

Topology optimization and lattice Boltzmann methods

Nørgaard, Sebastian Arlund; Sigmund, Ole; Lazarov, Boyan Stefanov; Engelbrecht, Kurt

Publication date:
2017

Document Version
Publisher's PDF, also known as Version of record

[Link back to DTU Orbit](#)

Citation (APA):

Nørgaard, S. A., Sigmund, O., Lazarov, B. S., & Engelbrecht, K. (2017). Topology optimization and lattice Boltzmann methods. Kgs. Lyngby: Technical University of Denmark (DTU). (DCAMM Special Report; No. S230).

DTU Library

Technical Information Center of Denmark

General rights

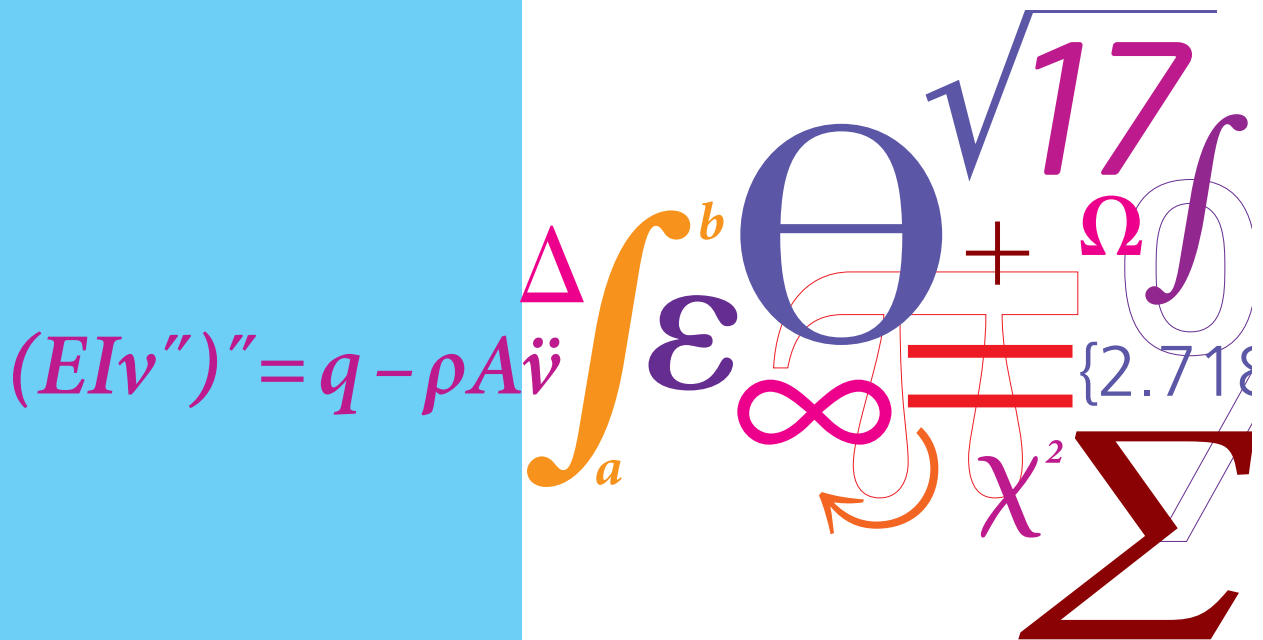
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal

If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Topology optimization and lattice Boltzmann methods

PhD Thesis



Sebastian Arlund Nørgaard
 DCAMM Special Report No. S230
 October 2017

Topology optimization and lattice Boltzmann methods

Sebastian Arlund Nørgaard

Kgs. Lyngby 2017
DCAMM Special Report No. S230

Topology optimization and lattice Boltzmann methods

October, 2017

PhD student:

Sebastian Arlund Nørgaard

Main supervisor:

Professor Ole Sigmund

Department of Mechanical Engineering, DTU

Co-supervisors:

Senior Researcher Boyan Stefanov Lazarov

Department of Mechanical Engineering, DTU

Associate Professor Kurt Engelbrecht

Department of Energy Conversion and Storage, DTU

© 2017 Sebastian Arlund Nørgaard

Technical University of Denmark

Department of Mechanical Engineering

Building 404, DK-2800 Kgs. Lyngby, Denmark

Phone: +45 45 25 25 25, Fax: +45 45 25 19 61

E-mail: info@mek.dtu.dk, URL: www.mek.dtu.dk

MEK-PHD: ISSN 0903-1685, ISBN 978-87-7475-506-7

This one is for Elena, my Sunday smile.

Preface

This thesis is submitted as part of the requirements for obtaining the degree of Ph.D. from the Technical University of Denmark (DTU). The work was carried out between September 1, 2014 and October 31, 2017, at the Department of Mechanical Engineering, DTU. The supervisors on the project were Professor dr.techn. Ole Sigmund, Senior Researcher Boyan Stefanov Lazarov (both DTU Mechanical Engineering), and Associate Professor Kurt Engelbrecht (DTU Energy). The research was conducted as part of the TopTen project, funded by the Danish Council for Independent Research, grant DFF-4005-00320.

In addition to the time spent at DTU Lyngby Campus, a month-long research stay was conducted in the SciComp group at TU Kaiserslautern, Germany, under the supervision of Prof. Dr. Nicolas R. Gauger.

Completing this thesis would have been impossible without the generous help and friendship offered by many kind individuals. I would like to thank my supervisors for the help and encouragement they have offered in the course of this work. I especially thank Ole Sigmund for his help and guidance in the final months of finishing this thesis, and for believing in me when I myself did not.

I would like to thank Nicolas Gauger, Max Sagebaum, Tim Albring, and my office-mate Emre Özkaya for the hospitality and great help they offered at TU Kaiserslautern.

I would like to extend my thanks to a number of my colleagues or former colleagues, who have all provided professional help or simply good company at various stages of the project: Christian Lundgaard, Jeroen Groen, Rasmus Ellebæk Christiansen, Said Zeidan, Joe Alexandersen, Anders Clausen, Erik Andreassen, Christopher Nellemann, and Niels Frandsen. A special thanks to Said Zeidan for our numerous interesting conversations, and to Rasmus Ellebæk Christiansen for the best game of Tetris that Singapore Airlines has ever seen.

I offer my sincere gratitude to my wonderful friends and family: my parents Kim and Susanne, and my siblings Alexander and Amalie, for always supporting me and being there when I need you most. I also thank all of my extended family for all the good times we have had together. I thank my close friends Jens Lilleskov Nielsen and Tue Hammer Lerche for all the good times, as well as my “homegirls”, Astrid Elme Breum, Caroline Boeg, and Peter Brylov Christensen. Finally, a special thanks to my girlfriend Elena Bossolini, who enriches my life greatly with her warmth, humor, and generosity, and without whom I could not have finished this project. You are a true treasure.

Kgs. Lyngby, October 31, 2017
Sebastian Arlund Nørgaard

Abstract

This thesis demonstrates the application of the lattice Boltzmann method for topology optimization problems. Specifically, the focus is on problems in which time-dependent flow dynamics have significant impact on the performance of the devices to be optimized. The thesis introduces new topology optimization problems for both isothermal and thermal flows, and it is demonstrated that topology optimization can account for unsteady flow effects during the optimization process.

The introduced optimization problems are solved using a gradient based approach, and the design sensitivities are computed using a discrete adjoint approach. To handle the complexity of the discrete adjoint approach more easily, a method for computing it based on automatic differentiation is introduced, which can be adapted to any lattice Boltzmann type method. For example, while it is derived in the context of an isothermal lattice Boltzmann model, it is shown that the method can be easily extended to a thermal model as well.

Finally, the predicted behavior of an optimized design is compared to the equivalent prediction from a commercial finite element solver. It is found that the weakly compressible nature of the lattice Boltzmann method leads to a discrepancy in the predicted outcomes. Further research is required to determine which prediction is more accurate, and what implications the discrepancy has for the optimized designs.

Resumé

Denne afhandling demonstrerer anvendelsen af lattice Boltzmann metoden til topologioptimeringsproblemer. Mere specifikt er fokus på problemer for hvilke tidsafhængige væskestrømninger er betydningsfulde for ydeevnen af de apparater der skal optimeres. Afhandlingen introducerer nye topologioptimeringsproblemer for både isotermske og termiske væskestrømninger, og det demonstreres at topologioptimering kan tage højde for tidsafhængige strømningseffekter under optimeringsprocessen.

De introducerede optimeringsproblemer løses ved hjælp af en gradient baseret fremgangsmåde, og design sensitiviteterne beregnes med en diskret adjoint metode. For nemmere at håndtere kompleksiteten af den diskrete adjoint metode, introduceres en fremgangsmåde til at beregne den baseret på automatisk differentiation, som kan tilpasses til enhver lattice Boltzmann model. For eksempel vises det at fremgangsmåden nemt kan tilpasses til en termisk lattice Boltzmann model, på trods af at den er udledt for en isotermsk model.

Endelig sammenlignes den forudsete adfærd af et optimeret design med den ækvivalente forudsigtelse fra en kommerciel finite element løser. Det konstateres at lattice Boltzmann metodens svagt kompressible natur fører til en uoverensstemmelse mellem de forudsete resultater. Yderligere forskning er påkrævet for at afgøre hvilken forudsigtelse der er mest nøjagtig, samt hvilke konsekvenser uoverensstemmelsen har for de optimerede design.

List of publications

The following publications are submitted as part of this thesis.

- [P1] Sebastian Nørgaard, Ole Sigmund, and Boyan Lazarov. *Topology optimization of unsteady flow problems using the lattice Boltzmann method*. Journal of Computational Physics, 307:291–307, 2016.
- [P2] Sebastian A. Nørgaard, Max Sagebaum, Nicolas R. Gauger, and Boyan S. Lazarov. *Applications of automatic differentiation in topology optimization*. Structural and Multidisciplinary Optimization, 56(5):1135–1146, 2017.

Contents

| | |
|---|------------|
| Preface | i |
| Abstract | iii |
| Resumé | iv |
| List of publications | v |
| Contents | vii |
| 1 Introduction | 1 |
| 1.1 Motivation and goal | 1 |
| 1.2 Structure | 1 |
| 1.3 A readers guide | 2 |
| 2 Physical theory | 3 |
| 2.1 Continuum fluid dynamics | 3 |
| 2.1.1 Governing equations | 4 |
| 2.1.2 Dimensionless numbers | 5 |
| 2.1.3 Non-dimensionalization of the governing equations | 6 |
| 2.2 Kinetic theory | 7 |
| 2.2.1 The particle distribution function | 7 |
| 2.2.2 The Boltzmann equation | 8 |
| 3 The lattice Boltzmann method | 11 |
| 3.1 Introduction | 11 |
| 3.2 Velocity discretization | 12 |
| 3.2.1 The D2Q9 lattice | 13 |
| 3.3 The lattice Boltzmann equation | 14 |
| 3.4 Collision operators | 14 |
| 3.4.1 BGK operator | 15 |
| 3.4.2 MRT operator | 15 |

| | | |
|----------|--|-----------|
| 3.4.3 | Other collision operators | 16 |
| 3.5 | Implementation | 16 |
| 3.6 | Boundary conditions | 17 |
| 3.6.1 | No-slip boundaries | 18 |
| 3.6.2 | Velocity and pressure boundaries | 19 |
| 3.7 | Non-dimensionalization of lattice Boltzmann | 20 |
| 3.8 | Accuracy of the lattice Boltzmann method | 20 |
| 4 | Topology optimization | 23 |
| 4.1 | Introduction | 23 |
| 4.2 | Optimization problem | 25 |
| 4.3 | Sensitivity analysis | 26 |
| 4.4 | Filtering of the design variables | 27 |
| 4.4.1 | Projection filter | 27 |
| 4.4.2 | Robust formulation | 28 |
| 4.5 | Solving the optimization problem | 28 |
| 5 | Unsteady flow topology optimization [P1] | 31 |
| 5.1 | Lattice Boltzmann for porous media | 31 |
| 5.2 | The unsteady optimization problem | 33 |
| 5.3 | Obstacle flow control | 33 |
| 5.3.1 | Problem description | 33 |
| 5.3.2 | Summary of results | 34 |
| 5.3.3 | Discussion | 35 |
| 5.4 | Passive fluid pump | 35 |
| 5.4.1 | Problem description | 35 |
| 5.4.2 | Summary of results | 36 |
| 5.4.3 | Discussion | 37 |
| 5.5 | Extension of the pump problem | 39 |
| 5.5.1 | Simulation setup | 39 |
| 5.5.2 | Robust optimization of the pump | 40 |
| 5.5.3 | Pumping against an external force | 41 |
| 5.5.4 | Comparison to a finite element solution | 44 |
| 6 | Topology optimization with automatic differentiation [P2] | 47 |
| 6.1 | Automatic differentiation by example | 47 |
| 6.1.1 | Forward mode | 48 |
| 6.1.2 | Reverse mode | 48 |
| 6.1.3 | Closing remark | 49 |
| 6.2 | Adjoint method with automatic differentiation | 49 |
| 6.2.1 | Main result | 50 |
| 6.2.2 | A note on adjoint streaming | 51 |
| 6.2.3 | Checkpointing | 51 |

| | | |
|----------|---|-----------|
| 6.3 | Adjoint boundary conditions | 52 |
| 6.4 | Discussion | 53 |
| 7 | Unsteady thermal flow topology optimization | 55 |
| 7.1 | Thermal lattice Boltzmann model | 55 |
| 7.1.1 | Thermal lattice | 56 |
| 7.1.2 | Thermal collision operators | 57 |
| 7.1.3 | Thermal boundary conditions | 57 |
| 7.2 | Thermal lattice Boltzmann for topology optimization | 57 |
| 7.2.1 | Material interpolation | 58 |
| 7.2.2 | Adjoint thermal lattice Boltzmann | 58 |
| 7.3 | Regenerators | 58 |
| 7.4 | Topology optimization of regenerators | 59 |
| 7.4.1 | Simulation setup | 59 |
| 7.4.2 | Objective function | 61 |
| 7.4.3 | Results | 62 |
| 7.5 | Robust optimization of regenerators | 63 |
| 7.5.1 | Non-uniform variations | 63 |
| 7.5.2 | Results | 65 |
| 7.6 | Discussion | 66 |
| 8 | Closing discussion | 69 |
| 8.1 | Summary | 69 |
| 8.2 | Comments on implementation | 69 |
| 8.3 | Future work | 70 |
| 8.4 | Conclusion | 70 |
| | Bibliography | 71 |
| | Publications | 81 |

1.1 Motivation and goal

Fluid devices which manipulate or control the dynamic flow behavior of a variety of fluids are ubiquitous in the modern world, with many important applications in e.g. in-door climate control, refrigeration, printing, et cetera. Many of these applications involve some sort of temperature control, an obvious example being the heat sink and/or fans present in laptops. Due to the highly non-linear nature of fluid dynamics, many of these devices use designs which are based on engineering experience, but are not necessarily optimal.

The topology optimization method is capable of generating optimized designs without necessarily needing to start from a well defined initial design and optimizing from there. The method has been applied to fluid dynamics design problems in numerous works, but the majority of these exclusively focus on optimizing for steady state fluid flow. Many fluid systems of engineering interest are dynamic in time, and thus exhibit unsteady flow behavior. This cannot be accounted for when using steady state optimization. The available literature on unsteady flow optimization is relatively sparse, however, likely due to the high computational cost of simulating these systems.

The lattice Boltzmann method is an alternative method of simulating fluid flows, which is based on kinetic theory. It is an explicit computational scheme which is well suited for parallel execution, meaning it can potentially simulate unsteady flow systems more efficiently than traditional fluid flow solvers. The goal of this thesis is to apply the lattice Boltzmann method to unsteady fluid flow topology optimization problems, starting with problems without temperature dependence and later extending to the more complicated case of thermal problems. The parallel implementation of the method allows solving optimization problems at a larger scale than has previously been presented in the existing literature on unsteady flow topology optimization, potentially paving the way for the discovery of designs which perform better than existing, non-optimized variants.

1.2 Structure

The document is structured as follows: chapter 2–4 are theoretical, covering the basic physical theory, the lattice Boltzmann method, and topology optimization, respectively. These chapters are included to make the text self-contained, and can be skimmed or skipped if the reader is already familiar with their respective topics. The novel scientific contributions of the present work are presented in chapter 5–7, which cover topology optimization for unsteady isothermal flow problems, the application of

automatic differentiation for computing the design sensitivities, and unsteady thermal flow problems, respectively. Chapter 5 and 6 summarize the findings of the two papers published as part of this work, [P1] and [P2], respectively. In addition, chapter 5 documents some extensions of the work presented in [P1], which are unpublished. Meanwhile, chapter 7 is composed wholly of unpublished material. Finally, chapter 8 summarizes the findings presented in the thesis and offers some additional discussion, and suggestions for future work.

1.3 A readers guide

This thesis covers a broad range of scientific topics, each deep enough that entire books could be (and have been) written about them. It has been the goal of the author to make the text reasonably self-contained, but since no-one would want to read a thesis which is 500 pages long, the coverage given here is by necessity not comprehensive.

Because of this, each chapter begins with a short overview of the material covered, and—just as importantly—the material which will *not* be covered; in both cases references will be provided so that the interested reader might pursue further details. The text assumes that the reader has some degree of familiarity with partial differential equations, optimization theory, numerical methods, and linear algebra. Knowledge of computational fluid dynamics would also be advantageous.

The goal of this chapter is to cover the physical concepts needed for further development, as well as the partial differential equations (PDEs) which govern the dynamics of the physical systems of interest. The purpose is not to go into the details of the derivation of these equations, but merely state them for future reference. The chapter will cover two major topics: continuum fluid dynamics, covered in section 2.1, and kinetic theory, covered in section 2.2. The material on continuum fluid dynamics is based on [1, 2], while the material on kinetic theory is based on [3–5]. These sources contain much more extensive details on the equations given below, including derivations, should this be of interest to the reader.

In addition to the core PDEs of continuum fluid dynamics, section 2.1 also covers non-dimensionalization of said PDEs, as well as the essential non-dimensional numbers associated with this process. It does not cover any of the conventional methods for solving the governing equations numerically, such as the finite difference method [6], the finite element method [7–9] or the finite volume method [10, 11]. Since the lattice Boltzmann method derives from the kinetic theory of gases, this document would be incomplete if some exposition of this topic was not given. Despite this, it should be strongly emphasized that the lattice Boltzmann method—for the purposes of this thesis—is a means to an end, namely the efficient simulation of *continuum* dynamics. Therefore, the section on kinetic theory is kept brief, covering only the key ideas and equations necessary to present the lattice Boltzmann method in a self-contained manner. Readers interested in an in-depth study of kinetic theory may consult the references given in the first paragraph above, as well as [12, 13].

As stated above, the end goal is to simulate continuum dynamics by means of a method which derives from kinetic theory. This implies that there is some connection between the two physical models. A common mathematical technique for linking them is the so-called Chapman-Enskog expansion [14–16], which is a type of asymptotic expansion. In the modern literature, this technique is often applied directly to the discretized lattice Boltzmann equation, and results derived from this application will be discussed in chapter 3. That said, the method was originally developed to analyze the continuous Boltzmann equation directly. For details on this, the reader is referred to [17].

2.1 Continuum fluid dynamics

The continuum treatment of fluids approximates the fluid medium to be modelled as an infinitely fine collection of fluid particles. This view fits well with our intuitive, everyday experience of fluids. Despite the modern understanding that fluids are, in fact, comprised of a finite number of discrete particles, i.e. atoms and molecules,

the continuum model is widely applicable across numerous engineering disciplines. Generally speaking, the continuum model is valid if the smallest fluid volume of interest—which may be small compared to the overall system size—is still very large compared to the size of individual fluid particles. To put this in perspective, a 0.05 mL droplet of water contains of order 1.5 sextillion ($= 1.5 \times 10^{21}$) water molecules; indeed, the miniscule size of atomic particles means that the continuum model remains a very good approximation even for system sizes at the μm level.

2.1.1 Governing equations

The continuum model allows the fluid to be modelled in terms of macroscopic quantities: the density ρ , the pressure p , the velocity \mathbf{u} , and if needed, the temperature T . The temporal and spatial variation of these quantities are modelled using a set of PDEs.

The first PDE is the *continuity equation*, which expresses the conservation of mass:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0. \quad (2.1)$$

A large class of fluidic systems can be modelled as *incompressible*, meaning that the density ρ is constant for all times and spatial positions. In this case, (2.1) simplifies to

$$\nabla \cdot \mathbf{u} = 0. \quad (2.2)$$

Compressible fluid dynamics is in itself a huge area of study, and shall not be covered in detail in this thesis. Nonetheless, it should be noted at this stage that the lattice Boltzmann method is *weakly compressible*, which means that in practice one cannot use it to simulate a fluid with ρ being truly constant. More details on this will be given in chapter 3.

The second PDE is the celebrated Navier-Stokes equation, which expresses the conservation of momentum. Here, it is given only in its incompressible form relevant to this thesis:

$$\rho \left(\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla) \mathbf{u} \right) = -\nabla p + \mu \nabla^2 \mathbf{u}, \quad (2.3)$$

where μ is the fluid *viscosity*. Given appropriate initial and boundary conditions, (2.2) and (2.3) form a solvable system of equations for the dynamics of p and \mathbf{u} . However, if the density ρ is not constant—as in the lattice Boltzmann method—an additional equation is required. This requirement is fulfilled by the *equation of state*, which relates the pressure p to the density ρ and temperature T . The most well-known equation of state is the *ideal gas law*, given by

$$p = \rho RT, \quad (2.4)$$

where R is the *ideal gas constant*.

In the case where the fluid system to be modeled is not of uniform temperature (isothermal), an additional equation is needed to solve for the temperature T . The simplest such equation is the temperature *advection-diffusion* equation:

$$\frac{\partial T}{\partial t} + \nabla \cdot (\mathbf{u}T) = D\nabla^2 T, \quad (2.5)$$

where D is the temperature *diffusivity*. Like the continuity and Navier-Stokes equations, (2.5) expresses a conservation law, in this case the conservation of energy. The validity of (2.5) rests on the assumption that temperature gradients do not significantly influence the velocity field \mathbf{u} . In this type of flow—referred to as a *forced convection* flow—there is only a one-way coupling between the velocity field and the temperature field; that is, the velocity field affects the dynamics of the temperature field, but the converse is not true.

2.1.2 Dimensionless numbers

This section will briefly introduce the various dimensionless numbers relevant for the characterization of the fluid flow simulations presented later. An in-depth discussion of the flow characteristics at various values of these numbers are beyond the scope of this thesis.

As discussed, the continuum model makes the assumption that the relevant fluid domain sizes are large compared to the size of the individual fluid particles. Following the exposition by Krüger *et al.* [5], this is formalized by considering the following hierarchy of length scales:

1. The size of the fluid atom or molecule, L_a .
2. The mean distance traveled by fluid particles between successive particle collision, also called the mean free path, L_{mfp} .
3. The typical length scale of gradients for relevant macroscopic properties, L_0 .
4. The system size, L_S .

These length scales have the typical ordering $L_a \ll L_{\text{mfp}} \ll L_0 \leq L_S$. However, for very small scale system sizes, it is feasible that $L_{\text{mfp}} \sim L_0$. The ratio between these two length scale defines the *Knudsen number*:

$$\text{Kn} = \frac{L_{\text{mfp}}}{L_0}. \quad (2.6)$$

For the continuum model to be accurate, it is required that $\text{Kn} \ll 1$; conversely, for $\text{Kn} \geq 1$, a more fine grained model of the fluid is appropriate. This model is provided by kinetic theory, which is the subject of section 2.2.

Another assumption made by equations (2.2)–(2.3) is that the fluid is incompressible. For this to be a good approximation, compression waves in the fluid must

travel much faster than the bulk advection in the fluid. The *Mach number* provides a measure of this. It is given by

$$\text{Ma} = \frac{|\mathbf{u}_0|}{c_s}, \quad (2.7)$$

where $|\mathbf{u}_0|$ is the characteristic velocity magnitude for the flow, and c_s is the speed of sound. In practice, the incompressibility assumption is usually valid if $\text{Ma} \leq 0.1$.

While the Knudsen and Mach numbers provide some measure of the accuracy of the simplifying assumptions underlying the Navier-Stokes equations, there are other factors which influence the actual dynamics described by the Navier-Stokes equations. One important such factor to consider is the well-known *Reynolds number*, which is defined as the ratio of inertial to viscous forces acting in the fluid. It is given by

$$\text{Re} = \frac{|\mathbf{u}_0|L_0}{\nu}, \quad (2.8)$$

where $\nu = \mu/\rho$ is the *kinematic viscosity*. The Reynolds number is important because the dynamics of moving fluids vary wildly at different Reynolds numbers. At low Reynolds number, the flow is *laminar*, which is characterized by smooth, slowly varying fluid motion. Meanwhile, high Reynolds number flows are *turbulent*, characterized by rapidly varying—even chaotic—flow behavior.

In the case of thermal fluid flow, the *Prandtl number* is also relevant. It is defined as the ratio of momentum to thermal diffusivity, i.e.

$$\text{Pr} = \frac{\nu}{D}. \quad (2.9)$$

As a final comment, note that the choice of characteristic length and velocity scales are somewhat arbitrary. Often, the length scale is chosen as the length of some geometrical feature in the fluid domain, while the velocity scale is chosen to be equal to the velocity at a boundary where it is known. The important part is that one must always document these choices.

2.1.3 Non-dimensionalization of the governing equations

In order to decouple numerical solutions of the Navier-Stokes equations from the arbitrary choice of units, it is common practice to put them in so-called *dimensionless*

form. This is achieved by introducing the following dimensionless quantities:

$$\tilde{\mathbf{x}} = \frac{1}{L_0} \mathbf{x}, \quad (2.10a)$$

$$\tilde{t} = \frac{1}{t_0} t, \quad (2.10b)$$

$$\tilde{\mathbf{u}} = \frac{1}{|\mathbf{u}_0|} \mathbf{u} = \frac{t_0}{L_0} \mathbf{u}, \quad (2.10c)$$

$$\tilde{p} = \frac{1}{\rho |\mathbf{u}_0|^2} p. \quad (2.10d)$$

With proper rescaling of the derivative operators, this results in the dimensionless Navier-Stokes equations, given by

$$\nabla_{\tilde{\mathbf{x}}} \cdot \tilde{\mathbf{u}} = 0, \quad (2.11)$$

$$\frac{\partial \tilde{\mathbf{u}}}{\partial \tilde{t}} + (\tilde{\mathbf{u}} \cdot \nabla_{\tilde{\mathbf{x}}}) \tilde{\mathbf{u}} = -\nabla_{\tilde{\mathbf{x}}} \tilde{p} + \frac{1}{\text{Re}} \nabla_{\tilde{\mathbf{x}}}^2 \tilde{\mathbf{u}}. \quad (2.12)$$

The advantage of this is that it allows decoupling the results of fluid dynamics simulations from the (arbitrary) choice of units. Indeed, the *law of similarity* states that two incompressible flow systems have the same dynamics (in dimensionless units) if they have the same geometry and Reynolds number [18].

2.2 Kinetic theory

As covered above, in the continuum model of fluid dynamics, the fluid is modeled at a macroscopic level. This means that the bulk dynamics of the fluid can be described in terms of the spatial and temporal variations of pressure, velocity, and temperature. It is not necessary to know or care about the microscopic fluid particles, even though the macroscopic behavior emerges from their dynamics. This is crucial because the sheer number of atomic particles means that it is completely impractical to model fluid systems of realistic size by considering each particle individually.

While the continuum model is widely applicable, there are nonetheless cases in which the approximation is not sufficiently accurate. Rather than drop down completely to the scale of individual atoms, kinetic theory considers the statistical distribution of particles. This is often referred to as the *mesoscopic* scale in the literature.

2.2.1 The particle distribution function

In kinetic theory, the fundamental state variable is the particle distribution function, $f(\mathbf{x}, \boldsymbol{\xi}, t)$. It can be interpreted as a generalization of the density $\rho(\mathbf{x}, t)$: the distribution function $f(\mathbf{x}, \boldsymbol{\xi}, t)$ represents the density of particles moving with velocity

$\boldsymbol{\xi}$ at position \mathbf{x} and time t . As a consequence, the macroscopic density can be found as the zeroth order moment

$$\rho(\mathbf{x}, t) = \int f(\mathbf{x}, \boldsymbol{\xi}, t) d\boldsymbol{\xi}. \quad (2.13)$$

Likewise, the macroscopic momentum density can be found as the first order moment

$$\rho(\mathbf{x}, t)\mathbf{u}(\mathbf{x}, t) = \int \boldsymbol{\xi} f(\mathbf{x}, \boldsymbol{\xi}, t) d\boldsymbol{\xi}, \quad (2.14)$$

while the macroscopic energy density is given by the second order moment

$$\rho(\mathbf{x}, t)E(\mathbf{x}, t) = \frac{1}{2} \int \boldsymbol{\xi}^2 f(\mathbf{x}, \boldsymbol{\xi}, t) d\boldsymbol{\xi}, \quad (2.15)$$

where $E(\mathbf{x}, t)$ is the total energy of the fluid, which is comprised of two components: the kinetic energy, $\frac{1}{2}\rho\mathbf{u}^2$, associated with the bulk motion of the fluid, and the internal energy due to the thermal motion of the particles. The internal energy density associated with the latter can be found by

$$\rho(\mathbf{x}, t)e(\mathbf{x}, t) = \int \mathbf{v}^2 f(\mathbf{x}, \boldsymbol{\xi}, t) d\boldsymbol{\xi}, \quad (2.16)$$

where $e(\mathbf{x}, t)$ is the internal energy, and $\mathbf{v} = \boldsymbol{\xi} - \mathbf{u}$ is the *relative velocity*.

2.2.2 The Boltzmann equation

For a dilute monoatomic gas, the dynamics of the distribution function $f(\mathbf{x}, \boldsymbol{\xi}, t)$ can be described by the Boltzmann equation:

$$\frac{\partial f}{\partial t} + \boldsymbol{\xi} \cdot \nabla f = \Omega(f), \quad (2.17)$$

where $\Omega(f)$ is the *collision operator*. The full collision operator as derived by Boltzmann is a complicated integral expression which will not be reproduced here because it does not yield any particular insight relevant to this thesis; in practice, the collision operator is always approximated in order to make solving (2.17) tractable.

One important consequence of the Boltzmann equation is that in the absence of external forces, the distribution function will tend toward an *equilibrium distribution*, $f^{\text{eq}}(\mathbf{x}, \mathbf{v}, t)$. This is the famous *Maxwell-Boltzmann distribution*, given by

$$f^{\text{eq}}(\mathbf{x}, \mathbf{v}, t) = \rho \left(\frac{1}{2\pi RT} \right)^{3/2} \exp \left(-\frac{\mathbf{v}^2}{2RT} \right), \quad (2.18)$$

where R is the ideal gas constant.

In order for an approximate collision operator Ω to be physically valid, it must satisfy the following requirements:

1. The distribution function f must tend towards the equilibrium function (2.18).
2. The collision operator must conserve mass, momentum, and total as well as internal energy. That is,

$$\int \Omega(f) \, d\xi = 0, \quad (2.19a)$$

$$\int \xi \Omega(f) \, d\xi = \mathbf{0}, \quad (2.19b)$$

$$\int \xi^2 \Omega(f) \, d\xi = 0, \quad (2.19c)$$

$$\int v^2 \Omega(f) \, d\xi = 0. \quad (2.19d)$$

A very simple—and very common—example of such a collision operator is the Bhatnagar-Gross-Krook (BGK) operator [19], given by

$$\Omega(f) = -\frac{1}{\tau}(f - f^{\text{eq}}), \quad (2.20)$$

where the time constant τ is the *relaxation time*. For the purposes of this thesis, it should be noted that (2.20) is used extensively in lattice Boltzmann simulations.

3

The lattice Boltzmann method

In this chapter, the key ideas necessary for implementing and applying the lattice Boltzmann method will be presented. Since the application of the lattice Boltzmann method for topology optimization constitutes a large part of the novelty of the research presented in this thesis, the details of the method will be presented in some detail. As a result, this is the longest theoretical chapter of this thesis. Nevertheless, the coverage in this chapter is far from comprehensive. Because the focus of this work is on application, the theoretical results presented will be—as in chapter 2—simply stated without derivation.

