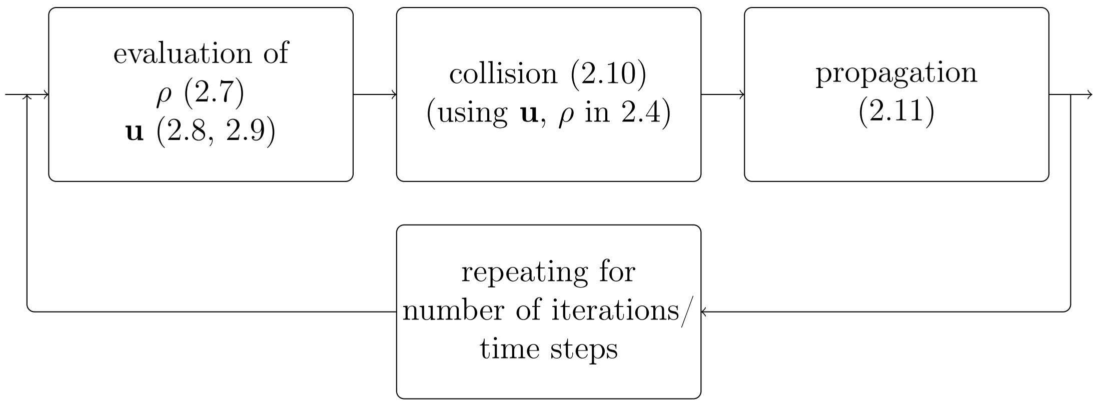


Figure 2.2: flowchart of the Lattice-Boltzmann method (for non-boundary cells)

2.1. Boundary conditions

From the general Lattice-Boltzmann algorithm, it is apparent that boundary cells have to be treated separately. This is obvious, since some incoming and outgoing distributions have no neighbors for propagation.
In the simplest case, boundary conditions are assigned fixed values for the density or velocities, the so called Dirichlet boundary condition. These fixed values enable the calculation of equilibrium distributions, which are then used as new values for the addressed distributions.
In cells next to a wall, there are also distributions which come from the outside. The standard solution is to reflect the outgoing distributions as incoming with various interpolation schemes [10].

2.2. LBM units and real units

For simulations we set all the parameters in dimensionless form. A fixed Machnumber should be chosen at about 0.1 or less to get a good compromise between calculation cost and discretization error, as it also determines the time step.
With $\bar{u}$ as the average velocity and by fixing the Reynolds number, the kinematic viscosity is specified.

$$Re = \frac{\bar{u} D_{LB}}{\nu} \tag{2.12}$$

The BGK equation can be expanded to the Navier-Stokes equations through the Chapman-Enskog expansion [13, p. 143ff]. Its result couples the collision frequency

ω and the viscosity as follows:

$$\omega = \frac{2}{1 + 6\nu} \tag{2.13}$$

Through numerical stability analysis, it can be shown that ω has to be lower than two for it to lead to a stable calculation [4].
The space in LBM is measured as multiples of the smallest cell length [4]:

$$\delta_x = \frac{l_0}{N} \tag{2.14}$$

where l_0 is the length of the domain and N is the number of (smallest) cell lengths fitting into this length.
The velocity is expressed in terms of the speed of sound c_s which here gets the fixed value

$$c_s = \frac{1}{\sqrt{3}} \tag{2.15}$$

Whereas the time is defined as:

$$\delta_t = \frac{\delta_x}{\sqrt{3}\, c_s} \tag{2.16}$$

Since dimensionless values of a system calculated in real units and Lattice-Boltzmann units have to be equal, it is possible to convert units into the other system. For a pressure drop this leads to:

$$\Delta p_{dimles} = \frac{\Delta p_{LB}}{\rho_{LB} \bar{u}_{LB}^2} = \frac{\Delta p_{real}}{\rho_{real} \bar{u}_{real}^2} \tag{2.17}$$

with both ρ's chosen to hold the value 1 it is:

$$\Delta p_{real} = \Delta p_{LB} \frac{\bar{u}_{real}^2}{\bar{u}_{LB}^2} \tag{2.18}$$

ρ and the pressure are connected via

$$p = \frac{1}{c_s^2} \rho \tag{2.19}$$

since the system is almost incompressible.
The drag coefficient is defined as follows:

$$c_{D\_real} = c_{D\_LB} = \frac{2F_{D,LB}}{\rho_{LB}\bar{u}_{LB}^2 D_{LB}} \tag{2.20}$$

The lift coefficient is defined similarly:

$$c_{L\_real} = c_{L\_LB} = \frac{2F_{L,LB}}{\rho_{LB}\bar{u}_{LB}^2 D_{LB}} \tag{2.21}$$

3. Simulation setup

In order to accomplish shape optimization in fluid flows, practically every geometry seems to be conceivable to start with. But since the geometry mapping with porosities has to be validated, the first of two test cases is closely related to the 2D test case of Schaefer and Turek [11]. Most of its data is depicted in Figure 3.1.

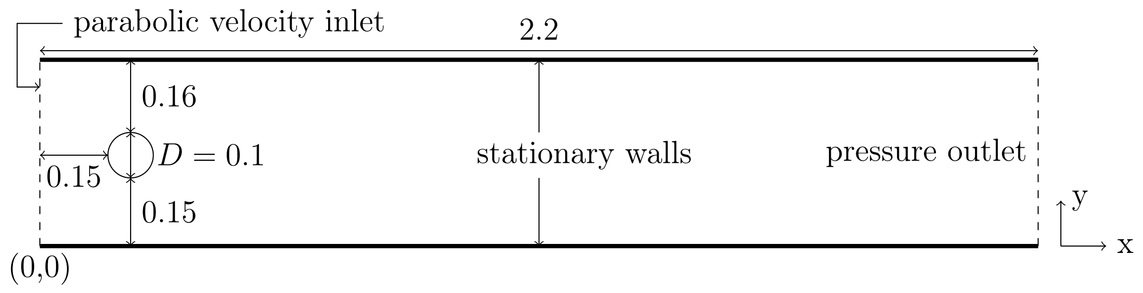


Figure 3.1: flow channel and cylinder

As can be seen, the cylinder is placed a bit below the center in a perpendicular direction. In this way numerical instabilities, such as those that lead to the Karman vortex street, would emerge earlier in simulation.
A fixed value is incorporated as a boundary condition at the inlet. Therefore, a parabolic inflow profile is given. The distributions without propagation sources then receive their value by performing a pseudo collision step. This collision step uses the predefined velocities for the calculation of the equilibrium function. The inlet velocity profile is defined through:

$$u(y) = \frac{3}{2} Ma \cdot c_s \cdot 4(y - y^2) \tag{3.1}$$

Where the perpendicular velocity v is set to zero.
At the outlet the pressure is prescribed. The distributions without propagation source (the one effected by the condition) are set to the equilibrium distribution. This in turn depends on the calculated u, v and the fixed density which is:

$$\rho' = (1 + \sum_{\alpha=0}^{8} f_\alpha)/2 \tag{3.2}$$

As a wall bounce back condition, the half-way, no-slip bounce back condition is chosen. This condition just lets distributions which would hit the wall, propagate to their opposing distribution on the same cell. Thereby all distributions in the wall cells are set and no change in the collision step has to be implemented.
The four corner cells are only treated as inlet/outlet cells and not as wall cells.
To validate the geometry mapping through porosities, a pressure drop between the points at (0.15, 0.2) and (0.25, 0.2) is defined like in [11].

Furthermore, a second geometry is used as a test case. In this case the elimination of the wall influence is intended. Therefore, instead of stationary walls, periodic boundaries are used. This can then be seen as an infinite cascade of the geometric object. Details can be seen in Figure 3.2. The boundary conditions for the inlet and outlet are the same as in the above case, in addition to the fact that the inlet velocity now has a uniform profile:

$$u = \frac{3}{2} Ma \cdot c_s \tag{3.3}$$

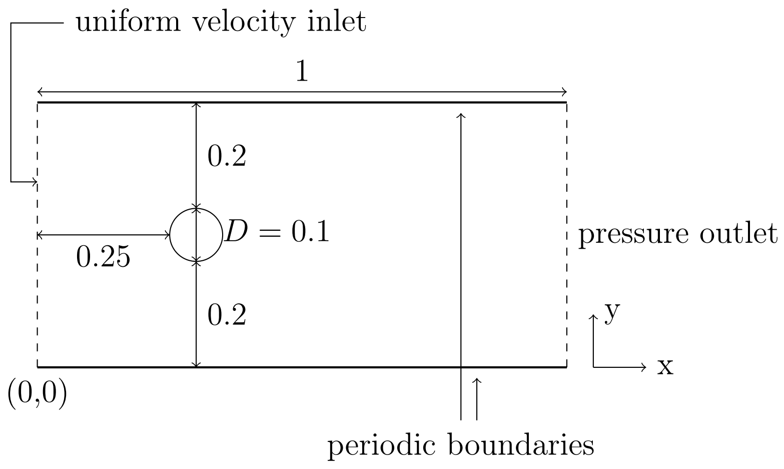


Figure 3.2: setup of the periodic domain

While the first domain will be applied for validations and a first case of application, the last shown domain will only be used for some cases of applications.

In both shown cases the following facts are pertained: In order to achieve comparable results for a set of calculations, the collision frequency has to be similar. In the current studies it receives the value 1.8 for all calculations.
The Reynolds number also receives fixed values and is 20, if not defined differently.
For the numerical accuracy, double precision was chosen.

4. Porous media for geometry mapping

In the current chapter the fundamentals of porous media simulation and its use for the current work shall be presented. Then a porosity scaling to obtain better posed problems for geometry optimization will be introduced. Lastly this scaling, together with boundary smoothing, will be validated by comparison to the aforementioned test case of Schaefer and Turek [11].

For optimization of shape as well as topology it is important to find an appropriate approach to describe solid regions and their boundaries. Since these regions are updated during optimization, special requirements are imposed on these approaches. Emerging and vanishing boundaries are not suitably described by standard wall boundary methods. Standard wall boundary methods switch cells on or off, and then have to rearrange boundary cells. A more continuous method is better conditioned for optimization. Spaid and Phelan [12] showed that porous media is promising for describing geometry in optimization, because they can be continuously altered from solid to fluid.

4.1. Simulation of porous media

Porous media are objects containing pores which can be perfused by a fluid. This type of medium could be directly simulated by applying fine geometries, but this would lead to complex geometry mapping and, therefore, high cost. If the influence on the flow variables is known, it will then be possible to express this influence in the flow equations.
In simulations where fluid and porous regions exist simultaneously, by default Stokes and Brinkman equations are used [8]. The Brinkman equation then deals with the porous regions. It is:

$$\mu_e \nabla^2 \mathbf{u} - \mu \mathbf{K}^{-1} \mathbf{u} = \nabla p \tag{4.1}$$

Where μ is the viscosity, μ_e is the effective viscosity, $\mathbf{u}$ and p are volume averaged velocities and pressures and $\mathbf{K}$ is the permeability tensor.
For the LBM, an approach to solve the Brinkman equation for porous media is needed. This has to be achieved by lowering the momentum at porous sites while conserving its direction. Spaid and Phelan [12] show that through a small change in the collision step, wall boundary conditions can be formulated algorithmically. Lowering of the momentum is accomplished by decreasing the local velocities, which are used for the calculation of the equilibrium distributions. The decrease depends on the magnitude of the local porosity p (compare [10]).
From now on $\tilde{\mathbf{u}}$ will represent the unscaled velocity and $\mathbf{u}$ the velocity with porosity scaling:

$$\mathbf{u}(\mathbf{x},t) = (1 - p(\mathbf{x}))\,\tilde{\mathbf{u}}(\mathbf{x},t) \tag{4.2}$$

where the porosity has a value of one for solid sites, zero for fluid sites and intermediate values at sites which are not completely solid or fluid:

$$0 \leq p(\mathbf{x}) \leq 1 \tag{4.3}$$

The loss of momentum accompanies it with a force on the porous sites. For a single cell it is:

$$\mathbf{F}(\mathbf{x},t) = \omega p(\mathbf{x})\rho(\mathbf{x},t)\tilde{\mathbf{u}}(\mathbf{x},t) \tag{4.4}$$

This force can be summed up over all porous cells, which then results in the force acting on the geometry.

4.2. Porosity scaling

The porous geometry mapping was introduced to achieve rather continuous geometry changes. Solutions where the porosity holds values equal to zero or one, representing fully fluid and solid sites respectively, are preferable. This kind of solution is called a 0-1 solution. The request for such a binary solution seems to be in contrast to the continuous approach through porosities, and in fact it is. But the porosities can be implemented in a way that the optimization algorithm tends to 0-1 solutions.
Moreover in the following chapter the approach of mapping the geometry through porosities will be compared to the 2D test case of Schaefer and Turek [11].
To reach a solution approximated to a 0-1 solution, the optimization algorithm has to treat the influence of a change of intermediate porosity values, higher than for the one being close to zero or one.
The standard porosity scaling, discussed in Chapter 4.1, can be enhanced for pro-

viding a stronger tendency to binary solutions. This is done here (compare [10]) by introducing an exponent κ for the porosity in the prefactor, resulting in:

$$\mathbf{u}(\mathbf{x},t) = \left(1 - p\left(\mathbf{x}\right)^{\kappa}\right)\tilde{\mathbf{u}}(\mathbf{x},t) \tag{4.5}$$

For low porosities, the sensitivity of the prefactor magnitude for a porosity change is high for the standard scaling. Figure 4.1 shows this fact and the influence of other exponents κ.

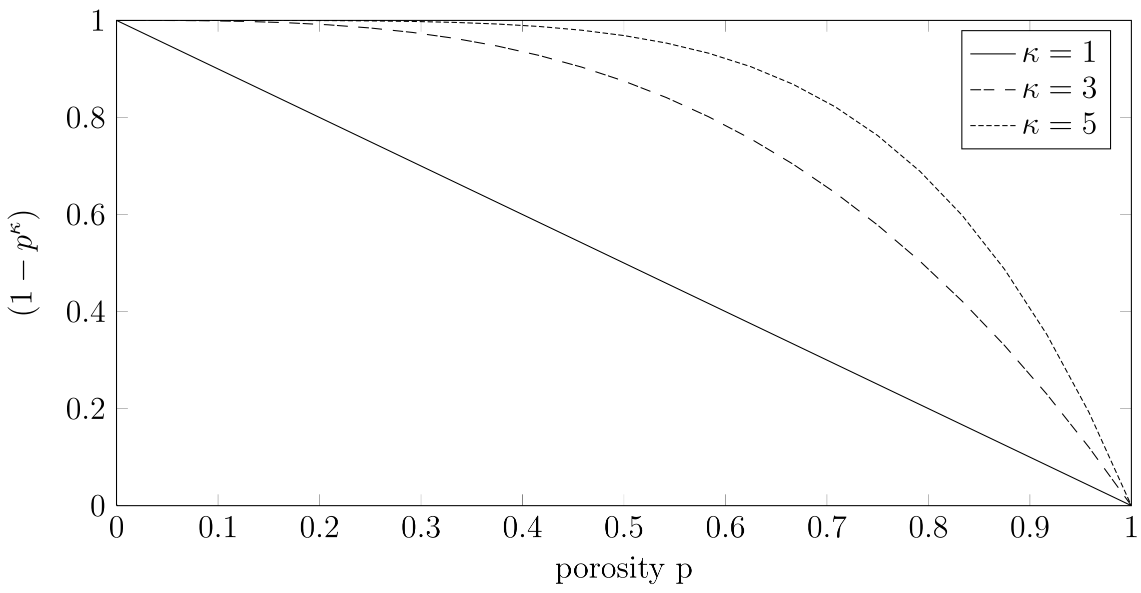


Figure 4.1: the porosity prefactor for different exponents κ

The gradient of the scaled velocity with respect to the porosity follows:

$$\frac{\partial \mathbf{u}}{\partial p} = \frac{\partial \tilde{\mathbf{u}}(1 - p^{\kappa})}{\partial p} = \begin{cases} -\tilde{\mathbf{u}}\kappa p^{\kappa-1} & \text{if } \kappa > 1 \\ -\tilde{\mathbf{u}} & \text{if } \kappa = 1 \end{cases} \tag{4.6}$$

At low porosities a small gradient in the prefactor is desired to compensate for the high velocities in these sites. In this way the overall gradient stays at small values. At fully fluid nodes ($p = 0$) the gradient is zero for $\kappa > 1$. This will disable the optimization algorithm to produce a solid state somewhere in the fluid away from the boundary. For some optimization goals, like topology optimization, achieving this can be desirable, but not for the current shape optimization.
The velocities at solid sites, which are near the boundary, should be close to zero. Therefore, a high gradient in the prefactor is favorable to compensate for the small velocities in the gradient equation. Due to non existing velocities for solid sites in the inner geometry, the gradient is practically zero at these locations. Whereas in

regions where the porosity is intermediate, the velocities should also have intermediate values, which should then lead to the highest overall gradients. This was the aim of the adjusted porosity scaling.
As this reveals, the gradient depends on the correlation of $\tilde{\mathbf{u}}$ and the porosity value. But this connection is certainly varying for different regions of the flow domain. Therefore, an optimal value for κ doesn't exist for the whole domain at once.
Since the gradient in the prefactor grows unfavorably strong, even for high κ's at high porosities (as seen in Figure 4.1), for the current work a value of 3 for κ was chosen. A more detailed reflection of this topic could be accomplished in the future.

4.3. Boundary smoothing of porous geometry

Before the porous boundary treatment will be validated, a smoothing of the boundary of the porous cylinder is introduced. This is because we are going to use intermediate values, for the representation of the geometry boundary, for optimization purposes as well. In order to achieve such smoothing, boundary cells, which are within a certain distance range around the geometry contour, receive intermediate porosity values. This range is defined as two cell lengths, since this leads to an comparable boundary to the one obtained by the filtering method for design variables, which is discussed in the next chapter. The distance is checked by calculating the shortest distance of each cell center to the geometry contour. As functional representation, a smoothed Heaviside function is used. This causes fully differentiable behavior. The function is then described by:

$$p = \begin{cases} \text{0 if d} > 1 \text{ (in flow region)} \\ \frac{1}{2} - \frac{1}{2} * sin\left(\frac{\pi}{2} * d\right) \text{ if } -1 \leq \text{d} \leq 1 \text{ (in smoothing region)} \\ \text{1 if d} < -1 \text{ (in solid region)} \end{cases} \tag{4.7}$$

With d being a dimensionless length and is, therefore, expressed as multiples of the smallest cell length. This can be depicted as in Figure 4.3. The corresponding mapping can be seen in Figure 4.2.

4.4. Validation of porous geometry mapping

To validate the porous geometry mapping, at first the flow solution is examined for grid independence. This is done by accomplishing simulations at different grid levels while fixing the flow conditions. A grid refinement of one level conforms to a splitting of a cell into four. Here 3072 cells are present at level 7 in the whole domain. The

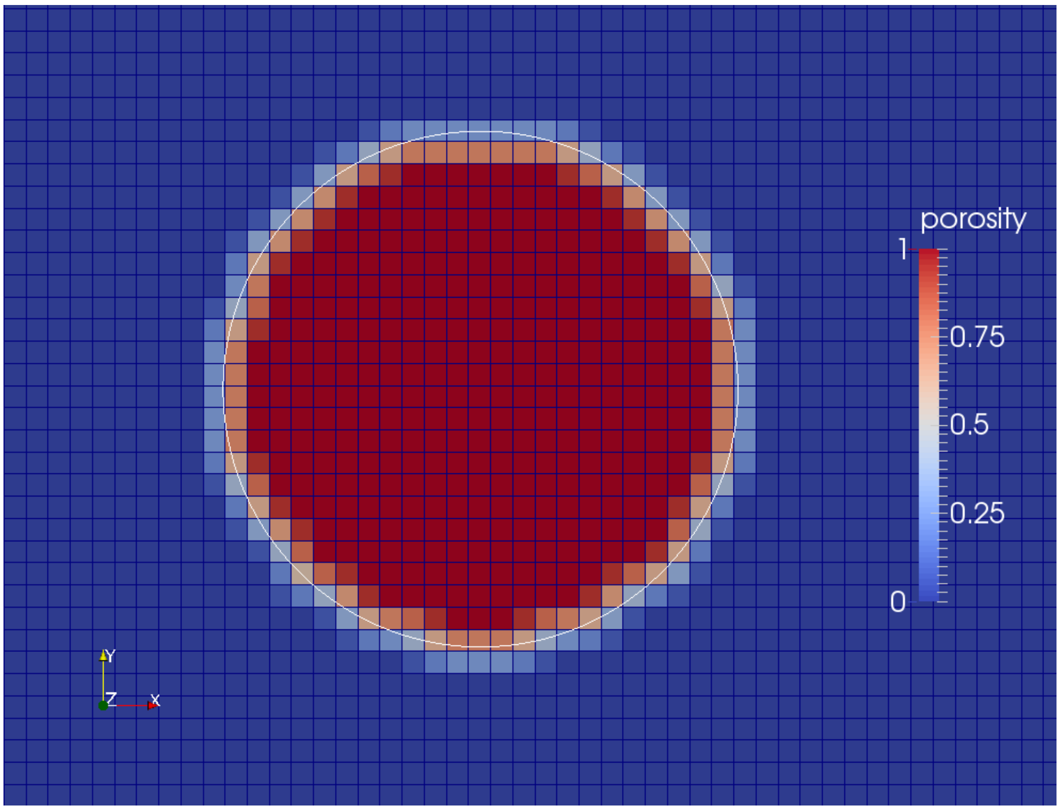


Figure 4.2: cylinder mapped via porous media, at grid refinement level 9; in white the original geometry of the cylinder

Reynolds number is 20 in every case and the respective Mach number is calculated by using the fact, that ω is set to 1.8 for all simulations, as discussed in Chapter 3. Next the drag coefficient c_D is calculated, as explained in Chapter 4.1, and compared to the reference value of Schaefer and Turek [11, p. 12]. This value for the drag coefficient represents the average of all the values they collected in their work for the same test case. The resulting diagram can be observed in Figure 4.4. As can be seen, the result shows an asymptotic behavior towards the value of Schaefer and Turek. From level 10 on the solution shows grid independence, as the relative change of the result compared to the one of the next level, is near zero. At this level the cylinder diameter (based on the given geometry contour) conforms to 46.5454545455 cell lengths.

Next we compare different values for κ with a standard wall boundary condition at different Reynolds numbers in figure 4.5. It can be seen that a value of three for κ fits the bounce back condition the most. Moreover, the geometry mapping through porosities delivers good results at all investigated Reynolds numbers with this κ value.

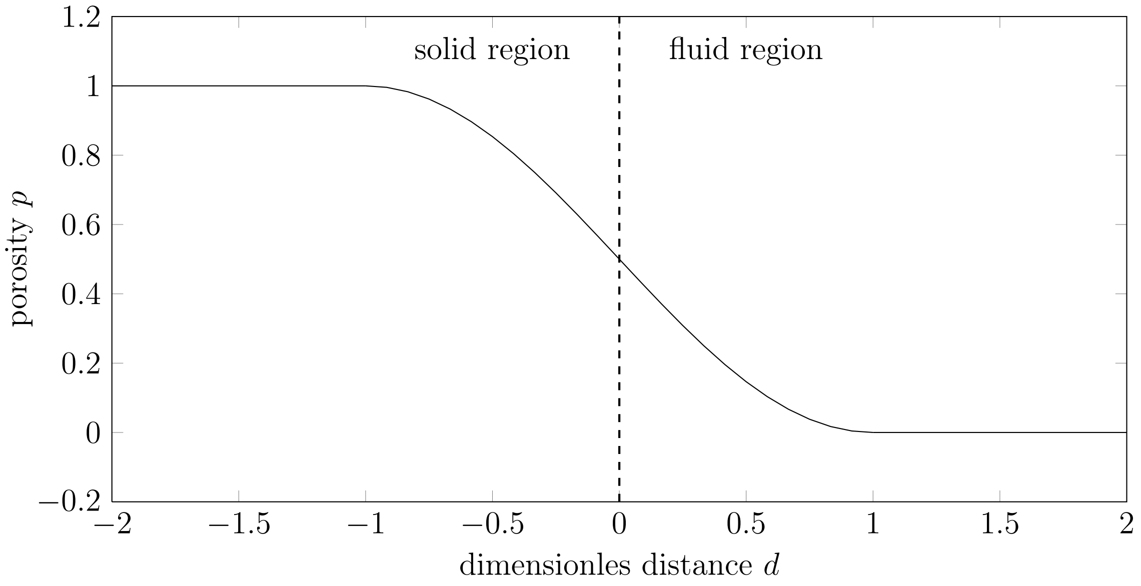


Figure 4.3: porosity smoothing region via smoothed Heaviside function

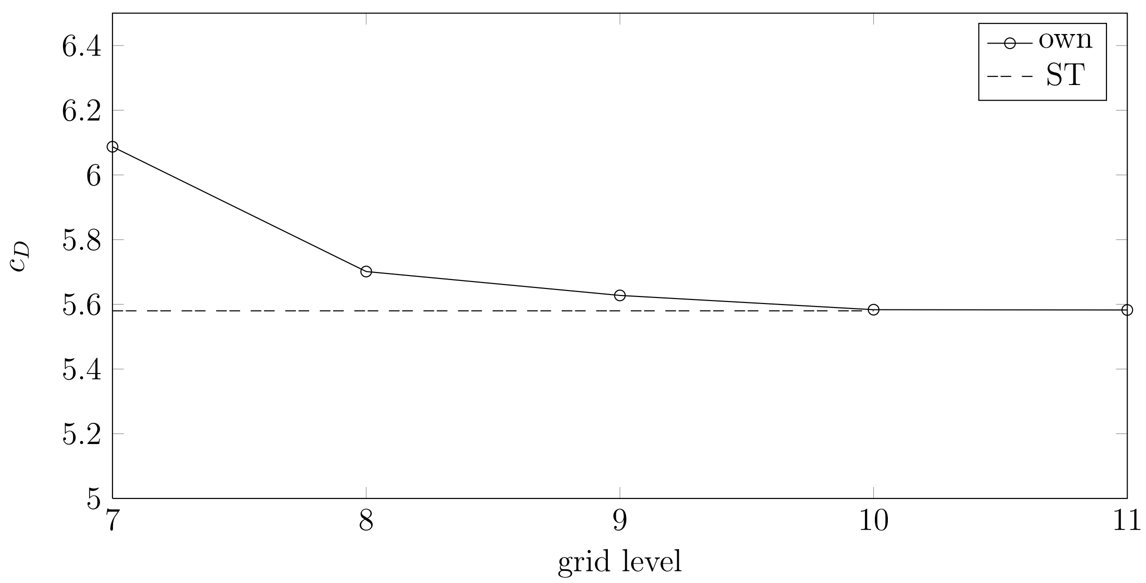


Figure 4.4: checking of grid convergence for $Re = 20$ through consideration of drag coefficient c_D; comparing own results with the average result 5.58 out of Schaefer and Turek [11, p. 12]