The chapter covers the following topics: section 3.1 gives a brief introduction to the history of the lattice Boltzmann method; section 3.2 explains the discretization of particle velocities and describes the most commonly used lattice; section 3.3 states the lattice Boltzmann equation itself, and shows how to compute the macroscopic solution variables from the lattice Boltzmann state variables; section 3.4 discusses the choice of collision operator, and states the two most commonly used operators; section 3.5 provides some details on how to actually implement the lattice Boltzmann method, and refers to further resources for the reader interested in highly efficient implementation strategies; section 3.6 discusses how to enforce boundary conditions in the lattice Boltzmann method; section 3.7 shows how to non-dimensionalize lattice Boltzmann simulations; finally, section 3.8 details the formal accuracy of the method, and provides some discussion of the additional sources of errors inherent in the method. Much of the exposition is based on the books [3, 5], as well as on numerous journal papers which will be referred in the relevant sections. In addition, each section will refer to additional resources in the literature should the reader wish to delve deeper into a particular topic.

As a final comment, note that this chapter will only cover the “classical” lattice Boltzmann method, which numerically solves the incompressible Navier-Stokes equations (2.2) and (2.3). The method can be extended to also cover thermal flows, i.e. equation (2.5). However, discussion of this topic will be deferred to chapter 7, since the material presented beforehand does not require it.

3.1 Introduction

The lattice Boltzmann method originated from so-called lattice gas cellular automata [20], which—as the name implies—is a type of cellular automata, a model in the vein of Conways famous “game of life”, which was found to be capable of simulating fluid dynamics. This type of model seems to have been largely displaced by modern lattice Boltzmann methods, which can in fact be derived independently from lattice gas cellular automata, though early papers on the method were writ-

ten from the perspective that the lattice Boltzmann method is an evolution of the cellular automata method [21, 22]. The interested reader may refer to the book by Wolf-Gladrow [23] for a detailed account of the connection between the two methods.

The lattice Boltzmann has been the subject of a fair amount of research interest since its introduction. This is motivated by its algorithmic simplicity, suitability for parallel implementation, and potential for faster performance over conventional methods [24]¹. In addition, the lattice Boltzmann method is well suited for complex geometries, and can be relatively easily extended from incompressible Navier-Stokes flow to more complicated physical systems, e.g. porous media [25–27], or multiphase flows [28, 29]. Due to the surge of interest in GPU computing, a number of papers on lattice Boltzmann GPU implementations have also appeared, since the highly parallel nature of the method lends itself well to this type of hardware [30–32].

3.2 Velocity discretization

If one wishes to solve the Boltzmann equation (2.17) numerically, it is not sufficient to simply discretize it in time and space, as is the case for the Navier-Stokes equations. This is because the Boltzmann equation has an additional continuous component: the particle velocity $\boldsymbol{\xi}$. The first order of business is therefore to somehow discretize the particle velocity space; essentially, the velocities which the particles are “allowed” to move with need to be restricted to a finite set. This discrete set of velocities is what is denoted as the “lattice” in “lattice Boltzmann”. Deriving the requirements which ensure that the necessary physics are retained in the lattice is an involved theoretical exercise, and will be omitted. Instead, the main results are simply stated.

A velocity lattice consists of a finite set of velocities \mathbf{c}_i , a corresponding set of numerical *weights* w_i , and a constant c_s such that

$$\sum_i w_i = 1, \quad (3.1a)$$

$$\sum_i w_i c_{i\alpha} = 0, \quad (3.1b)$$

$$\sum_i w_i c_{i\alpha} c_{i\beta} = c_s^2 \delta_{\alpha\beta}, \quad (3.1c)$$

$$\sum_i w_i c_{i\alpha} c_{i\beta} c_{i\gamma} = 0, \quad (3.1d)$$

$$\sum_i c_{i\alpha} c_{i\beta} c_{i\gamma} c_{i\delta} = c_s^4 (\delta_{\alpha\beta} \delta_{\gamma\delta} + \delta_{\alpha\gamma} \delta_{\beta\delta} + \delta_{\alpha\delta} \delta_{\beta\gamma}), \quad (3.1e)$$

$$\sum_i w_i c_{i\alpha} c_{i\beta} c_{i\gamma} c_{i\delta} c_{i\eta} = 0. \quad (3.1f)$$

¹Note that the cited paper is from 2006, its conclusions are likely outdated. To the authors knowledge, no newer benchmarks exist.

Above, Greek letters correspond to Cartesian coordinates of the velocity vector \mathbf{c}_i and $\delta_{\alpha\beta}$ is the Kronecker delta. It can be shown that the parameter c_s is equal to the speed of sound in the lattice Boltzmann simulation, hence why the notation has been reused.

For most practical lattice Boltzmann simulations described in the literature, a limited set of “standard” lattices which are already known to satisfy the requirements (3.1) are almost always used. Interestingly, there exist usable lattices which do not satisfy all the requirements (3.1) [33], but these require modification of the collision operator to compensate, and have not been considered for this thesis.

3.2.1 The D2Q9 lattice

For the 2D simulations presented later, the so-called D2Q9 lattice (two dimensional, nine velocities) will be used. The weights and velocities associated with this lattice are listed in equation (3.2), and depicted in figure 3.1.

$$\mathbf{c}_i = \frac{\Delta x}{\Delta t} \begin{cases} (0, 0), & i = 0, \\ (\pm 1, 0), (0, \pm 1), & i = 1, 2, 3, 4, \\ (\pm 1, \pm 1), & i = 5, 6, 7, 8; \end{cases} \quad (3.2a)$$

$$w_i = \begin{cases} \frac{4}{9}, & i = 0, \\ \frac{1}{9}, & i = 1, 2, 3, 4, \\ \frac{1}{36}, & i = 5, 6, 7, 8; \end{cases} \quad (3.2b)$$

$$c_s = \frac{\Delta x}{\Delta t} \frac{1}{\sqrt{3}}. \quad (3.2c)$$

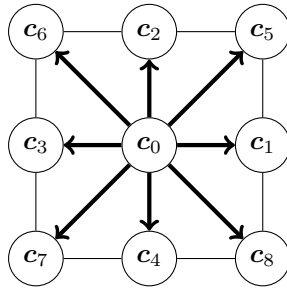


Figure 3.1: Visual illustration of the D2Q9 model. This figure originally appeared in the publications [P1] and [P2].

On an equidistant Cartesian grid with grid point separation Δx , the D2Q9 lattice is simply the collection of velocities that can travel to the nearest neighbors of a given point in the time Δt . For the purpose of lattice Boltzmann simulations, a common choice for Δx and Δt are the so-called “lattice units”, in which $\Delta x = \Delta t = 1$.

3.3 The lattice Boltzmann equation

At its core, the lattice Boltzmann equation is simply a finite difference discretization of the Boltzmann equation (2.17). The key result is that if the fluid system of interest is operating at sufficiently low Mach and Knudsen number, one can evaluate the integrals (2.13) and (2.14) numerically in order to obtain the values of the macroscopic quantities ρ and \mathbf{u} . As mentioned in section 2.1.1, the lattice Boltzmann method is weakly compressible, meaning that there will be density fluctuations in the fluid domain; these fluctuations directly relate to pressure differences through the state equation

$$p = c_s^2 \rho. \quad (3.3)$$

The values of p and \mathbf{u} thus obtained will approximately satisfy the Navier-Stokes equations (2.2) and (2.3).

Like in the previous section, the derivation of the lattice Boltzmann equation will not be covered in this thesis. Instead, the result will simply be stated. The equation is given by

$$f_i(\mathbf{x}_j + \mathbf{c}_i \Delta t, t + \Delta t) = f_i(\mathbf{x}_j, t) + \Omega_i(\mathbf{f}(\mathbf{x}_j, t)), \quad (3.4)$$

where f_i is the distribution value corresponding to the particle velocity \mathbf{c}_i , and Ω_i is a discretized collision operator. Equation (3.4) is an explicit time-stepping scheme for the distribution function \mathbf{f} , given appropriate initial and boundary conditions.

Once the distribution values $\mathbf{f}(\mathbf{x}, t)$ is known at each grid point, the macroscopic values can be evaluated by

$$\rho(\mathbf{x}_j, t) = \sum_i f_i(\mathbf{x}_j, t), \quad (3.5)$$

$$\rho(\mathbf{x}_j, t) \mathbf{u}(\mathbf{x}_j, t) = \sum_i \mathbf{c}_i f_i(\mathbf{x}_j, t). \quad (3.6)$$

This is the basic framework of the lattice Boltzmann method, though well-defined collision operators and boundary conditions are still needed to actually use it in practice.

3.4 Collision operators

The choice of collision operator is an important one for the purpose of performing lattice Boltzmann simulations. It has implications for the accuracy, stability, and execution time of the simulation.

3.4.1 BGK operator

The BGK operator was already introduced in section 2.2.2. For the lattice Boltzmann method, it retains the same basic form as the continuous version, (2.20), that is

$$\Omega_i^{\text{BGK}} = -\frac{1}{\tau}(f_i - f_i^{\text{eq}}), \quad (3.7)$$

where the equilibrium function f_i^{eq} is generally truncated up to second order in \mathbf{u} :

$$f_i^{\text{eq}} = w_i \rho \left(1 + \frac{\mathbf{c}_i \cdot \mathbf{u}}{c_s^2} + \frac{(\mathbf{c}_i \cdot \mathbf{u})^2}{2c_s^4} - \frac{\mathbf{u} \cdot \mathbf{u}}{2c_s^2} \right). \quad (3.8)$$

The BGK is the most common collision operator used for lattice Boltzmann simulations due to its simplicity and ease of implementation.

Relating the lattice Boltzmann method to the Navier-Stokes equations is a theoretical exercise that will not be covered in this thesis; the interested reader may refer to e.g. [23, 34] for these details. The main result needed from this theoretical ground work is the relation between the relaxation time τ and the kinematic viscosity ν . It is given by

$$\tau = c_s^2 \nu \frac{\Delta t}{\Delta x^2} + \frac{1}{2}. \quad (3.9)$$

With this τ , the dynamics simulated by the lattice Boltzmann method will approximate that of the incompressible Navier-Stokes equations with kinematic viscosity ν to second order accuracy in time and space. However, the issue of accuracy in the lattice Boltzmann method is more subtle than the simple statement above would suggest. This is discussed in further detail in section 3.8.

3.4.2 MRT operator

The idea of the *multiple relaxation time* (MRT) operator is that the distribution values f_i are mapped to a vector of derived moments; for instance, the density (3.5) is one such moment. These moments can then be relaxed with individual relaxation times, unlike the BGK operator, in which all moments are relaxed with the rate $1/\tau$. This allows for increased accuracy and stability, at the cost of a higher computational complexity of the collision operator. The theoretical details of this operator may be found in [35, 36].

The collision operator is given by

$$\Omega^{\text{MRT}}[\mathbf{f}(\mathbf{x}, t)] = \mathbf{M}^{-1} \mathbf{S} \mathbf{M}[\mathbf{f}(\mathbf{x}, t) - \mathbf{f}^{\text{eq}}(\mathbf{x}, t)], \quad (3.10)$$

where \mathbf{M} and \mathbf{S} are matrices. \mathbf{M} is the mapping from distribution into moment space, while \mathbf{S} contains the relaxation times for the various moments. For the D2Q9

lattice, they are given by

$$\mathbf{M} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ -4 & -1 & -1 & -1 & -1 & 2 & 2 & 2 & 2 \\ 4 & -2 & -2 & -2 & -2 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & -1 & 0 & 1 & -1 & -1 & 1 \\ 0 & -2 & 0 & 2 & 0 & 1 & -1 & -1 & 1 \\ 0 & 0 & 1 & 0 & -1 & 1 & 1 & -1 & -1 \\ 0 & 0 & -2 & 0 & 2 & 1 & 1 & -1 & -1 \\ 0 & 1 & -1 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & -1 & 1 & -1 \end{pmatrix}, \quad (3.11)$$

$$\mathbf{S} = \text{diag}(0, s_1, s_2, 0, s_{46}, 0, s_{46}, s_{78}, s_{78}), \quad (3.12)$$

where the s values are the different relaxation rates for the individual moments. For the purpose of simulations, s_{78} is the equivalent of τ , so this relaxation rate is determined by the kinematic viscosity ν via (3.9). The other relaxation rates can be tuned independently in order to maximize stability and accuracy. The reader may refer to [36,37] for linear stability analyses.

3.4.3 Other collision operators

The two collision operators described above are the most commonly used in applications, and have been used almost exclusively for the problems presented in this thesis. Of course, these two are not the only available options. Notable examples are the so-called entropic collision operators [38,39], lattice Boltzmann with regularized pre-collision [40], and the cascaded collision operator [41,42]. Of these, only the cascaded collision operator has been considered in the present work, and even so it was mostly used as a data point for the discussion of performance in [P2]. Lattice Boltzmann with the cascaded operator does seem to be remarkably stable though, so further investigation of its potential would definitely be warranted.

3.5 Implementation

In concrete implementations, the lattice Boltzmann method is often split into two steps: collision and streaming. The *collision step* is purely local, and transforms the distributions at each grid point according to the collision operator Ω :

$$\tilde{f}_i(\mathbf{x}_j, t) = f(\mathbf{x}_j, t) + \Omega_i(\mathbf{f}(\mathbf{x}_j, t)), \quad (3.13)$$

where tilde is used to denote the post-collision state. The transformed distributions are then shifted to neighboring grid nodes in the *streaming step*:

$$f_i(\mathbf{x}_j + \mathbf{c}_i \Delta t, t + \Delta t) = \tilde{f}_i(\mathbf{x}_j, t). \quad (3.14)$$

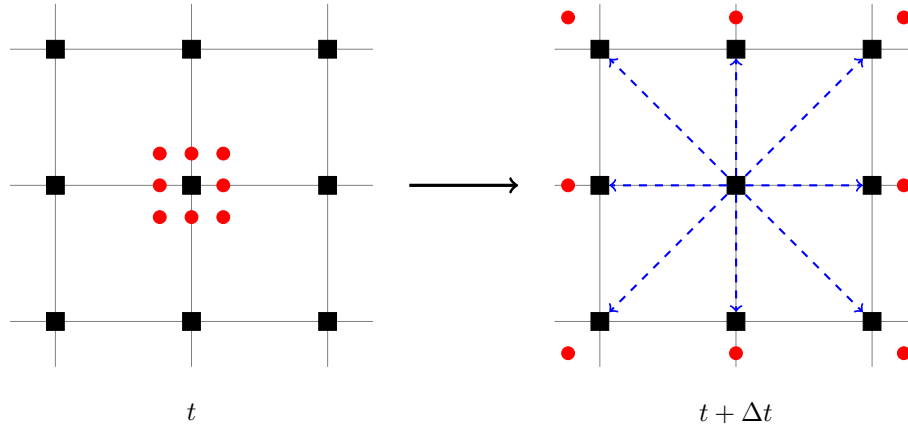


Figure 3.2: Illustration of streaming on a D2Q9 lattice. Left: prior to streaming, the distribution values, represented by red dots, are gathered at the central node. Right: following the streaming step, the distributions are shifted along the velocity vectors \mathbf{c}_i to the nearest neighbor nodes. Note that the resting velocity \mathbf{c}_0 is not depicted, since its associated distribution simply remains in-place.

The shift of distributions according to the streaming step is illustrated in figure 3.2.

The simplest way to implement the collision and streaming steps is using two separate arrays. Collision is then performed on the first array, and the subsequent streaming step shifts the distributions by copying from the collision array to a second streaming array. This approach is nice because it is very easy to implement, and it is still reasonably efficient. On the other hand, using two arrays means that the memory requirement of the method is doubled, and it is necessary to traverse the grid twice in order to complete a single time-step. There are more sophisticated algorithms available [43–45], which allow the collision and streaming steps to be computed using a single pass over a single array. These algorithms inevitably increase the complexity of the implementation though. While the author did implement the algorithm by Latt [43], the code was eventually reverted to using the naive algorithm, since the performance gain was found to be negligible compared to the increase in complexity.

3.6 Boundary conditions

The topic of boundary conditions in the lattice Boltzmann method is complicated by the fact that one cannot straightforwardly impose boundary conditions directly on the distribution functions f_i themselves. This is because the end goal is to simulate Navier-Stokes dynamics, where boundary conditions are generally imposed on the macroscopic quantities p or \mathbf{u} . The distribution values, however, are comprised of a

greater number of state variables. This implies that while the macroscopic quantities can be uniquely determined from the distribution values, the reverse is not true. Therefore, some sort of simplifying assumptions have to be made in order to derive boundary conditions for the distribution values from boundary conditions for the macroscopic quantities.

Boundary conditions for the lattice Boltzmann method are typically applied immediately following the streaming step. Since the streamed distributions arrive from all immediate neighbor nodes, only those distributions which would have arrived from “outside” the simulation domain are unknown following the streaming step, see figure 3.3. Because of this it is generally not necessary to determine all the distribution values at the boundaries following the streaming step, but only a subset.

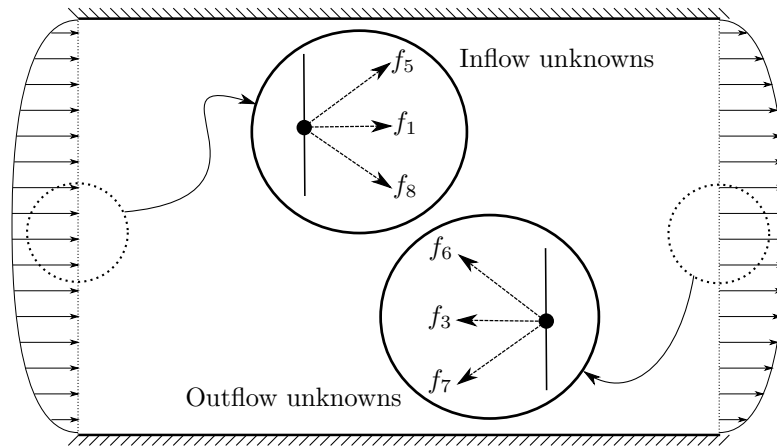


Figure 3.3: Illustration of the unknowns at the boundaries of a simple flow geometry, using the D2Q9 lattice. Note that the unknowns depend on the location of the boundary. This figure originally appeared in [P1].

Boundary conditions for lattice Boltzmann is a large topic, to the point where it could be considered its own sub-field. Here only some very simple boundary conditions which have been used extensively for the work presented in this thesis are covered.

3.6.1 No-slip boundaries

No-slip boundary conditions can be dealt with very easily using the so-called *bounce-back* boundary conditions. Under these conditions, it is assumed that the distributions which would propagate out of the domain during the streaming step hit the no-slip wall and are reflected back in the opposite direction. That is, the unknown

distribution values are determined as

$$f_i(\mathbf{x}_j, t + \Delta t) = f_{i'}(\mathbf{x}_j, t), \quad (3.15)$$

where $f_{i'}$ is the distribution associated with the opposite velocity of f_i (e.g. $f_{1'} = f_3$). The bounceback boundary conditions are a very useful tool because they are simple to implement for all boundary locations, and because they handle corner nodes (where there are five unknown distributions rather than three) without any additional modifications.

3.6.2 Velocity and pressure boundaries

Besides no-slip boundary conditions, the most common type of boundary conditions are so-called Dirichlet conditions on velocity or pressure. That is, some known value of the velocity or pressure is imposed on the boundary. Indeed, no-slip conditions are simply a specific case of this, with $\mathbf{u} = \mathbf{0}$. Note that in the lattice Boltzmann method, fixed pressure boundary conditions are equivalent with fixed density boundary conditions, due to the relation (3.3),

A simple method for imposing Dirichlet boundary conditions has been introduced by Zou and He [46]. Here, the idea is demonstrated by showing how to implement a velocity boundary condition, but the method can also be used for pressure boundaries.

The idea is illustrated by means of a concrete example: consider the “west” boundary on the left of figure 3.3. Following the streaming step, the unknowns are ρ , f_1 , f_5 , and f_8 . To compute these, a system of four equations is needed to solve for the four unknowns. The macroscopic variables equations (3.5)–(3.6) provide three such equations. The system is closed by what Zou and He refer to as the *non-equilibrium bounceback* assumption, which asserts

$$f_1(\mathbf{x}_j, t) - f_1^{\text{eq}}(\mathbf{x}_j, t) = f_3(\mathbf{x}_j, t) - f_3^{\text{eq}}(\mathbf{x}_j, t). \quad (3.16)$$

Solving this system of equations yields the Zou/He velocity boundary conditions on the west boundary:

$$\rho(\mathbf{x}_j, t) = \frac{f_0 + f_2 + f_4 + 2(f_3 + f_6 + f_7)}{1 - \bar{u}_x}, \quad (3.17a)$$

$$f_1(\mathbf{x}_j, t) = f_3 + \frac{2}{3}\rho\bar{u}_x, \quad (3.17b)$$

$$f_5(\mathbf{x}_j, t) = f_7 + \frac{1}{2}(f_4 - f_2) + \frac{1}{6}\rho\bar{u}_x + \frac{1}{2}\rho\bar{u}_y, \quad (3.17c)$$

$$f_8(\mathbf{x}_j, t) = f_6 + \frac{1}{2}(f_2 - f_4) + \frac{1}{6}\rho\bar{u}_x - \frac{1}{2}\rho\bar{u}_y. \quad (3.17d)$$

Above, \bar{u}_x and \bar{u}_y are the x and y components of the fixed boundary velocity. Note that the spatial and temporal arguments for the variables have been omitted on the right-hand side for brevity. Similar boundary conditions may be derived for velocity

boundary conditions of different orientations, as well as for pressure boundaries, though they will not be listed here.

3.7 Non-dimensionalization of lattice Boltzmann

As mentioned in section 2.1.3, non-dimensionalization of the Navier-Stokes equations is a useful way to decouple numerical solutions from the choice of units. Of course, the same applies to the lattice Boltzmann method. However, as mentioned in section 3.2.1, it is convenient to implement the lattice Boltzmann method in terms of “lattice units” in which the distance between nodes Δx and the time-step Δt are both taken to be unity. As shown by Latt [47], it is possible to map between these two representations easily. If a non-dimensional system has been defined in which the reference length L_0 and reference time t_0 are both unity (which also implies $|u_0| = 1$), then the relevant parameters in lattice units are computed with the mapping

$$\begin{aligned} u_{0,\text{lb}} &= \frac{\Delta t_d}{\Delta x_d}, \\ \nu_{\text{lb}} &= \frac{\Delta t_d}{\Delta x_d^2} \frac{1}{\text{Re}}, \end{aligned} \tag{3.18}$$

where Δx_d and Δt_d are the sizes of the of the lattice units in the original non-dimensional reference units. For example, if the reference length L_0 was discretized into $N_{\mathbf{x}}$ lattice nodes, then

$$\Delta x_d = 1/(N_{\mathbf{x}} - 1). \tag{3.19}$$

This mapping is useful since it makes it relatively easy to make consistent comparisons between numerical solutions obtained from the Navier-Stokes equations and those obtained by the lattice Boltzmann method.

3.8 Accuracy of the lattice Boltzmann method

As mentioned in the opening of chapter 2, linking the lattice Boltzmann method to the incompressible Navier-Stokes equations requires asymptotic expansion techniques, e.g. the Chapman-Enskog expansion. Carrying out this asymptotic expansion on the lattice Boltzmann equation reveals that the lattice Boltzmann method is a second order accurate approximation of the incompressible Navier-Stokes equations in both time and space [5, 34].

This simple assessment is nowhere close to the complete story regarding the accuracy of the lattice Boltzmann method, however. There are additional error sources which arise because the lattice Boltzmann method is not a direct discretization of the incompressible Navier-Stokes equations. In addition to the truncation error related to the time and space discretization, there is also an error term related to the relaxation time τ (see [5] for a detailed discussion of this for several collision

operators), as well as the so called *compressibility error*, which arises because the lattice Boltzmann method is weakly compressible.

The compressibility error scales as $\mathcal{O}(\text{Ma}^2)$, meaning that ideally the Mach number should be kept as low as possible. An interesting consequence of this is that since the Mach number is grid independent, it is not possible to improve the accuracy of a lattice Boltzmann simulation indefinitely by refining the discretization arbitrarily in either space or time, since eventually the compressibility error would dominate. Decreasing the compressibility error at the same rate as the truncation error requires scaling the time-step according to

$$\Delta t \propto \Delta x^2, \quad (3.20)$$

which is often referred to as *diffusive scaling* in the literature. Interestingly, this means that the lattice Boltzmann is in practice only first order accurate in time. Furthermore, to the authors understanding, it is possible to completely eliminate the compressibility error in the case of steady state simulations [46, 48], but this does not seem to apply for unsteady simulations [49, 50].

4

Topology optimization

The purpose of this chapter is to document the essential ideas of topology optimization which will be needed in the subsequent chapters. It is meant as a brief overview of the topic and is by no means comprehensive. The reader is referred to the book [51] for thorough exposition, as well as the various journal papers referenced in this chapter.

The chapter is structured as follows: section 4.1 gives an informal introduction to topology optimization with the density method and provides some literature review; section 4.2 goes more into the mathematical details of the optimization problem, and provides more formal exposition on density based topology optimization; section 4.3 covers the adjoint method for computing the gradient; section 4.4 introduces the projection filter, which is used extensively in the optimization problems presented later, as well as the robust formulation of a topology optimization problem; finally, section 4.5 briefly discusses the method of moving asymptotes, which is the algorithm used for solving the optimization problem.

This chapter only discusses the density based method of topology optimization. For discussion of other approaches, the reader is referred to the review paper [52]. Furthermore, the reader is assumed to have some knowledge of numerical optimization, which is covered in many textbooks, e.g. [53].

4.1 Introduction

Topology optimization is a sub-discipline of the engineering field of structural optimization. Informally, the goal of structural optimization is to optimize the structure or shape¹ of some engineering feature, in order to maximize its capacity to fulfill its desired purpose. This capacity is typically constrained by the underlying physics, which is modeled using PDEs. Thus, structural optimization problems are formulated as PDE constrained optimization problems. Structural optimization can be split into three sub-disciplines: size optimization, shape optimization, and topology optimization. The differences between them are illustrated in figure 4.1, using a circular geometry as a basis.

In size optimization, only the radius of the circle can change, the basic form of the circle is always preserved. Conversely, shape optimization may change the external boundary of the circle but may not introduce new internal boundaries by creating new holes or features separated from the external boundary. Finally, topology optimization allows full design freedom: the circle can in principle be changed into any arbitrarily complicated geometry.

In order to set up the problem mathematically, some way of representing the state of the design is needed. For numerical optimization, the design is represented

¹The word “shape” is used here in its informal sense of the external appearance of an object.

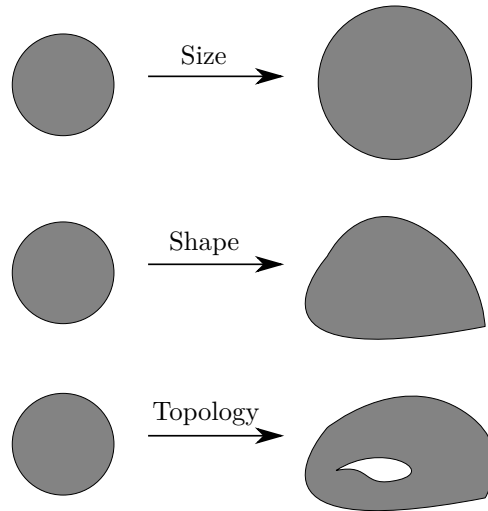


Figure 4.1: Conceptual illustration of the different forms of structural optimization on a circular design.

by a finite vector of *design variables*, which describes the design in some manner. For size optimization, the design variables would simply be the size or sizes in question, e.g. the radius in the example of the circle. For shape optimization, the choice of design variables is less obvious. One possibility would be to use a finite collection of points which lie on the boundary of the shape. This representation would of course only be approximate, since interpolation between the points would be required to recover the shape. In the case relevant for this thesis—topology optimization—there is likewise a number of possibilities available that (approximately) represent the design topology as a collection of design variables. In this thesis only the so-called *density based* approach is discussed.

Informally, the idea of this approach is to partition the design area—often called the *design domain*—into a discrete mesh of elements, each of which are assigned a numerical value corresponding to the element being either “solid”, i.e. containing design material, or “void”, i.e. without design material. These numerical values then comprise the design variables. For fluid flow problems, the void phase corresponds to a fluid element, i.e. an element in which fluid can flow freely, whereas solid corresponds to rigid material where no fluid can flow. In order to solve the problem numerically, the requirement that each element must be either solid or fluid is relaxed to allow “in-between” states. This allows solving the optimization problem using gradient based methods.

Topology optimization as a scientific field originates with the seminal paper by Bendsøe and Kikuchi [54], and has since been applied to a wide variety of

physics constrained design problems, including problems in fluid dynamics. The first application of topology optimization to fluid problems was published by Borrvall and Petersson in 2003 [55], who investigated Stokes flow problems. Their work has since been extended to laminar Navier-Stokes flows [56, 57]. Using the lattice Boltzmann method to solve problems in topology optimization was pioneered by Pingen *et al.* [58], solving problems similar to those presented by Borrvall and Petersson. Others have investigated topology optimization with the lattice Boltzmann method using a level-set approach [59–61].

Several more recent publications have focused on extending the work presented in the above papers to more complicated fluid dynamics, notably thermal flows. Andreasen *et al.* applied topology optimization to a thermofluidic mixer [62], while Alexandersen *et al.* have investigated passive heat sinks [63, 64]. In addition, there are publications which have investigated thermal problems using the lattice Boltzmann method [65].

A commonality of all the papers presented above is that only steady state flow solutions have been considered. At present, there are only few publications which have considered topology optimization for unsteady flow problems, likely due to the high computational cost associated with these problems. The seminal papers in this area are by Kreissl *et al.* [66], and Deng *et al.* [67], published within two months of each other in 2011. Since then, the works by Deng *et al.* has been extended to flows with body forces [68]. To the best of the authors knowledge, no other publications have subsequently appeared, except for the contributions presented in this thesis.

4.2 Optimization problem

Mathematically, a topology optimization problem may be formulated as follows:

$$\begin{aligned} & \min_{\mathbf{s} \in \Omega_d} \varphi(\mathbf{v}, \mathbf{s}), \\ \text{s.t. } & \begin{cases} R(\mathbf{v}, \mathbf{s}) = \mathbf{0}, & \text{PDE constraint,} \\ 0 \leq s_j \leq 1, \forall j \in \{1, \dots, N_d\} & \text{design variable constraint,} \\ C_k(\mathbf{v}, \mathbf{s}) \leq 0, \forall k \in \{1, \dots, N_c\}, & \text{additional constraints,} \end{cases} \end{aligned} \quad (4.1)$$

where φ is some function which measures the capacity of the structure to perform its desired purpose, often referred to as the *objective function* in the literature; \mathbf{v} is a vector of physical state variables associated with the system, and \mathbf{s} is the vector of design variables, which characterize the topology of the system. These variables are defined on the discretized physical domain Ω , and design domain $\Omega_d \subseteq \Omega$, respectively. As mentioned in section 4.1, the optimization problem is constrained by the physics of the problem, which are modeled using PDEs. This is formulated as an equality constraint, $R(\mathbf{v}, \mathbf{s}) = \mathbf{0}$, where R is the residual of some numerical discretization of a PDE (or system of PDEs), properly arranged to have a zero right-hand side. The design domain is discretized into N_d design variables, each with a value in the

range $s_j \in [0, 1]$. For the purpose of this thesis, $s_j = 0$ shall always denote a solid domain, while $s_j = 1$ shall always denote a fluid domain. Finally, the optimization problem may have an arbitrary number N_c of additional constraints, given by the functions C_j . A constraint commonly seen in the topology optimization literature is the *volume constraint*, which states that only a limited fraction of the design domain can be occupied by the optimized structure.

A solution to equation (4.1) should ideally be a vector of design variables where all values are either zero or one. However, the design variables are allowed to vary continuously between these two end-points; this makes the derivative $\frac{d\varphi}{ds_j}$ well defined, enabling the use of gradient-based numerical methods for solving the optimization problem iteratively. This also implies that the model PDE represented by $R(\mathbf{v}, \mathbf{s}) = \mathbf{0}$ must allow each element to continuously transition from the solid state to the fluid state, with values in $s_j \in]0, 1[$ representing some fictitious state in-between. In some cases, these transitional states can be ascribed physical meaning [69], but otherwise various penalization methods are used to ensure that the optimization algorithm will converge to a fully discrete 0, 1-solution, often called “black and white” solutions in the literature, contrary to solutions which contain “grey” elements. Chapter 5 discusses the details of how this is achieved with the lattice Boltzmann method.