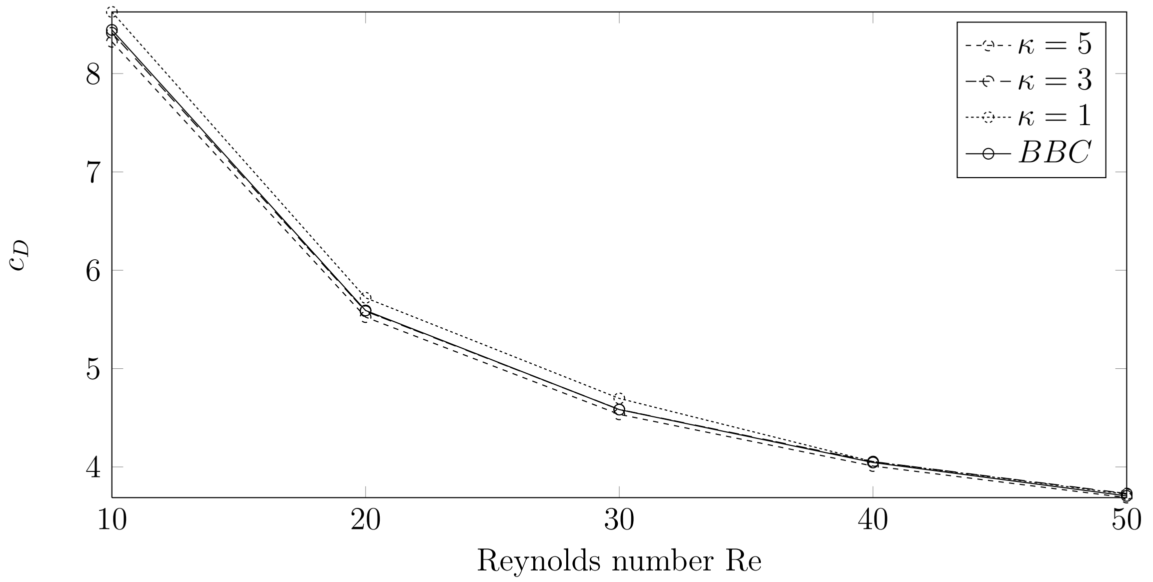


Figure 4.5: comparison of different scaling factors κ with a boundary condition by Mei et. al. [7] through consideration of drag coefficient c_D at varying Reynolds numbers

5. Adjoint optimization for the Lattice-Boltzmann method

Before we dive into detailed implementation of the adjoint optimization, some general information and components for optimization and adjoint optimization will be introduced. The resulting optimization gradient will then be validated via comparison to a discrete gradient obtained through finite differences. Next, the optimization algorithm will be described in detail and finally a few cases of application will be given.

5.1. Optimization and adjoint optimization

A general optimization problem in mathematics is defined as the minimization or maximization of a given function or functional. Such a mathematical principle can be applied to all kind of problems where a configuration can be changed to influence an outcome.
Following these definitions a fluid optimization problem is defined by a performance functional $F(s, f(s))$, which is to be maximized or minimized (e.g. lift or drag of an airfoil). This performance functional can be influenced by changing one or more design variables s (like shape parameters of the airfoil for example). Design parameters in turn alter the flow state $f(s)$, which in LBM is defined by a set of values for the distribution functions. Furthermore, there can be one or more constraints $g(s, f(s))$ which have to be fulfilled (such as a maximum, minimum or an exact amount of weight or volume).

An optimality system can be described using the Lagrangian functional with the Lagrange multipliers λ (all parameters are treated as scalars firstly; nevertheless we are using the transpose sign $()^t$ to lighten the switching to vector formulation):

$$L(s, f(s), \lambda) = F(s, f(s)) - \lambda^{*t} g(s, f(s)) \tag{5.1}$$

The product $\lambda^* g(s, f(s))$ is an inner product.
Then the first order optimality condition states that all derivatives have to be equal to zero once the optimal point is reached:

$$\frac{\delta L(s, f(s), \lambda^*)}{\delta s} = \frac{\delta F(s, f(s))}{\delta s} - \lambda^{*t} \frac{\delta g(s, f(s))}{\delta s} = 0 \tag{5.2}$$

$$\frac{\delta L(s, f(s), \lambda^*)}{\delta f} = \frac{\delta F(s, f(s))}{\delta f} - \lambda^{*t} \frac{\delta g(s, f(s))}{\delta f} = 0 \tag{5.3}$$

$$\frac{\delta L(s, f(s), \lambda^*)}{\delta \lambda^*} = -g(s, f(s)) = 0 \tag{5.4}$$

whereby equation 5.2 gives the gradient, equation 5.3 provides the adjoint and equation 5.4 gives the constraint function.
The resulting system can be solved by three approaches [3]:

1. Either by solving the complex, coupled and nonlinear system to obtain the optimal system in one step, hence the name “single-shot method”. Since it is expensive enough to solve the nonlinear fluid flow system without the additional unknowns, this method is withdrawn here.
2. Another possibility is to solve these equations is using an iterative method. Then a gradient based method is incorporated, assuming the given equations are smooth enough. In this method, at first the current gradient of the performance functional with respect to the design variables will be calculated. Next it will be used to update the design variables. This procedure is repeated until sufficient convergence will be achieved. For this gradient based method, sensitivities can in turn be used which are calculated for each design variable.
3. Another approach is similar to 2. but with a difference in determining the gradient. It is specified by using an adjoint system, which only has to be calculated once, independently from the number of design variables. The optimization algorithm based on this method is depicted in Figure 5.1.

The approach to receive the gradient is then described as follows. The variation of the Lagrange functional is (compare [2]):

$$dL = \frac{\partial F}{\partial f} df + \frac{\partial F}{\partial s} ds - \lambda^{*t} \left(\frac{\partial g}{\partial f} df + \frac{\partial g}{\partial s} ds \right) \tag{5.5}$$

This can be converted to:

$$dL = \left(\frac{\partial F}{\partial f} - \frac{\partial g}{\partial f}^{t} \lambda^* \right)^{t} df + \left(\frac{\partial F}{\partial s} - \lambda^{*t} \frac{\partial g}{\partial s} \right) ds \tag{5.6}$$

Since the state variables non-linearly depend on the design variables (geometry), df is costly to calculate. For thousands of design parameters, also thousands of flow states would have to be calculated to receive df. Therefore, we set $\left(\frac{\partial F}{\partial f} - \frac{\partial g}{\partial f}^t \lambda^*\right)^t$ to zero which makes df unnecessary to calculate.
This results to:

$$\frac{\partial g}{\partial f}^t \lambda^* = \frac{\partial F}{\partial f} \tag{5.7}$$

This equation can be solved for the Lagrange multipliers, which are also called adjoints, since they connect the performance functional and the constraints.
The rest of equation 5.6 then gives the gradient:

$$\frac{dL}{ds} = \frac{\partial F}{\partial s} - \lambda^{*t} \frac{\partial g}{\partial s} \tag{5.8}$$

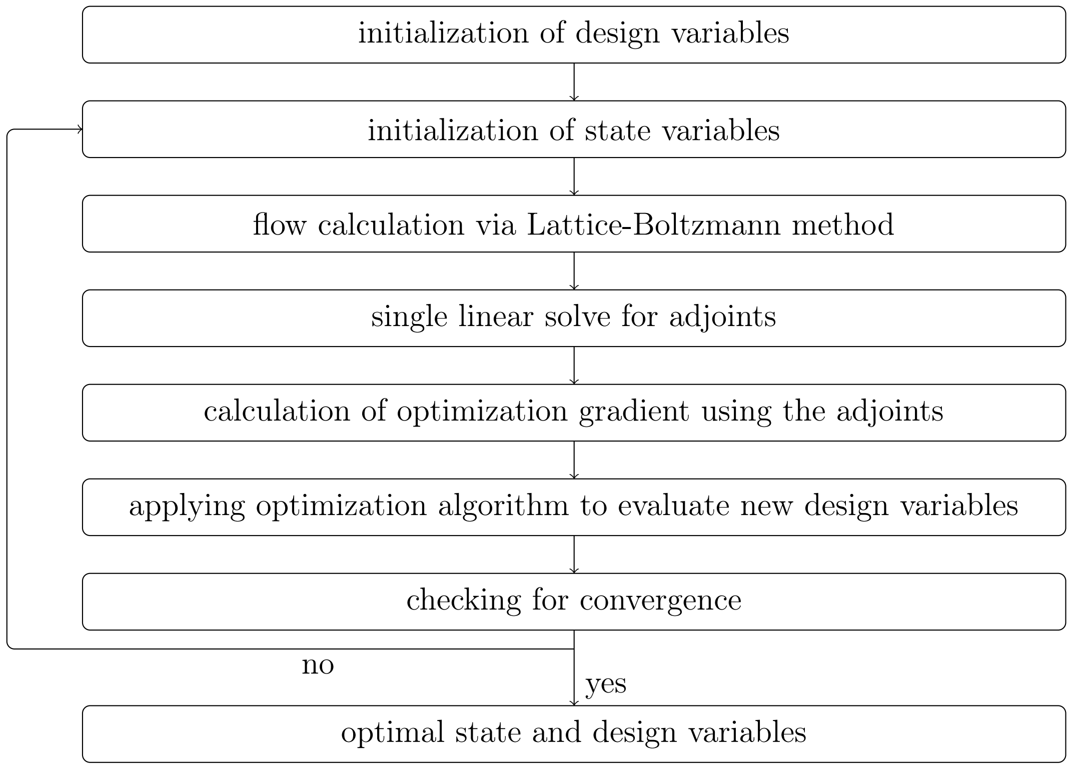

initialization of design variables
initialization of state variables
flow calculation via Lattice-Boltzmann method
single linear solve for adjoints
calculation of optimization gradient using the adjoints
applying optimization algorithm to evaluate new design variables
checking for convergence
no
yes
optimal state and design variables

Figure 5.1: flowchart of adjoint optimization (compare to [3] p. 56)

This gradient can then be applied in different optimization algorithms. In this case the steepest descent or conjugated gradient method (gradient based optimization methods) can be chosen.
For more information the reader is referred to [3].

5.2. Defining the variables

First we identify the design variables, also called the controlling parameters. These parameters can be designed to control a performance (or output), which is defined later on so that the gradients of this functional could be calculated analytically. Furthermore, constraint functions have to be determined.

5.2.1. Design variables

The choice of design variables depends on the given problem. If the problem is to optimize a whole domain, for example in respect to the pressure drop, this problem is called a topology optimization. These problems generally are not described through geometry parameters. Therefore, the porosities could change without being dependent on other design parameters, like the radius of a cylinder. Then the design variables in the simplest case could be the porosities themselves and thus, cause as many design variables as cells in the domain to optimize:

$$p_i = s_i \tag{5.9}$$

By using this method one can calculate the gradient of the performance functional with respect to the porosity in each cell and, therefore, with a localized effect. This can be changed through using a filtering method where the porosity of one cell depends on values which are common for multiple cells. In order to achieve this, we define vertex values which give their respective enclosed cell its porosity value, therefore, reaching a staggered grid. Figure 5.2 depicts this fact.

$$p_i = \frac{s_{x,y} + s_{x-\Delta x,y} + s_{x,y-\Delta y} + s_{x-\Delta x,y-\Delta y}}{4} \tag{5.10}$$

We will later see that this filtering method enables the boundary to grow into regions where the porosity is zero (fully fluid).

In shape optimization one wants to optimize the shape of an geometrical object. This object is often, but not necessarily, described by parameters, such as the parameters of an airfoil. Then the porosities are given through these parameters and the number of design variables equals the number of free geometrical parameters:

$$p_i = p_i(s_j) \tag{5.11}$$

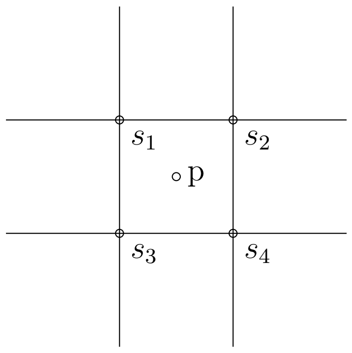


Figure 5.2: depiction of possible design variables; here: $p = \frac{s_1+s_2+s_3+s_4}{4}$

5.2.2. Performance functional

The performance functional analytically describes the calculation of the goal magnitudes.
In this study the pressure drop from inlet to outlet, the drag and lift force or a combination of these are used as performance functionals.
The pressure drop $\Delta p_{pressure}$ (not to be confused with porosity p) by using equation 2.19 is given as:

$$F(\mathbf{f}) = \Delta p_{pressure} = \frac{1}{3}\left(\sum_{i}^{inlet\,cells} \rho_i - \sum_{i}^{outlet\,cells} \rho_i\right) = \\ \frac{1}{3}\left(\sum_{i}^{inlet\,cells}\sum_{\alpha=0}^{8} f_\alpha - \sum_{i}^{outlet\,cells}\sum_{\alpha=0}^{8} f_\alpha\right) \tag{5.12}$$

Using equation 4.4 the drag force for the complete geometry is defined by:

$$F(\mathbf{p}(\mathbf{s}), \mathbf{f}) = F_D = \sum_{i}^{number\,of\,cells} (\omega p_i^\kappa \rho_i \tilde{u}_i) \tag{5.13}$$

The lift force is also defined similarly:

$$F(\mathbf{p}(\mathbf{s}), \mathbf{f}) = F_L = \sum_{i}^{number\,of\,cells} (\omega p_i^\kappa \rho_i \tilde{v}_i) \tag{5.14}$$

As seen the performance functional varies depending on the choices above, therefore, just the label F is assigned to tag the performance functional hereafter.

5.2.3. The steady state constraint

A Lattice-Boltzmann method simulation advances in time by repeating the described solution steps collision and propagation. A steady state can be achieved (requiring flow conditions appropriate for a steady state solution) by repeating this procedure until the change in all distributions between two consecutive time steps has fallen below a chosen value. Under ideal conditions this value would be zero. This statement can be expressed via the following equation:

$$\mathbf{R}(\mathbf{p}(\mathbf{s}), \mathbf{f}) = \mathbf{M}(\mathbf{p}(\mathbf{s}), \mathbf{f}) - \mathbf{f} = \mathbf{0} \tag{5.15}$$

$\mathbf{R}(\mathbf{p}(\mathbf{s}), \mathbf{f})$ describes the residuum which depends on the design variables, the porosities $\mathbf{p}(\mathbf{s})$ and the distribution functions, and should be $\mathbf{0}$ at steady state up to machine precision. The operator $\mathbf{M}(\mathbf{p}(\mathbf{s}), \mathbf{f})$ depicts the Lattice-Boltzmann algorithm and, thus, the combination of collision and propagation step like seen in Figure 2.2 and can be formulated as:

$$\mathbf{M}(\mathbf{p}(\mathbf{s}), \mathbf{f}) = \mathbf{\Pi}(\mathbf{\Omega}(\mathbf{p}(\mathbf{s}), \mathbf{f})) \tag{5.16}$$

Vividly the residuum equation can be interpreted as the difference of the current value of a certain distribution function and the value this distribution function is updated to. Whereby the updating is performed through the next collision and propagation step. Attention has to be paid to distributions which are given through a boundary condition and not through the LB-algorithm. These distributions are later treated separately.
The steady state condition serves as constraint functional $\mathbf{g}(\mathbf{s}, \mathbf{f}(\mathbf{s}))$ in the Lagrangian.

5.3. Determining the derivatives

Until now the Lagrange functional and its resulting equations are present in a more general and scalar form. Nevertheless, design variables, distributions and state variables are vectors in this case. Furthermore, from here on the Lagrange multipliers λ^* will be called adjoints $\mathbf{a}$ and the steady state constraint serves as constraint $\mathbf{g}(\mathbf{s}, \mathbf{f}(\mathbf{s}))$ in the Lagrangian functional (equation 5.1):

$$\mathbf{L}(\mathbf{s}, \mathbf{f}(\mathbf{s}), \mathbf{a}) = \mathbf{F} - \mathbf{a}^t\mathbf{R}(\mathbf{p}(\mathbf{s}), \mathbf{f}) \tag{5.17}$$

Accordingly, the needed gradient is now:

$$\frac{d\mathbf{L}(\mathbf{s},\mathbf{f}(\mathbf{s}),\mathbf{a})}{d\mathbf{s}} = \frac{d\mathbf{F}}{d\mathbf{s}} - \mathbf{a}^t \left(\frac{d\mathbf{R}(\mathbf{p}(\mathbf{s}),\mathbf{f})}{d\mathbf{s}}\right)^t \tag{5.18}$$

If the porosities are not the design variables, both terms on the right hand side of the last equation will have to be expanded:

$$\frac{d\mathbf{F}}{d\mathbf{s}} = \frac{\partial\mathbf{F}}{\partial\mathbf{s}} + \frac{\partial\mathbf{F}}{\partial\mathbf{p}(\mathbf{s})}\frac{d\mathbf{p}(\mathbf{s})}{d\mathbf{s}} \tag{5.19}$$

and

$$\frac{d\mathbf{R}(\mathbf{p}(\mathbf{s}),\mathbf{f}}{d\mathbf{s}} = \frac{\partial\mathbf{R}(\mathbf{p}(\mathbf{s}),\mathbf{f})}{\partial\mathbf{s}} + \frac{\partial\mathbf{R}(\mathbf{p}(\mathbf{s}),\mathbf{f})}{\partial\mathbf{p}(\mathbf{s})}\frac{d\mathbf{p}(\mathbf{s})}{d\mathbf{s}} \tag{5.20}$$

where at both expansions the first term on the right hand side would be zero (if the performance functional does not depend on the design variables).
The adjoints **a** can be calculated through derivation of the Lagrangian equation with respect to the state variables (as discussed in Chapter 5.1). In the current case the adjoint operator simply is the transpose of the basic term. Therefore equation 5.7 results in:

$$\mathbf{a} = \left(\frac{\partial\mathbf{R}(\mathbf{p}(\mathbf{s}),\mathbf{f})}{\partial\mathbf{f}}\right)^{-t}\frac{\partial\mathbf{F}}{\partial\mathbf{f}} \tag{5.21}$$

Solving this equation for the vector with adjoints takes the major amount of computation time together with the flow solving. For the current work deal.II [1] was used for solving this system. As appropriate solver GMRES or Bicgstab has been incorporated together with SparseILU or SSOR as preconditioners. The Bicgstab solver won't be able to solve the equation if the system is not well conditioned. However, together with the SparseILU preconditioner which receives some extra off diagonal entries, it is able to solve most systems in a minimum amount of time.
Equations 5.19, 5.20 and 5.21 in the gradient equation 5.18 result in the final form of the equation to solve:

$$\frac{d\mathbf{L}(\mathbf{s},\mathbf{f}(\mathbf{s}),\mathbf{a})}{d\mathbf{s}} = \frac{\partial\mathbf{F}}{\partial\mathbf{p}(\mathbf{s})}\frac{d\mathbf{p}(\mathbf{s})}{d\mathbf{s}} - \left[\frac{\partial\mathbf{F}}{\partial\mathbf{f}}\left(\frac{\partial\mathbf{R}(\mathbf{p}(\mathbf{s}),\mathbf{f})}{\partial\mathbf{f}}\right)^{-1}\right]^t \left(\frac{\partial\mathbf{R}(\mathbf{p}(\mathbf{s}),\mathbf{f})}{\partial\mathbf{p}(\mathbf{s})}\frac{d\mathbf{p}(\mathbf{s})}{d\mathbf{s}}\right)^t \tag{5.22}$$

local and global indexing

To assemble the above system, a FEM-like approach is chosen in this work. Therefore, to prevent confusion, the difference between local and global indexing of distributions is shown here. The local indices of a cell always range from zero to eight (D2Q9-model). Whereas global indices also depend on the cell number. One can enumerate all distributions contained in all cells in a way that every distribution receives a unique indice. This is done by:

$$global\_index = local\_index + cell\_number \cdot n\_o\_distributions \tag{5.23}$$

with $n\_o\_distributions$ being the distributions of one cell and, in fact, is 9 in this case (since D2Q9-model is used). Therefore, the global indices range from zero to $(n \cdot n\_o\_distributions - 1)$ with n being the number of cells in the whole domain. Thus, for example for the second cell the global indices have to start at 9 and range to 17.

5.3.1. Determination of $\partial\mathbf{R}/\partial\mathbf{f}$

This is the most complex of all needed derivatives since the residuum equation, which contains the Lattice-Boltzmann algorithm, has to be derived with respect to all distributions. Using equation 5.15 it is:

$$\frac{\partial\mathbf{R}(\mathbf{p}(\mathbf{s}),\mathbf{f})}{\partial\mathbf{f}} = \frac{\partial\mathbf{M}(\mathbf{p}(\mathbf{s}),\mathbf{f})}{\partial\mathbf{f}} - \frac{\partial\mathbf{f}}{\partial\mathbf{f}} \tag{5.24}$$

where the result is a matrix of $(n{\cdot}n\_o\_distribution){\cdot}(n{\cdot}n\_o\_distributions)$ entries. The term $\partial\mathbf{f}/\partial\mathbf{f}$, or in global index notation $\partial f_\alpha/\partial f_\beta$, explains this fact clearly: α names the row and β the column of its matrix. Hence each row of the matrix can be associated with a certain distribution. This property holds for other derivatives, too. Coming back to the aforementioned term which is written in index notation:

$$\frac{\partial f_\alpha}{\partial f_\beta} = \begin{cases} 1 \text{ if } \alpha = \beta \\ 0 \text{ otherwise} \end{cases} \tag{5.25}$$

This just describes the identity matrix $\mathbf{I}$.
Now the first term on the right hand side can be examined. The propagation operator Π just reallocates the result of the collision operator $\mathbf{\Omega}$. As discussed in Chapter 2, this happens in two separate steps. Therefore, we can begin by evaluating $\partial\mathbf{\Omega}(\mathbf{p}(\mathbf{s}),\mathbf{f})/\partial\mathbf{f}$ while switching to index notation:

$$\frac{\partial\mathbf{\Omega}(p_i, f_\alpha)}{\partial f_\beta} = \frac{\partial\tilde{f_\alpha}}{\partial f_\beta} = \frac{\partial f_\alpha}{\partial f_\beta} - \omega\left(\frac{\partial f_\alpha}{\partial f_\beta} - \frac{\partial f_\alpha^{eq}}{\partial f_\beta}\right) \tag{5.26}$$

Now that $\partial f_\alpha / \partial f_\beta$ is known, $\partial f_\alpha^{eq} / \partial f_\beta$ should be evaluated. The definition of the equilibrium distribution function has been given back in equation 2.4:

$$f_\alpha^{eq} = w_\alpha \rho \left[1 + 3(c_{\alpha,i} u_i) + \frac{9}{2}(c_{\alpha,i} u_i)^2 - \frac{3}{2} u_i^2 \right]$$

It can be seen that u_i and ρ are functions depending on the distributions. The derivative can be expanded to:

$$\frac{\partial f_\alpha^{eq}}{\partial f_\beta} = \frac{\partial f_\alpha^{eq}}{\partial \rho} \frac{\partial \rho}{\partial f_\beta} + \frac{\partial f_\alpha^{eq}}{\partial u} \frac{\partial u}{\partial f_\beta} + \frac{\partial f_\alpha^{eq}}{\partial v} \frac{\partial v}{\partial f_\beta} \tag{5.27}$$

Where $\partial f_\alpha^{eq} / \partial \rho$, $\partial f_\alpha^{eq} / \partial u$ and $\partial f_\alpha^{eq} / \partial v$ are:

$$\frac{\partial f_\alpha^{eq}}{\partial \rho} = w_\alpha \left[1 + 3(c_{\alpha,i} u_i) + \frac{9}{2}(c_{\alpha,i} u_i)^2 - \frac{3}{2} u_i^2 \right] \tag{5.28}$$

$$\frac{\partial f_\alpha^{eq}}{\partial u} = w_\alpha \rho \left[3(c_{\alpha,1}) + 9 c_{\alpha,1}(c_{\alpha,1} u + c_{\alpha,2} v) - 3u \right] \tag{5.29}$$

$$\frac{\partial f_\alpha^{eq}}{\partial v} = w_\alpha \rho \left[3(c_{\alpha,2}) + 9 c_{\alpha,2}(c_{\alpha,1} u + c_{\alpha,2} v) - 3v \right] \tag{5.30}$$

For the derivative of ρ, u and v, we recall their definitions. By using the local indexing, ρ (equation 2.7), u (equation 2.8) and v (equation 2.9) are:

$$\rho = f_0 + f_1 + f_2 + f_3 + f_4 + f_5 + f_6 + f_7 + f_8 \tag{5.31}$$

$$u = (1 - p^\kappa) \frac{\rho \tilde{u}}{\rho} = (1 - p^\kappa) \frac{f_1 + f_4 + f_5 - f_0 - f_6 - f_7}{f_0 + f_1 + f_2 + f_3 + f_4 + f_5 + f_6 + f_7 + f_8} \tag{5.32}$$

$$v = (1 - p^\kappa) \frac{\rho \tilde{v}}{\rho} = (1 - p^\kappa) \frac{f_3 + f_4 + f_7 - f_2 - f_5 - f_6}{f_0 + f_1 + f_2 + f_3 + f_4 + f_5 + f_6 + f_7 + f_8} \tag{5.33}$$

The according derivatives are (using quotient rule for derivatives of u and v):

$$\frac{\partial \rho}{\partial f_\beta} = [1, 1, 1, 1, 1, 1, 1, 1] \tag{5.34}$$

$$\frac{\partial u}{\partial f_\beta} = (1 - p^\kappa) \left[\frac{-\tilde{u}}{\rho} [1, 1, 1, 1, 1, 1, 1, 1] + \frac{1}{\rho} [-1, 1, 0, 0, 1, 1, -1, -1, 0] \right] \tag{5.35}$$

$$\frac{\partial v}{\partial f_\beta} = (1 - p^\kappa) \left[\frac{-\tilde{v}}{\rho} [1, 1, 1, 1, 1, 1, 1, 1] + \frac{1}{\rho} [0, 0, -1, 1, 1, -1, -1, 1, 0] \right] \tag{5.36}$$

These formulations are only valid for local derivatives where β is in the range of the current cell. For all other β's, the derivatives are zero. This is caused by the local calculation of these values and the fully localized collision step. Accordingly the global matrix is a sparse matrix, a matrix which mostly contains zeros. It results by combining the listed derivatives into the derivative of the collision step (equation 5.26).