4.3 Sensitivity analysis

In order to run any gradient-based optimization algorithm, the actual gradient vector $\frac{d\varphi}{d\mathbf{s}}$ obviously needs to be evaluated. The derivatives collected in the vector are often termed the *sensitivities* in the literature. An efficient way of computing the sensitivities is the so-called *adjoint method* [70]. With this approach, the computational cost of evaluating the sensitivities is similar to the cost of determining the physical solution vector \mathbf{v} , i.e. solving $R(\mathbf{v}, \mathbf{s}) = \mathbf{0}$.

A way to explain the adjoint method is to add a *Lagrange multiplier* times the residual to objective function, that is

$$\hat{\varphi}(\mathbf{v}, \mathbf{s}) = \varphi(\mathbf{v}, \mathbf{s}) + \boldsymbol{\lambda}^T R(\mathbf{v}, \mathbf{s}), \quad (4.2)$$

where $\boldsymbol{\lambda}$ is a vector of multipliers. Since $R = 0$, this does not change the value of φ . Differentiation of (4.2) with respect to an arbitrary design variable s_j yields

$$\frac{d\hat{\varphi}}{ds_j} = \frac{\partial\varphi}{\partial s_j} + \frac{\partial\varphi}{\partial\mathbf{v}} \frac{d\mathbf{v}}{ds_j} + \boldsymbol{\lambda}^T \left(\frac{\partial R}{\partial s_j} + \frac{\partial R}{\partial\mathbf{v}} \frac{d\mathbf{v}}{ds_j} \right). \quad (4.3)$$

The term $\frac{d\mathbf{v}}{ds_j}$ is difficult to evaluate, but rearranging (4.3) to

$$\frac{d\hat{\varphi}}{ds_j} = \frac{\partial\varphi}{\partial s_j} + \boldsymbol{\lambda}^T \frac{\partial R}{\partial s_j} + \left(\frac{\partial\varphi}{\partial\mathbf{v}} + \boldsymbol{\lambda}^T \frac{\partial R}{\partial\mathbf{v}} \right) \frac{d\mathbf{v}}{ds_j}, \quad (4.4)$$

it becomes apparent that the troublesome terms with $\frac{dv}{ds_j}$ can be eliminated by solving the following *adjoint problem*:

$$\left(\frac{\partial R}{\partial s_j}\right)^T \boldsymbol{\lambda} = -\left(\frac{\partial \varphi}{\partial \mathbf{v}}\right)^T. \quad (4.5)$$

While the adjoint problem is simple to state, in practice it can be quite challenging to derive the adjoint problem for a concrete numerical scheme. Specifically, this is true for the lattice Boltzmann method due to the highly non-linear nature of the collision step. In chapter 6, a generalized approach for adjoint sensitivity analysis of the lattice Boltzmann method, utilizing automatic differentiation, will be presented.

4.4 Filtering of the design variables

One problem which can arise in density based topology optimization is that the optimizing algorithm is free to modify the design on the length scale of a single design element, which can lead to undesirable features such as checkerboard patterns [71]. A common way of mitigating this issue is to apply a transformation—commonly called a *filter*—to the design variables in order to smoothen the final design. The interested reader is referred to [51] for detailed discussion of this issue.

4.4.1 Projection filter

In this thesis, the focus will be on the so-called *projection filter* [72]. This filter works by projecting a smoothened version of the design field according to

$$s_j^p = \frac{\tanh(\beta\eta) + \tanh(\beta(s_j^w - \eta))}{\tanh(\beta\eta) + \tanh(\beta(1 - \eta))}, \quad (4.6)$$

as given in [73]. Here, s_j^w is the value of the smoothed design field at node j , β controls the strength of the projection around the threshold, and η determines the threshold location, see figure 4 in [P1].

The smoothened design field \mathbf{s}^w is obtained by applying a weighted averaging around each element [74, 75], that is

$$s_j^w = \frac{\sum_{k \in w\mathbb{N}_{e,j}} H(\mathbf{x}_k) s_k}{\sum_{k \in \mathbb{N}_{e,j}} H(\mathbf{x}_k)}, \quad (4.7)$$

where s_j^w is the value of the design variable after smoothing, $\mathbb{N}_{e,j}$ is comprised of the design elements within radius r of s_j , $H(\mathbf{x}_k)$ is the weighting, given by

$$H(\mathbf{x}_k) = r - |\mathbf{x}_k - \mathbf{x}_j|, \quad (4.8)$$

and \mathbf{x}_j and \mathbf{x}_k are the center point coordinates of the elements s_j and s_k . Note that the smoothening operation defined by (4.7)–(4.8) can be implemented more efficiently [76], but this was not done for this work since it was never found to be a significant contributor to the total run-time.

4.4.2 Robust formulation

An extension of the projection filter is the so-called *robust formulation* of the topology optimization problem (4.1). In this formulation, the optimization problem is written as a min-max problem:

$$\begin{aligned} & \min_{\mathbf{s} \in \Omega_d} \max \{ \varphi(\mathbf{v}, \mathbf{s}_e^p), \varphi(\mathbf{v}, \mathbf{s}_i^p), \varphi(\mathbf{v}, \mathbf{s}_d^p) \}, \\ & \text{s.t.} \begin{cases} R(\mathbf{v}_e, \mathbf{s}_e^p) = R(\mathbf{v}_i, \mathbf{s}_i^p) = R(\mathbf{v}_d, \mathbf{s}_d^p) = \mathbf{0}, \\ 0 \leq s_j \leq 1, \forall j \in \{1, \dots, N_{\text{design}}\}, \\ C_k(\mathbf{v}, \mathbf{s}) \leq 0, \forall k \in \{1, \dots, N_c\}. \end{cases} \end{aligned} \quad (4.9)$$

Here, the subscript e, i , and d correspond to eroded, intermediate, and dilated design fields, which are obtained with an application of the projection filter with $\eta < 0.5$, $\eta = 0.5$, and $\eta > 0.5$, respectively. That is, one design vector \mathbf{s} is mapped to three different design realizations. The min-max formulation ensures that it is always the worst design realization driving the optimization. This means that the end result of the optimization is a design which performs well for the η values of the intermediate, eroded and dilated design. If the end result also performs well in the entire range of η values spanned by the eroded and dilated design, and there are no topology changes when η varies, then the design performs robustly under uniform spatial variations, though none of this is guaranteed. For detailed discussion, see the papers [73, 77]. This method of robust optimization can be further extended to account for non-uniform variations [78] by using a spatially varying projection value, $\eta(\mathbf{x})$, rather than a constant one. An application of this will be demonstrated in chapter 7.

As a final comment, note that the robust formulation is in no way restricted to three realizations; this was merely chosen because it is a common choice in the literature, but any number of realizations can be used, given sufficient computational power.

4.5 Solving the optimization problem

In this thesis, the optimization algorithm of choice is the method of moving asymptotes (MMA), introduced by Svanberg [79]. This algorithm and its extension, the globally convergent MMA (GCMMA) [80], are extensively used in the topology optimization field to the point of being almost ubiquitous. The implementation of MMA documented in [81] has been used for all the problems presented later. The full

mathematical details of the MMA are beyond the scope of this thesis; the interested reader may refer to the original papers. Here we simply note a few details of the MMA for later reference.

The MMA approximates the solution to the general nonlinear programming problem

$$\begin{aligned} & \min_{\mathbf{s} \in \mathbb{R}^n, \mathbf{y}, z} f_0(\mathbf{s}), \\ \text{s.t.} & \begin{cases} s_j^{\min} \leq s_j \leq s_j^{\max}, \forall j \in \{1, \dots, n\}, \\ C_k(\mathbf{s}) \leq 0, \forall k \in \{1, \dots, N_c\}, \end{cases} \end{aligned} \quad (4.10)$$

by finding an approximate solution to the modified problem

$$\begin{aligned} & \min_{\mathbf{s} \in \mathbb{R}^n, \mathbf{y}, z} f_0(\mathbf{s}) + z + \sum_{k=1}^{N_c} \left(c_k y_k + \frac{d_k}{2} y_k^2 \right), \\ \text{s.t.} & \begin{cases} s_j^{\min} \leq s_j \leq s_j^{\max}, \forall j \in \{1, \dots, n\}, \\ C_k(\mathbf{s}) - a_k z - y_k \leq 0, \forall k \in \{1, \dots, N_c\}, \\ \mathbf{y} \geq 0, z \geq 0, \end{cases} \end{aligned} \quad (4.11)$$

where \mathbf{y} and z are “artificial” variables, and \mathbf{a} , \mathbf{c} , and \mathbf{d} are user-defined constants. Ideally, this modification should not affect the final solution, though it depends somewhat on the choice of constants. The choice of constants also allows some flexibility in the kinds of problems which can be solved. Specifically, the robust problem (4.9) can be solved using MMA. This is done by re-writing (4.9) to

$$\begin{aligned} & \min_{\mathbf{s} \in \mathbb{R}^n, z} z \\ \text{s.t.} & \begin{cases} \varphi(\mathbf{v}, \mathbf{s}_r) + K - z \leq 0, \forall r \in \{e, i, d\}, \\ s_j^{\min} \leq s_j \leq s_j^{\max}, \forall j \in \{1, \dots, n\}, \\ C_k(\mathbf{s}) - a_k z - y_k \leq 0, \forall k \in \{1, \dots, N_c\}, \\ \mathbf{y} \geq 0, z \geq 0, \end{cases} \end{aligned} \quad (4.12)$$

where $K \geq 0$ is a constant value which ensures that $\varphi(\mathbf{v}, \mathbf{s}_r) + K > 0$. This problem may be solved with MMA by defining $f_0(\mathbf{s}) = 0$, as well as proper choice of the constants \mathbf{a} , \mathbf{c} , and \mathbf{d} .

5

Unsteady flow topology optimization [P1]

This chapter summarizes the essential results presented in [P1]. In addition, unpublished results which further extend the pump problem introduced in [P1] are presented.

The chapter is structured as follows: section 5.1 briefly introduces the lattice Boltzmann model for porous media used to distinguish between fluid and solid nodes in the optimization; section 5.2 states the form of the objective function used for all of the subsequently presented problems; section 5.3 summarizes the findings for the first example problem introduced in [P1], using topology optimization to design an obstacle which results in a specific downstream flow pattern; section 5.4 discusses the findings of the second example introduced in [P1], topology optimization for a simple model of a fluid pump; finally, section 5.5 describes extensions of the pump problem which have been implemented following the publication of [P1].

The references relevant to this chapter are given in the individual sections. Note that the adjoint approach to lattice Boltzmann used in [P1] is not covered here, since it was found to be difficult to extend and maintain the code implementing this approach. A more robust and general approach was subsequently developed; this approach is covered in chapter 6.

5.1 Lattice Boltzmann for porous media

As mentioned in section 4.2, it is necessary to distinguish between which design nodes are fluid, solid, or “grey”. Since the standard lattice Boltzmann method described in chapter 3 can only represent the fluid state, some modification of the method is needed in order to achieve a smooth transition between states.

In previous work on fluid topology optimization with Navier-Stokes based finite element solvers, this problem has been solved by modeling the design domain as a porous material, with varying degrees of porosity in each fluid element. In this approach, the Brinkman equation for fluid flow in porous media should be satisfied in solid regions. It is given by:

$$\frac{1}{\rho} \nabla \rho = -\frac{\nu}{K} \mathbf{u} + \tilde{\nu} \nabla^2 \mathbf{u}, \quad (5.1)$$

where K is the *permeability* of the medium, and $\tilde{\nu}$ the *effective viscosity*. The permeability controls how easily fluid can flow through the porous medium. When using this model for topology optimization, the solid regions are assigned a very low permeability, approximately making the porous medium equivalent to a solid

medium in which no fluid can flow. Note that choosing a too low permeability for solid regions can lead to numerical problems, however. Modeling the solid regions in this way works well in many cases, but is not perfect since it does not prevent diffusion in the solid regions. For further details on how to implement this approach in a finite element based Navier-Stokes solver, the reader is referred to the literature, e.g. the original paper by Borrvall and Petersson [55].

Implementing the porous media based approach in the lattice Boltzmann method does not require any extensive modification of the core method. In fact, one of the much touted advantages of the method is that it handles complicated porous media well. In the literature, various porous media modifications of the method are available [25–27, 82]. In the present work, the so-called *partial bounceback* model presented by Zhu and Ma [27] is used, since it has been found to be very robust and stable. This method is inspired by the bounceback boundary condition described in section 3.6. It works by modifying the collision step, so that in porous nodes, a fraction of the post-collision distributions is bounced back in the opposing velocity direction. In fully solid nodes, the distributions are scattered equally in each direction, which theoretically leads to a permeability of zero. Specifically, the collision step (3.13) is modified as follows:

$$\tilde{f}_i(\mathbf{x}_j, t) = f_i^c + \frac{1}{2}g(s_j)[f_{i'}^c(\mathbf{x}_j, t) - f_i^c(\mathbf{x}_j, t)], \quad (5.2)$$

where

$$f_i^c(\mathbf{x}_j, t) = f_i(\mathbf{x}_j, t) + \Omega(\mathbf{f}(\mathbf{x}_j, t))$$

is the unmodified post-collision state, i' is the index corresponding to the velocity opposite of i , and $g(s_j)$ is a continuous function $g : [0, 1] \rightarrow [0, 1]$. This function is used to control the transition from solid to fluid nodes, ideally in such a manner that intermediate states (“grey” nodes) will be sub-optimal. In structural mechanics, the SIMP (Solid Isotropic Material with Penalization) power-law function [83] is near-ubiquitous for this purpose. Conversely, in fluid mechanics, the following convex function is similarly common:

$$g(s_j) = 1 - s_j \frac{1 + \gamma}{s_j + \gamma}, \quad (5.3)$$

where γ is an adjustable parameter which allows some degree of control over the penalization of grey nodes. This function was introduced by Borrvall and Petersson in their seminal 2003 paper. While (5.3) was originally formulated in the context of minimal pressure drop problems, it is quite versatile, and has been used for all the problems presented in this thesis.

5.2 The unsteady optimization problem

For the optimization problems presented in this thesis, the following general form of the objective function—based on the one given by Kreissl *et al.* [66]—is used:

$$\varphi = \sum_{t=0}^{N_t} z(t, \mathbf{v}, \mathbf{s}), \quad (5.4)$$

where z is some function of the state in each time-step and N_t is the number of time-steps. This general form is sufficiently flexible to express the objectives of all the problems presented in this thesis.

5.3 Obstacle flow control

This section summarizes and discusses the results for the first numerical example presented in [P1].

5.3.1 Problem description

The idea of this problem is to reproduce the downstream vortex flow pattern of flow past a cylinder with a topology optimized obstacle. The problem is setup by initially computing the flow past a cylinder and saving the average velocity in a predefined downstream area in memory. In the subsequent optimization iterations the realized profile of the current design is compared to this reference flow profile. The computational domain for the problem is illustrated in figure 5.1.

The full statement of the optimization problem is given by

$$\begin{aligned} \min_{\mathbf{s}} Z &= \log \sum_{n=N_0}^{N_t} \sum_{j \in M} \frac{\|\mathbf{u}(\mathbf{x}_j, t_n) - \mathbf{u}_{\text{ref}}(\mathbf{x}_j, t_n)\|^2}{N_m M_A}, \\ \text{s.t.} \quad &\begin{cases} G_0 = \frac{1}{N_s} \sum_{k=1}^{N_s} s_k - 0.9 \leq 0, \\ G_1 = \Delta p - \xi \Delta p_{\text{ref}} \leq 0, \\ 0 \leq s_k \leq 1, \quad \forall k = 1, \dots, N_s. \end{cases} \end{aligned} \quad (5.5)$$

Here, M denotes the nodes in the measuring domain, and M_A denotes its area. The deviation from the reference velocity is computed over $N_m = N_t - N_0$ time steps. The problem is constrained by the volume constraint G_0 which constrains the amount of fluid allowed in the design domain, corresponding to the fluid amount around the reference obstacle. Additionally, the constraint G_1 is a pressure drop constraint, with Δp being the time averaged pressure drop in the cavity. The reference pressure drop Δp_{ref} is the pressure drop in the cavity for the cylinder, and ξ is a constant

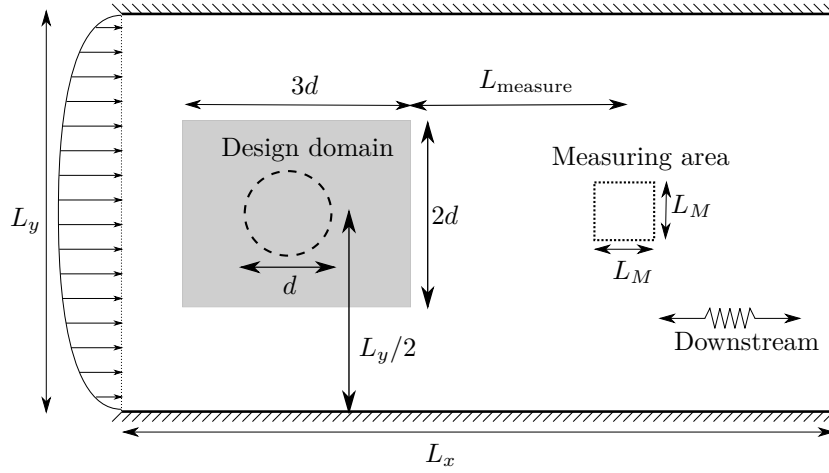


Figure 5.1: Illustration of the computational domain for the obstacle flow control problem. Originally appeared in [P1].

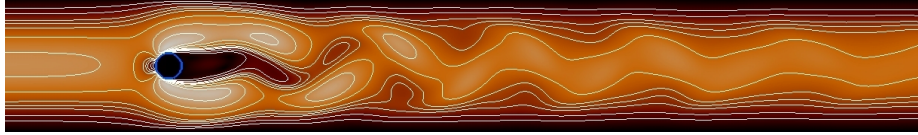
tolerance parameter (above paragraph quoted from [P1]). In words, the object of the problem is to minimize the difference between the flow profile computed with the designed obstacle, and the flow profile precomputed with the cylinder obstacle, i.e. the vortex shedding profile. The logarithm is used to prevent numerical issues close to a difference of zero.

The motivation for this problem was to construct a problem which was suitably challenging, and which had an explicit time-dependent component. The problem has the advantage that it has a known globally optimal solution, while still exhibiting interesting non-linear behavior in the form of the downstream vortex shedding. So while the problem is somewhat academic in nature, the aim was for it to serve as a non-trivial test problem for the unsteady optimization framework.

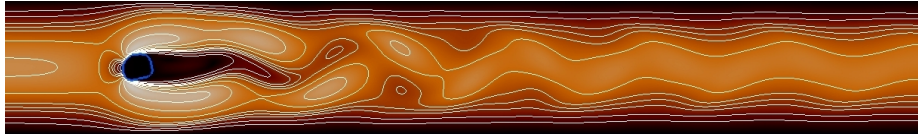
5.3.2 Summary of results

Despite the fact that the problem has a known globally optimal solution, the problem turned out to be more challenging than initially anticipated. Nonetheless, it was shown to be possible to achieve a fairly good reproduction of the desired flow profile. Figure 5.2 gives a comparison of the streamlines for the reference and an example design, while figure 5.3 gives a more quantitative comparison over all the simulation time steps.

It was not managed to recover the known (but not necessarily unique) global optimum—namely the original cylinder—even from initial designs which were only slightly perturbed from the reference.



(a) Reference profile.



(b) Obtained profile.

Figure 5.2: Qualitative comparison of streamlines from the cylinder reference and an example design. Originally appeared in [P1].

5.3.3 Discussion

In order to achieve the results depicted in figures 5.2 and 5.3, the design space had to be severely restricted. Specifically, the pressure drop restriction had to be introduced, and a very large filter radius compared to the size of the design domain was needed. Even with these restrictions in place, convergence to a well defined design was slow and required a very gradual increment of the projection value β (equation (4.6)), otherwise the design could effectively be unrecoverably ruined on the increment step. At Reynolds numbers sufficiently high for vortex shedding dynamics, any given (realistic) downstream flow profile can likely be reproduced to reasonable accuracy with a very large number of possible designs. Indeed, if we consider figure 5.3, it seems that the optimization has converged to a local minimum which reproduces the *period* of the vortex shedding well, but fails to hit the correct *amplitude*. These factors of non-linear behavior and non-uniqueness of solutions combine to make the problem quite challenging.

5.4 Passive fluid pump

This section summarizes and discusses the results for the second numerical example presented in [P1].

5.4.1 Problem description

This problem investigates a simplified model of a passive pump in which the goal is to redirect a cyclical input towards an output channel. The computational domain is depicted in figure 5.4.

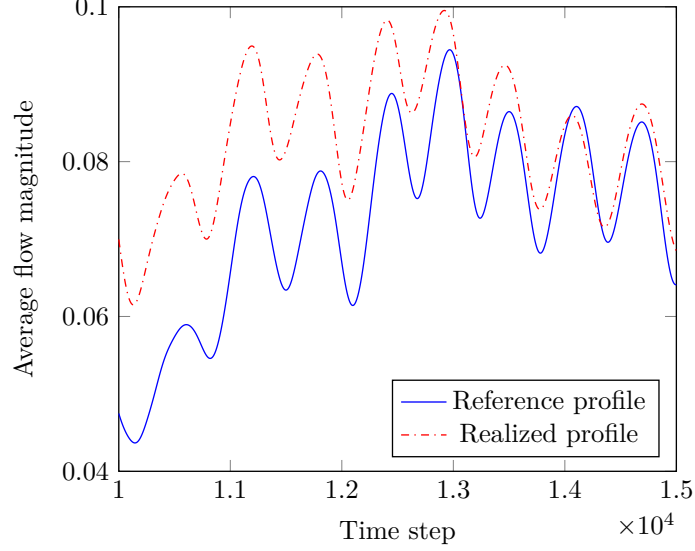


Figure 5.3: Plot of the average velocity magnitude in the downstream measuring area as a function of time, for the cylinder reference and an example design. Originally appeared in [P1].

The problem is formulated as follows:

$$\begin{aligned} \min_{\mathbf{s}} Z &= -\frac{1}{N_t L_{\text{out}}} \sum_{n=0}^{N_t} \sum_{\mathbf{x}_j \in \Gamma_{\text{out}}} u_x(\mathbf{x}_j, t_n), \\ \text{s.t.} \quad &\begin{cases} G_0 = \frac{1}{N_s} \sum_{k=1}^{N_s} s_k - V \leq 0, \\ 0 \leq s_k \leq 1, \quad \forall k = 1, \dots, N_s. \end{cases} \end{aligned} \quad (5.6)$$

Here, Γ_{out} denotes the set of grid points at the pump outflow boundary, and L_{out} denotes the length of the boundary. Once again, a volume constraint on the amount of allowed fluid in the design domain is applied to the problem (quoted from [P1]).

5.4.2 Summary of results

Compared to the cylinder problem described in section 5.3, achieving good results for the pump problem was significantly simpler. Figure 5.5 illustrates the working of an example design via streamlines from both the inflow and outflow cycle of the pump, while figure 5.6 contrasts the output of the pump with the cyclical inflow.

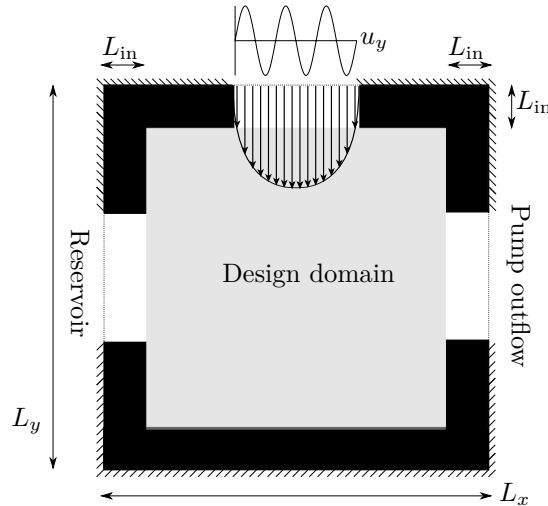


Figure 5.4: Illustration of the computational domain for the pump problem. The cyclical input at the top needs to be redirected towards the “Pump outflow” boundary. Originally appeared in [P1].

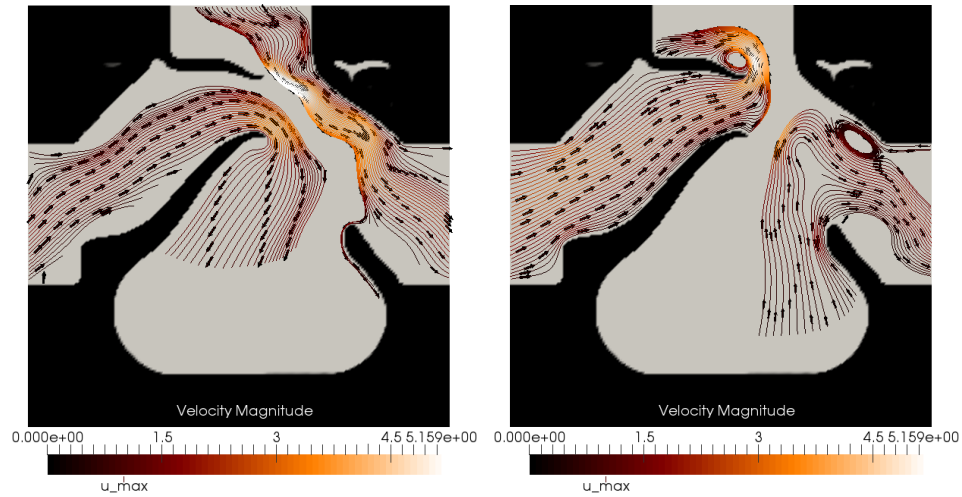
A set of similar results were found: all included the basic feature of an initial narrowing of the inflow channel into a central reservoir, with blocking “arms” on either side to passively control the motion of the fluid.

5.4.3 Discussion

The pump problem is interesting from a topology optimization perspective because the task in this case is to shape the design domain such that the flow exhibits a certain behavior. Essentially, the flow in this case is a means to an end, rather than being the end objective itself, meaning that there is a greater possibility for novel, possibly counterintuitive designs which might otherwise not be imagined. Interestingly, the final design does look to take advantage of vortex-like structures to avoid excessive fluid reentering from the output during the outflow phase.

It is also very interesting from an engineering perspective, since pumps are ubiquitous in many areas of application. Of course, real pumps are more complicated than the simplified model used here, but this also means that the problem is well suited for further extension and refinement. The work which has been done on this is documented in section 5.5.

The paper and the problem introduced within it has generated some interest from the research community. The pump problem was adapted to a 3D finite element implementation by Villanueva and Maute [84]. In addition, going by Google Scholar, at the time of this writing, the paper has been cited in seven other publications,



(a) Example streamlines during the inflow cycle. (b) Example streamlines during the outflow cycle.

Figure 5.5: Representative streamlines during the inflow and outflow cycle of an optimized design. Originally appeared in [P1].

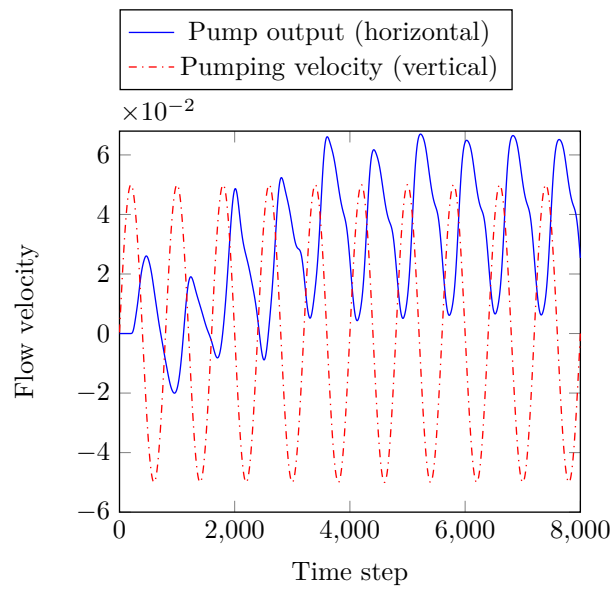


Figure 5.6: Plot of the average flow output on the outflow boundary compared to the cyclical input at the inflow boundary. Originally appeared in [P1].

mostly related to topology optimization for thermofluidic systems. This indicates that there is significant interest in unsteady flow optimization for thermal applications.

5.5 Extension of the pump problem

Following the publication of [P1], various ways to extend the pump problem have been investigated. Rather than giving a full account of the details of these extensions, the aim of this section is to summarize the major findings of this work, discuss which conclusions can be drawn from these findings, and provide an outline for possible future research.

The extensions investigated can be summarized as follows:

1. Applying the robust formulation described in section 4.4.2 in order to obtain a design for the pump which performs robustly under uniform variations.
2. Since real life pumps typically displace fluid against a pressure drop, gravity or other external forces, the pump problem was modified to account for this by implementing a way of modeling external forcing in the simulation.
3. Verifying the physical model by comparing it to the solution computed by a commercial finite element solver.

All these lines of investigation shared a similar simulation setup, which is described next.

5.5.1 Simulation setup

The simulation setup is similar to the one described in [P1] and shown in figure 5.4, but in order to facilitate direct comparison with the solution of a commercial finite element solver, the non-dimensional formulation of the lattice Boltzmann model introduced in section 3.7 is used.

The length of the inflow channel is set to be the reference of length unity in the model. Similarly, the time for completing a single cycle of the top inflow is set to be the reference time, so that one cycle takes one unit of non-dimensionalized time. With the choices in place, the node separation may be computed using (3.19), while the time-step is computed using the diffusive scaling (3.20). Then the number of time-steps for a cycle may be computed as $1/\Delta t$, while the maximal inflow velocity and viscosity may be computed using (3.18).

Unless otherwise noted, the Reynolds number in the results presented below is 80, and the simulation time is five cycles of the inflow. The characteristic length and velocity is the inflow length and maximal inflow velocity, respectively. The flow solutions were computed using the MRT collision operator. In all cases, the strength of the projection is controlled by using the β -continuation strategy described in [P1].

5.5.2 Robust optimization of the pump

The simple robust optimization of the pump is mostly a straightforward extension of the work already presented. Two examples of robustly optimized designs are shown in figures 5.7 and 5.8. The resulting designs are generally share many similar traits with the ones already presented in [P1]. Shown are two different designs at different

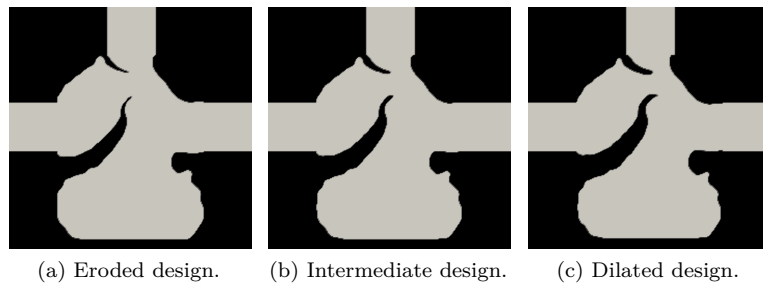


Figure 5.7: Example result of a robust optimization of the pump problem on a 235 by 235 grid. The eroded, intermediate, and dilated designs use the η values 0.3, 0.5, and 0.7, respectively.

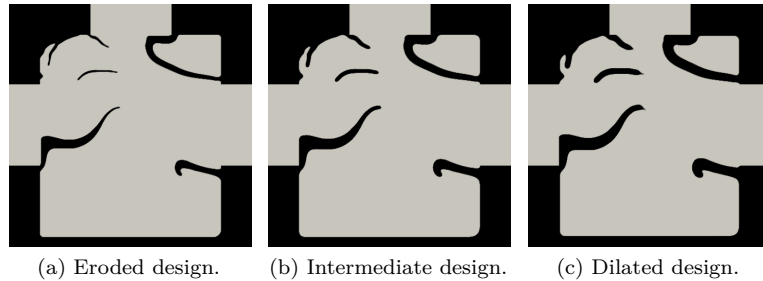


Figure 5.8: Example result of a robust optimization of the pump problem on a 468 by 468 grid. The eroded, intermediate, and dilated designs use the η values 0.3, 0.5, and 0.7, respectively.

levels of mesh refinement. The computational cost of simulating several realizations of the design is mitigated by the fact that each realization can be simulated in an embarrassingly parallel fashion. For example, the optimization result shown in figure 5.8 was computed using 72 cores, with 24 cores assigned to each realization. Once the objective and sensitivities have been computed for all realizations, the MMA algorithm is run on all 72 cores to update the underlying design field.