Hereafter the connection between lattice model and global Jacobian will be explained with respect to a simple example. A simple D1Q3-model with three nodes is introduced here (Figure 5.3). This results in a local Jacobian with $(3 \cdot 3)$ entries and, hence, a block-diagonal matrix as seen in Figure 5.4a.

The propagation operator propagates distributions according to their direction to

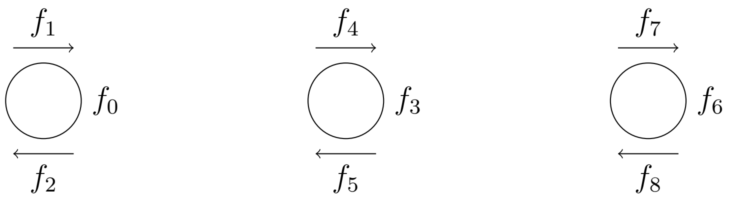


Figure 5.3: D1Q3 lattice model with three nodes and global indexing

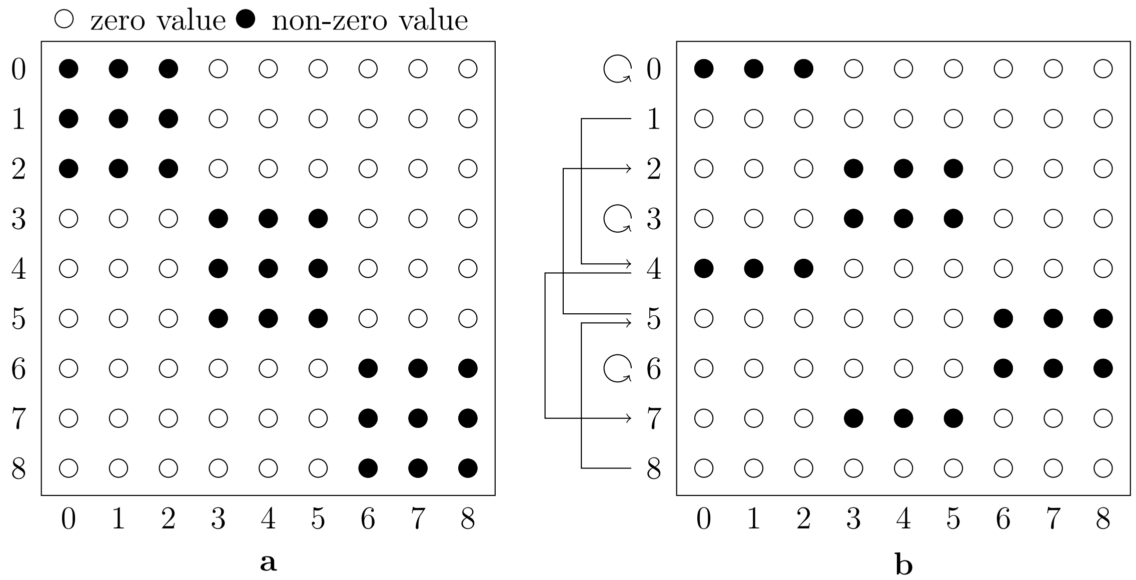


Figure 5.4: D1Q3 lattice Jacobian for Figure 5.3; a shows the collision Jacobian, b shows the result of propagation, propagation is indicated by arrows

their target distributions. By recalling the propagation equation 2.11

$$\mathbf{\Pi}(\mathbf{\Omega}(f_\alpha)) = f_{\alpha'}(x_i + c_{\alpha,i}\Delta t, t + \Delta t) = \tilde{f}_\alpha(x_i, t)$$

the derivative of this equation with respect to the distribution functions in index notation is:

$$\frac{\partial f_{\alpha'}(x_i + c_{\alpha,i}\Delta t, t + \Delta t)}{\partial f_\beta} = \frac{\partial \tilde{f}_\alpha(x_i, t)}{\partial f_\beta} \tag{5.37}$$

It can be seen that the result of the collision step is assigned to a new position in the matrix. In detail this means the entries at row α will be assigned to a new row corresponding to the global index of its propagation target (α'). Anyway, since ∂f_β as the denominator stays the same, the column of all entries doesn't change through propagation. As discussed before, every distribution has an according row and their propagation targets are known. Therefore, the propagation operator just has to shift each row to the row corresponding to its respective target distribution. This can be seen in Figure 5.4b. The distributions at rest (distributions with global indexes 0, 3 and 6) stay in their row, the others propagate to their target.
For example, row one is shifted to row four, like distribution one is propagated to distribution four. At the same time the content of row four is shifted to row seven. Therefore no overwriting occurs.

For the D2Q9 model, the only difference is that each cell has nine distributions, that leads to a local Jacobian with $(9 \cdot 9)$ entries. However, the rows are still shifted in the same way.

Compared to the original work [10], a minor improvement is made here by directly implementing the shifted Jacobian. In this original work, propagation and collision operators are defined as separate matrices, which are later combined through a costly matrix-matrix multiplication.

Two facts are apparent in the D1Q3 example above. Firstly: Distributions 2 and 7 don't have a propagation target. Their rows just vanish. And secondly: Rows 1 and 8 don't have a propagation source (they have no distributions propagating into them) leading to their empty rows after propagation. This is due to the fact that we didn't assign boundary conditions in this example case.

Treatment of boundary cells

As discussed in Chapter 2.1, boundary conditions take care of distributions, like those showed above, to receive a new value in each time step as well. For these distributions the steady state constraint has to be fulfilled. Therefore, we have to derive their defining equations instead of a collision and propagation step.

At the inlet a pseudo collision step with fixed values for u and v is used (see Chapter

3 for more information). Thus, the derivative reads:

$$\frac{\partial \mathbf{M}'(\mathbf{p}(\mathbf{s}),\mathbf{f})}{\partial \mathbf{f}} = \frac{\partial \boldsymbol{\Omega}'(\mathbf{p}(\mathbf{s}),f_\alpha)}{\partial f_\beta} = \frac{\partial \tilde{f}'_\alpha}{\partial f_\beta} = \frac{\partial f_\alpha}{\partial f_\beta} - \omega\left(\frac{\partial f_\alpha}{\partial f_\beta} - \frac{\partial f'^{eq}_\alpha}{\partial f_\beta}\right) \tag{5.38}$$

Since the velocities are no longer dependent on the values of the distributions in this cell, the expanded derivative of the equilibrium function (equation 5.27) is simplified to:

$$\frac{\partial f'^{eq}_\alpha}{\partial f_\beta} = \frac{\partial f'^{eq}_\alpha}{\partial \rho}\frac{\partial \rho}{\partial f_\beta} + \frac{\partial f'^{eq}_\alpha}{\partial u}\frac{\partial u}{\partial f_\beta} + \frac{\partial f'^{eq}_\alpha}{\partial v}\frac{\partial v}{\partial f_\beta} = \frac{\partial f'^{eq}_\alpha}{\partial \rho}\frac{\partial \rho}{\partial f_\beta} + \frac{\partial f'^{eq}_\alpha}{\partial u}0 + \frac{\partial f'^{eq}_\alpha}{\partial v}0 = \frac{\partial f'^{eq}_\alpha}{\partial \rho}\frac{\partial \rho}{\partial f_\beta} \tag{5.39}$$

For the outlet cells a pressure outlet condition was chosen (see Chapter 3). Hence the density ρ is given. It is set to (recalling equation 3.2):

$$\rho' = (1 + \sum_{\alpha=0}^{8} f_\alpha)/2$$

Then the distributions without propagation source are set to the according equilibrium distribution and are therefore:

$$\frac{\partial \mathbf{M}''(\mathbf{p}(\mathbf{s}),\mathbf{f})}{\partial \mathbf{f}} = \frac{\partial f''^{eq}_\alpha}{\partial f_\beta} \tag{5.40}$$

Hence, the expanded derivative of the equilibrium function in this case is:

$$\frac{\partial f''^{eq}_\alpha}{\partial f_\beta} = \frac{\partial f''^{eq}_\alpha}{\partial \rho'}\frac{\partial \rho'}{\partial f_\beta} + \frac{\partial f''^{eq}_\alpha}{\partial u}\frac{\partial u}{\partial f_\beta} + \frac{\partial f''^{eq}_\alpha}{\partial v}\frac{\partial v}{\partial f_\beta} \tag{5.41}$$

The only term which is now different to the standard derivative is:

$$\frac{\partial \rho'}{\partial f_\beta} = \frac{1}{2}[1,1,1,1,1,1,1,1,1] \tag{5.42}$$

For cells at the wall boundary the changes are even simpler. As discussed before, the chosen wall bounce back condition just combines the distributions in a cell through reflection on the wall. This scheme doesn't change the collision step (see Chapter 3). So no changes in the derivatives are needed.

Inserting the derivatives

To be able to test the derivative $\partial \mathbf{R}/\partial \mathbf{f}$, it will be shown how the final derivative appears:

$$\begin{aligned}\frac{\partial f_{\alpha'}(x_i + c_{\alpha,i}\Delta t, t+\Delta t)}{\partial f_\beta} &= \frac{\partial f_\alpha}{\partial f_\beta} - \omega\Bigg[\frac{\partial f_\alpha}{\partial f_\beta} - \\ \Bigg[w_\alpha\big[1 + 3(c_{\alpha,1}u + c_{\alpha,2}v) &+ \frac{9}{2}(c_{\alpha,1}u + c_{\alpha,2}v)^2 - \frac{3}{2}(u^2+v^2)\big] + \\ w_\alpha\big[3(c_{\alpha,1}) + 9(c_{\alpha,1}u &+ c_{\alpha,2}v)c_{\alpha,1} - 3u\big]\big(-u + c_{\alpha,1}(1-p^\kappa)\big) + \\ w_\alpha\big[3(c_{\alpha,2}) + 9(c_{\alpha,1}u &+ c_{\alpha,2}v)c_{\alpha,2} - 3v\big]\big(-v + c_{\alpha,2}(1-p^\kappa)\big)\Bigg]\Bigg]\end{aligned} \tag{5.43}$$

The second row of the equation conforms to $\partial f_\alpha^{eq}/\partial\rho \cdot \partial\rho/\partial f_\beta$ where $\partial\rho/\partial f_\beta$ is just the vector containing ones and, therefore, was omitted. The third row is $\partial f_\alpha^{eq}/\partial u \cdot \partial u/\partial f_\beta$ and the fourth row $\partial f_\alpha^{eq}/\partial v \cdot \partial v/\partial f_\beta$. ρ doesn't occur since it is either derived or canceled away. Notice that this equation will be valid only if indices α and β are in the range of the same cell. Otherwise the derivative is always zero, as discussed before.

5.3.2. Determination of $\partial F/\partial \mathbf{f}$

This derivative again is dependent on the chosen performance functional.

For the pressure drop from inlet to outlet (see equation 5.12) it is:

$$\frac{\partial F}{\partial \mathbf{f}} = \frac{1}{3}\frac{\partial\left(\sum_i^{inlet\,cells}\sum_{\alpha=0}^{8} f_\alpha - \sum_i^{outlet\,cells}\sum_{\alpha=0}^{8} f_\alpha\right)}{\partial f_\beta} \tag{5.44}$$

This results in a vector with 1/3 at all inlet distributions and $-1/3$ at all outlet distributions. Therefore it contains mostly zeros.

Deriving the drag force (equation 5.13) results in:

$$\frac{\partial F_D}{\partial \mathbf{f}} = \frac{\partial \sum_i^{number\,of\,cells}(\omega p_i^\kappa \rho_i \tilde{u}_i)}{\partial f_\beta} = \sum_i^{number\,of\,cells} (\omega p_i^\kappa \frac{\partial \rho_i \tilde{u}_i}{\partial f_\beta}) \tag{5.45}$$

Which is a vector where entries are zero at sites with zero porosity. The local derivative conforms to

$$\frac{\partial \rho \tilde{u}}{\partial f_\beta} = \frac{\partial (f_1 + f_4 + f_5 - f_0 - f_6 - f_7)}{\partial f_\beta} = [-1, 1, 0, 1, 1, -1, -1, 0] \quad \text{for } 0 \leq \beta \leq 8 (local\, indexing) \tag{5.46}$$

and is otherwise zero.

5.3.3. Determination of $\partial \mathbf{R}/\partial \mathbf{p}$

The porosities were defined to lower the velocities in the equilibrium distribution. Hence the only occurrence of the porosity in the residuum function (equation 5.15) is in the calculation of this equilibrium distribution. For the index i running over all cells this leads to:

$$\frac{\partial \mathbf{R}(\mathbf{p}(\mathbf{s}), \mathbf{f})}{\partial \mathbf{p}} = \frac{\partial \mathbf{M}(\mathbf{p}(\mathbf{s}), \mathbf{f})}{\partial \mathbf{p}} - \frac{\partial \mathbf{f}}{\partial \mathbf{p}} = \frac{\partial \mathbf{\Pi}(\mathbf{\Omega}(f_\alpha))}{\partial p_i} = \frac{\partial \tilde{f}_\alpha}{\partial p_i} = \omega \frac{\partial f_\alpha^{eq}}{\partial p_i} \tag{5.47}$$

Performing this derivative a matrix will emerge with $n \cdot (n \cdot n\_o\_distributions)$ entries. Because only u and v are functions of the porosities this results in:

$$\frac{\partial f_\alpha^{eq}}{\partial p_i} = \frac{\partial f_\alpha^{eq}}{\partial u} \frac{\partial u}{\partial p} + \frac{\partial f_\alpha^{eq}}{\partial v} \frac{\partial v}{\partial p} \tag{5.48}$$

where

$$\frac{\partial u}{\partial p} = \frac{\partial \tilde{u}(1 - p^\kappa)}{\partial p} = -\tilde{u} \kappa p^{\kappa - 1} \tag{5.49}$$

and

$$\frac{\partial v}{\partial p} = -\tilde{v} \kappa p^{\kappa - 1} \tag{5.50}$$

Since $\partial f_\alpha^{eq}/\partial u$ and $\partial f_\alpha^{eq}/\partial v$ were discussed earlier (Chapter 5.3.1), the given derivatives can be combined.
One has to pay attention to the fact that the propagation operator is still active. Thus, the calculated entries have to be set into the correct row.

If the design values are the porosities or vertex values, a high amount of optimization steps will be anticipated for the geometry to develop. This is due to the fact that in cells with a porosity of zero, also the gradient $\frac{\partial \mathbf{u}}{\partial p}$ is zero (compare equations above). Therefore, only the aforementioned filtering method (vertex values as design variables) enables the geometry to grow into unoccupied regions next to porous sites. Hence, this growth has a maximal speed of one cell per optimization step.

5.3.4. Determination of $\partial \mathbf{p}/\partial \mathbf{s}$

This derivative depends on the chosen design variable. In the current work three different kinds of design variables are used, each of them with a different influence on the porosities. Therefore, the derivative of the porosities with respect to the particular design variable(s) have to be determined:

1. If the design variables s are just the porosities, the derivative is just 1 (leading to the identity matrix $\mathbf{I}$).
2. If the design variables are chosen to be the filtering method discussed in Chapter 5.2.1, the derivative will be:

   $$\frac{\partial \mathbf{p}(\mathbf{s})}{\partial \mathbf{s}} = \left[\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}\right] \tag{5.51}$$

   for the $s_{x,y}$, $s_{x-\Delta x,y}$, $s_{x,y-\Delta y}$ and $s_{x-\Delta x,y-\Delta y}$ (surrounding vertex values) of each cell and 0 for every other design variable. Therefore, it is a quadratic matrix with $(n \cdot n)$ entries (n number of cells).

3. If the design variable is the radius of the cylinder (see Chapter 4.3), the derivative will result by deriving the smoothed Heaviside function (equation 4.7):

   $$\frac{\partial p(s)}{\partial s} = \begin{cases} -\frac{\pi}{4} cos(\frac{\pi}{2} d) & \text{if } d < 1/2 \cdot \text{smoothing\_region} \\ 0 & \text{otherwise} \end{cases} \tag{5.52}$$

5.3.5. Determination of $\partial F/\partial \mathbf{p}$

Again the chosen performance functional F affects the derivative (see Chapter 5.2.2 for used functions).

For the overall pressure difference as performance functional, this derivative is zero since it doesn't depend on any porosity.

If the performance functional is the drag force or lift force it yields (no Einstein summation convention):

$$\frac{\partial F_D}{\partial p_i} = \omega \kappa p_i^{\kappa-1} \rho_i \tilde{u}_i \tag{5.53}$$

and

$$\frac{\partial F_L}{\partial p_i} = \omega \kappa p_i^{\kappa-1} \rho_i \tilde{v}_i \tag{5.54}$$

5.4. Optimization algorithm

The general optimization procedure was described and depicted in Chapter 5.1. But it is still to be clarified how the calculated gradient is used to determine the new design variables and additionally it is to define stopping criteria of the optimization loop.

New design variables have to be received in each optimization step. In order to achieve this, a search direction for these variables has to be defined. Two options shall be briefly described and implemented here. The reader is referred to [9] for more information about these methods.

At first the simplest gradient based optimization procedure is used: The steepest descent method. As its name implies, this method searches new design variables in the direction of the steepest descent. Therefore, for a minimization this is just the negative gradient $-\frac{d\mathbf{L}}{d\mathbf{s}}$, or for a maximization the positive one.

For the conjugated gradient method a varying optimization gradient is chosen. Every new search direction is chosen in a way that it is conjugated to all previous gradients. Only a few additions have to be made to the algorithm described in Chapter 5.1. Figure 5.5 shows those two additions. The content of the upper box has to be added after the first calculation of the adjoint gradient and the content of the second box has to be performed each time before "applying the optimization algorithm to evaluate new design variables".

After determining the optimization gradient through one of above methods, a step length α has to be chosen. The gradient is multiplied with this step length to be added to the old design variables:

$$s_{i,new} = s_{i,old} + \alpha \xi_i \tag{5.55}$$

The step length should be chosen carefully. A step length being too high causes the leaving of the approximately linear area around the current stance. Whereas if the chosen step length is too small, the optimization algorithm would be slow to progress. Therefore, some conditions were developed to minimize these risks. The so called line search methods are about choosing a search direction and the step length. Since here, the backtracking condition is used as line search method, only one condition add to initialization: $\xi = \frac{d\mathbf{L}}{d\mathbf{s}}$, $\nabla = -\xi$, $h = \nabla$, $\xi = h$

calculate conjugated gradient

$$\xi = \frac{d\mathbf{L}}{d\mathbf{s}}$$
$$\gamma = \frac{(\xi + \nabla)\xi}{\nabla\nabla}$$
$$\nabla = -\xi$$
$$h = \nabla + \gamma h$$
$$\xi = h$$

Figure 5.5: the conjugated gradient method (compare [6]); only extensions to algorithm in Chapter 5.1 are depicted

has to be checked. This method is depicted in Figure 5.6. The satisfaction of the aforementioned condition then has to be proven through a new flow calculation. The condition is:

$$\mathbf{L}(\mathbf{s} + \alpha\boldsymbol{\xi}) \leq \mathbf{L}(\mathbf{s}) + c_1\alpha\left(\boldsymbol{\xi} \cdot \frac{d\mathbf{L}}{d\mathbf{s}}\right) \tag{5.56}$$

The factor c_1 has to be chosen by trying out some values. Here it is set to 0.5 at first. If this condition is not fulfilled, α has to get a new, smaller value. Here this is done by:

$$\alpha_{new} = \frac{1}{2}\alpha_{old} \tag{5.57}$$

For the design variables, being the filtered porosities (see Chapter 5.2.1), the initial α is set through to the fact that $\alpha\xi_i$ defines the updating value of the design variables (equation 5.55). Hence, an appropriate updating value can be estimated. At first the entry with the absolute maximum of the gradient vector is determined. Then the product $\alpha\xi_{max}$ is set to an appropriate value, for example 0.7. This then can be solved for the initial step length $\alpha_{initial}$. The initial step length generally should be chosen in a way that the aforementioned condition scarcely is undershot, therefore, the initial step length is chosen to start with a high value and from there on will be lowered until the condition is fulfilled.

The first stopping criterion of the optimization loop is that the L2-norm of the gradient has fallen under a fraction of the starting L2-norm of the gradient. Additionally the algorithm will be stopped, if the line search method has looped more than a couple of times without finding an appropriate step length.

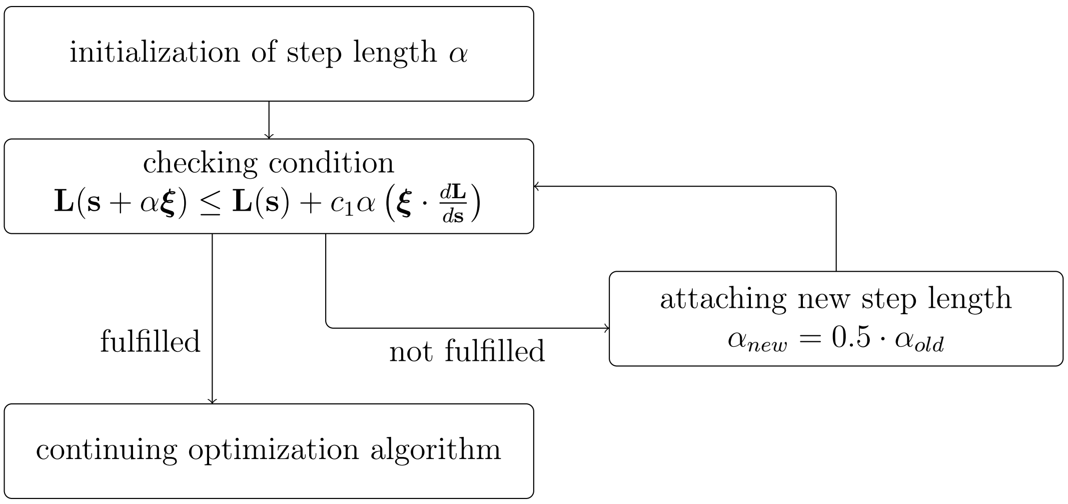

initialization of step length α
checking condition
$\mathbf{L}(\mathbf{s}+\alpha\boldsymbol{\xi}) \leq \mathbf{L}(\mathbf{s}) + c_1\alpha\left(\boldsymbol{\xi}\cdot\frac{d\mathbf{L}}{d\mathbf{s}}\right)$
attaching new step length
$\alpha_{new} = 0.5\cdot\alpha_{old}$
fulfilled
not fulfilled
continuing optimization algorithm

Figure 5.6: the line search method "backtracking"

Attention has to be paid to the fact that for the vertex values being the design variables, values smaller than zero or higher than one can lead to porosities which are not in the feasible region ($0 \leq \mathbf{p} \leq 1$). Several methods could be used to inhibit this behavior. Here just a limiting method is applied. Two options arise to conduct this: Either the vertex values or the porosities resulting out of these values can be limited to the feasible and physical region. This means: If a porosity or vertex value exceeds one or is under zero, it just will be set back to the limit which will be violated. This method is called clipping.
It remains to be seen which of both parameters have to be clipped to achieve the best results in optimization, and this will, therefore, be a topic in Chapter 5.6.