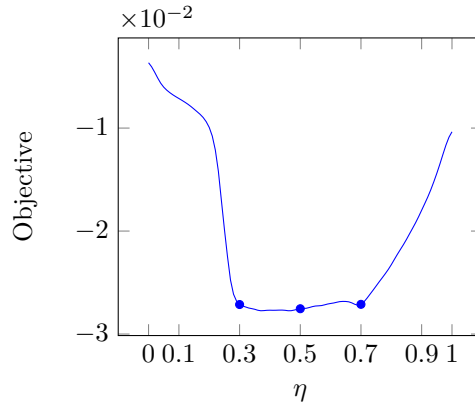


Figure 5.9: Computed objective of the pump design in figure 5.8 as a function of the projection value η . There is a clear degradation in performance outside the range spanned by the three realizations. Each realization is indicated by the marked data points.

The result of the optimization procedure is robust across the range $\eta \in [0.3, 0.7]$, as shown in figure 5.9. Outside of this range, the performance degrades noticeably, although the pumping action (i.e. a net outflow at the right boundary) is retained.

5.5.3 Pumping against an external force

As described in the opening of this section, pumps typically work against an external force which counteracts the fluid displacement induced by the pump. Besides the motivating factor of making the pump model slightly more realistic, it is also interesting from a topology optimization perspective to see if the introduction of such an external force changes the topology of the optimized design in a significant way.



Figure 5.10: Example of an undesirable optimized design for the pump problem with enforced pressure drop.

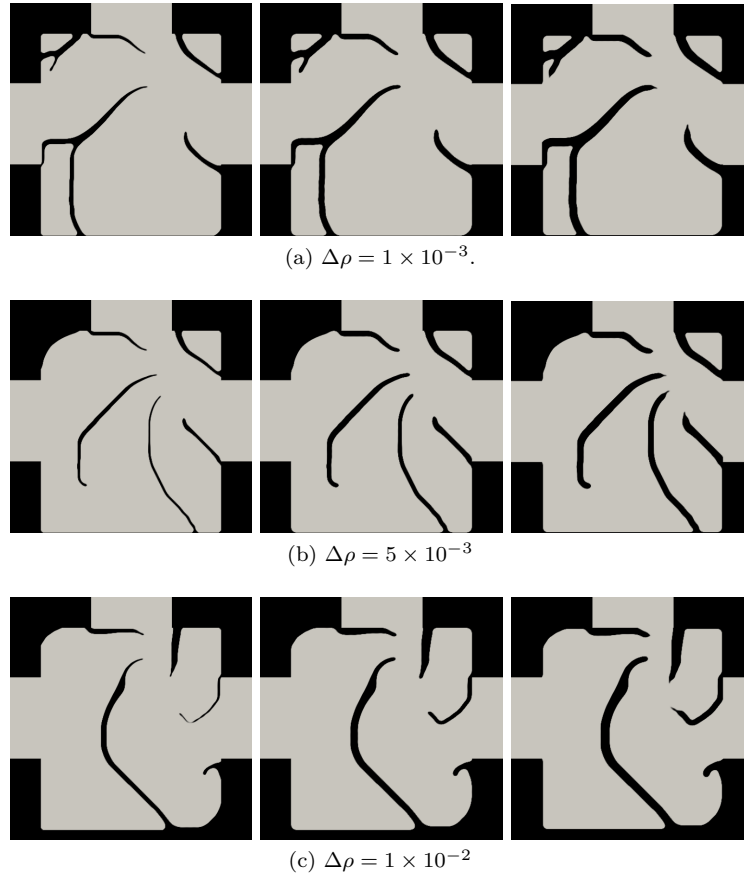


Figure 5.11: Examples of optimized designs on a 468 by 468 grid, for different values of $\Delta\rho$. Like in previous figures, the η values are 0.3, 0.5, and 0.7, respectively.

One way to introduce the external forcing is to enforce a pressure drop across the design domain by using density boundary conditions (recall that density and pressure are proportional in the lattice Boltzmann method) at the two outflows, with a difference $\Delta\rho$ between them. That is, the left outflow will have a constant density $\rho_0 = 1$, while the right will have the density $\rho_0 + \Delta\rho$. One problem with this approach is that it can lead to undesired local minima in which the left boundary is “sealed off”, see figure 5.10 for an example.

This pathological behavior becomes much more likely when $\Delta\rho$ is increased. A likely cause of this is that the cyclical pumping action needs of order one cycle to “get going”, while the initial pressure drop will cause an immediate burst of flow towards the left outflow, causing the optimizer to close it off to prevent the formation

of a strong flow in an undesired direction.

This undesired optimization behavior is mitigated by the following strategy: an initial optimization is performed at a low value of the pressure drop, in this case $\Delta\rho = 1 \times 10^{-3}$. The resulting design variables are then used as the initial guess in a subsequent optimization, this time with $\Delta\rho = 2 \times 10^{-3}$. This procedure is repeated, with $\Delta\rho$ being incremented by 1×10^{-3} , up to a final value of $\Delta\rho = 1 \times 10^{-2}$.

With this strategy in place, no undesired “sealed off” local minima were observed. Some examples of the obtained designs are shown in figure 5.11. Not all the designs in the sequence described are shown, merely representative examples. To further classify the performance of the different designs, in figure 5.12, the performance is plotted as a function of $\Delta\rho$.

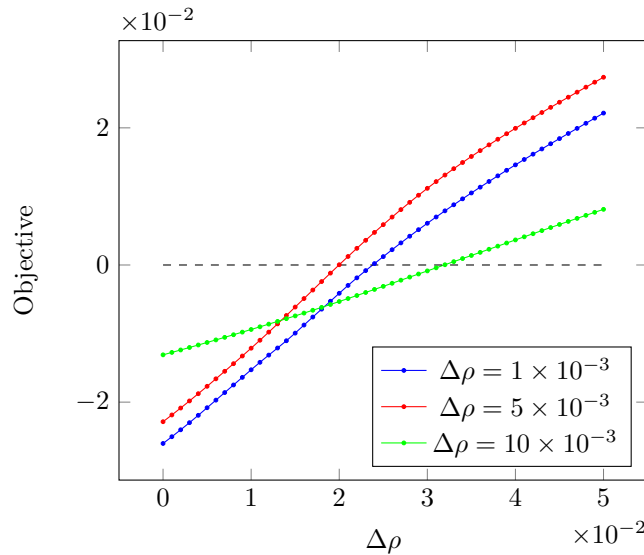


Figure 5.12: Plot of the performance of the designs optimized at different values of $\Delta\rho$, as a function of $\Delta\rho$. The dashed line indicates the threshold for a net outflow from the pump.

The plot reveals a few things of note. First, the design optimized for $\Delta\rho = 5 \times 10^{-3}$ is globally *worse* than the one optimized for $\Delta\rho = 1 \times 10^{-3}$. This indicates that, unsurprisingly, the problem of sub-optimal local minima is still present. Secondly, while the design optimized for $\Delta\rho = 1 \times 10^{-2}$ retains the pumping action for comparatively larger values of $\Delta\rho$ it performs worse than the other designs for lower values of $\Delta\rho$. Note also that the designs optimized for higher $\Delta\rho$ do not actually have a net outflow at the values they are optimized for.

5.5.4 Comparison to a finite element solution

In the course of implementing the extensions described above, it was discovered that in some cases the resulting designs exhibited unphysical behavior, in which the average output of the pump would grow without bound as the number of time-steps increased. This is in some sense an unavoidable consequence of formulation of the problem combined with the potential for unstable solutions in computational fluid dynamics. That is, if during the optimization there is a design which causes numerical instability, that optimization run will likely result in an unphysical result because the instability is beneficial to the objective function. Nonetheless, the occurrence of such designs motivated investigating how well the flow solution as computed by the lattice Boltzmann method matched that of a more conventional method, in the case the finite element method.

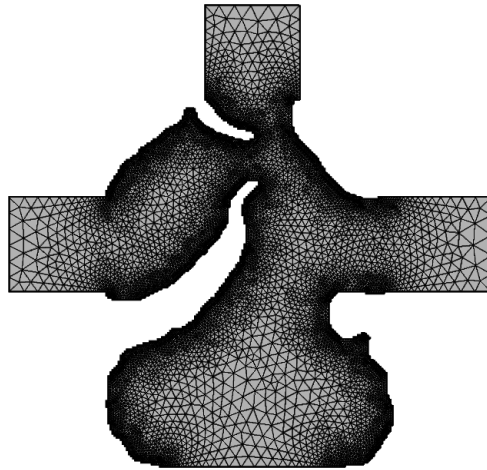


Figure 5.13: Mesh for the pump design as generated by COMSOL Multiphysics.

The design which will be used for comparison is the one shown in figure 5.7b. The finite element solution is computed using COMSOL Multiphysics version 5.2. The COMSOL model is implemented using non-dimensional units, which can be mapped to the corresponding lattice Boltzmann solution as described in section 3.7. The design is converted from the VTK to the STL file format (which COMSOL supports) using Paraview version 4.3.1. The mesh generated by COMSOL is shown in figure 5.13.

The COMSOL model has been solved using the default solver settings found in the laminar flow module. In order to compare the two solutions, the average outflow at the pumping outlet has been plotted as a function of time in figure 5.14.

As is readily apparent from figure 5.14, there is a large discrepancy between the COMSOL and lattice Boltzmann solutions. The exact cause of this is not

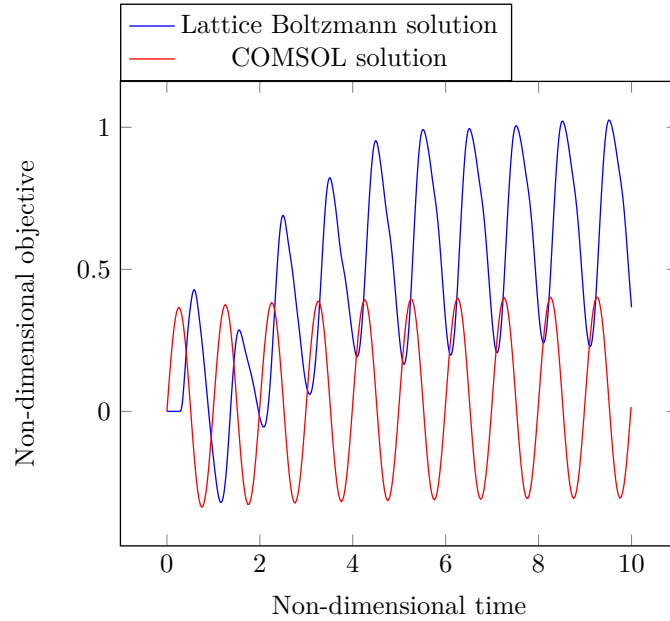


Figure 5.14: Comparison of the average outflow as a function of time, as computed by the lattice Boltzmann method and COMSOL.

known as of this writing. However, figure 5.14 does reveal a crucial difference in the two solutions: the weak compressibility of the lattice Boltzmann method causes a “build-up time” before the final cyclical behavior emerges, whereas the finite element solution immediately exhibits the final cyclical solution. Indeed, discounting the phase shift caused by the initial delay of information transfer from the inflow to the outflow, the final cyclical behavior of the two solutions seem to match up rather well, albeit at different amplitudes. This points to the discrepancy between the two solutions being caused by the weakly compressible nature of the lattice Boltzmann method. It was already mentioned in section 3.8 that completely eliminating the compressibility error associated of the lattice Boltzmann method is known to be impossible, and the result shown here may well be a manifestation of this fact. Which of the two simulations is more accurate—both from a theoretical as well as practical perspective—is currently unknown, and would be an interesting and highly relevant avenue of further research.

6

Topology optimization with automatic differentiation [P2]

This chapter summarizes the automatic differentiation based adjoint approach to the lattice Boltzmann method presented in [P2]. As discussed in the opening of chapter 5, naive application of the adjoint method as presented in section 4.3 results in adjoint code which is difficult to extend and maintain. The approach presented here enables the generation of adjoint code for arbitrary lattice Boltzmann models, though some caveats do apply, especially with regards to the computational efficiency of said adjoint code.

This chapter covers the following: section 6.1 gives a brief introduction to automatic differentiation by means of simple examples, based primarily on the textbook by Griewank and Walther [85]; section 6.2 summarizes the main result from [P2], and provides some more in-depth details not found in the paper; similarly, section 6.3 provides details on how to implement adjoint boundary conditions, which was also not covered in great detail in the original paper; finally, section 6.4 offers some closing comments and discussion.

As a final comment, note that the method presented here (as well as the naive approach used in [P1]) are examples of *discrete* adjoint methods; this means that the adjoint problem (4.5) is derived from the discretized version of the PDE to be solved. This is also sometimes referred to as a “discretize then optimize” approach in the literature. Conversely, an “optimize then discretize”, or *continuous* adjoint approach involves deriving the adjoint problem directly from the continuous PDE. This approach is not covered in this thesis, but for the interested reader a few journal papers describing continuous adjoint methods for lattice Boltzmann are available [61, 65, 86].

6.1 Automatic differentiation by example

Consider a function $F : \mathbb{R}^n \rightarrow \mathbb{R}^m$, with a well-defined Jacobian $F' : \mathbb{R}^n \rightarrow \mathbb{R}^{m \times n}$. This function is referred to as the *primal function* (i.e. the function to be differentiated). We can imagine evaluation of this function taking place on some abstract computational machine in a finite number of underlying computations; this set of computations is denoted the *evaluation trace*. The reader may refer to [85] for a formal definition of the evaluation trace. Each step in the evaluation trace should be simple to differentiate on their own. It is then possible to differentiate the (arbitrarily complicated) function F by computing a series of simple derivatives and propagating them along the evaluation trace via the chain rule. This powerful idea is demonstrated by example in the following. These examples originally appeared in [P2].

6.1.1 Forward mode

In the *forward mode*, the computation proceeds starting from the input vector $x \in \mathbb{R}^n$. Consider the function

$$\begin{aligned} f(x) : \mathbb{R} &\rightarrow \mathbb{R}^2, \\ y_1 &= \cos(\cos(x)), \\ y_2 &= \exp(y_1). \end{aligned} \tag{6.1}$$

An evaluation trace for (6.1) may be written like so:

$$\begin{aligned} v_1 &= \cos(x), \\ v_2 &= y_1 = \cos(v_1), \\ v_3 &= y_2 = \exp(v_2). \end{aligned} \tag{6.2}$$

The entire trace may then be differentiated by differentiating the intermediate variable v_i and applying the chain rule:

$$\begin{aligned} \dot{v}_1 &= -\sin(x), \\ \dot{v}_2 &= \dot{y}_1 = -\sin(v_1)\dot{v}_1, \\ \dot{v}_3 &= \dot{y}_2 = \exp(v_2)\dot{v}_2, \end{aligned} \tag{6.3}$$

in this way, even though the full expression for $f'(x)$ is never “written down”, its exact value is nevertheless obtained at x .

In general, the forward mode evaluates the expression:

$$\dot{y} = F'(x)\dot{x}, \tag{6.4}$$

where $\dot{x} \in \mathbb{R}^n$ is denoted the *seed direction*.

6.1.2 Reverse mode

As one might expect, in the *reverse mode*, the differentiation starts from the output vector $y = f(x) \in \mathbb{R}^m$. Define the *adjoint variable*, $\bar{v}_i = \partial y / \partial v_i$, which may be computed by going through the evaluation trace in reverse order¹.

Consider the function

$$\begin{aligned} g(x_1, x_2) : \mathbb{R}^2 &\rightarrow \mathbb{R}, \\ y &= \cos(\cos(x_1 x_2^2)), \end{aligned} \tag{6.5}$$

¹Formally, the adjoint variables actually denote the *variation*, $\bar{v}_i = \partial y / \partial \delta_i$. See [85] for full details.

with the evaluation trace

$$\begin{aligned}
v_1 &= x_1, \\
v_2 &= x_2, \\
v_3 &= v_2^2, \\
v_4 &= \cos(v_1 v_3), \\
v_5 &= \cos(v_4).
\end{aligned} \tag{6.6}$$

The reverse mode is now computed by evaluating \bar{v}_i , starting from the end:

$$\begin{aligned}
\bar{v}_5 &= 1, \\
\bar{v}_4 &= -\bar{v}_5 \sin(v_4), \\
\bar{v}_3 &= -\bar{v}_4 \sin(v_1 v_3) v_1, \\
\bar{v}_2 &= \frac{\partial g}{\partial x_2} = 2\bar{v}_3 v_2, \\
\bar{v}_1 &= \frac{\partial g}{\partial x_1} = -\bar{v}_4 \sin(v_1 v_2) v_3.
\end{aligned} \tag{6.7}$$

In general, reverse mode evaluates the expression:

$$\bar{x}^T = \bar{y}^T F'(x), \tag{6.8}$$

where $\bar{y} \in \mathbb{R}^m$ is denoted the *weight functional*.

6.1.3 Closing remark

The point of these simple examples is not to show some alternative way to derive derivatives, but that the machinery powering the forward and reverse modes can be automated and implemented in software. Discussing implementation details of automatic differentiation packages will be omitted here, but the reader may refer to [P2] for a slightly more in-depth discussion, and of course the already cited [85] for thorough coverage. In addition, a list of implementations for many programming languages may be found at the website [87].

6.2 Adjoint method with automatic differentiation

As mentioned, this section summarizes the main result of [P2]. This summary is given in subsection 6.2.1. The following subsection provides some additional details not found in [P2], namely an alternative way of deriving the adjoint streaming step, while subsection 6.2.3 discusses checkpointing strategies.

6.2.1 Main result

The main idea of the derivation, following Liu *et al.* [82], is to split the residual $R(\mathbf{v}, \mathbf{s})$ into a series of computational steps, and then introduce Lagrange multipliers for each step. That is, for the lattice Boltzmann method, the method is split into the collision, streaming, and boundary condition step, yielding the following Lagrange multiplier objective:

$$\hat{\varphi} = \sum_{t=0}^{N_t} z(t, \mathbf{f}, \mathbf{s}) + \boldsymbol{\lambda}_t^T R_t^{\text{stream}} + \boldsymbol{\sigma}_t^T R_t^{\text{bc}} + \boldsymbol{\tau}_t^T R_t^{\text{collision}}, \quad (6.9)$$

where the t subscript denotes the value of the vector in time-step t , and $\boldsymbol{\lambda}$, $\boldsymbol{\sigma}$, and $\boldsymbol{\tau}$ are all distinct Lagrange multipliers. Note that the notation \mathbf{f} is here preferred over \mathbf{v} for the state variable, since this is the common notation in the lattice Boltzmann literature. From here, the derivation proceeds equivalently to the one starting from (4.2) in section 4.3, and the reader is referred to [P2] for the full details. The end result is the following adjoint collision step:

$$\boldsymbol{\sigma}(\mathbf{x}_j, t)^T = \boldsymbol{\tau}(\mathbf{x}_j, t)^T \frac{\partial \Omega[\mathbf{f}(\mathbf{x}_j, t), s_j]}{\partial \mathbf{f}(\mathbf{x}_j, t)} + \frac{\partial z(t, \mathbf{f}_t, \mathbf{s})}{\partial \mathbf{f}(\mathbf{x}_j, t)}. \quad (6.10)$$

Following the adjoint collision step, we have the adjoint boundary step

$$\boldsymbol{\lambda}(\mathbf{x}_j, t)^T = \boldsymbol{\sigma}(\mathbf{x}_j, t)^T \frac{\partial \psi[\mathbf{f}^{\text{stream}}(\mathbf{x}_j, t)]}{\partial \mathbf{f}^{\text{stream}}(\mathbf{x}_j, t)}. \quad (6.11)$$

and finally the adjoint streaming step

$$\boldsymbol{\tau}_i(\mathbf{x}_j, t) = \lambda_i(\mathbf{x}_i - \mathbf{e}_i \Delta x, t - \Delta t), \quad (6.12)$$

that is, the adjoint streaming step is backwards in time and in the opposite direction of the primal streaming step. Above \mathbf{f} denotes the distribution values at the beginning of a new time step, while $\mathbf{f}^{\text{stream}}$ denotes the distribution values following the streaming step, but prior to the application of boundary conditions. The operator ψ denotes a generic boundary condition operator (see section 6.3 for a specific example of adjoint boundary conditions).

Finally, the sensitivities may be evaluated by the expression

$$\frac{\partial \hat{\varphi}}{\partial s_j} = \sum_{t=0}^{N_t} \frac{\partial z}{\partial s_j} + \boldsymbol{\tau}^T(\mathbf{x}_j, t) \frac{\partial \Omega[\mathbf{f}(\mathbf{x}_j, t), s_j]}{\partial s_j}. \quad (6.13)$$

The important point is that the expressions (6.10) and (6.13) (which are both highly non-linear) both contain terms which are exactly of the form (6.8), and can therefore be evaluated with the reverse mode of automatic differentiation.

It is important to note that since the collision step is purely local, automatic differentiation is only applied on local operations as well. In this way a large limitation of many automatic differentiation packages is bypassed: namely that they do not directly support external libraries or parallel primitives such as MPI calls. This is ideal in our case since the parallel implementation is based on the PETSc framework [81].

6.2.2 A note on adjoint streaming

The adjoint streaming step (6.12) is derived in [82], and the result was stated without derivation in [P2]. Their derivation—while correct—makes use of a slightly counter-intuitive assumption about the extent of the computational domain, and also does not describe how to deal with bounce-back boundaries of the form described in section 3.6.1, at least to the authors understanding. Here, an outline of a slightly different derivation is given, which accounts explicitly for the bounce-back boundary conditions as described in this thesis.

The idea of the derivation is that the streaming and bounce-back steps are simply a permutation of the vector of post-collision distribution values. Thus, the residual of the streaming step may be written as a matrix-vector product, like so:

$$R_t^{\text{stream}} = \mathbf{M}^{\text{stream}} \mathbf{f}_t - \mathbf{f}_t^{\text{stream}} = \mathbf{0}, \quad (6.14)$$

where $\mathbf{M}^{\text{stream}}$ is the sparse permutation matrix for the streaming and bounce-back step. When deriving the adjoint method, the relevant term is then the summation

$$\sum_{t=0}^{N_t} \frac{\partial \hat{\varphi}}{\partial \mathbf{f}_t} \frac{\partial \mathbf{f}_t}{\partial s_j} = \sum_{t=0}^{N_t} \boldsymbol{\lambda}_t^T \mathbf{M}^{\text{stream}} - \boldsymbol{\tau}_t^T = \mathbf{0}, \quad (6.15)$$

which, since the summands are mutually independent, implies the adjoint streaming step

$$\boldsymbol{\tau}_t = (\mathbf{M}^{\text{stream}})^T \boldsymbol{\lambda}_t. \quad (6.16)$$

In other words, the permutation matrix for the adjoint streaming and bounce-back step is simply the transpose of the primal matrix. This results in exactly the streaming step (6.12) in the interior, and an adjoint bounce-back which is opposite in direction to the primal bounce-back step, as one might intuitively expect.

6.2.3 Checkpointing

Though it might not be immediately clear from the above equations, the above adjoint method (indeed any adjoint method for a time-stepping scheme) must be evaluated “backwards in time”. That is, the evaluation of the Lagrange multipliers must start at the final time-step N_t . Since the collision operator Ω is non-linear in \mathbf{f} , the adjoint collision step (6.10) will itself be a function of \mathbf{f} . This implies

that the time history of \mathbf{f} must be either stored in memory or made available by recomputation during the adjoint evaluation. Note that this is not the case for the adjoint streaming step since it is not a function of \mathbf{f} , while for the adjoint boundary step it depends on whether ψ is non-linear in \mathbf{f} , see section 6.3.

Setting aside the question of the adjoint boundary conditions for the moment, the pre-collision distribution values must in any case be available to compute the adjoint collision step. Naively storing the whole history in memory does not scale well, but recomputation increases the computational cost of the adjoint method. Fortunately, there are algorithms available which minimize the amount of recomputation necessary to complete the full adjoint evaluation. An early example of such an algorithm is REVOLVE [88], which computes a provably optimal layout of *checkpoints*, from which recomputation takes place. For the work presented in this thesis, the newer algorithm by Wang *et al.* [89] has been used. This computes the same optimal checkpoint layout as REVOLVE, but has the added feature that the total number of time-steps need not be known in advance. We shall use this to our advantage in the thermal problem presented in chapter 7.

6.3 Adjoint boundary conditions

Like the adjoint collision step (6.10), the adjoint boundary step (6.11) is of the form (6.8), which means it can be evaluated by the reverse mode of algorithmic differentiation. For relatively simple boundary conditions such as the Zou/He boundary conditions (3.17), which are linear in \mathbf{f} , it is generally relatively easy to derive and implement the adjoint boundary conditions by hand. For the concrete example at hand—equation (3.17)—one may derive the following adjoint boundary condition:

$$\begin{aligned}
 C(\mathbf{x}_j, t) &= \frac{1}{1 - \bar{u}_x}, \\
 \tilde{\rho}(\mathbf{x}_j, t) &= \left(\frac{\bar{u}_x}{6} - \frac{\bar{u}_y}{2} \right) C\sigma_1 + \left(\frac{\bar{u}_x}{6} + \frac{\bar{u}_y}{2} \right) C\sigma_7 + \frac{2\bar{u}_x}{3} C\sigma_8, \\
 \lambda_0(\mathbf{x}_j, t) &= \sigma_0 + \tilde{\rho}, \\
 \lambda_2(\mathbf{x}_j, t) &= \sigma_2 + \frac{1}{2}(\sigma_7 - \sigma_1) + \tilde{\rho}, \\
 \lambda_3(\mathbf{x}_j, t) &= \sigma_3 + \sigma_7 + 2\tilde{\rho}, \\
 \lambda_4(\mathbf{x}_j, t) &= \sigma_4 + \sigma_8 + 2\tilde{\rho}, \\
 \lambda_5(\mathbf{x}_j, t) &= \sigma_5 + \sigma_1 + 2\tilde{\rho}, \\
 \lambda_6(\mathbf{x}_j, t) &= \sigma_6 + \frac{1}{2}(\sigma_1 - \sigma_7) + \tilde{\rho}, \\
 \lambda_1(\mathbf{x}_j, t) &= \lambda_7(\mathbf{x}_j, t) = \lambda_8(\mathbf{x}_j, t) = 0.
 \end{aligned} \tag{6.17}$$

Note that the spatial and temporal argument has been omitted from the right-hand side, as in (3.17). Since (3.17) is linear in \mathbf{f} , the adjoint boundary condition is independent of \mathbf{f} . This is advantageous because it means that only the initial state of the distributions variables at time-step t needs to be known to evaluate the adjoint time-step. Thus only this state needs to be considered in the checkpointing algorithm. While there certainly are non-linear boundary conditions available in the literature (e.g. Latt *et al.* [90], Malaspinas *et al.* [91]), and these do have better stability properties than the linear Zou/He boundary conditions, using them would add an additional layer of complexity to the adjoint code, since the post-streaming distribution values would need to be available when computing the adjoint boundary step. This in turn means that these values either need to be stored as a checkpoint or recomputed from other checkpointed values. All of this is certainly possible, but—in the view of the author—the added complexity is not worth it unless the additional stability of these boundary conditions was *absolutely* necessary.

6.4 Discussion

While automatic differentiation has been applied quite extensively for other types of design optimization than topology optimization [92–94], to the authors knowledge, only few papers have applied it to topology optimization [95,96]. Its applicability in any given concrete case of course depends on the difficulty of implementing hand derived adjoints, but the potential utility of completely automating sensitivity computations can scarcely be understated. In the present work, utilizing automatic differentiation did require some up front work, but implementing the approach described above *significantly* reduced the work required for implementing adjoint code going forward, since the approach is generic and can be adapted to any lattice Boltzmann model which follows the “collide and stream” time-stepping structure. A specific example will be shown in chapter 7, where the approach is adapted to a thermal lattice Boltzmann model.

7

Unsteady thermal flow topology optimization

This chapter documents the work done on topology optimization of heat regenerators as part of the thesis work. These results are unpublished, and therefore can be considered wholly new material distinct from what is covered in the publications submitted with this thesis.

Unlike the results presented in the previous chapters, the problem presented here requires simulation of a thermofluidic system. Therefore, section 7.1 covers how to simulate such a system using the lattice Boltzmann method, by means of a so-called double distribution model. Section 7.2 follows on this by discussing the consequences the extension of the lattice Boltzmann method to thermal systems has from a topology optimization perspective. That is, how solid and fluid nodes are handled in the thermal model, and how to adapt the adjoint method presented in chapter 6 to the extended model. Subsequent sections then focus on the actual topology optimization problem: section 7.3 gives a brief introduction to regenerator devices; section 7.4 describes how a topology optimization problem for the regenerator was formulated and implemented; based on these initial findings, section 7.5 extends the problem to robust optimization for non-uniform variations; finally, section 7.6 discusses various ways to further investigate the problem beyond the initial findings presented here.

The theoretical sections of this chapter are mainly based on various papers on thermal lattice Boltzmann models [97–99], as well as the book [100]. Like in previous chapters, the main results are simply stated, without going into the details of the theoretical derivation. The focus is exclusively on the double distribution approach to thermal lattice Boltzmann. An alternative method for implementing thermal lattice Boltzmann models is the so-called *multi-speed* approach, in which the lattice is expanded to include velocities that travel farther than to nearest neighbor nodes in a single time-step [101]. This type of model has not been considered for this work since it forces a constant Prandtl number and is reported to have poor stability properties [102].

7.1 Thermal lattice Boltzmann model

The *double distribution* lattice Boltzmann model, as the name implies, adds another set of distribution values to the lattice Boltzmann model, which in this case governs the dynamics of the temperature field in a thermofluidic model. The additional set of distribution values are advanced in time according to their own collision and streaming step, coupling to the isothermal model through the macroscopic values. In this thesis, the temperature dynamics of interest are governed by the advection-

diffusion equation (2.5), with a one-way coupling achieved by the presence of the fluid velocity \mathbf{u} in (2.5).

To separate the distributions associated with the temperature from the “standard” distributions associated with density and velocity, the following notation shall be used:

$$g_i(\mathbf{x}_j + \mathbf{c}_i \Delta t, t + \Delta t) = g_i(\mathbf{x}_j, t) + \Pi(\mathbf{g}(\mathbf{x}_j, t)), \quad (7.1)$$

where $\Pi(\mathbf{g}(\mathbf{x}_j, t))$ is the thermal collision operator. The temperature T is computed like the density in standard lattice Boltzmann, that is

$$T(\mathbf{x}_j, t) = \sum_i g_i(\mathbf{x}_j, t). \quad (7.2)$$

7.1.1 Thermal lattice

Because the advection-diffusion equation is a simpler dynamical equation than the Navier-Stokes equation (2.3), a correspondingly simpler lattice than the D2Q9 lattice can be used. For this thesis, the D2Q5 lattice has been used in the thermal simulations. Its weights and velocities are listed in equation (7.3) and it is illustrated in figure 7.1.

$$\mathbf{c}_i = \frac{\Delta x}{\Delta t} \begin{cases} (0, 0), & i = 0, \\ (\pm 1, 0), (0, \pm 1), & i = 1, 2, 3, 4; \end{cases} \quad (7.3a)$$

$$w_i = \begin{cases} \frac{1}{3}, & i = 0, \\ \frac{1}{6}, & i = 1, 2, 3, 4; \end{cases} \quad (7.3b)$$

$$c_s = \frac{\Delta x}{\Delta t} \frac{1}{\sqrt{3}}. \quad (7.3c)$$

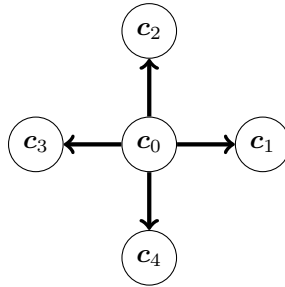


Figure 7.1: Visual illustration of the D2Q5 model.

Clearly, the D2Q5 lattice is simply the D2Q9 without diagonal velocities (and different corresponding weights).

7.1.2 Thermal collision operators

The simplest thermal collision operator available is similar to the BGK operator (3.7). It is given by

$$\Pi_i^{\text{BGK}} = -\frac{1}{\tau'}(g_i - g_i^{\text{eq}}), \quad (7.4)$$

with the equilibrium function

$$g_i^{\text{eq}} = w_i T \left(1 + \frac{\mathbf{c}_i \cdot \mathbf{u}}{c_s^2} \right). \quad (7.5)$$

Likewise, an MRT-like operator is available for the thermal lattice Boltzmann model [98]. It is given by

$$\Pi^{\text{MRT}}(\mathbf{g}(\mathbf{x}_j, t)) = \mathbf{M}^{-1} \mathbf{S} \mathbf{M} (\mathbf{g} - \mathbf{g}^{\text{eq}}), \quad (7.6)$$

with

$$\mathbf{M} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 & -1 \\ -4 & 1 & 1 & 1 & 1 \\ 0 & 1 & -1 & 1 & -1 \end{pmatrix}, \quad (7.7)$$

$$\mathbf{S} = \text{diag}(0, s_{12}, s_{12}, s_3, s_4). \quad (7.8)$$

Like in the standard lattice Boltzmann method, the relaxation time is related to a macroscopic material parameter, in this case the diffusivity:

$$\tau' = s_{12} = c_s^2 D \frac{\Delta t}{\Delta x^2} + \frac{1}{2}. \quad (7.9)$$

7.1.3 Thermal boundary conditions

Since the D2Q5 does not contain diagonal velocities, there is only one unknown distribution value after streaming on non-corner boundaries. This makes it trivial to implement a Zou/He style boundary conditions for fixed temperature boundaries. In addition, bounce-back boundary conditions can be used to approximate Neumann boundary conditions [98]. This approach has been found to be inaccurate, however, in cases where there is a strong fluid flow (i.e. a high value of $|\mathbf{u}|$) on the boundary. In that case, it is better to use the method for Neumann boundaries given by Malaspinas [103].

7.2 Thermal lattice Boltzmann for topology optimization

In the present case of applying the thermal lattice Boltzmann method to topology optimization problems, there are a few additional considerations to take into account.

7.2.1 Material interpolation

Like in the case of the isothermal problems, where the partial bounceback model described in section 5.1 was used to facilitate a smooth transition between the solid and fluid states, the thermal model needs to somehow distinguish between these states as well. This is achieved by adapting the method used by Alexandersen *et al.* [63] to the lattice Boltzmann model. In this approach, the thermal diffusivity varies smoothly between the diffusivity of the fluid and that of the material. The ratio between their diffusivities are denoted

$$C_D = \frac{D}{D_{\text{mat}}}, \quad (7.10)$$

where D_{mat} is the diffusivity of the material. The diffusivity of node j is then given by

$$h(s_j) = \frac{s_j(C_D(1 + \xi) - 1) + 1}{C_D(1 + \xi s_j)}, \quad (7.11)$$

$$D(s_j) = Dh(s_j), \quad (7.12)$$

where $h(s_j)$ is a so-called RAMP interpolation function [104], and ξ is a convexity factor which can be adjusted to penalize intermediate variables. Unlike the partial bounceback model, this approach does not directly alter the thermal collision operator Π directly. Rather, the prior to the collision step, the diffusivity in the node is computed with (7.12), and this value is then used to compute a node-local relaxation time using (7.9). The collision step is then computed as normal.

7.2.2 Adjoint thermal lattice Boltzmann

The automatic differentiation approach presented in chapter 6 is perfectly applicable to the thermal lattice Boltzmann model, since it does not change the fundamental “collide and stream” execution of the algorithm. In the adjoint streaming step, there are simply a larger set of distributions which will be streamed backwards in space and time, and the two collision steps can be concatenated into a single step like so:

$$\Omega_{\text{total}}([\mathbf{f}(\mathbf{x}_j, t) \quad \mathbf{g}(\mathbf{x}_j, t)]^T) = [\Omega(\mathbf{f}(\mathbf{x}_j, t)) \quad \Pi(\mathbf{g}(\mathbf{x}_j, t))]^T. \quad (7.13)$$

This “total” collision operator is then readily differentiated (as required in the adjoint collision step (6.10)) by the reverse mode of automatic differentiation.

7.3 Regenerators

In this chapter, the thermal devices for which optimization will be attempted are the so-called *regenerative heat exchangers*, commonly known simply as *regenerators*. As the name suggests, regenerators are a specific type of heat exchanger. In regenerators,

heat from a hot fluid is stored in an intermittent storage medium, called the regenerator matrix, before it is transferred to a cold fluid. The device functions by fluid flowing through it in two phases: first, hot fluid flows through the device, transferring its heat to the storage medium inside; second, cold fluid flows through, absorbing the heat stored in the medium. Regenerators are used in a variety of important applications, e.g. dehumidifiers, electronics cooling, and magnetic refrigeration.

Heat regenerators are typically grouped into three classes: monolithic, stacked parallel plates, and packed spheres. Detailed discussion of the advantages and disadvantages of each class of design is beyond the scope of this thesis, but the reader may refer to e.g. Duprat and Lopez for an analysis of the trade-offs associated with each design [105]. In this thesis, the focus will be on the parallel plate regenerator, and this will be the starting point from which optimization will be performed.

The performance of regenerators may be classified by means of a so-called *single blow* experiment [106]. The idea of this experiment is to pump fluid of a fixed temperature through the regenerator, then forcing a sudden change in the temperature of the fluid, and measuring the temperature response at the outlet of the regenerator. A good regenerator should then equilibrate to the new temperature as fast as possible, since a short equilibration time indicates that the transfer of heat from the regenerator matrix happens swiftly. A simple model of the single blow experiment shall be used as the basis for the optimization process.

7.4 Topology optimization of regenerators

In this section the basic setup of the optimization problem as well as some initial results are covered.

7.4.1 Simulation setup

We consider a simplified model of a single blow experiment for a parallel plate regenerator. The simulation domain is depicted in figure 7.2. In this model, the fluid is flowing from left to right, driven by a fixed density drop $\Delta\rho$ (recall that this is equivalent to a pressure drop) imposed by density boundary conditions on both ends. The full simulation proceeds according to the following steps:

1. If this is the first iteration, the domain is initialized with zero velocity everywhere, but with a linearly decreasing density from the inflow to the outflow. Otherwise, the steady state solution from the previous iteration is used to initialize the flow. The temperature is initialized to a uniform value in the entire domain.
2. A steady state flow solution is computed at uniform temperature. This is done by a “pseudo time-stepping” approach, i.e. the flow is evolved forward in time using the standard lattice Boltzmann time-stepping algorithm until a steady

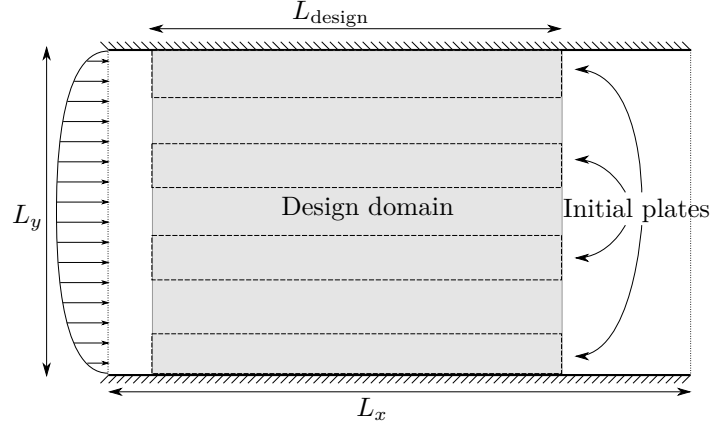


Figure 7.2: Illustration of the computational domain for the regenerator problem. The dashed lines show an example of an initial parallel plate design, but this is merely used as an initial guess for the optimization.

state is reached. The flow is considered to be in steady state once the criterion $\|\mathbf{f}_t - \mathbf{f}_{t-1}\|_\infty < \epsilon$ is satisfied, where ϵ is some small tolerance.

3. Once the steady state solution is found, the temperature at the inflow is changed suddenly, and the flow simulation is allowed to run for an additional fixed number of time steps, in order to analyze the transient response to the sudden temperature change.

The dynamics of both the isothermal and thermal distributions are simulated using their corresponding MRT collision operator. The advantage of the above approach is that it allows for full reuse of the machinery already developed for the isothermal problems. As mentioned in section 6.2.3, the checkpointing algorithm [89] which has been implemented for this work does not need to know a priori the exact number of time-steps performed in the simulation. If the steady state analysis takes N_{steady} time-steps, and the fixed number of time-steps for the transient analysis is $N_{\text{transient}}$, the adjoint analysis can step backward through the history of the $N_t = N_{\text{steady}} + N_{\text{transient}}$ steps “as if” the number N_t had been known all along.

As a final implementation detail, in order to avoid numerical difficulties, we do not use a true step change in the temperature at the inflow, but rather the temperature

change happens according to

$$T(t) = \begin{cases} T_0, & t \leq N_{\text{steady}}, \\ T_0 + \Delta T \sin\left(\frac{\pi(t - N_{\text{steady}})}{2N_{\text{ramp}}}\right), & N_{\text{steady}} < t \leq N_{\text{steady}} + N_{\text{ramp}}, \\ T_0 + \Delta T, & N_{\text{steady}} + N_{\text{ramp}} < t \leq N_t, \end{cases} \quad (7.14)$$

where N_{ramp} is some fixed (small) number of time-steps in which the temperature change happens, and ΔT is the temperature change.

7.4.2 Objective function

In order to actually optimize a design using the simulation method described above, some figure of merit for the performance of the regenerator is still needed. In other words, an objective function needs to be formulated. According to Jensen *et al.* [107], the most general figure of merit is the so-called *regenerator efficiency*, defined as the ratio between the total energy transferred from the regenerator matrix to the fluid in a single blow to the total energy stored in the matrix, i.e.

$$\chi = \frac{E_{\text{transferred}}}{E_{\text{total}}}, \quad (7.15)$$

such that a theoretical perfect regenerator would have an efficiency of 1. Jensen *et al.* compute the effectiveness as a function of time using the formula

$$\chi(t) = \frac{\int_0^t \dot{q}_t dt}{E_{\text{total}}}, \quad (7.16)$$

where \dot{q}_t is the heat transfer rate across the plate/channel interface. Computing the efficiency in this manner would be quite difficult in the present context of density based topology optimization, since some approximate scheme for tracking the boundary or boundaries between the fluid and the solid matrix would be necessary. The two papers [105, 106] offer some alternative figures of merit for heat regenerators. Based on these papers, we formulate the following approximation of the regenerator efficiency:

$$\tilde{\chi}(t) = \frac{\sum_{j \in \Omega_d} (1 - s_j)(T(t) - T_0)}{\sum_{j \in \Omega_d} (1 - s_j)\Delta T}. \quad (7.17)$$

In this formulation, the efficiency is approximated as the ratio of the weighted sum of the temperature change in each node, over the weighted sum of the theoretical maximum in each node. The weight $(1 - s_j)$ is there since only the temperature of the solid matrix is of interest. In the topology optimization problems where (7.17) was applied, the actual objective function is $\tilde{\chi}^{-1}$, evaluated in the final time-step. In addition, in all the results presented below, a volume constraint is in place which constrains the material fraction allowed in the design domain to be equal to the fraction of the initial guess.

7.4.3 Results

The result of a “standard” (i.e. not robust) optimization is shown in figure 7.3. The relevant numerical parameters are in table 7.1.

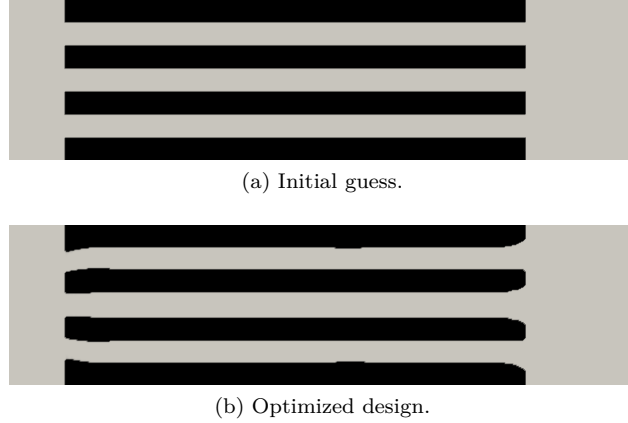


Figure 7.3: Result of simple topology optimization on the regenerator problem.

| | | | |
|------------------------|--------|-------------------|--------------------|
| L_x | 550 | L_y | 140 |
| L_{design} | 350 | $\Delta\rho$ | 0.05 |
| T_0 | 2 | ΔT | 2 |
| $N_{\text{transient}}$ | 25000 | N_{ramp} | 200 |
| ξ | 1 | γ | 1 |
| Re | 30 | Pr | 0.71 |
| C_D | 0.0025 | ϵ | 1×10^{-7} |

Table 7.1: Numerical parameters for the regenerator problem. All physical parameters are in lattice units. For the Reynolds number, the characteristic length is the inflow length, and the characteristic velocity is the maximum velocity of a Poiseuille flow at density drop $\Delta\rho$.

The result in figure 7.3 seems to indicate that the initial parallel plate solution is already very close to a local minimum. The predicted improvement in regenerator efficiency of the optimized design is less than one percent. This is supported by the following quote from Jensen *et al.* [107]:

“Parallel-plate regenerators with small dimensions (...) are receiving interest for several application because of their theoretically high thermal performance (...) with low pressure drops. (...) Although the theoretical performance of parallel plate regenerators is high, the experimentally

measured performance is typically less than what is expected. It is generally accepted that flow maldistribution of the heat transfer fluid caused by non-uniformity in the flow channel widths is one of the causes of this discrepancy.”

While the above quote indicates that the parallel plate design is already very good for its purpose, it suggests that the design is not robust to non-uniform variations. This motivates the work presented in the next section.

7.5 Robust optimization of regenerators

Motivated by the findings of the previous section, this section describes the application of non-uniform robust topology optimization [78] to the regenerator problem.

7.5.1 Non-uniform variations

The robust optimization for non-uniform variations still makes use of the robust formulation (4.9), but the projection variable η is now made a function of space, $\eta \mapsto \eta(\mathbf{x})$.

The projection field is computed in the following way: for each realization in the robust optimization, evaluate the function

$$Y(x, y) = \sum_{i=1}^m \sigma_i^x (A_i \cos(\omega_i^x x) + B_i \sin(\omega_i^x x)) + \sum_{j=1}^n \sigma_j^y (C_j \cos(\omega_j^y y) + D_j \sin(\omega_j^y y)), \quad (7.18)$$

where the factors $\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D} \sim \mathcal{N}(0, 1)$ are normally distributed random variables, and the constraint

$$\sum_{i=1}^m (\sigma_i^x)^2 + \sum_{j=1}^n (\sigma_j^y)^2 = 1, \quad (7.19)$$

is satisfied. The variables ω^x and ω^y are predefined frequencies at which variations occur. For each realization, $Y(x, y)$ is computed, and $\eta(x, y)$ is then determined as

$$\eta(x, y) = \frac{1}{2}(\eta_{\max} - \eta_{\min}) \left(1 + \operatorname{erf} \left(\frac{Y(x, y)}{\sqrt{2}} \right) \right) + \eta_{\min}, \quad (7.20)$$

where erf is the Gauss error function:

$$\operatorname{erf}(x) = \frac{1}{\sqrt{\pi}} \int_{-x}^x \exp(-t^2) dt, \quad (7.21)$$

which is used for computing the cumulative distribution function of the normal distribution. The parameters η_{\min} and η_{\max} are predefined minimum and maximum values for η .

The method of determining $\eta(x, y)$ allows modeling of “wave-like” non-uniform variations. An example application is shown in figure 7.4 for a parallel plate structure.

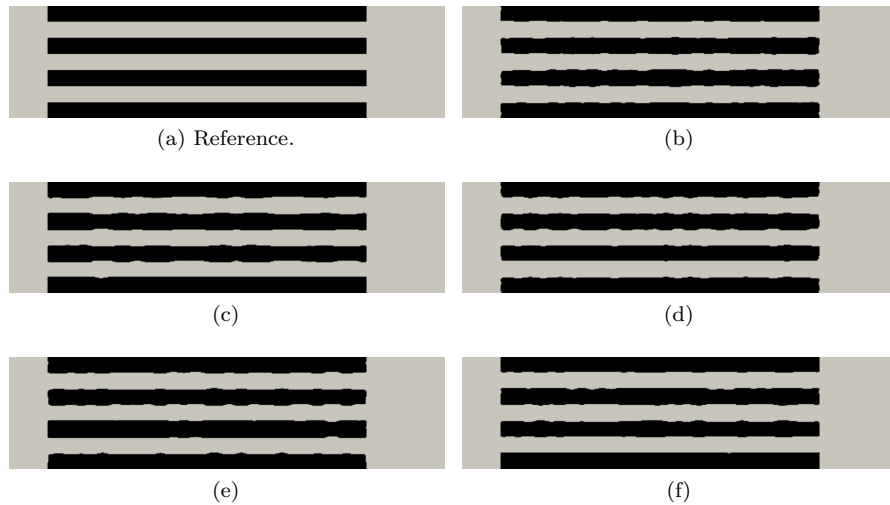


Figure 7.4: Example realizations created by applying the non-uniform projection filter to parallel plates shown in figure 7.3a. The wave-like nature of the non-uniform variations is clearly seen.

One challenge associated with this kind of robust optimization is that since random variables have been introduced, it is essentially a kind of Monte Carlo method. Such methods have the weakness that a large number of samples (of order hundreds) are generally needed in order for the method to converge. Since the equivalent of a “sample” in this case is a single design realization, the computational cost of having a hundred or more realizations would be enormous, even though the individual realizations can be computed in an embarrassingly parallel fashion, as described in section 5.5.2. To counteract this somewhat, rather than sampling using a pseudo-random number generator, the samples are generated from a Sobol sequence, a so-called quasi-random or low discrepancy sequence [108, 109]. Using this type of sequence speeds up the rate of convergence for Monte Carlo methods by traversing the sampling space in a more structured manner, ensuring that the space is “covered” using fewer samples.

7.5.2 Results

Now that the means to generate design realizations with non-uniform variations are in place, robust topology optimization can be performed on the regenerator problem. Two sets of results are presented. Both have the same simulation setup (as described in section 7.4.1), but differ in the initial guess used for the design. In both cases, the optimization has been performed on 20 realizations, though only a representative sample will be shown here. The two sets of results are shown in figures 7.5 and 7.6, respectively.

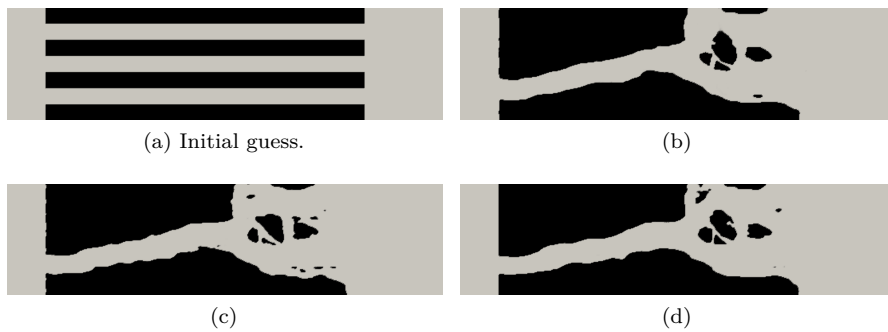


Figure 7.5: Example results obtained with a thick plate design used as initial guess.

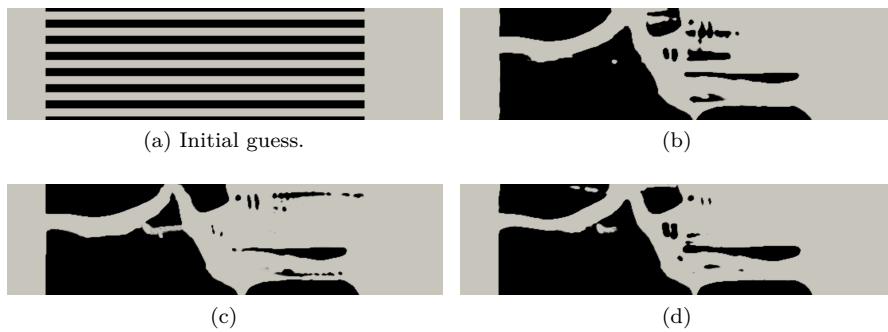


Figure 7.6: Example results obtained with a thin plate design used as initial guess.

The results shown should be considered preliminary since they exhibit a number of problematic features. The individual realizations all have topology differences, which is especially pronounced in the second set of results. This could potentially be mitigated by using the double filter introduced by Christiansen *et al.* [77]. Moreover, there is no particularly clear physical interpretation which can be made of the designs.

Nonetheless, from a purely numerical perspective, the lattice Boltzmann simulations predict an average of 22% improvement in the regenerator efficiency, compared to the initial guess, for the first set of results, and an average improvement of 30% for the second set of results. Despite this, more investigation into this problem is clearly needed.

7.6 Discussion

To conclude this chapter, various ways forward from the current state of the investigation into the regenerator problem are proposed.

We will postpone discussion of the problems already touched on for a moment and focus first on a purely practical matter: the optimization runs for the regenerator problem are very computationally costly. This is especially true in the early stages optimization, where computing the steady state solution can often take of order a hundred thousand time-steps, which then have to be reversed for the adjoint analysis (including recomputation done by the checkpointing algorithm). This problem diminishes as the optimization starts to converge, since the design change between iterations is quite small, which means that the steady state computations will converge in a relatively small number of time-steps. Nonetheless, steady state computations account for a very large portion of the computational time, and speeding up this part would most likely yield the greatest return on time invested to realize the speed-up. A multigrid method for computing steady state solutions using the lattice Boltzmann method has been proposed by Mavriplis [110], which is reported to compute steady state solutions much faster than the pseudo time-stepping approach. This was attempted implemented by the author, but unfortunately the implementation is not functional as of this writing.

Increasing the computational speed of the optimization runs would be a worthwhile pursuit because it would allow for a faster iteration time in producing new results. This would allow more rapid prototyping of potential solutions for the modeling issues discussed below. It would also be absolutely essential if one was to extend the problem to 3D.

Regarding the modeling of the problem, the thermal model used is quite simple, since it is based on the advection-diffusions equation (2.5), rather than a fully coupled model. It could well be that a more advanced model is necessary to fully capture the physics necessary to model and optimize the regenerator in a satisfying way. Considering the findings of section 5.5.4, it is likely necessary to conduct a more in-depth study of this type of thermal modeling with the lattice Boltzmann method, both from a theoretical and practical perspective. Another potential benefit of such a study would be additional insight into the best way to formulate the problem: there might well exist a more natural and/or precise objective function for the problem than (7.17), for example.

In conclusion, it seems that there is good potential for the application of topology optimization to this problem, but there are significant challenges associated with the modeling as well as the implementation quality which ideally must be overcome in order to proceed.

8

Closing discussion

In this chapter, the major findings presented in this thesis are summarized, along with discussion of remaining issues for the application of the lattice Boltzmann method for topology optimization as the author sees them. Finally, it offers concluding comments and possible avenues of future research.

8.1 Summary

This thesis presents a framework for topology optimization of unsteady fluid flow systems using the lattice Boltzmann method. The gradients needed for the optimization process are computed using a discrete adjoint approach which utilizes automatic differentiation to differentiate the non-linear collision steps in the method.

The applicability of the framework has been demonstrated on a number of unsteady flow problems, including both isothermal and thermal systems. The optimization process consistently converges to a design which is numerically superior to the initial design. Despite this, there are lingering issues regarding the efficacy of the physical simulation. It was shown in chapter 5 that predicted behavior of an optimized design can differ significantly from a conventional finite element simulation. In the case of the thermal problems, no direct comparison was made with a finite element simulation, but the obtained designs are hard to interpret physically.

Despite these as yet unresolved problems, the implemented simulation code is able to simulate fairly large unsteady flow systems in a reasonable amount of time, utilizing parallel computation, checkpointing, and in the case of robust optimization, distribution of each design realization to its own group of computational cores. There is definite potential for applications in topology optimization at a truly large scale.

8.2 Comments on implementation

A large part of the work for this thesis was spent on implementation work for the lattice Boltzmann framework as described. The code utilizes the PETSc library, which provides convenient parallel data structures for numerical codes. However, the functionality provided by PETSc is primarily centered around linear algebra solvers, which are not directly applicable for the lattice Boltzmann method. It is likely that significant flexibility and performance could be gained by implementing, or—even better—utilizing already implemented data structures specialized for lattice Boltzmann. Several open source lattice Boltzmann libraries exist, but it is not known by the author if the code could be easily adapted to topology optimization. In any case, some implementation effort would be needed to adapt other dependencies, e.g. MMA, to such a code. Such an effort might well be worth it, since it would allow

access to an ecosystem of lattice Boltzmann code, rather than having to implement everything from scratch.

8.3 Future work

As already discussed in some detail, understanding the discrepancy between solvers will be a very important issue to resolve going forward. Beyond that, a few possible extensions of the present work will be listed here.

The most obvious extension would be to implement a 3D version of the optimization framework described here. The tools developed for this thesis, e.g. the automatic differentiation based adjoint approach and the checkpointing algorithm, would transfer directly to a 3D implementation. The main issue, then, is the large increase in computational cost when transitioning to 3D.

While the increase in computational cost is inevitable, there are a few techniques available to reduce the problem somewhat. In the current code, only rectangular domains are supported. This leads to a fair amount of wasted computing time in e.g. the pump problem, which has fixed solid domains to account for the inflow and outflow channels. It would be more efficient to completely exclude these areas from the simulation, but this would require support for non-rectangular computational domains. Note that this would require a major restructuring of the code, since the current implementation depends on PETSc data structures which only support rectangular domains, cf. the discussion on implementation details above.

A more exotic extension would be investigating local grid refinement methods for the lattice Boltzmann method [111]. This would allow overall computational savings by placing more grid nodes in areas which are expected to exhibit more complicated dynamics. This is likely to greatly increase the complexity of the implementation, however.

Finally, regarding the thermal models, it would be interesting (possibly even necessary) to investigate more advanced thermal models [112, 113] than the simple advection-diffusion model used for this work.

8.4 Conclusion

In conclusion, this thesis shows the applicability of the lattice Boltzmann method as a tool for topology optimization, but further work is required to realize the full potential of the method. Since relatively little work on unsteady flow optimization has been done, it is not clear as of this writing whether the lattice Boltzmann method offers a significant advantage over conventional Navier-Stokes based methods. Certainly, the traditional methods are much more well established, which might make them more practical for the foreseeable future. In the view of the author, an in-depth study of the trade-offs between the two methods would be well warranted.

Bibliography

- [1] R. B. Bird, W. E. Stewart, and E. N. Lightfoot. *Transport phenomena*. J. Wiley, 2007.
- [2] B. Lautrup. *Physics of continuous matter*. CRC Press, 2011.
- [3] S. Succi. *The lattice Boltzmann equation for fluid dynamics and beyond*. Oxford University Press, 2001.
- [4] Gilberto Medeiros Kremer. *An Introduction to the Boltzmann Equation and Transport Processes in Gases*. Springer Berlin Heidelberg, 2010.
- [5] Timm Krüger, Halim Kusumaatmaja, Alexandr Kuzmin, Orest Shardt, Goncalo Silva, and Erlend Magnus Viggen. *The Lattice Boltzmann Method: Principles and Practice*. Springer, 2017.
- [6] Randall J. LeVeque. *Finite Difference Methods for Ordinary and Partial Differential Equations*. Society for Industrial and Applied Mathematics, 2007.
- [7] R.D. Cook, D.S. Malkus, and M.E. Plesha. *Concepts and applications of finite element analysis*. Wiley, 2002.
- [8] Jean Donea and Antonio Huerta. *Finite Element Methods for Flow Problems*. John Wiley & Sons, Ltd, 2003.
- [9] Susanne C. Brenner and L. Ridgway Scott. *The Mathematical Theory of Finite Element Methods*. Springer New York, 2008.
- [10] Robert Eymard, Thierry Gallouët, and Raphaèle Herbin. Finite volume methods. In *Handbook of Numerical Analysis*, pages 713–1018. Elsevier, 2000.
- [11] H.K. Versteeg and W. Malalasekera. *An introduction to computational fluid dynamics*. Pearson, 2007.
- [12] Harold Grad. On the kinetic theory of rarefied gases. *Communications on Pure and Applied Mathematics*, 2(4):331–407, 1949.
- [13] Richard L. Liboff. *Kinetic Theory: Classical, Quantum, and Relativistic Descriptions*. Springer-Verlag, 2003.
- [14] D. Enskog. The numerical calculation of phenomena in fairly dense gases. *Arkiv Mat. Astr. Fys*, 16(1):1–60, 1921.

- [15] Sydney Chapman. On the law of distribution of molecular velocities, and on the theory of viscosity and thermal conduction, in a non-uniform simple monatomic gas. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, 216:279–348, 1916.
- [16] Sydney Chapman. On the kinetic theory of a gas. part II: a composite monatomic gas: diffusion, viscosity, and thermal conduction. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, 217:115–197, 1918.
- [17] Henning Struchtrup. *Macroscopic Transport Equations for Rarefied Gas Flows*. Springer-Verlag, 2005.
- [18] Landau and Lifshitz. *Fluid mechanics*. Pergamon Press, 1959.
- [19] P. L. Bhatnagar, E. P. Gross, and M. Krook. A model for collision processes in gases. I: Small amplitude processes in charged and neutral one-component systems. *Physical Review*, 94(3), 1954.
- [20] Uriel Frisch, Brosl Hasslacher, and Yves Pomeau. Lattice-gas automata for the Navier-Stokes equation. *Physical review letters*, 56(14):1505, 1986.
- [21] Guy R. McNamara and Gianluigi Zanetti. Use of the boltzmann equation to simulate lattice-gas automata. *Physical review letters*, 61(20):2332, 1988.
- [22] F.J. Higuera and J. Jimenez. Boltzmann approach to lattice gas simulations. *EPL (Europhysics Letters)*, 9(7):663, 1989.
- [23] D. A. Wolf-Gladrow. *Lattice-gas cellular automata and lattice Boltzmann models - an introduction*. Springer, 2005.
- [24] Sebastian Geller, Manfred Krafczyk, Jonas Tölke, Stefan Turek, and Jaroslav Hron. Benchmark computations based on lattice-Boltzmann, finite element and finite volume methods for laminar flows. *Computers & Fluids*, 35(8):888–897, 2006.
- [25] M. A. A. Spaid and F.R. Phelan. Lattice Boltzmann methods for modeling microscale flow in fibrous porous media. *Physics of Fluids*, 9(9):2468–2474, 1997.
- [26] S. D. C. Walsh, H. Burwinkle, and M. O. Saar. A new partial-bounceback lattice-Boltzmann method for fluid flow through heterogeneous media. *Computers and Geosciences*, 35(6):1186–1193, 2009.
- [27] J. Zhu and J. Ma. An improved gray lattice Boltzmann model for simulating fluid flow in multi-scale porous media. *Advances in Water Resources*, 56:61, 2013.

- [28] T. Inamuro, T. Ogata, S. Tajima, and N. Konishi. A lattice Boltzmann method for incompressible two-phase flows with large density differences. *Journal of Computational Physics*, 198(2):628–644, 2004.
- [29] P. Asinari. Semi-implicit-linearized multiple-relaxation-time formulation of lattice Boltzmann schemes for mixture modeling. *Physical Review E*, 73(5):056705, 2006.
- [30] Ye Zhao. Lattice Boltzmann based PDE solver on the GPU. *The visual computer*, 24(5):323–333, 2008.
- [31] Massimo Bernaschi, Massimiliano Fatica, Simone Melchionna, Sauro Succi, and Efthimios Kaxiras. A flexible high-performance lattice Boltzmann GPU code for the simulations of fluid flows in complex geometries. *Concurrency and Computation: Practice and Experience*, 22(1):1–14, 2010.
- [32] Wang Xian and Aoki Takayuki. Multi-GPU performance of incompressible flow computation by lattice Boltzmann method on GPU cluster. *Parallel Computing*, 37(9):521–535, 2011.
- [33] Dominique d’Humières, M’hamed Bouzidi, and Pierre Lallemand. Thirteen-velocity three-dimensional lattice Boltzmann model. *Physical Review E*, 63(6):066702, 2001.
- [34] Michael Junk, Axel Klar, and Li-Shi Luo. Asymptotic analysis of the lattice Boltzmann equation. *Journal of Computational Physics*, 210(2):676–704, 2005.
- [35] Dominique d’Humières. Generalized lattice-Boltzmann equations. *Progress in Astronautics and Aeronautics*, 159:450–450, 1994.
- [36] Dominique d’Humières. Multiple-relaxation-time lattice Boltzmann models in three dimensions. *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 360(1792):437–451, 2002.
- [37] Pierre Lallemand and Li-Shi Luo. Theory of the lattice Boltzmann method: Dispersion, dissipation, isotropy, Galilean invariance, and stability. *Physical Review E*, 61(6):6546, 2000.
- [38] Bruce M. Boghosian, Jeffrey Yeppez, Peter V. Coveney, and Alexander Wager. Entropic lattice Boltzmann methods. In *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, volume 457, pages 717–766. The Royal Society, 2001.
- [39] Santosh Ansumali, Iliya V Karlin, and Hans Christian Öttinger. Minimal entropic kinetic models for hydrodynamics. *EPL (Europhysics Letters)*, 63(6):798, 2003.