5.5. Validation of gradient

Before the optimization algorithm can be used for its purpose, its validity has to be checked. For this purpose the analytically derived gradient will be compared to a discrete gradient. This discrete gradient is determined with an approximation by a first order forward finite difference. With varying perturbations ϵ it is:

$$\frac{dF}{ds} = \frac{F(s) - F(s\cdot(1+\epsilon))}{s - (s\cdot(1+\epsilon))} \tag{5.58}$$

Hereby a performance functional F, the pressure from inlet to outlet is chosen and for the design parameter, the diameter is selected. Since the steady state condition will hold, $\frac{dF}{ds}$ can be compared to $\frac{dL}{ds}$. As geometry the test case containing a cylinder in channel flow, described in Chapter 3, is used again, as well as the porosity scaling and boundary smoothing out of Chapter 4. The Mach number is a chosen constant and the Reynolds number varies around 20 according to the perturbation. The results can be seen in Figure 5.7. For high perturbations, the discrete gradient was expected

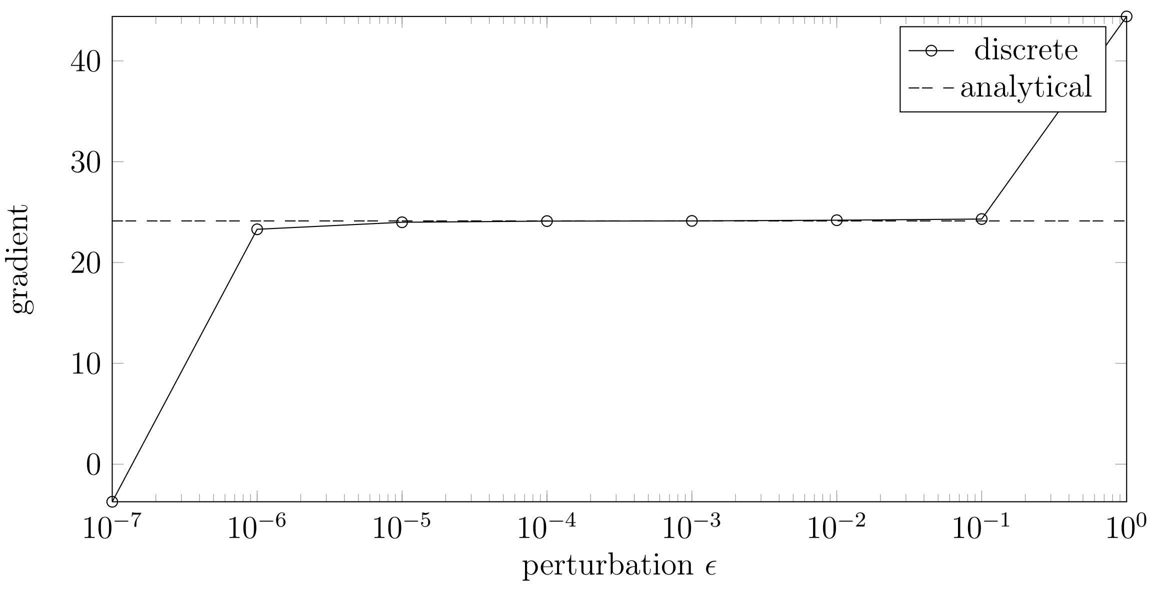


Figure 5.7: comparison of discrete and analytical gradient

to differ from the analytical. This fact can be explained by the behavior of the performance functional, since it is only approximately linear for small stencils. Very low perturbations on the other hand, lead to a difference which can be explained by numerical errors due to limited decimal places. Most important is that the gradients agree well for medium perturbations. It follows that the analytical gradient can be used for the optimization algorithm.

In fact this method of validating the gradient by a discrete approximation has to be done every time the performance functional or the design variables change. This makes sure that their derivatives are implemented in the simulation code correctly. E.g. if the design variables are the filtered porosity values (see Chapter 5.2.2), a single design variable will be changed slightly for obtaining the discrete gradient. Therefore, this validation is done for all cases of application without mentioning it explicitly.

5.6. Cases of application

After implementing the optimization algorithm, it should be applied to some cases of applications below. The design variables are now chosen to be the variables implemented at the vertices for all the following optimizations (compare Chapter 5.2.1).

5.6.1. Channel domain

Clipping method

It still has to be shown which clipping method discussed in Chapter 5.4 is more efficient to avoid the leaving of the physical area ($0 \leq p \leq 1$). In order to investigate this, an optimization with both the channel domain and the cylinder as starting geometry was chosen. The aim of the algorithm is to minimize the performance functional. This is defined as the difference between drag and lift force:

$$F(\mathbf{p}(\mathbf{s}), \mathbf{f}) = F_D - F_L = \sum_{i}^{number\,of\,cells} (\omega p_i^{\kappa} \rho_i \tilde{u}_i) - \sum_{i}^{number\,of\,cells} (\omega p_i^{\kappa} \rho_i \tilde{v}_i) \qquad (5.59)$$

The change of the Lagrangian value during optimization is shown in Figure 5.8. It shows a faster and longer optimization process for the clipping of the porosities. This can be explained with the fact that just limiting the porosities gives the optimization algorithm more degrees of freedom in varying the design variables, leading to better results. The Lagrangian value can still be compared with the value of the performance functional, since the steady state constraint will hold.

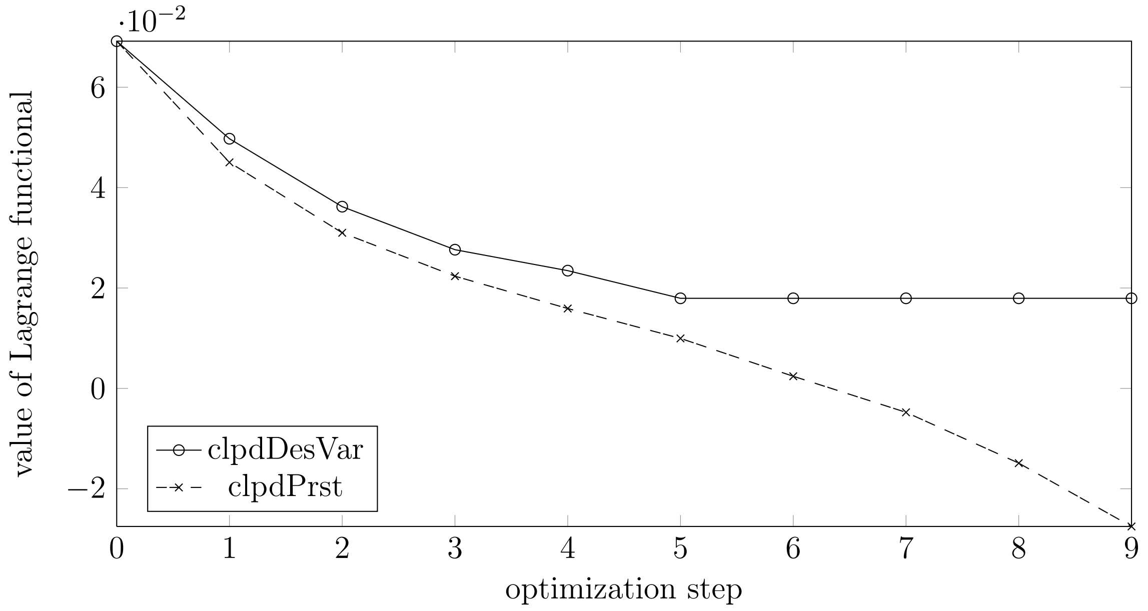


Figure 5.8: comparison of vertex value and porosity clipping

5.6.2. Periodic Domain

To eliminate the influence of the walls in the channel flow domain, a second domain was introduced in Chapter 3. By the use of periodic boundary conditions instead of walls, this domain represents an infinite cascade of the design geometry. This domain will be used in the following investigations.

Reference optimization

Firstly a simple optimization with a cylinder as starting geometry and the difference between drag and lift force as the performance functional was performed. The results are presented in Figures 5.9 and 5.10. The algorithm works as expected as the Lagrangian value (and the geometry changes in a way that it can redirect the flow to create lift and in the same time has a thin contour for low drag.

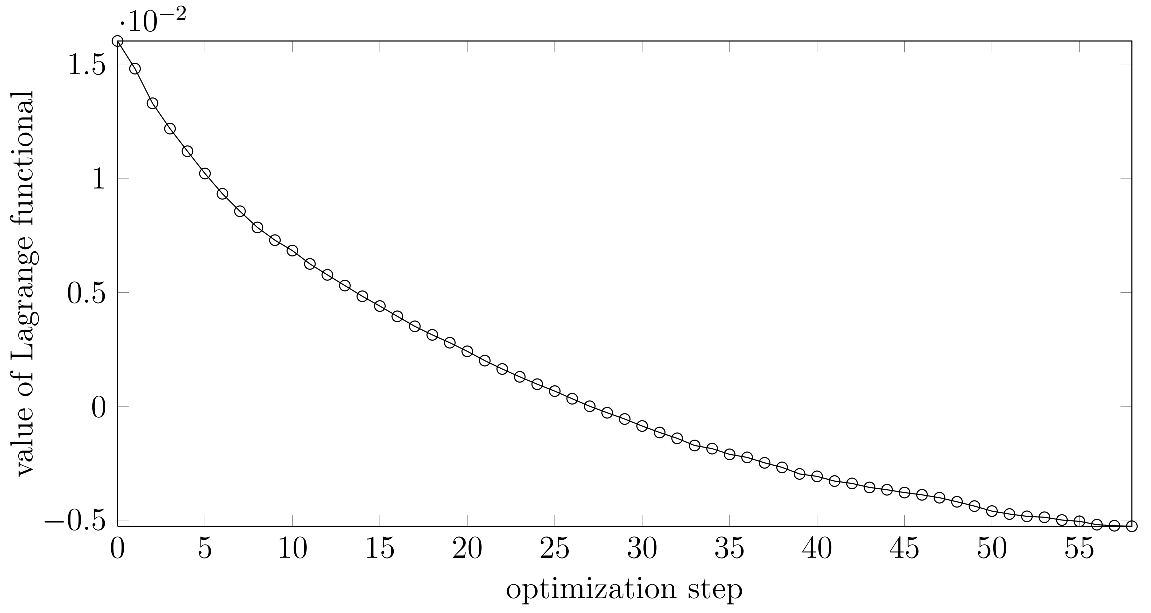


Figure 5.9: optimization progress with periodic Domain and cylinder as starting geometry

Penalty term

To influence the geometry development, a penalty term can be implemented. Just to test this possibility, a simple penalty term is implemented in the performance

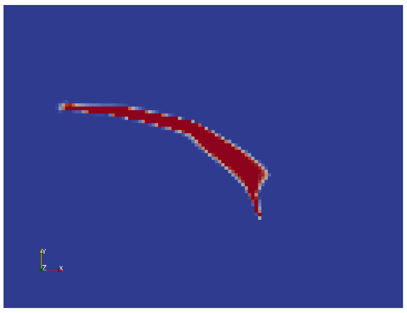


Figure 5.10: final geometry (step 58) of the reference optimization with the periodic Domain and the cylinder as starting geometry

function:

$$F(\mathbf{p}(\mathbf{s}),\mathbf{f}) = F_D - F_L - penaltyTerm = \sum_{i}^{number\,of\,cells} (\omega p_i^{\kappa}\rho_i\tilde{u}_i) - \sum_{i}^{number\,of\,cells} (\omega p_i^{\kappa}\rho_i\tilde{v}_i) - \frac{1}{800000}p\cdot\left(\frac{d_i}{r}-1\right) \tag{5.60}$$

with d_i being the distance of a cell to the center of the starting cylinder and r being the radius of the cylinder. The prefactor $\frac{1}{800000}$ is chosen in a way that the penalty term is at the start about one sixteenth of the magnitude of the original performance value (difference of drag and lift force). This term shall penalize porosity which emerges outside the original starting geometry and reward geometry inside it. In this way the growth shall be limited. Figures 5.11 and 5.12 illustrate that this seems to work. Certainly, this is just to prove that penalty terms can be implemented. More expedient terms can be found (like quadratic terms). Moreover constraints could be added through slack variables and an additional Lagrange multiplier.

Change in starting geometry

Furthermore, it would be interesting to know if the starting geometry has an effect on the final geometry under otherwise the same conditions. Therefore, an optimization with the same options as in the reference case was executed. As starting geometry serves a triangle. It is defined to be the upper left half of a square. This square has the same area as the previous cylinder. Figure 5.13 depicts the progression of the Lagrangian minimization. The effect of the optimization on the geometry is shown in Figure 5.14. This investigation displays the fact that, independently from the starting geometry, the optimization algorithm seems to asymptotically converge to the same result. The comparison to the reference case (between the Lagrangian progression and the geometries), by considering the respective figures, illustrates this. Differences can be explained by canceling the calculation through the stopping criteria.