- [40] Jonas Latt and Bastien Chopard. Lattice Boltzmann method with regularized pre-collision distribution functions. *Mathematics and Computers in Simulation*, 72(2):165–168, 2006.
- [41] Martin Geier, Andreas Greiner, and Jan G. Korvink. Cascaded digital lattice Boltzmann automata for high Reynolds number flow. *Physical Review E*, 73(6):066705, 2006.
- [42] Martin Geier, Andreas Greiner, and Jan G. Korvink. Properties of the cascaded lattice Boltzmann automaton. *International Journal of Modern Physics C*, 18(04):455–462, 2007.
- [43] Jonas Latt. Technical report: How to implement your DdQq dynamics with only q variables per node (instead of 2q). Technical report, Technical report, Tufts University, 2007.
- [44] Markus Wittmann, Thomas Zeiser, Georg Hager, and Gerhard Wellein. Comparison of different propagation steps for lattice Boltzmann methods. *Computers & Mathematics with Applications*, 65(6):924–935, 2013.
- [45] Martin Geier and Martin Schönherr. Esoteric twist: An efficient in-place streaming algorithm for the lattice Boltzmann method on massively parallel hardware. *Computation*, 5(2):19, 2017.
- [46] Q.S. Zou and X.Y. He. On pressure and velocity boundary conditions for the lattice Boltzmann BGK model. *Physics of Fluids*, 9(6):1591–1598, 1997.
- [47] Jonas Latt. Choice of units in lattice Boltzmann simulations. *Freely available online at http://lbmethod.org/_media/howtos:lbunits.pdf*, 2008.
- [48] Shuling Hou, Qisu Zou, Shiyi Chen, Gary Doolen, and Allen C. Cogley. Simulation of cavity flow by the lattice Boltzmann method. *Journal of computational physics*, 118(2):329–347, 1995.
- [49] Paul J. Dellar. Lattice kinetic schemes for magnetohydrodynamics. *Journal of Computational Physics*, 179(1):95–126, 2002.
- [50] Gábor Házi and C. Jiménez. Simulation of two-dimensional decaying turbulence using the “incompressible” extensions of the lattice Boltzmann method. *Computers & fluids*, 35(3):280–303, 2006.
- [51] M. P. Bendøse and O. Sigmund. *Topology Optimization*. Springer, 2003.
- [52] Ole Sigmund and Kurt Maute. Topology optimization approaches. *Structural and Multidisciplinary Optimization*, 48(6):1031–1055, aug 2013.
- [53] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer New York, 2006.

- [54] M. P. Bendsøe and N. Kikuchi. Generating optimal topologies in structural design using a homogenization method. *Computer Methods in Applied Mechanics and Engineering*, 71(2):197–224, November 1988.
- [55] T. Borrvall and J. Petersson. Topology optimization of fluids in Stokes flow. *International Journal for Numerical Methods in Fluids*, 41(1):77–107, 2003.
- [56] A. Gersborg-Hansen, O. Sigmund, and R. B. Haber. Topology optimization of channel flow problems. *Structural and Multidisciplinary Optimization*, 30(3):181–192, 2005.
- [57] L. H. Olesen, F. Okkels, and H. Bruus. A high-level programming-language implementation of topology optimization applied to steady-state Navier-Stokes flow. *International Journal for Numerical Methods in Engineering*, 65(7):975–1001, 2006.
- [58] G. Pingen, M. Waidmann, A. Evgrafov, and K. Maute. A parametric level-set approach for topology optimization of flow domains. *Structural and Multidisciplinary Optimization*, 41(1):117–131, 2010.
- [59] G. Pingen, A. Evgrafov, and K. Maute. Topology optimization of flow domains using the lattice Boltzmann method. *Structural and Multidisciplinary Optimization*, 34(6):507–524, 2007.
- [60] S. Kreissl, G. Pingen, and K. Maute. An explicit level set approach for generalized shape optimization of fluids with the lattice Boltzmann method. *International Journal for Numerical Methods in Fluids*, 65(5):496–519, 2011.
- [61] K. Yaji, T. Yamada, and M. Yoshino. Topology optimization using the lattice Boltzmann method incorporating level set boundary expressions. *Journal of Computational Physics*, 274:158, 2014.
- [62] C. S. Andreasen, A. R. Gersborg, and O. Sigmund. Topology optimization of microfluidic mixers. *International Journal for Numerical Methods in Fluids*, 61(5):498–513, 2009.
- [63] Joe Alexandersen, Niels Aage, Casper Schousboe Andreasen, and Ole Sigmund. Topology optimisation for natural convection problems. *International Journal for Numerical Methods in Fluids*, 76(10):699–721, 2014.
- [64] Joe Alexandersen, Ole Sigmund, and Niels Aage. Large scale three-dimensional topology optimisation of heat sinks cooled by natural convection. *International Journal of Heat and Mass Transfer*, 100:876–891, 2016.
- [65] Kentaro Yaji, Takayuki Yamada, Masato Yoshino, Toshiro Matsumoto, Kazuhiro Izui, and Shinji Nishiwaki. Topology optimization in thermal-fluid flow using the lattice Boltzmann method. *Journal of Computational Physics*, 307:355–377, feb 2016.

- [66] S. Kreissl, G. Pingen, and K. Maute. Topology optimization for unsteady flow. *International Journal for Numerical Methods in Engineering*, 87(13), 2011.
- [67] Y. Deng, Z. Liu, P. Zhang, Y. Liu, and Y. Wu. Topology optimization of unsteady incompressible Navier-Stokes flows. *Journal of Computational Physics*, 230(17):6688–6708, 2011.
- [68] Yongbo Deng, Zhenyu Liu, and Yihui Wu. Topology optimization of steady and unsteady incompressible Navier-Stokes flows driven by body forces. *Structural and Multidisciplinary Optimization*, 47(4):555–570, 2013.
- [69] Martin P. Bendsøe and Ole Sigmund. Material interpolation schemes in topology optimization. *Archive of Applied Mechanics*, 69:635–654, 1999.
- [70] Daniel A. Tortorelli and Panagiotis Michaleris. Design sensitivity analysis: overview and review. *Inverse problems in Engineering*, 1(1):71–105, 1994.
- [71] Alejandro Díaz and Ole Sigmund. Checkerboard patterns in layout optimization. *Structural and Multidisciplinary Optimization*, 10(1):40–45, 1995.
- [72] J.K. Guest, J.H. Prevost, and T. Belytschko. Achieving minimum length scale in topology optimization using nodal design variables and projection functions. *International Journal for Numerical Methods in Engineering*, 61(2):238–254, 2004.
- [73] F. Wang, B. S. Lazarov, and O. Sigmund. On projection methods, convergence and robust formulations in topology optimization. *Structural and Multidisciplinary Optimization*, 43(6):767–784, 2011.
- [74] T.E. Bruns and D.A. Tortorelli. Topology optimization of non-linear elastic structures and compliant mechanisms. *Computer Methods in Applied Mechanics and Engineering*, 190(26-27):3443–3459, 2001.
- [75] B. Bourdin. Filters in topology optimization. *International Journal for Numerical Methods in Engineering*, 50(9):2143–2158, 2001.
- [76] Boyan Stefanov Lazarov and Ole Sigmund. Filters in topology optimization based on Helmholtz-type differential equations. *International Journal for Numerical Methods in Engineering*, 86(6):765–781, 2011.
- [77] Rasmus E. Christiansen, Boyan S. Lazarov, Jakob S. Jensen, and Ole Sigmund. Creating geometrically robust designs for highly sensitive problems using topology optimization. *Structural and Multidisciplinary Optimization*, 52(4):737–754, 2015.

- [78] Mattias Schevenels, Boyan Stefanov Lazarov, and Ole Sigmund. Robust topology optimization accounting for spatially varying manufacturing errors. *Computer Methods in Applied Mechanics and Engineering*, 200(49):3613–3627, 2011.
- [79] K. Svanberg. The method of moving asymptotes - a new method for structural optimization. *International Journal for Numerical Methods in Engineering*, 24(2):359–373, 1987.
- [80] Krister Svanberg. A class of globally convergent optimization methods based on conservative convex separable approximations. *SIAM journal on optimization*, 12(2):555–573, 2002.
- [81] Niels Aage, Erik Andreassen, and Boyan Stefanov Lazarov. Topology optimization using PETS: An easy-to-use, fully parallel, open source topology optimization framework. *Structural and Multidisciplinary Optimization*, 51(3):565–572, 2015.
- [82] Geng Liu, Martin Geier, Zhenyu Liu, Manfred Krafczyk, and Tao Chen. Discrete adjoint sensitivity analysis for fluid flow topology optimization based on the generalized lattice Boltzmann method. *Computers and Mathematics With Applications*, 68(10):1374–1392, 2014.
- [83] Martin P. Bendsøe. Optimal shape design as a material distribution problem. *Structural and multidisciplinary optimization*, 1(4):193–202, 1989.
- [84] Carlos H. Villanueva and Kurt Maute. CutFEM topology optimization of 3d laminar incompressible flow problems. *Computer Methods in Applied Mechanics and Engineering*, 320:444–473, 2017.
- [85] A. Griewank and A. Walther. *Automatic differentiation of algorithms*. SIAM, 2008.
- [86] Mathias J. Krause, Gudrun Thäter, and Vincent Heuveline. Adjoint-based fluid flow control and optimisation with lattice Boltzmann methods. *Computers and Mathematics with Applications*, 65(6):945–960, mar 2013.
- [87] autodiff.org: Community portal for automatic differentiation, 2017. Accessed: 2017-10-02.
- [88] A. Griewank and A. Walther. Algorithm 799: Revolve: an implementation of checkpointing for the reverse or adjoint mode of computational differentiation. *Acm Transactions on Mathematical Software*, 26(1):19–45, 19–45, 2000.
- [89] Qiqi Wang, Parviz Moin, and Gianluca Iaccarino. Minimal repetition dynamic checkpointing algorithm for unsteady adjoint calculation. *Siam Journal on Scientific Computing*, 31(4):2549–2567, 2009.

- [90] Jonas Latt, Bastien Chopard, Orestis Malaspinas, Michel Deville, and Andreas Michler. Straight velocity boundaries in the lattice Boltzmann method. *Physical Review E*, 77(5):056703, 2008.
- [91] O. Malaspinas, B. Chopard, and J. Latt. General regularized boundary condition for multi-speed lattice Boltzmann models. *Computers and Fluids*, 49(1):29–35, 2011.
- [92] Anil Nemili, Emre Özkaya, Nicolas R. Gauger, Felix Kramer, Tobias Höll, and Frank Thiele. Optimal design of active flow control for a complex high-lift configuration. In *Proceedings of 7th AIAA Flow Control Conference*, 2014-2515, 2014.
- [93] Emre Özkaya, Junis Abdel Hay, Nicolas R. Gauger, Norbert Schönwald, and Frank Thiele. A two-level approach for design optimization of acoustic liners. In *Proceedings of 9th International Conference on Computational Fluid Dynamics*, ICCFD9-2016-184, 2016.
- [94] Beckett Y. Zhou, Tim Albring, Nicolas R. Gauger, Carlos R. Illario da Silva, Thomas D. Economon, and Juan J. Alonso. A discrete adjoint approach for jet-flap interaction noise reduction. In *Proceedings of 58th AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, AIAA SciTech Forum*, AIAA 2017-0130, 2017.
- [95] L. Łaniewski Wołk and J. Rokicki. Adjoint lattice Boltzmann for topology optimization on multi-gpu architecture. *Computers and Mathematics With Applications*, 71(3):833–848, 2016.
- [96] Cetin B. Dilgen, Sumer B. Dilgen, David R. Fuhrman, Ole Sigmund, and Boyan S. Lazarov. Topology optimization of turbulent flows. *In review*, 2017.
- [97] Zhaoli Guo, Baochang Shi, and Chuguang Zheng. A coupled lattice BGK model for the Boussinesq equations. *International Journal for Numerical Methods in Fluids*, 39(4):325–342, 2002.
- [98] Ahmed Mezrhab, Mohammed Amine Moussaoui, Mohammed Jami, Hassan Naji, and M’hamed Bouzidi. Double MRT thermal lattice Boltzmann method for simulating convective flows. *Physics Letters A*, 374(34):3499–3507, 2010.
- [99] Zheng Li, Mo Yang, and Yuwen Zhang. Lattice Boltzmann method simulation of 3-D natural convection with double MRT model. *International Journal of Heat and Mass Transfer*, 94:222–238, 2016.
- [100] Abdulmajeed A. Mohamad. *Lattice Boltzmann method: fundamentals and engineering applications with computer codes*. Springer Science & Business Media, 2011.

- [101] Xiaowen Shan, Xue-Feng Yuan, and Hudong Chen. Kinetic theory representation of hydrodynamics: a way beyond the Navier–Stokes equation. *Journal of Fluid Mechanics*, 550:413–441, 2006.
- [102] Guy R. McNamara, Alejandro L. Garcia, and Berni J. Alder. Stabilization of thermal lattice Boltzmann models. *Journal of Statistical Physics*, 81(1):395–408, 1995.
- [103] Orestis Malaspinas. How to impose a Neumann boundary condition with the lattice Boltzmann method. *Freely available online at http://wiki.palabos.org/_media/howtos:neumann.pdf*, 2008.
- [104] Mathias Stolpe and Krister Svanberg. An alternative interpolation scheme for minimum compliance topology optimization. *Structural and Multidisciplinary Optimization*, 22(2):116–124, 2001.
- [105] Françoise Duprat and Guadalupe Lopez Lopez. Comparison of performance of heat regenerators: relation between heat transfer efficiency and pressure drop. *International journal of energy research*, 25(4):319–329, 2001.
- [106] P.J. Heggs and D. Burns. Single-blow experimental prediction of heat transfer coefficients: A comparison of four commonly used techniques. *Experimental Thermal and Fluid Science*, 1(3):243–251, 1988.
- [107] Jesper B. Jensen, Kurt Engelbrecht, Christian R.H. Bahl, Nini Pryds, Gregory F. Nellis, Sanford A. Klein, and Brian Elmegaard. Modeling of parallel-plate regenerators with non-uniform plate distributions. *International Journal of Heat and Mass Transfer*, 53(23):5065–5072, 2010.
- [108] Il’ya Meerovich Sobol’. On the distribution of points in a cube and the approximate evaluation of integrals. *Zhurnal Vychislitel’noi Matematiki i Matematicheskoi Fiziki*, 7(4):784–802, 1967.
- [109] Lauwerens Kuipers and Harald Niederreiter. *Uniform distribution of sequences*. Courier Corporation, 2012.
- [110] Dimitri J. Mavriplis. Multigrid solution of the steady-state lattice Boltzmann equation. *Computers & fluids*, 35(8):793–804, 2006.
- [111] Daniel Lagrava, Orestis Malaspinas, Jonas Latt, and Bastien Chopard. Advances in multi-domain lattice Boltzmann grid refinement. *Journal of Computational Physics*, 231(14):4808–4822, 2012.
- [112] Li-Hsin Hung and Jaw-Yen Yang. A coupled lattice Boltzmann model for thermal flows. *IMA journal of applied mathematics*, 76(5):774–789, 2011.

- [113] Q. Li, K.H. Luo, Y.L. He, Y.J. Gao, and W.Q. Tao. Coupling lattice Boltzmann model for simulation of thermal flows on standard lattices. *Physical Review E*, 85(1):016710, 2012.

Publications

Publication [P1]

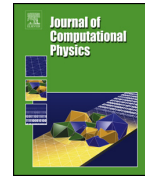
Topology optimization of unsteady
flow problems using the lattice
Boltzmann method



Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp



Topology optimization of unsteady flow problems using the lattice Boltzmann method



Sebastian Nørgaard*, Ole Sigmund, Boyan Lazarov

Department of Mechanical Engineering, Technical University of Denmark, DK-2800 Lyngby, Denmark

ARTICLE INFO

Article history:

Received 14 August 2015
 Received in revised form 13 November 2015
 Accepted 9 December 2015
 Available online 15 December 2015

Keywords:

Topology optimization
 Unsteady flow
 Lattice Boltzmann

ABSTRACT

This article demonstrates and discusses topology optimization for unsteady incompressible fluid flows. The fluid flows are simulated using the lattice Boltzmann method, and a partial bounceback model is implemented to model the transition between fluid and solid phases in the optimization problems. The optimization problem is solved with a gradient based method, and the design sensitivities are computed by solving the discrete adjoint problem. For moderate Reynolds number flows, it is demonstrated that topology optimization can successfully account for unsteady effects such as vortex shedding and time-varying boundary conditions. Such effects are relevant in several engineering applications, i.e. fluid pumps and control valves.

© 2015 Elsevier Inc. All rights reserved.

1. Introduction

The lattice Boltzmann method (LBM) has gained increasing attention in recent years as a viable alternative to Navier–Stokes (NS) based methods for computational fluid dynamics. In this paper, we present a method for topology optimization of fluid domains subject to unsteady flow conditions using the LBM. The topology of the domain is represented using the density approach, in which each point of the discretized domain is assigned a value between 0 and 1, representing a fully solid and fluid node, respectively [1]. In contrast to previously published work on fluid domain optimization with the LBM, which has focused on steady-state formulations of the optimization problem, this work considers an unsteady flow formulation, and presents optimization problems in which unsteady effects have a significant influence on the optimized topologies.

The topology optimization methodology was originally developed as a design tool for structural mechanics [1,2], and since its inception it has been applied to a variety of physical domains, including the optimal control of fluid flow problems. Borrvall and Petersson first applied topology optimization to Stokes flow problems in 2003 [3], and since then, additional studies have extended their work to laminar Navier–Stokes flow at moderate Reynolds number [4,5], and to large scale problems solved using parallel computation [6]. Furthermore, there are numerous published studies applying the methodology to more complicated fluidic devices, such as fluid switches [4], and microfluidic mixers [7,8].

In the majority of previous works, the flow solutions are approximated by numerically solving the NS equations. Recently, however, the LBM has become a popular alternative to traditional solvers. For a thorough introduction, the reader is referred to the books by Sukop and Thorne [9], and Wolf-Gladrow [10]. In contrast to the continuum assumption of the NS based approach, the starting point of the LBM is the Boltzmann transport equation of kinetic theory. Discretization of this equation

* Corresponding author.

E-mail address: sebnorg@mek.dtu.dk (S. Nørgaard).

<http://dx.doi.org/10.1016/j.jcp.2015.12.023>

0021-9991/© 2015 Elsevier Inc. All rights reserved.

yields an explicit time marching scheme which can compute approximate solutions to the incompressible NS equations for low Mach number flows. The method is attractive because it is algorithmically simple, lends itself well to parallel implementation, and is relatively easy to extend to more complicated physics, such as porous media [11–13], or multiphase flows [14,15]. The use of the LBM for topology optimization was pioneered by Pingen et al. [16], who used the density approach to topology optimization. The work is extended to multiphase flow problems by Makhija et al. [8]. In addition, a number of studies have investigated a level-set based optimization approach using the LBM [17–19].

For gradient-based optimization, a common technique for computing the necessary gradients is the adjoint method. There are two main approaches to this technique, the “discretize then optimize” approach, in which the adjoint problem is derived from the discrete lattice Boltzmann equation, and the “optimize then discretize” approach, in which the adjoint problem is derived from the continuous Boltzmann equation and then discretized. In this paper, we adopt the first approach. The discretize then optimize approach was first used by Tekitek et al. [20] for finding optimal parameters for a lattice Boltzmann model, while Krause et al. [21] applied the discretize then optimize approach to flow control problems.

Regardless of the computational method used to obtain flow profiles, the vast majority of studies on fluid topology optimization have only considered steady-state flow. On the other hand, many fluidic systems of interest are dominated by unsteady flows and do not permit steady-state solutions. Only few studies have been published treating such systems, however. An implementation of unsteady flow topology optimization using a discrete formulation of the objective function has been presented by Kreissl et al. [22], while Deng et al. presented a continuous formulation [23]. Both of these studies computed the flow profiles by a finite element discretization of the NS equations. While Yonekura and Kanno [24] have presented a method for computing steady state designs using transient information, to the authors’ knowledge, there are no published works which apply the LBM to topology optimization of unsteady flow problems. In addition, the above works consider problems in which unsteady effects have little influence on the topology of the computed solutions. In the present work, problems with inherent unsteady characteristics will be presented. The problems considered are restricted to two spatial dimensions, but the method is readily extensible to three-dimensional problems.

Following the work of Kreissl et al. [22], the unsteady flow topology optimization problems considered in this paper may be written in the time-discrete form:

$$\begin{aligned} \min_{\mathbf{s}} Z(\mathbf{f}^0, \dots, \mathbf{f}^{N_t}, \mathbf{s}) &= \chi \left(\sum_{n=0}^{N_t} z^n(t^n, \mathbf{f}^n, \mathbf{s}^{\text{ph}}) \right), \\ \text{s.t. } \begin{cases} \mathbf{s}, & \text{satisfies design constraints } G_j \leq 0, \\ \mathbf{f}^n, & \text{satisfies the governing equations, } \mathbf{R}^n = \mathbf{0}, \\ & \text{for the given } \mathbf{s}, \forall n \in \{0, \dots, N_t\}, \end{cases} \end{aligned} \quad (1)$$

where \mathbf{s} is a vector of design variables, and \mathbf{s}^{ph} is a vector of physical variables associated with the fluid medium at each design element. The physical variables are obtained from \mathbf{s} by a continuous mapping $X: \mathbf{s} \rightarrow \mathbf{s}^{\text{ph}}$. The vector \mathbf{f}^n is the fluid state at time step $n \in \{0, \dots, N_t\}$, and $\mathbf{R}^n = \mathbf{0}$ is the residual vector of the governing equations at time step n . The objective Z is given by a differentiable function χ which depends on a sum of contributions z^n in each time step. The functions G_j represent the design constraints.

The rest of the paper is organized as follows: section 2 contains a brief overview of the governing equations of the lattice Boltzmann method, including the treatment of boundary conditions and modeling of fluid and solid domains. Section 3 covers the adjoint sensitivity analysis for the case of an unsteady objective function. In section 4 a brief overview of filtering techniques by means of a proper choice of the mapping X is given. The introduced concepts are demonstrated in section 5 with two numerical examples. Finally, section 6 offers a summary of the results as well as concluding remarks.

2. Governing equations

Unlike the Navier–Stokes equations, the LBM models the motion of a fluid as an ensemble of microscopic particles. The state of the system is given in terms of a distribution function, from which macroscopic quantities such as density and fluid velocity can be obtained by computing its moments.

2.1. The lattice Boltzmann method

As shown by He and Luo [25,26], the discrete lattice Boltzmann equation with Bhatnagar–Gross–Krook (BGK) collision operator [27] reads:

$$f_{\alpha}(\mathbf{x}_i + \mathbf{c}_{\alpha} \Delta t, t + \Delta t) = f_{\alpha}(\mathbf{x}_i, t) - \frac{1}{\tau} [f_{\alpha}(\mathbf{x}_i, t) - f_{\alpha}^{\text{eq}}(\mathbf{x}_i, t)], \quad (2)$$

where τ is the relaxation time, \mathbf{c}_{α} belongs to some discrete set of lattice velocities, and $f_{\alpha} = f(\mathbf{x}_i, \mathbf{c}_{\alpha}, t)$ and $f_{\alpha}^{\text{eq}} = f^{\text{eq}}(\mathbf{x}_i, \mathbf{c}_{\alpha}, t)$ is respectively the distribution function and local equilibrium associated with the corresponding velocity. For this work, the common D2Q9 scheme (Fig. 1), which partitions two-dimensional velocity space into nine discrete velocities,

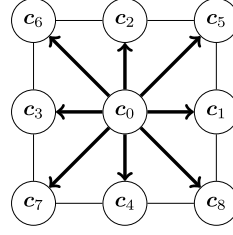


Fig. 1. The D2Q9 model.

has been applied. The discrete velocities are given by:

$$\begin{aligned} & [c_0, c_1, c_2, c_3, c_4, c_5, c_6, c_7, c_8] \\ &= \frac{\Delta x}{\Delta t} \begin{bmatrix} 0 & 1 & 0 & -1 & 0 & 1 & -1 & -1 & 1 \\ 0 & 0 & 1 & 0 & -1 & 1 & 1 & -1 & -1 \end{bmatrix}. \end{aligned} \quad (3)$$

For simplicity, the step sizes Δx and Δt may be taken to be in “lattice units”, meaning that $\Delta x = \Delta t = 1$. Typically, equation (2) is separated into a local collision step

$$\tilde{f}_\alpha(\mathbf{x}_i, t) = f_\alpha(\mathbf{x}_i, t) - \frac{1}{\tau} [f_\alpha(\mathbf{x}_i, t) - f_\alpha^{\text{eq}}(\mathbf{x}_i, t)], \quad (4a)$$

followed by a global propagation step

$$f_\alpha(\mathbf{x}_i + \mathbf{c}_\alpha \Delta t, t + \Delta t) = \tilde{f}_\alpha(\mathbf{x}_i, t). \quad (4b)$$

From equation (4) the corresponding residual vectors may be written as

$$\begin{aligned} \mathbf{R}^0 &= \mathbf{f}^0 - \mathbf{f}^{\text{init}}, \\ \mathbf{R}^n &= \mathbf{f}^n - S[\mathbf{f}^{n-1} + \frac{1}{\tau}(\mathbf{f}^{n-1} - \mathbf{f}^{\text{eq}, n-1})], \end{aligned} \quad (5)$$

where S is the streaming operator, which shifts the distribution values as in (4b), and \mathbf{f}^{init} is the vector of initial distribution values.

In this work we apply the incompressible lattice Boltzmann method introduced by He and Luo [28]. In this model the equilibrium distribution f^{eq} is given by the following second order expansion of the Maxwell–Boltzmann distribution:

$$f_\alpha^{\text{eq}}(\mathbf{x}_i, t) = w_\alpha \left(\rho + \rho_0 \left[3(\mathbf{c}_\alpha \cdot \mathbf{u}) + \frac{9}{2}(\mathbf{c}_\alpha \cdot \mathbf{u})^2 - \frac{3}{2}\mathbf{u}^2 \right] \right), \quad (6)$$

where ρ is the macroscopic density, which fluctuates slightly around the constant value ρ_0 , \mathbf{u} is the macroscopic velocity, and w_α are weights that depend on the discretization in velocity space. For the D2Q9 lattice, the weights are given by

$$w_\alpha = \begin{cases} 4/9, & \text{for } \alpha = 0, \\ 1/9, & \text{for } \alpha \in \{1, 2, 3, 4\}, \\ 1/36, & \text{for } \alpha \in \{5, 6, 7, 8\}. \end{cases} \quad (7)$$

Without loss of generality, the constant term ρ_0 may be taken to be unity. The macroscopic parameters, i.e. density, velocity, pressure, and viscosity can then be evaluated as:

$$\rho(\mathbf{x}_i, t) = \sum_{\alpha=0}^8 f_\alpha(\mathbf{x}_i, t), \quad (8a)$$

$$\rho_0 \mathbf{u}(\mathbf{x}_i, t) = \sum_{\alpha=0}^8 \mathbf{c}_\alpha f_\alpha(\mathbf{x}_i, t), \quad (8b)$$

$$p(\mathbf{x}_i, t) = c_s^2 \rho(\mathbf{x}_i, t), \quad (8c)$$

$$\nu = \left(\tau - \frac{1}{2} \right) c_s^2 \frac{(\Delta x)^2}{\Delta t}, \quad (8d)$$

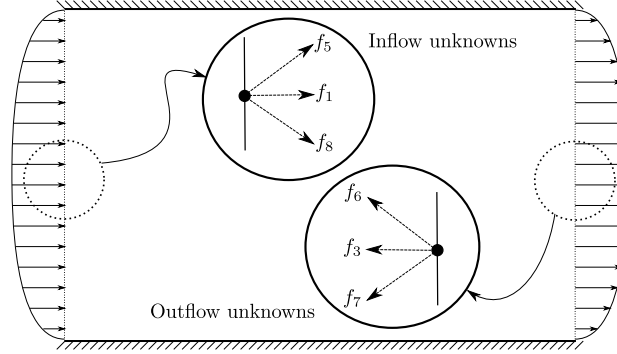


Fig. 2. Illustration of boundary conditions in a rectangular domain. At the boundary nodes, the incoming distributions need to be determined by some scheme depending on the type of boundary.

where p is the scalar pressure, ν is the kinematic viscosity, and c_s is the lattice speed of sound for the D2Q9 lattice. It can be shown that, in lattice units, $c_s^2 = 1/3$.

Assuming low Mach number flow, it can be shown—e.g. by the Chapman–Enskog expansion [29,10] or asymptotic analysis [30]—that the macroscopic values obtained by the above numerical scheme solve the incompressible NS equations for an isothermal fluid to first order accuracy in time and second order accuracy in space.

2.2. Boundary conditions

Enforcing boundary conditions in the LBM requires special treatment, since the governing equations describe the dynamics of the distributions functions f_i , but hydrodynamic boundary conditions are typically given in terms of the macroscopic properties of the fluid. For a general introduction to the implementation of boundary conditions in the LBM, the reader is referred to the book by Succi [31]. In this section, only the boundary conditions relevant to the numerical examples shall be discussed.

The basic idea of the implementation is that on the boundaries, there are certain velocity distributions that are unknown after the streaming step, since they can be thought of as having entered the computational domain from “outside”. Which distribution values are unknown depends on the orientation of the boundary, as illustrated in Fig. 2.

For inflow boundaries, a velocity inlet condition is imposed, using the method described by Zou and He [32]. Assuming a bounceback condition in the non-equilibrium part of the velocity distributions normal to the boundary (e.g. $f_1 - f_1^{\text{eq}} = f_3 - f_3^{\text{eq}}$ for an inflow on the west boundary as in Fig. 2), a closed system of equations is obtained. In the case of an inflow on the west boundary, solving this system for the unknowns yields

$$f_1 = f_3 + \frac{2}{3}\rho_0\bar{u}_x, \quad (9a)$$

$$f_5 = \frac{1}{6}\rho_0\bar{u}_x + \frac{1}{2}\rho_0\bar{u}_y + \frac{1}{2}(f_4 - f_2) + f_7, \quad (9b)$$

$$f_8 = \frac{1}{6}\rho_0\bar{u}_x - \frac{1}{2}\rho_0\bar{u}_y + \frac{1}{2}(f_2 - f_4) + f_6, \quad (9c)$$

where \bar{u}_x and \bar{u}_y is the prescribed velocity in the x - and y -direction, respectively. Similarly structured equations may be derived for other boundary orientations. These boundary conditions are also applied for no-slip closed boundaries, by simply setting $\bar{u}_x = \bar{u}_y = 0$ in (9).

For open boundaries, the zero normal shear stress (ZNS) boundary condition, also known as the no-friction condition, is imposed. This is a Neumann type boundary condition, and a method for implementing it in the LBM has been derived by Junk and Yang using asymptotic analysis [33]. Their derivation gives a more general scheme for the computation of boundary values, but here the equations shall simply be given in the form relevant to this paper, i.e. two dimensional flow with BGK collision operator. For an outflow located on the east boundary (Fig. 2), the unknown distributions may be computed as:

$$f_3(\mathbf{x}, t) = F_3^{\text{eq}}(1, \mathbf{u}(\mathbf{x}, t - \Delta t)) - (2\nu\tau^{-1} - 1) \times (f_1(\mathbf{x}, t - \Delta t) - f_1^{\text{eq}}(\mathbf{x}, t - \Delta t)), \quad (10a)$$

$$f_6(\mathbf{x}, t) = F_6^{\text{eq}}(1, \mathbf{u}(\mathbf{x}, t - \Delta t)) - 2\nu\tau^{-1}(w_6/w_3) \times (f_1(\mathbf{x}, t - \Delta t) - f_1^{\text{eq}}(\mathbf{x}, t - \Delta t)), \quad (10b)$$

$$f_7(\mathbf{x}, t) = F_7^{\text{eq}}(1, \mathbf{u}(\mathbf{x}, t - \Delta t)) - 2\nu\tau^{-1}(w_7/w_3) \\ \times (f_1(\mathbf{x}, t - \Delta t) - f_1^{\text{eq}}(\mathbf{x}, t - \Delta t)), \quad (10c)$$

where

$$F_i^{\text{eq}}(\rho, \mathbf{u}) = w_i \left(\rho + \rho_0 \left[3(\mathbf{c}_i \cdot \mathbf{u}) + \frac{9}{2}(\mathbf{c}_i \cdot \mathbf{u})^2 - \frac{3}{2}\mathbf{u}^2 \right] \right).$$

Again, equations of similar structure can be applied for different boundary orientations.

As a final remark, we note that in general, corner nodes need to be treated as special cases, since there are effectively five unknown distribution values rather than three. In the numerical examples, all corner nodes are no-slip nodes, and the on-grid bounceback scheme as described in [31] is applied. In this scheme the unknown distribution values are simply set equal to the known values in the reverse directions, i.e.:

$$f_1 \leftrightarrow f_3, \quad f_2 \leftrightarrow f_4, \quad f_5 \leftrightarrow f_7, \quad f_6 \leftrightarrow f_8. \quad (11)$$

This simple scheme ensures that the no-slip condition is satisfied provided that the initial distribution values for opposite velocities are equal.

2.3. Modeling fluid and solid domains in the LBM

For the purpose of topology optimization, the above LBM needs to be modified to allow for a continuous transition between fluid and solid nodes in the optimization procedure. One way to achieve this is to modify the method such that in solid regions, the Brinkman equation for flow in porous media is satisfied:

$$\frac{1}{\rho} \nabla p = -\frac{\nu}{K} \mathbf{u} + \tilde{\nu} \nabla^2 \mathbf{u}, \quad (12)$$

where K is the permeability of the medium, and $\tilde{\nu}$ is the so-called effective viscosity.

One approach which obeys the Brinkman equation in solid nodes is to adopt a partial bounceback formulation of the LBM. Partial bounceback models were first suggested by Dardis and McCloskey [34,35]; they are probabilistic meso-scale models in which a bounceback like term dependent on the permeability of the lattice node is added to the collision step. In this work, the partial bounceback model presented by Zhu and Ma [13] is adopted. The modified collision step has the form

$$\tilde{f}_\alpha(\mathbf{x}, t) = f_\alpha^c(\mathbf{x}_i, t) + g(s_i^{\text{ph}})[f_{\alpha'}^c(\mathbf{x}_i, t) - f_\alpha^c(\mathbf{x}_i, t)], \quad (13)$$

where

$$f_\alpha^c = f_\alpha(\mathbf{x}_i, t) - \frac{1}{\tau}[f_\alpha(\mathbf{x}_i, t) - f_\alpha^{\text{eq}}(\mathbf{x}_i, t)]$$

is the original post-collision value, $f_{\alpha'}^c$ is the post-collision value in the direction opposite f_α , as per (11), and $g \in [0, 0.5]$ is the fraction of particles bounced back at the node i . One can then choose $g: [0, 1] \rightarrow [0, 0.5]$ to be a continuous function such that

$$g(0) = 0.5, \quad g(1) = 0,$$

with $s_i^{\text{ph}} = 1$ corresponding to a fluid node, which recovers the standard collision step, and $s_i^{\text{ph}} = 0$ corresponding to a solid node, where a bounceback like collision step is obtained. Following Borrvall and Petersson [3], the following convex interpolation function is chosen:

$$g(s_i^{\text{ph}}) = 0.5 \left(1 - s_i^{\text{ph}} \frac{1 + \gamma}{s_i^{\text{ph}} + \gamma} \right), \quad (14)$$

where γ is an adjustable parameter which allows controlled penalization of intermediate permeabilities in the optimized designs. Increasing the value of γ results in a more linear scaling, which causes a greater penalization of intermediate values, see Fig. 3. While the function (14) was formulated by Borrvall and Petersson specifically for minimal dissipation type problems, it has also been found to work well for the numerical examples given in the present work.

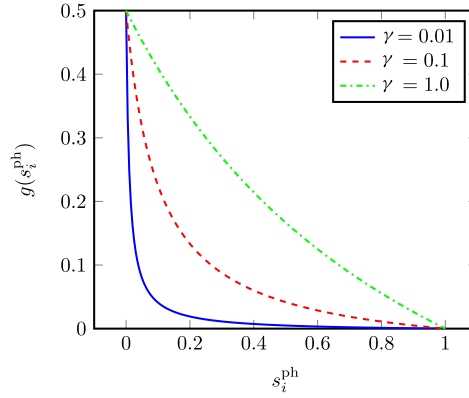


Fig. 3. Plot of the interpolation function (14) for different values of γ . Higher values result in more linear scaling and thus penalize “grey” design variables more.

3. Adjoint sensitivity analysis

A key component of the topology optimization procedure is the efficient evaluation of the gradients of the objective function and constraints with respect to the design variables. A method commonly used in topology optimization for determining the gradients is the adjoint method, which allows for all of the sensitivities to be computed from the transposed solution of a single linear system of equations.

Following the derivation by Kreissl et al. [22], the time discrete quantities are collected in column vectors:

$$\begin{aligned}\hat{\mathbf{z}} &= [z_0, \dots, z^{N_t}]^T, \\ \hat{\mathbf{R}} &= [\mathbf{R}^0, \dots, \mathbf{R}^{N_t}]^T, \\ \hat{\mathbf{f}} &= [\mathbf{f}^0, \dots, \mathbf{f}^{N_t}]^T, \\ \hat{\boldsymbol{\lambda}} &= [\boldsymbol{\lambda}^0, \dots, \boldsymbol{\lambda}^{N_t}]^T,\end{aligned}\tag{15}$$

where $\boldsymbol{\lambda}^n, n \in \{0, \dots, N_t\}$ is the vector of Lagrange multipliers at time step n . The adjoint equation for objective functionals of the form (1) is then given by

$$\left(\frac{\partial \hat{\mathbf{R}}}{\partial \hat{\mathbf{f}}}\right)^T \hat{\boldsymbol{\lambda}} = - \left(\frac{\partial \hat{\mathbf{z}}}{\partial \hat{\mathbf{f}}}\right)^T \frac{\partial \chi}{\partial \hat{\mathbf{z}}}.\tag{16}$$

Since the LBM is an explicit time marching scheme dependent only on the state in the previous time step, the Jacobian $\partial \hat{\mathbf{R}} / \partial \hat{\mathbf{f}}$ is a band matrix. For the initial step \mathbf{R}^0 we have

$$\frac{\partial \mathbf{R}^0}{\partial \mathbf{f}^j} = \begin{cases} \mathbf{I}, & \text{for } j = 0, \\ \mathbf{0}, & \text{for } j \neq 0, \end{cases}\tag{17}$$

where \mathbf{I} is the identity matrix. For subsequent time steps, $n > 0$, the following notation is introduced:

$$\frac{\partial \mathbf{R}^n}{\partial \mathbf{f}^j} = \begin{cases} \mathbf{C}^n, & \text{for } j = n - 1, \\ \mathbf{D}^n, & \text{for } j = n, \\ \mathbf{0}, & \text{for } j \notin \{n - 1, n\}. \end{cases}\tag{18}$$

Equation (16) may then be written in matrix form by performing the substitutions

$$\left(\frac{\partial \hat{\mathbf{R}}}{\partial \hat{\mathbf{f}}}\right)^T = \begin{pmatrix} \mathbf{I} & (\mathbf{C}^1)^T & & & \\ & (\mathbf{D}^1)^T & (\mathbf{C}^2)^T & & \\ & & (\mathbf{D}^2)^T & \ddots & \\ & & & \ddots & (\mathbf{C}^{N_t})^T \\ & & & & (\mathbf{D}^{N_t})^T \end{pmatrix},\tag{19}$$

$$\left(\frac{\partial \hat{\mathbf{z}}}{\partial \hat{\mathbf{f}}}\right)^T \frac{\partial \chi}{\partial \hat{\mathbf{z}}} = \begin{pmatrix} (\partial z^0 / \partial \mathbf{f}^0)^T (\partial \chi / \partial z^0) \\ \vdots \\ (\partial z^{N_t} / \partial \mathbf{f}^{N_t})^T (\partial \chi / \partial z^{N_t}) \end{pmatrix}. \quad (20)$$

The band structure of the matrix (19) allows the adjoint problem to be conveniently solved backward in time, starting by solving for λ^{N_t} :

$$\lambda^{N_t} = (\mathbf{D}^{N_t})^{-T} \left(\frac{\partial z^{N_t}}{\partial \mathbf{f}^{N_t}}\right)^T \frac{\partial \chi}{\partial z^{N_t}}. \quad (21a)$$

Subsequent λ^n may then be solved by backward substitution:

$$\lambda^n = (\mathbf{D}^n)^{-T} \left[\left(\frac{\partial z^n}{\partial \mathbf{f}^n}\right)^T \frac{\partial \chi}{\partial z^n} - (\mathbf{C}^{n+1})^T \lambda^{n+1} \right]. \quad (21b)$$

Once $\hat{\lambda}$ has been fully determined, the gradient with respect to the physical variable \mathbf{s}^{ph} may be computed by

$$\frac{dZ}{d\mathbf{s}_k^{\text{ph}}} = \frac{\partial \mathcal{Z}}{\partial \mathbf{s}_k^{\text{ph}}} + \left(\frac{\partial \hat{\mathbf{z}}}{\partial \hat{\mathbf{z}}}\right)^T \frac{\partial \hat{\mathbf{z}}}{\partial \mathbf{s}_k^{\text{ph}}} + (\hat{\lambda})^T \frac{\partial \mathbf{R}}{\partial \mathbf{s}_k^{\text{ph}}}. \quad (22)$$

While the matrix form of equation (16) neatly illustrates the overall structure of the adjoint problem, implementing the adjoint analysis using matrix routines requires assembling the matrices \mathbf{D}^n and \mathbf{C}^n in each time step. The computational cost of such an implementation quickly becomes intractable as the number of time steps considered increases. Alternatively, it is possible to exploit the local nature of the lattice Boltzmann equation to derive explicit expressions for the Lagrange multipliers in each time step, thereby bypassing the need for matrix routines entirely; in this way the adjoint problem can be solved on the same time scale as the forward problem. A detailed derivation of a discrete adjoint lattice Boltzmann method has already been given by Liu et al. [36], who applied it to steady-state fluid topology optimization problems. Below we give an outline of the derivation of the explicit adjoint expressions, including the treatment of the boundary conditions described in section 2.2, which to our knowledge have not been covered before.

3.1. Derivation of adjoint equations

For the purpose of this derivation, we will consider an arbitrary objective of the form (1). From (21) it is apparent that there will be a contribution from the source term $(\partial z^n / \partial \mathbf{f}^n)(\partial \chi / \partial z^n)$ for each multiplier $\lambda_\alpha(\mathbf{x}, t)$. We will denote this contribution

$$G_\alpha(\mathbf{x}, t) = - \left(\frac{\partial z^t}{\partial f_\alpha(\mathbf{x}, t)}\right)^T \frac{\partial \chi}{\partial z^t}. \quad (23)$$

Beyond this ubiquitous contribution, it is necessary to derive different equations for interior nodes and each type of boundary node. The various types will be considered in turn.

3.1.1. Interior nodes

Since the interior nodes only require information about distribution values in the previous time step, the factor $(\mathbf{D}^n)^{-T}$ in (21b) can effectively be considered to be an identity term for the purpose of the interior nodes. What is left to consider is then the term $(\mathbf{C}^{n+1})^T \lambda^{n+1} = (\partial \mathbf{R}^{n+1} / \partial \mathbf{f}^n)^T \lambda^{n+1}$. The non-zero terms for a given node will be at the neighboring residuals, i.e. the nodes where information has been propagated to. Specifically, the individual Lagrange multipliers may be obtained as

$$\lambda_\alpha(\mathbf{x}, t) = G_\alpha(\mathbf{x}, t) - \sum_{j=0}^8 \frac{\partial R_j(\mathbf{x} + \mathbf{c}_j \Delta t, t + \Delta t)}{\partial f_\alpha(\mathbf{x}, t)} \lambda_j(\mathbf{x} + \mathbf{c}_j \Delta t, t + \Delta t), \quad (24)$$

where $\alpha \in \{0, \dots, 8\}$ corresponds to a multiplier for each distribution value at the node. Because of the non-linearity of the collision step, the sum of differentials in (24) results in a quite formidable expression. A practical way to implement this is to automatically generate the code for evaluating it using a computer algebra system, e.g. Maple.

3.1.2. Inflow boundaries

We will consider an inflow on the western boundary of the domain, such that the unknown distributions are computed by (9). The necessary adjoint computations on this boundary differ from those of the interior nodes in one significant way: since three distributions are computed using distribution values from the same time step, the matrix factor $(\mathbf{D}^n)^{-T}$ can no

longer be considered an identity. This can be handled by multiplying the set of nine Lagrange multipliers in each inflow point by a 9×9 matrix. For a west inflow, the matrix is

$$A = \left(\frac{\partial R_j(\mathbf{x}, t)}{\partial \mathbf{f}(\mathbf{x}, t)} \right)^{-T} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & -\frac{1}{2} & 0 & 0 & \frac{1}{2} \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & \frac{1}{2} & 0 & 0 & -\frac{1}{2} \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}, \tag{25}$$

that is, the inverse transpose of the residual Jacobian at the point \mathbf{x} . The nine multipliers at each inflow node are then found by computing (24) and multiplying the result with the matrix (25). For the bounceback condition on the corners, the Lagrange multipliers may be computed similarly.

3.1.3. Open boundaries

For the open boundaries, the unknown distributions are computed using only information from the previous time step, just like the interior nodes. Thus, a matrix like (25) is not needed. Rather, equation (24) needs a slight adjustment:

$$\begin{aligned} \lambda_\alpha(\mathbf{x}, t) &= G_\alpha(\mathbf{x}, t) \\ &- \sum_{j \in \Lambda_s} \frac{\partial R_j(\mathbf{x} + \mathbf{c}_j \Delta t, t + \Delta t)}{\partial f_\alpha(\mathbf{x}, t)} \lambda_j(\mathbf{x} + \mathbf{c}_j \Delta t, t + \Delta t) \\ &- \sum_{j \in \Lambda_u} \frac{\partial R_j(\mathbf{x}, t + \Delta t)}{\partial f_\alpha(\mathbf{x}, t)} \lambda_j(\mathbf{x}, t + \Delta t) \end{aligned} \tag{26}$$

here Λ_s refers to the set of velocities that can stream to a neighbor within the domain, and Λ_u is the set of unknown distributions. For an open boundary on the east part of the domain, $\Lambda_s = \{0, 2, 3, 4, 6, 7\}$, and $\Lambda_u = \{3, 6, 7\}$.

3.2. Implementation

The resulting algorithm for obtaining the objective and sensitivities is summarized by pseudo-code in Algorithm 1. Note that equation (24) results in expressions which depend on $\mathbf{f}(\mathbf{x}, t)$, meaning that the backward time stepping in the adjoint problem requires the value of the state variable \mathbf{f}^n in every time step $n \in \{N_t, N_t - 1, \dots\}$. In this work, we have taken the naive approach of simply storing the full history in memory. For larger scale problems (i.e. in 3D), storing the entire history of the forward analysis would be prohibitively memory intensive. However, the memory requirement can be significantly reduced—at the cost of extra computation—by implementing a checkpointing algorithm, cf. [37,38].

Algorithm 1 Compute Z and dZ/ds .

```

for all  $n \in \{1 \dots N_t\}$  do
  Advance current time step from  $n - 1$  to  $n$  by the lattice Boltzmann equation (2).
  Save the vector  $\mathbf{f}^n$  in memory.
end for
Compute the objective  $Z$  by equation (1).
for all  $n \in \{N_t \dots 1\}$  do
  if  $n = N_t$  then
    Compute  $\lambda^{N_t}$  by (21a).
  else
    Advance current time step from  $n$  to  $n - 1$  by equation (24).
  end if
end for
Compute  $dZ/ds^{ph}$  by equation (22).
Compute  $dZ/ds$  from  $dZ/ds^{ph}$  by the chain rule (see section 4).

```

4. Filtering

For topology optimization problems in unsteady fluid dynamics, taking the physical and design variables to be equal, i.e. $\mathbf{s} = \mathbf{s}^{ph}$, can lead to numerical problems in the optimization procedure. Since the design and physical variables are equivalent, the optimizer can add features to the design on the length scale of a single grid point. This makes it very difficult

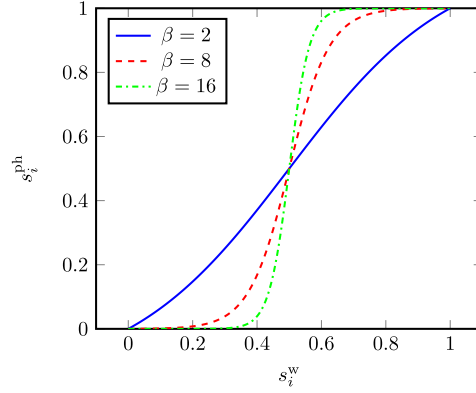


Fig. 4. Plot of the projection function (27) with a threshold of $\eta = 0.5$ for different values of β .

to obtain reasonable solutions, because the large search space combined with the objective functions non-linear response to the design variables makes it unlikely that the optimization procedure will converge to a well defined design. Rather, designs will typically converge to local minima with poorly resolved topologies. This pathology arises in both the numerical examples given below. To alleviate such issues, several regularization techniques have been proposed in the literature [1].

For this work, in order to regularize the optimized design and obtain a discrete 0/1 result, a Heaviside projection filter is applied. Several variations of this idea have been described in published works [39–41]. The steps necessary for applying the projection filter are briefly described below.

4.1. Threshold projection

In order to obtain a black and white solution, the following smoothed Heaviside function given by Wang et al. [42] is applied:

$$s_i^{\text{ph}} = \frac{\tanh(\beta\eta) + \tanh(\beta(s_i^{\text{w}} - \eta))}{\tanh(\beta\eta) + \tanh(\beta(1 - \eta))}. \quad (27)$$

Here, β is a control parameter for the function which determines the steepness of the region around the threshold, see Fig. 4, and η gives the threshold for the projection, so that values above η are projected toward 1 and values below are projected toward 0. Note that in the limit $\beta \rightarrow 0$, $s_j^{\text{w}} = s_j^{\text{ph}}$, so that the unprojected variable is recovered. The variable s_j^{w} refers to a smoothing of the design variables given by the weighted averaging

$$s_j^{\text{w}} = \frac{\sum_{k \in \mathbb{N}_{e,j}} H(\mathbf{x}_k) s_k}{\sum_{k \in \mathbb{N}_{e,j}} H(\mathbf{x}_k)}, \quad (28)$$

where $\mathbb{N}_{e,j}$ is the neighborhood of elements within a filter radius r , and $H(\mathbf{x}_k)$ is the weighting factor given by

$$H(\mathbf{x}_k) = r - |\mathbf{x}_k - \mathbf{x}_j|, \quad (29)$$

where \mathbf{x}_j and \mathbf{x}_k are the center point coordinates of the design elements j and k .

Once the design variables have been mapped to the physical variables by applying the projection technique, the sensitivities of the objective function with respect to the original design variables can be computed using the chain rule:

$$\frac{\partial Z}{\partial s_k} = \sum_{j \in \mathbb{N}_{e,k}} \frac{\partial Z}{\partial s_j^{\text{ph}}} \frac{\partial s_j^{\text{ph}}}{\partial s_j^{\text{w}}} \frac{\partial s_j^{\text{w}}}{\partial s_k}, \quad (30)$$

where

$$\frac{\partial s_j^{\text{w}}}{\partial s_k} = \frac{H(\mathbf{x}_k)}{\sum_{k \in \mathbb{N}_{e,j}} H(\mathbf{x}_k)}. \quad (31)$$

We emphasize that the projection filter does not necessarily ensure mesh independence in the final design [42]. This can be alleviated by employing the robust formulation given by Wang et al. [42], or by geometric constraints as presented by Zhou et al. [43], but application of these techniques to fluid topology optimization has yet to be investigated.

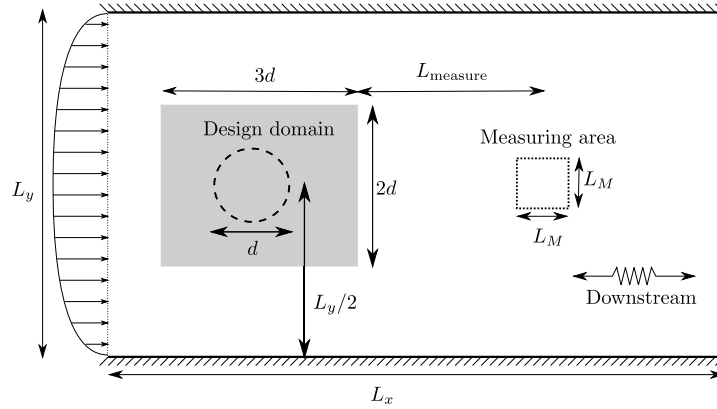


Fig. 5. Illustration of the computational domain for the flow control problem. Note that the downstream section of the computational domain is much longer than the figure would indicate, but this is irrelevant for illustrative purposes.

4.2. β -continuation

As shown in Fig. 4, increasing the value of the parameter β has the result that the threshold function (27) will project a greater range of values towards a pure solid or fluid node. Therefore, ideally the value of β should be large in order to ensure a crisp black and white design. However, using an initially large value of β typically leads to numerical problems such as design oscillations, since the sensitivities for design variables near the threshold η will be comparatively large. Instead, a common strategy is the β -continuation approach. The optimization is run with an initially small value of β ; the parameter is then periodically increased to facilitate proper convergence to a black and white design. This strategy has been employed for both the numerical examples given below.

5. Numerical examples

The topology optimization framework is now exemplified by two numerical examples. In both cases, a domain is considered where the fluid is initially at rest, and all distribution functions are at their equilibrium values.

The design is updated in each iteration by the method of moving asymptotes (MMA), due to Svanberg [44]. The domain is initialized to some initial design guess (which will be detailed below), and the optimization procedure is considered to be converged if

$$\|\mathbf{s}^m - \mathbf{s}^{m-1}\|_\infty < \epsilon, \quad (32)$$

where m is the current iteration step, and ϵ is some small tolerance parameter. For the examples below, the value $\epsilon = 10^{-4}$ has been used.

5.1. Controlling flow past an obstacle

For the first numerical example, we consider a problem that exhibits significant unsteady effects, but in which an exact (but not necessarily unique) global optimum is known to exist. It is well known that even at relatively moderate Reynolds numbers, in the wake of an obstacle, complicated flow patterns can arise. The goal of this problem is to control the topology of an obstacle in order to recover a given spatial and temporal flow profile. The geometry of the problem is shown in Fig. 5. To obtain a reference flow profile, an initial simulation is run with a cylindrical obstacle (as shown with a dashed contour in Fig. 5). The flow is allowed to develop for a set number of time steps to ensure that vortex shedding has started. For the remaining time steps, the temporal profile of the velocities in the measuring area is then saved. During the optimization procedure, we seek to minimize the time-averaged difference between the reference velocity profile and the profile of the obstacle generated by topology optimization.

Mathematically, the optimization problem is formulated as follows:

$$\begin{aligned} \min_{\mathbf{s}} Z &= \log \sum_{n=N_0}^{N_t} \sum_{h \in M} \frac{\|\mathbf{u}(\mathbf{x}_h, t_n) - \mathbf{u}_{\text{ref}}(\mathbf{x}_h, t_n)\|^2}{N_m M_A}, \\ \text{s.t. } \begin{cases} G_0 = \frac{1}{N_s} \sum_{k=1}^{N_s} s_k^{\text{ph}} - 0.9 \leq 0, \\ G_1 = \Delta p - \xi \Delta p_{\text{ref}} \leq 0, \\ 0 \leq s_k \leq 1, \quad \forall k = 1, \dots, N_s. \end{cases} \end{aligned} \quad (33)$$

Table 1
Numerical parameters used for the flow control example.

| | |
|----------------------|--------------|
| L_x | 400 |
| L_y | 100 |
| L_{measure} | $L_x/4$ |
| d | $L_y/5$ |
| N_0 | 10 000 |
| N_t | 15 000 |
| γ | 1.0 |
| ξ | 1.5 |
| r | 0.1 |
| β_{max} | $(1.4)^{15}$ |

Here, M denotes the nodes in the measuring domain, and M_A denotes its area. The deviation from the reference velocity is computed over $N_m = N_t - N_0$ time steps. The problem is constrained by the volume constraint G_0 which constrains the amount of fluid allowed in the design domain, corresponding to the fluid amount around the reference obstacle. Additionally, the constraint G_1 is a pressure drop constraint, with Δp being the time averaged pressure drop in the cavity. The reference pressure drop Δp_{ref} is the pressure drop in the cavity for the cylinder, and ξ is a constant tolerance parameter.

Because the spatial and temporal flow profile is dependent on the obstacle in a highly non-linear way, design features on length scales down to one node can have significant effects on the flow in the wake of the obstacle. Therefore, in order to regularize the problem, filtering is applied.

To ensure convergence to a black and white design, the β -continuation approach described in section 4.2 is employed. The initial value of $\beta = 1$ is used, which is increased by the constant factor 1.4 every 50 iterations until a maximal value $\beta_{\text{max}} = (1.4)^{15}$ is reached. For the projection, the value $\eta = 0.5$ has been used. Note that in order to ensure the existence of a globally optimal solution, the reference velocity is obtained by performing projection on the initial cylinder, with the value of β set to its maximum value, and then measuring the resulting flow profile.

As the initial design, the design domain is initialized to be equal to the volume fraction everywhere. For the purpose of the filter, the vertical length L_y will be considered scaled to unity, so that a filter radius $0 < r < 1$ may be chosen independently of the mesh refinement. The remaining numerical values used for the problem are listed in Table 1, and example obstacle designs are shown in Fig. 6. For these examples, the Reynolds number with respect to the initial cylinder diameter is 60.

From the examples in Fig. 6(b)–6(c), it is clear that the projection filter as well as the β -continuation strategy is essential to ensure convergence to a well defined black and white solution. In all cases, the known global optimum is not found, but with projection a reasonable obstacle is obtained for different sizes of the measuring area.

The convergence history for two of the examples is given in Fig. 7. The spikes in the plot are due to the β -continuation, but it is apparent that these oscillations become smaller and smaller as the optimization progresses. For a larger measuring area we observe that the final objective has a higher value, since increasing the amount of points in which the velocities should match makes the optimization problem harder. That is, increasing the measurement area results in a mathematically stronger objective which is more difficult to minimize. It is not immediately clear which one of the two examples provides the “best” reproduction of cylindrical vortex shedding, however. We discuss this further below. In both cases, the pressure drop constraint is initially violated, but this is handled within 10 iterations of the MMA algorithm, after which both constraints are fulfilled for the remainder of the optimization procedure.

Note that for both examples in Fig. 6(d)–6(e), the obstacle is shifted to the left compared to the reference obstacle. As can be seen in Fig. 8, the phases of the vortex shedding match up quite well in spite of this, even though the magnitudes differ. As is apparent from Fig. 9, the more elongated shape of the computed obstacle results in a longer “tail” in the immediate wake of the obstacle, so that the vortices still match up fairly well in spite of the computed obstacle being shifted to the left.

To conclude this section, some additional comments should be made about the formulation (33). If we consider the two examples in Fig. 6(d) and 6(e), it is clear that the relative difference between their shape is quite small, and yet the difference in terms of the objective function is substantial. Indeed, it turns out that if the objective of the design optimized for a 5×5 measuring area is computed using a 9×9 measuring area, this design actually performs better than the design specifically optimized for the 9×9 area. Given the relatively small difference in the two designs, this demonstrates the non-linear nature of the problem and the likelihood that the optimization algorithm will end up in a local minimum. However, as Figs. 8 and 9 demonstrate, the performance of such minima can still be quite good.

5.2. A fluid pump

For the second example, we consider the problem of optimizing a simplified fluid pump. The basic idea is to have a vertical inflow on which a periodic inflow is imposed, and then optimize the interior domain to ensure that fluid is

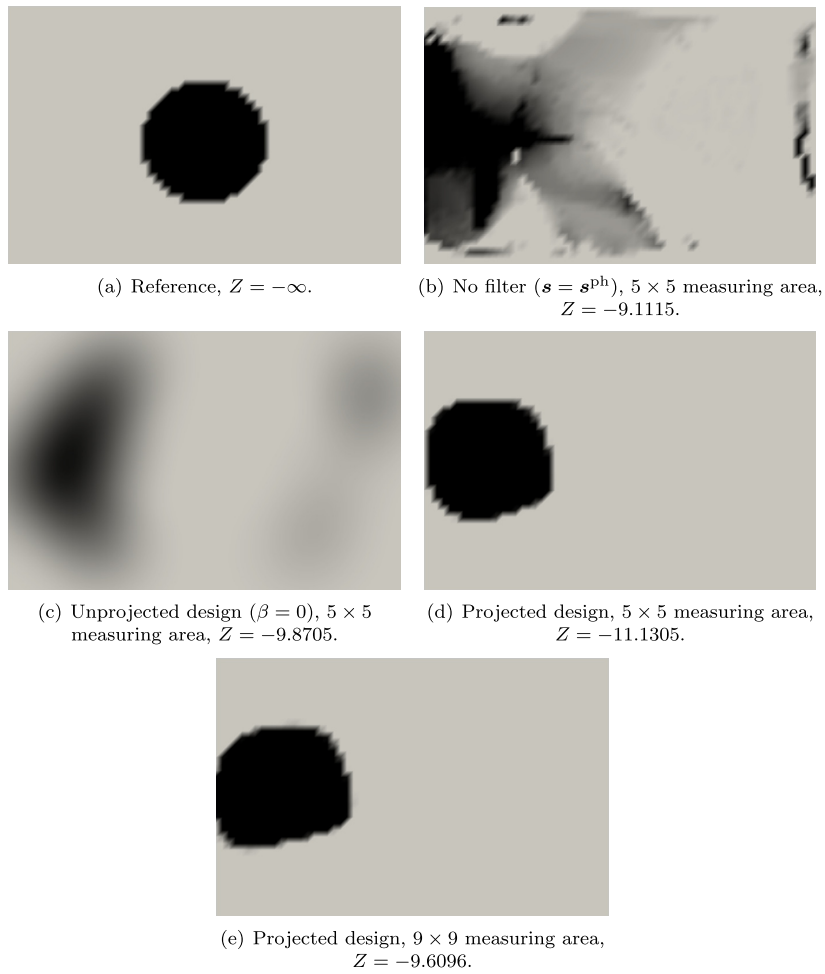


Fig. 6. Example designs for the flow control problem. Only the design domain is shown.

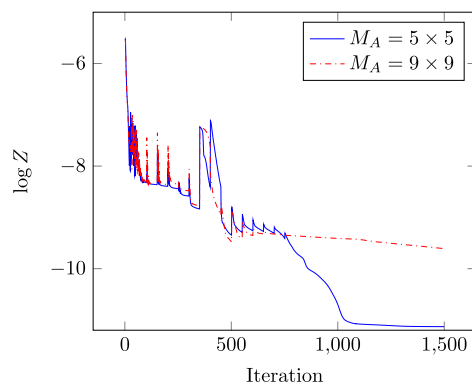


Fig. 7. Convergence plot of the objective for the examples in Fig. 6(d) and 6(e).

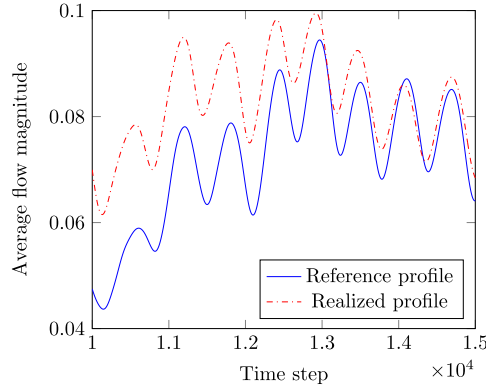


Fig. 8. Comparison of the reference and attained profile for the obstacle shown in Fig. 6(e).