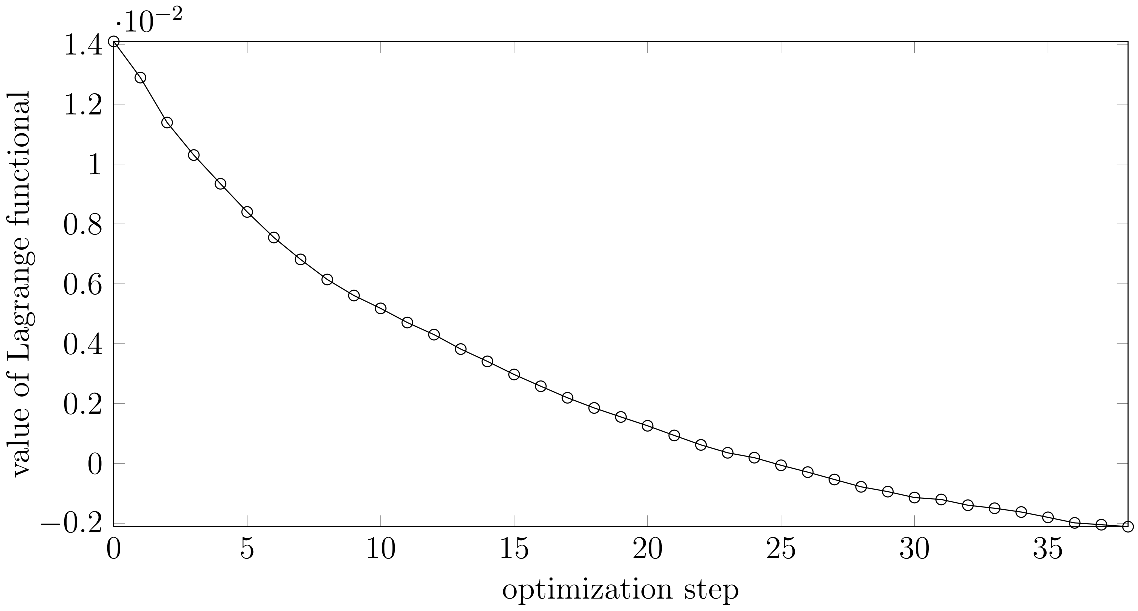


Figure 5.11: optimization progress with cylinder and penalty term

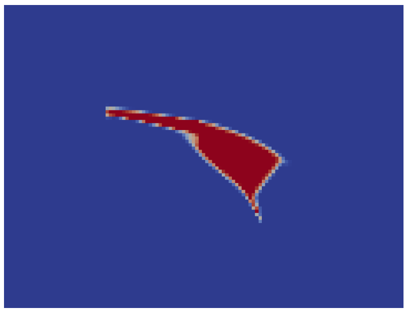


Figure 5.12: final geometry with penalty term

The stopping criteria and optimization method

In this last investigation two questions shall be answered. Firstly the question of which optimization method, steepest descent or conjugated gradients (see Chapter 5.4), is better for the current optimization problem. And secondly: How long will the optimization go on with loose stopping criteria?
Therefore, the stopping criteria were intensely loosened. Again, the triangle is the starting geometry and the performance functional is the difference between drag and lift force. Figure 5.15 shows that the conjugated gradient method performs a slower progression in the first optimization steps. Whereas it clearly outpaces the steepest descent method at a higher number of optimization steps. It also leads to a more optimal Lagrangian value (lower value) than the steepest descent method, even in a lower amount of optimizations steps. The stopping criteria and line search method

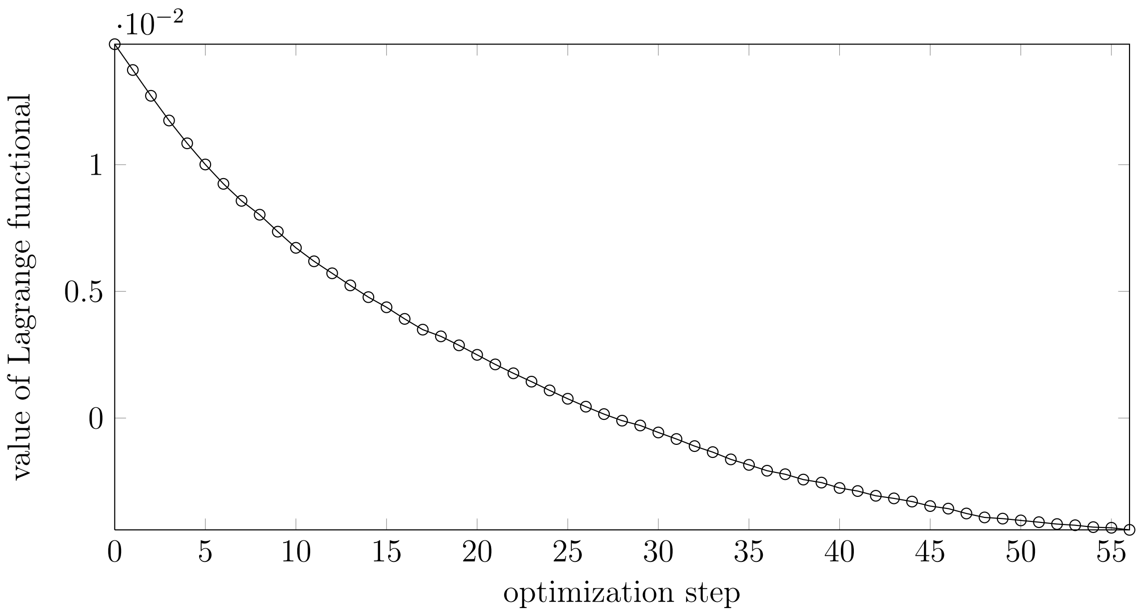


Figure 5.13: optimization progress with triangle as starting geometry

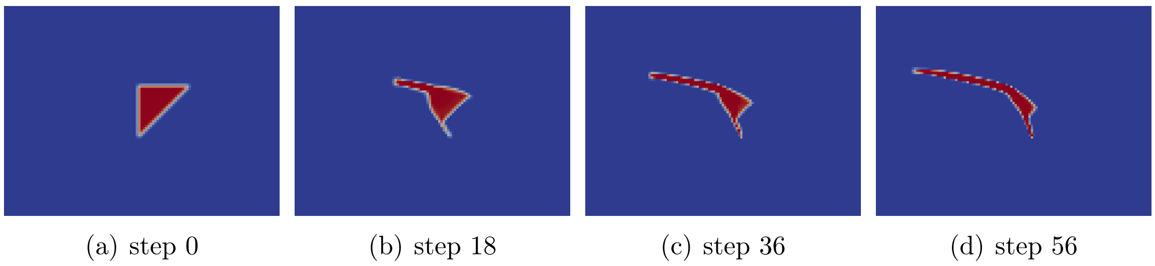


Figure 5.14: geometry change during optimization with triangle as starting geometry

were the same. Probably the used parameters of the line search method or the method itself were less efficient with the conjugated gradient method. Figure 5.16 shows the final geometry for both methods. More investigations have to be performed here.
It can also be seen that the optimization probably would go on further even though the improvements are minor. Overall the optimization algorithm shows the expected slow progression rate as discussed in Chapter 5.3.3.

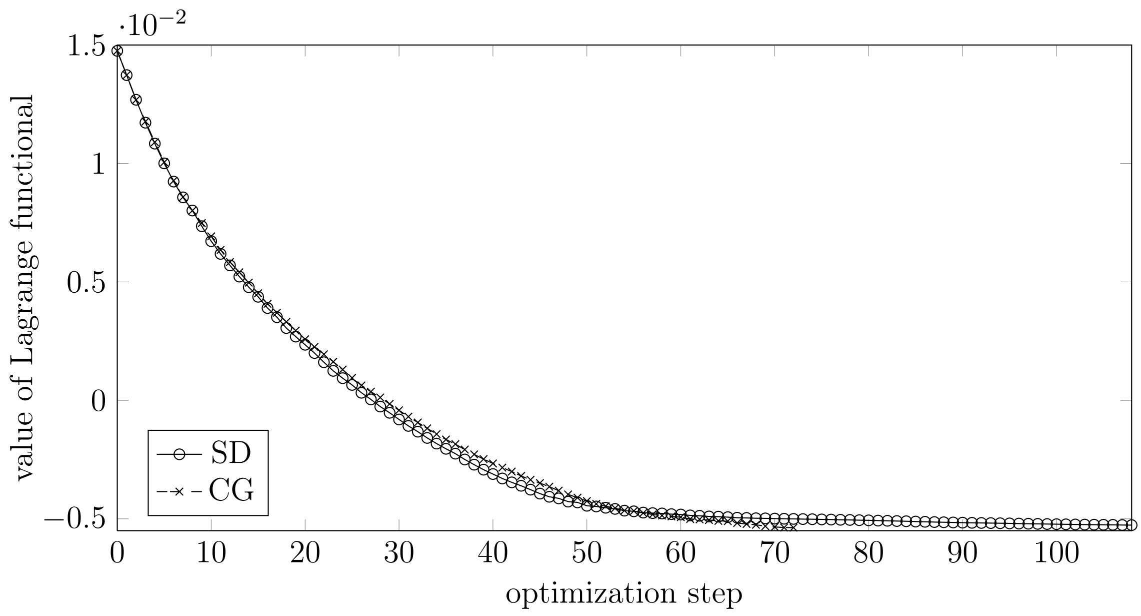
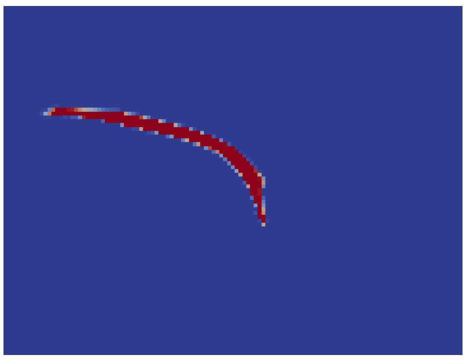


Figure 5.15: optimization progress with loose stopping criteria; comparison between steepest descent and conjugated gradient method


(a) steepest descent; step 108

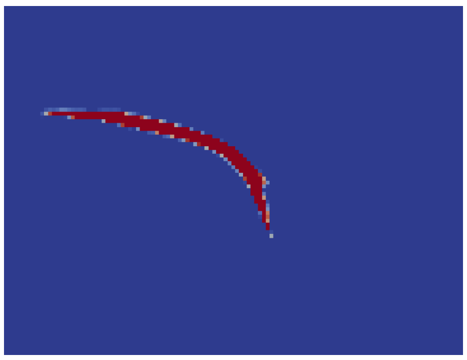

(b) conjugated gradient; step 72

Figure 5.16: comparison of final geometry

6. Summary and outlook

The aim of the project was to implement and apply an adjoint shape optimization algorithm in a Lattice-Boltzmann solver based on the work of Pingen et al. [10]. In order to achieve a continuous approach in altering a geometry for optimization, porous media were introduced, implemented and validated. Furthermore, an adjoint optimization algorithm was derived, validated and used. This algorithm was then successfully applied to a few cases of application. Thereby a filtering method was used to slightly delocalize the optimization effect. In order to achieve this, new parameters at the cells' vertices were defined. The cases of application showed that the described algorithm is able to iteratively optimize a given shape with respect to chosen performance functionals. This was respectively proven by viewing the magnitudes of the Lagrangian in the course of optimization.

Moreover the cases of application generated some more insights.
A method had to be chosen to limit porosity values to physical values. It was shown that for the aforementioned filtering method which provided the design variables, limiting the porosity values instead of the design variables appeared to be more efficient. More degrees of freedom in choosing the values for the design variables seems to be the explanation for this fact.
For the determination of new design variables in optimization, two methods have been examined. In the first comparison, the conjugated gradient method proves to be less effective during the first optimization steps, but it clearly outpaces the steepest descent method later on. Hence, the parameters of the line search method or the method itself, don't seem to be ideal for the conjugated gradient method to unfold its full potential. Therefore, the line search method and further algorithms have to be examined in more detail.

A minor improvement in the assembling of the optimization algorithm was introduced in the current thesis. This improvement emerged through the prevention of a costly matrix-matrix multiplication which was used originally.
Furthermore, the treatment of boundary conditions was applied and described in the presented optimization algorithm to increase accuracy, which was neglected, especially for inlet and outlet cells in the original work.

There are still some matters to be considered in further works. More constraints, like mass constraints, can be tested. Additionally the algorithm could be applied for more complex, parameter described shapes like airfoil profiles. However, the major challenge and improvement would be to parallelize the programming code. Since the adjoint calculations are not parallelized, even the flow calculation had to run on a single processor. Therefore, a parallelization would lead to a distinct acceleration of the optimization algorithm. Only this would enable us to perform optimizations with vast refinements or even 3D optimizations within an acceptable amount of time.

Bibliography

[1] W. Bangerth, T. Heister, Kanschat G., and et al. The deal.ii finite element library, 2013.

[2] H. Foysi. Lecture flow control. Lecture, 2013.

[3] M. D. Gunzburger. *Perspectives in Flow Control and Optimization.* Society for Industrial and Applied Mathematics, 2003.

[4] D. Haenel. *Molekulare Gasdynamik.* Springer, 2004.

[5] D. Makhija, G. Pingen, Yang R., and K. Maute. Topology optimization of multi-component flows using a multi-relaxation time lattice boltzmann method. *Computers & Fluids 67*, 2012.

[6] D. Marinc. Analysis of adjoint based optimal control applied to noise reduction of plane jets. Dissertation, 2013.

[7] R. Mei, D. Yu, and W. Shyy. Force evaluation in the lattice boltzmann method involving curved geometry. *PHYSICAL REVIEW E, VOLUME 65*, 2002.

[8] D. A. Nield and A. Bejan. *Convection in Porous Media.* Springer, 2013.

[9] J. Nocedal and S. J. Wright. *Numerical Optimization.* Springer, 1999.

[10] G. Pingen, A. Evgrafov, and K. Maute. Adjoint parameter sensitivity analysis for the hydrodynamic lattice boltzmann method with applications to design optimization. *Computers & Fluids 38*, 2009.

[11] M. Schaefer and S. Turek. *Benchmark Computations of Laminar Flow Around a Cylinder*, pages 547–566. Vieweg+Teubner Verlag, 1996.

[12] M. A. A. Spaid and F. R. Phelan Jr. Lattice boltzmann methods for modeling microscale flow in fibrous porous media. *Physics of Fluids 9*, 1997.

[13] D. A. Wolf-Gladrow. *Lattice-Gas Cellular Automata and Lattice Boltzmann Models - An Introduction.* Springer, 2005.