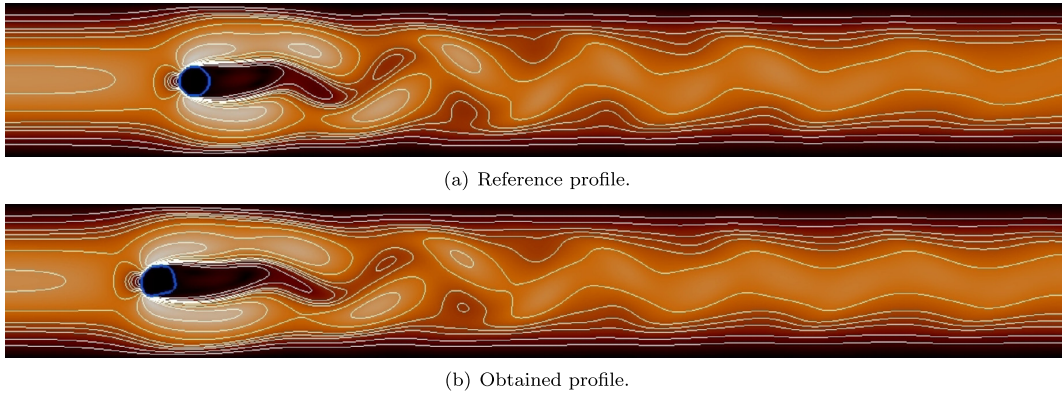


Fig. 9. Qualitative comparison of the velocity magnitude contours of the reference design (Fig. 6(a)) and a computed obstacle (Fig. 6(e)) in a single time step. The blue contour line indicates the obstacle. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

transported horizontally from the left inlet of the domain to the right outlet. More precisely, a sinusoidally varying parabolic inflow with maximal velocity u_{\max} given by

$$u_y(t^n) = u_{\max} \sin(\pi t^n / \nu), \tag{34}$$

is imposed, where ν is some frequency. The computational domain is illustrated in Fig. 10.

The goal is to maximize the horizontal velocity at the open boundary marked “pump outflow”. The second open boundary will effectively act as a reservoir from which additional fluid can enter the domain. The optimization problem is formulated as follows:

$$\begin{aligned} \min_s Z &= -\frac{1}{N_t L_{\text{out}}} \sum_{n=0}^{N_t} \sum_{\mathbf{x}_i \in \Gamma_{\text{out}}} u_x(\mathbf{x}_i, t_n) \\ \text{s.t.} \quad &\begin{cases} G_0 = \frac{1}{N_s} \sum_{k=1}^{N_s} s_k^{\text{ph}} - V \leq 0, \\ 0 \leq s_k \leq 1, \quad \forall k = 1, \dots, N_s. \end{cases} \end{aligned} \tag{35}$$

Here, Γ_{out} denoted the set of grid points at the pump outflow boundary, and L_{out} denoted the length of the boundary. Once again, a volume constraint on the amount of allowed fluid in the design domain is applied to the problem.

As with the obstacle design problem, projection is applied in order to regularize the design and ensure a black and white solution. Once again, the value $\eta = 0.5$ is used for the projection function, and the value of β is iteratively increased during the course of the optimization procedure. The design variables are initialized to be equal to the volume fraction everywhere. The remaining numerical values for this problem are listed in Table 2. The Reynolds number with respect to the top inflow length is 80.

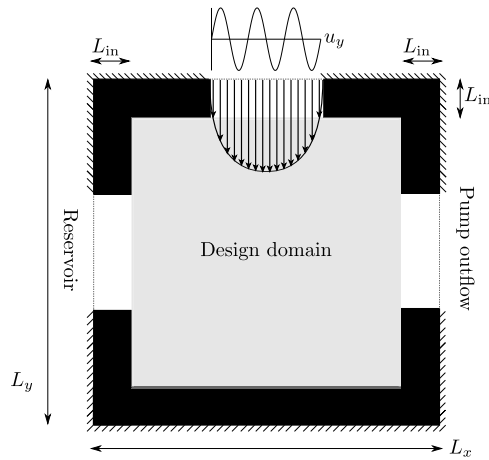
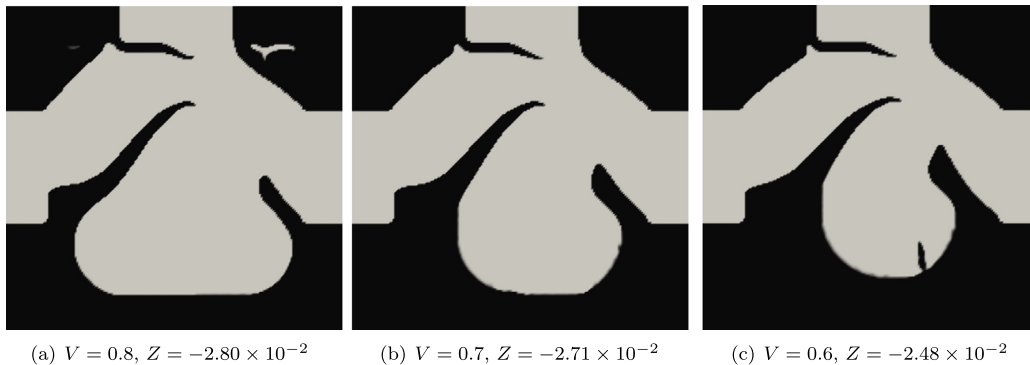


Fig. 10. Illustration of the computational domain for the fluid pump problem.

Table 2
Numerical parameters used for the pump example.

| | |
|---------------|--------------|
| L_x | 204 |
| L_y | 204 |
| L_{in} | 25 |
| N_t | 8000 |
| γ | 0.5 |
| r | 0.03 |
| ν | 400 |
| u_{max} | 0.05 |
| β_{max} | $(1.4)^{15}$ |



(a) $V = 0.8$, $Z = -2.80 \times 10^{-2}$ (b) $V = 0.7$, $Z = -2.71 \times 10^{-2}$ (c) $V = 0.6$, $Z = -2.48 \times 10^{-2}$

Fig. 11. Example pump designs for different values of the volume fraction.

Example designs for this problem for different values of the volume constraint V are shown in Fig. 11. The imposed flow from the top is cyclical, consisting of an inflow phase in which fluid is flowing into the domain from the top, and an outflow phase in which fluid is sucked out from the top; an optimized design needs to account for both these phases. Each design in Fig. 11 exhibits the same basic feature of a narrowing of the inflow channel, and a reservoir in the middle which is “shielded” by a protrusion on the left. The performance of the design example in Fig. 11(a) is illustrated by the Figs. 12 and 13. Fig. 12 shows the pump output as a function of the imposed pumping velocity. Note that after the first few cycles, the output is never negative, meaning that fluid never flows back into the domain at the output, even during the outflow cycle. The mechanism by which this is achieved is shown by streamlines in Fig. 13. During the inflow cycle, fluid from the top flows towards the pumping outlet, while additional fluid is pulled from the left reservoir (left open boundary) towards the central reservoir. During the outflow phase, fluid from the central reservoir flows toward the pumping outlet, while fluid

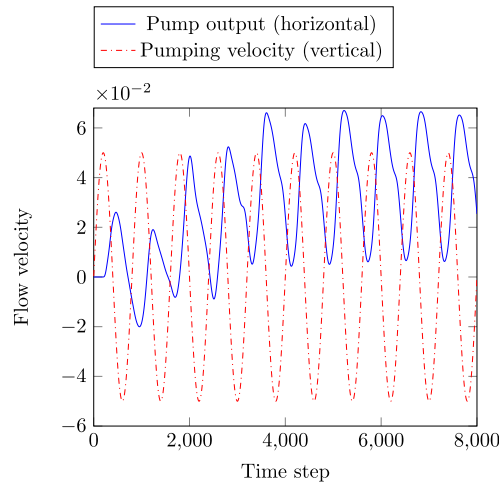


Fig. 12. Performance of the pump design in Fig. 11(a) as a function of time.

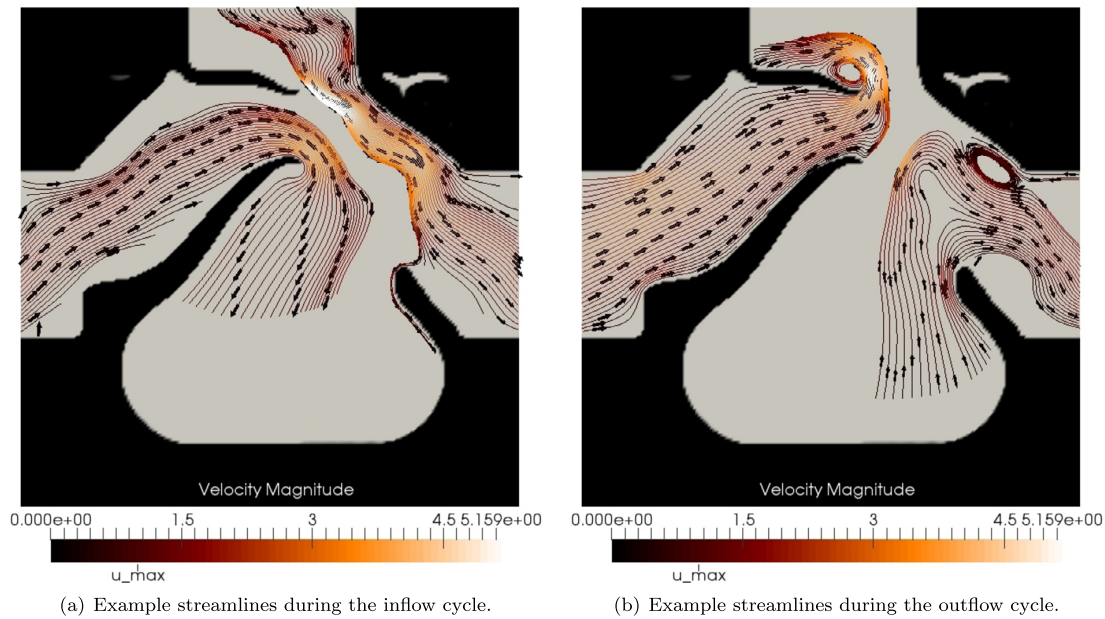


Fig. 13. Streamlines of the flow in the design in Fig. 11(a) during selected timesteps in the inflow and outflow phase. Note that the velocity magnitude has been scaled so that $\|u_{\max}\| = 1$.

from the left reservoir flows in a vortex like path around the top inlet; this fluid is then transported towards the pumping outlet during the next inflow phase.

6. Conclusion

Fluid topology optimization using the lattice Boltzmann method has been successfully applied to problems which exhibit significant unsteady effects. The topologies of the computed solutions were described by the density approach, while the fluid-solid boundary conditions were imposed by a partial bounceback model. By solving the lattice Boltzmann equation, and the corresponding discrete adjoint equation, the objective function as well as the design sensitivities were determined in a computationally efficient manner, which allowed solving problems where many time steps were needed to resolve the dynamical effects. In addition, the necessity of a transformation of the design variables in order to ensure convergence to

well defined designs has been demonstrated. It is emphasized that the applied map from design to physical variables does not guarantee mesh independence of solutions, but is nonetheless necessary for obtaining well defined designs. While the proposed formulation can be extended to 3D problems, the increase in both computational cost and memory footprint is likely to pose a significant challenge.

Acknowledgements

The authors acknowledge the financial support received from the TopTen project sponsored by the Danish Council for Independent Research (DFF-4005-00320), from the NextTop project, sponsored by the Villum Foundation, and the Hypercool project sponsored by the Innovation Fund Denmark (117-2014-1).

References


- [1] M.P. Bendsoe, O. Sigmund, *Topology Optimization: Theory, Methods and Applications*, Springer, 2004.
- [2] M.P. Bendsoe, N. Kikuchi, Generating optimal topologies in structural design using a homogenization method, *Comput. Methods Appl. Mech. Eng.* (ISSN 0045-7825) 71 (2) (1988) 197–224.
- [3] T. Borrvall, J. Petersson, Topology optimization of fluids in Stokes flow, *Int. J. Numer. Methods Fluids* 41 (1) (2003) 77–107, <http://dx.doi.org/10.1002/flid.426>, ISSN 14333015, 02683768, 10970363, 02712091.
- [4] A. Gersborg-Hansen, O. Sigmund, R.B. Haber, Topology optimization of channel flow problems, *Struct. Multidiscip. Optim.* 30 (3) (2005) 181–192, <http://dx.doi.org/10.1007/s00158-004-0508-7>, ISSN 16151488, 1615147x.
- [5] L.H. Olesen, F. Okkels, H. Bruus, A high-level programming-language implementation of topology optimization applied to steady-state Navier–Stokes flow, *Int. J. Numer. Methods Eng.* 65 (7) (2006) 975–1001, <http://dx.doi.org/10.1002/nme.1468>, ISSN 10970207, 00295981.
- [6] N. Aage, A. Gersborg-Hansen, O. Sigmund, Topology optimization of large scale Stokes flow problems, *Struct. Multidiscip. Optim.* 35 (2008) 175–180, <http://dx.doi.org/10.1007/s00158-007-0128-0>, ISSN 16151488, 1615147x.
- [7] C.S. Andreasen, A.R. Gersborg, O. Sigmund, Topology optimization of microfluidic mixers, *Int. J. Numer. Methods Fluids* 61 (5) (2009) 498–513, <http://dx.doi.org/10.1002/flid.1964>, ISSN 10970363, 02712091.
- [8] D. Makhija, G. Pinggen, R. Yang, K. Maute, Topology optimization of multi-component flows using a multi-relaxation time lattice Boltzmann method, *Comput. Fluids* 67 (2012) 104–114, <http://dx.doi.org/10.1016/j.compfluid.2012.06.018>, ISSN 18790747, 00457930, 10960341, 00426822.
- [9] M.C. Sukop, D.T. Thorne Jr., *Lattice Boltzmann Modelling*, 2nd edn., Springer, 2007.
- [10] D.A. Wolf-Gladrow, *Lattice-Gas Cellular Automata and Lattice Boltzmann Models – An Introduction*, Springer, 2005.
- [11] M.A.A. Spaid, F. Phelan, Lattice Boltzmann methods for modeling microscale flow in fibrous porous media, *Phys. Fluids* 9 (9) (1997) 2468–2474, <http://dx.doi.org/10.1063/1.869392>, ISSN 10897666, 10706631.
- [12] S.D.C. Walsh, H. Burwinkle, M.O. Saar, A new partial-bounceback lattice-Boltzmann method for fluid flow through heterogeneous media, *Comput. Geosci.* 35 (6) (2009) 1186–1193, <http://dx.doi.org/10.1016/j.cageo.2008.05.004>, ISSN 18737803, 00983004.
- [13] J. Zhu, J. Ma, An improved gray lattice Boltzmann model for simulating fluid flow in multi-scale porous media, *Adv. Water Resour.* 56 (2013) 61, ISSN 03091708, 18729657.
- [14] T. Inamuro, T. Ogata, S. Tajima, N. Konishi, A lattice Boltzmann method for incompressible two-phase flows with large density differences, *J. Comput. Phys.* 198 (2) (2004) 628–644, <http://dx.doi.org/10.1016/j.jcp.2004.01.019>, ISSN 10902716, 00219991, 10957103, 00219797.
- [15] P. Asinari, Semi-implicit-linearized multiple-relaxation-time formulation of lattice Boltzmann schemes for mixture modeling, *Phys. Rev. E* 73 (5) (2006) 056705, <http://dx.doi.org/10.1103/physreve.73.056705>, ISSN 15502376, 15393755.
- [16] G. Pinggen, A. Evgrafov, K. Maute, Topology optimization of flow domains using the lattice Boltzmann method, *Struct. Multidiscip. Optim.* 34 (6) (2007) 507–524, <http://dx.doi.org/10.1007/s00158-007-0105-7>, ISSN 16151488, 1615147x.
- [17] G. Pinggen, M. Waidmann, A. Evgrafov, K. Maute, A parametric level-set approach for topology optimization of flow domains, *Struct. Multidiscip. Optim.* 41 (1) (2010) 117–131, <http://dx.doi.org/10.1007/s00158-009-0405-1>, ISSN 16151488, 1615147x.
- [18] S. Kreissl, G. Pinggen, K. Maute, An explicit level set approach for generalized shape optimization of fluids with the lattice Boltzmann method, *Int. J. Numer. Methods Fluids* 65 (5) (2011) 496–519, <http://dx.doi.org/10.1002/flid.2193>, ISSN 10970363, 02712091.
- [19] K. Yajii, T. Yamada, M. Yoshino, Topology optimization using the lattice Boltzmann method incorporating level set boundary expressions, *J. Comput. Phys.* 274 (2014) 158.
- [20] M.M. Tekitek, M. Bouzidi, F. Dubois, P. Lallemand, Adjoint lattice Boltzmann equation for parameter identification, *Comput. Fluids* 35 (8–9) (2006) 805–813, <http://dx.doi.org/10.1016/j.compfluid.2005.07.015>, ISSN 18790747, 00457930, 10960341, 00426822.
- [21] M.J. Krause, G. Thaefer, V. Heuveline, Adjoint-based fluid flow control and optimisation with lattice Boltzmann methods, *Comput. Math. Appl.* 65 (6) (2013) 945–960, <http://dx.doi.org/10.1016/j.camwa.2012.08.007>, ISSN 18737668, 08981221, 00199958.
- [22] S. Kreissl, G. Pinggen, K. Maute, Topology optimization for unsteady flow, *Int. J. Numer. Methods Eng.* 87 (13) (2011), <http://dx.doi.org/10.1002/nme.3151>, ISSN 10970207, 00295981.
- [23] Y. Deng, Z. Liu, P. Zhang, Y. Liu, Y. Wu, Topology optimization of unsteady incompressible Navier–Stokes flows, *J. Comput. Phys.* 230 (17) (2011) 6688–6708, <http://dx.doi.org/10.1016/j.jcp.2011.05.004>, ISSN 00219991, 10902716.
- [24] K. Yonekura, Y. Kanno, A flow topology optimization method for steady state flow using transient information of flow field solved by lattice Boltzmann method, *Struct. Multidiscip. Optim.* 51 (1) (2015) 159–172, <http://dx.doi.org/10.1007/s00158-014-1123-x>, ISSN 1615147x, 16151488, 1615147x.
- [25] X. He, L. Luo, A priori derivation of the lattice Boltzmann equation, *Phys. Rev. E* 55 (1997) R6333–R6336, <http://dx.doi.org/10.1103/PhysRevE.55.R6333>, ISSN 10953787, 1063651x.
- [26] X. He, L. Luo, Theory of the lattice Boltzmann method: from the Boltzmann equation to the lattice Boltzmann equation, *Phys. Rev. E* 56 (6) (1997) 6811–6817, ISSN 10953787, 1063651x.
- [27] P.L. Bhatnagar, E.P. Gross, M. Krook, A model for collision processes in gases. I: small amplitude processes in charged and neutral one-component systems, *Phys. Rev.* 94 (3) (1954), ISSN 0031899x, 15366065.
- [28] X. He, L. Luo, Lattice Boltzmann model for the incompressible Navier–Stokes equation, *J. Stat. Phys.* 88 (3–4) (1997) 927–944, ISSN 15729613, 00224715.
- [29] S. Chapman, T. Cowling, *The Mathematical Theory of Non-Uniform Gases*, Cambridge University Press, 1970.
- [30] M. Junk, A. Klar, L.S. Luo, Asymptotic analysis of the lattice Boltzmann equation, *J. Comput. Phys.* 210 (2) (2005) 676–704, <http://dx.doi.org/10.1016/j.jcp.2005.05.003>, ISSN 10902716, 00219991, 10957103, 00219797.
- [31] S. Succi, *The Lattice Boltzmann Equation for Fluid Dynamics and Beyond*, Oxford University Press, 2001.
- [32] Q. Zou, X. He, On pressure and velocity boundary conditions for the lattice Boltzmann BGK model, *Phys. Fluids* 9 (6) (1997) 1591–1596, ISSN 10897666, 10706631.

- [33] M. Junk, Z. Yang, Outflow boundary conditions for the lattice Boltzmann method, *Prog. Comput. Fluid Dyn.* 8 (1–4) (2008) 38–48, <http://dx.doi.org/10.1504/pcfd.2008.018077>, ISSN 17415233, 14684349.
- [34] O. Dardis, J. McCloskey, Lattice Boltzmann scheme with real numbered solid density for the simulation of flow in porous media, *Phys. Rev. E* 54 (4) (1998) 4834–4837, ISSN 10953787, 1063651x.
- [35] O. Dardis, J. McCloskey, Permeability porosity relationships from numerical simulations of fluid flow, *Geophys. Res. Lett.* 25 (9) (1998) 1471–1474, ISSN 19448007, 00948276.
- [36] G. Liu, M. Geier, Z. Liu, M. Krafczyk, T. Chen, Discrete adjoint sensitivity analysis for fluid flow topology optimization based on the generalized lattice Boltzmann method, *Comput. Math. Appl.* 68 (10) (2014) 1374–1392, ISSN 08981221, 18737668.
- [37] A. Griewank, A. Walther, Algorithm 799: revolve: an implementation of checkpointing for the reverse or adjoint mode of computational differentiation, *ACM Trans. Math. Softw.* 26 (1) (2000) 19–45, <http://dx.doi.org/10.1145/347837.347846>, ISSN 15577295, 00983500.
- [38] Q. Wang, P. Moin, G. Iaccarino, Minimal repetition dynamic checkpointing algorithm for unsteady adjoint calculation, *SIAM J. Sci. Comput.* 31 (4) (2009) 2549–2567, <http://dx.doi.org/10.1137/080727890>, ISSN 10957197, 10648275.
- [39] J. Guest, J. Prevost, T. Belytschko, Achieving minimum length scale in topology optimization using nodal design variables and projection functions, *Int. J. Numer. Methods Eng.* 61 (2) (2004) 238–254, <http://dx.doi.org/10.1002/nme.1064>, ISSN 10970207, 00295981.
- [40] O. Sigmund, Morphology-based black and white filters for topology optimization, *Struct. Multidiscip. Optim.* 33 (4–5) (2007) 401–424, <http://dx.doi.org/10.1007/s00158-006-0087-x>, ISSN 16151488, 1615147x.
- [41] S. Xu, Y. Cai, G. Cheng, Volume preserving nonlinear density filter based on heaviside functions, *Struct. Multidiscip. Optim.* 41 (4) (2010) 495–505, <http://dx.doi.org/10.1007/s00158-009-0452-7>, ISSN 16151488, 1615147x.
- [42] F. Wang, B.S. Lazarov, O. Sigmund, On projection methods, convergence and robust formulations in topology optimization, *Struct. Multidiscip. Optim.* 43 (6) (2011) 767–784, <http://dx.doi.org/10.1007/s00158-010-0602-y>, ISSN 16151488, 1615147x.
- [43] M. Zhou, B.S. Lazarov, F. Wang, O. Sigmund, Minimum length scale in topology optimization by geometric constraints, *Comput. Methods Appl. Mech. Eng.* 293 (2015) 266–282, <http://dx.doi.org/10.1016/j.cma.2015.05.003>, ISSN 18792138, 00457825.
- [44] K. Svanberg, The method of moving asymptotes – a new method for structural optimization, *Int. J. Numer. Methods Eng.* 24 (2) (1987) 359–373, ISSN 10970207, 00295981.

Publication [P2]

**Applications of automatic
differentiation in topology
optimization**

Applications of automatic differentiation in topology optimization

Sebastian A. Nørgaard¹  · Max Sagebaum² · Nicolas R. Gauger² · Boyan S. Lazarov¹

Received: 21 February 2017 / Revised: 3 April 2017 / Accepted: 26 April 2017
© Springer-Verlag Berlin Heidelberg 2017

Abstract The goal of this article is to demonstrate the applicability and to discuss the advantages and disadvantages of automatic differentiation in topology optimization. The technique makes it possible to wholly or partially automate the evaluation of derivatives for optimization problems and is demonstrated on two separate, previously published types of problems in topology optimization. Two separate software packages for automatic differentiation, CoDi-Pack and Tapenade are considered, and their performance and usability trade-offs are discussed and compared to a hand coded adjoint gradient evaluation process. Finally, the resulting optimization framework is verified by applying it to a non-trivial unsteady flow topology optimization problem.

Keywords Topology optimization · Automatic differentiation · Lattice Boltzmann

1 Introduction

Automatic differentiation, also at times called algorithmic differentiation, is a technique that, according to Griewank and Walther (2008) “has been rediscovered and implemented many times, yet its application still has not reached its full potential”. Automatic differentiation (AD) allows

for the exact evaluation of the Jacobian of an arbitrarily complicated differentiable function, by partitioning the function into a sequence of simple operations, which are by themselves trivially differentiable. This process can be automated by software, allowing developers to focus on the solution of the problems requiring differentiation, rather than the derivation and implementation of code for evaluating derivatives. This potential for easily evaluated derivatives makes AD very useful for design optimization, especially for highly non-linear problems (Albring et al. 2016; Nemili et al. 2014; Zhou et al. ; Özkaya et al. 2016). Despite this, to the authors knowledge, there have been only few applications of AD for density based topology optimization—the only example the authors are aware of is the paper by Łaniewski Wołstrok and Rokicki (2016). Thus, the aim of the presentation is to discuss the application details and to demonstrate AD for two topology optimization problems in computational mechanics.

An extensive review of topology optimization itself is beyond the scope of this paper, but the interested reader is referred to the monograph by Bendsøe and Sigmund (2004), as well as the more recent review paper by Sigmund and Maute (2013).

1.1 Automatic differentiation

The goal of this section is to give a brief introduction to AD. For an extensive and more general treatment, the reader is referred to the introductory text by Griewank and Walther (2008). To simplify the discussion, assume a continuous function $F : \mathbb{R}^n \rightarrow \mathbb{R}^m$, with the Jacobian matrix $F' : \mathbb{R}^n \rightarrow \mathbb{R}^{m \times n}$. Further assume that a routine (i.e. a particular computer implementation) exists to evaluate F . In the AD literature, F is often called the *primal function*. Even though F may be arbitrarily complicated, its

✉ Sebastian A. Nørgaard
sebnorg@mek.dtu.dk

¹ Department of Mechanical Engineering, Technical University of Denmark, 2800 Lyngby, Denmark

² TU Kaiserslautern, Chair for Scientific Computing, 67663 Kaiserslautern, Germany

concrete implementation may be decomposed into a series of simple operations (e.g. additions, multiplication, elementary functions such as the trigonometric functions) which are individually easy to differentiate exactly. The differentiated value of each operation can then be propagated to the next by the chain rule. This idea of propagation can be applied in two ways, either starting from the input vector $x \in \mathbb{R}^n$, which results in the *forward mode*, or from the output vector $y \in \mathbb{R}^m$, which results in the *reverse mode*. Since the full mathematical details of AD are beyond the scope of the paper, each mode will be demonstrated by means of a very simple example.

For the forward mode, consider the function

$$\begin{aligned} f(x) : \mathbb{R} &\rightarrow \mathbb{R}^2, \\ y_1 &= \cos(\cos(x)), \\ y_2 &= \exp(y_1). \end{aligned} \tag{1}$$

An implementation of (1) might evaluate the function like so:

$$\begin{aligned} v_1 &= \cos(x), \\ v_2 &= y_1 = \cos(v_1), \\ v_3 &= y_2 = \exp(v_2), \end{aligned} \tag{2}$$

where the variables v_i can be considered intermediate values or “computational steps” taken to evaluate the function. Using these steps, the derivative of f with respect to x can be obtained as:

$$\begin{aligned} \dot{v}_1 &= -\sin(x), \\ \dot{v}_2 &= \dot{y}_1 = -\sin(v_1)\dot{v}_1, \\ \dot{v}_3 &= \dot{y}_2 = \exp(v_2)\dot{v}_2, \end{aligned} \tag{3}$$

where the dot denotes differentiation with respect to x . In general, the forward mode allows the evaluation of the expression:

$$\dot{y} = F'(x)\dot{x}, \tag{4}$$

where $\dot{x} \in \mathbb{R}^{n \times 1}$ is called the *seed direction*.

For the reverse mode, consider the function:

$$\begin{aligned} g(x_1, x_2) : \mathbb{R}^2 &\rightarrow \mathbb{R}, \\ y &= \cos(\cos(x_1 x_2^2)), \end{aligned} \tag{5}$$

a possible evaluation procedure for this function is:

$$\begin{aligned} v_1 &= x_1, \\ v_2 &= x_2, \\ v_3 &= v_2^2, \\ v_4 &= \cos(v_1 v_3), \\ v_5 &= y = \cos(v_4). \end{aligned} \tag{6}$$

The *adjoint variables*, $\bar{v}_i = \partial y / \partial v_i$, may now be evaluated by stepping through the evaluation (6) in reverse order:

$$\begin{aligned} \bar{v}_5 &= 1, \\ \bar{v}_4 &= -\bar{v}_5 \sin(v_4), \\ \bar{v}_3 &= -\bar{v}_4 \sin(v_1 v_3) v_1, \\ \bar{v}_2 &= \partial g / \partial x_2 = 2\bar{v}_3 v_2, \\ \bar{v}_1 &= \partial g / \partial x_1 = -\bar{v}_4 \sin(v_1 v_3) v_3. \end{aligned} \tag{7}$$

In general, the reverse mode evaluates the expression:

$$\bar{x}^T = \bar{y}^T F'(x), \tag{8}$$

where $\bar{y} \in \mathbb{R}^{m \times 1}$ is termed the *weight functional*.

Note that the two examples given above are intentionally simplistic, as they serve only to demonstrate the principle of AD at the most basic level. Evaluating more sophisticated functions, one has to consider issues such as branching, potential instabilities caused by differentiation close to singularities or discontinuities, and the influence of round-off errors on the final result. Dealing with these things is an active area of research which is beyond the scope of this paper. The authors simply note that none of the examples shown in the following sections exhibit pathological behavior, and that the gradients obtained with AD in all cases have been verified by a finite difference check.

The expressions (4) and (8) above are general, but by choosing a standard basis seed direction or weight functional (e.g. $\dot{x} = [1 \ 0 \ 0 \ \dots]^T$), (4) and (8) allow the evaluation of a column or row of the Jacobian matrix, respectively. While both modes have similar mathematical properties, evaluating the reverse mode requires more memory since intermediate values and operations must be stored in order to step through them in reverse order. AD packages supSigmund2013porting the reverse mode generally provide a storage object—often called a *tape*—which is responsible for storing the information necessary to reverse the function F .

For topology optimization, the function of interest is typically the objective function, $F_{\text{objective}} : \mathbb{R}^{N_d} \rightarrow \mathbb{R}$, where N_d is the number of design variables. This makes the reverse mode the obvious choice, since the sensitivities

$$F'_{\text{objective}} = \left[\frac{dF'_{\text{objective}}}{ds_1} \quad \frac{dF'_{\text{objective}}}{ds_2} \quad \dots \right],$$

can be computed with a single evaluation of the reverse mode (8), in the same manner that hand derived adjoints allow. It should be stressed that for topology optimization, N_d is typically much larger than in other structural optimization problems, since in size and shape optimization the geometry of the design is represented by a much smaller number of parameters. In addition, note that the reverse mode is not fundamentally different from a hand derived discrete adjoint approach; its purpose is to reduce the burden of implementing the adjoint.

Generally, there are two approaches to implementing an AD package: source transformation and object overloading. Source transformation, as the name implies, provides a program which takes as input the source code to be differentiated, and outputs new source code which evaluates the derivative of the original source. An example of this type of implementation is Tapenade (Hascoët and Pascual 2013; Tapenade website 2016), which provides a convenient online server on which users can upload their source code, which will then get differentiated and served back. The advantage of this approach is that the differentiated code can be inspected directly, and if necessary, the user can manually optimize it to improve the execution speed of the application. Of course, if one chooses to do this, some convenience is sacrificed since the differentiation procedure is no longer fully automatic. Additionally, should the source code for the primal function change, the source transformation procedure—possibly including hand optimization—must be repeated.

The second approach, operator overloading, takes advantage of a feature of certain programming languages (notably C++ and Fortran 90) which allows the user to define basic operations such as addition and multiplication on user-defined types. This is exploited in AD libraries to provide types which perform both primal and differentiated computations. The function to be differentiated is then overloaded to accept these library types as input—rather than intrinsic floating point types such as `double` in C++. The resulting values can then be queried for their gradient as well as primal values. While this approach is typically slower than source transformation, it was shown by Hogan (2014) that in C++, expression templates could be used to achieve execution speeds which are competitive with source transformed code. The great advantage of this approach is the convenience. The code for evaluating the primal function can be reused without further implementation effort to evaluate the gradients. Furthermore, any modifications made to the primal code will be immediately reflected in the differentiated output, without requiring further involvement from the user. An example of this type of implementation is CoDiPack (CoDiPack website 2016). The CoDiPack library is header only, meaning that the code can simply be included in the application code to be differentiated, without any pre-compilation step. For further information, the interested reader is referred to the CoDiPack website cited above as well as Albring et al. (2015a, b). The two packages presented above will be used to solve the optimization problems presented in this paper. For a much more complete list of AD packages, the interested reader is referred to the online list available at the AutoDiff website (2016).

The remainder of this paper is organized as follows: first AD is demonstrated for a relatively simple 1D wave propagation problem in Section 2. The example allows for

easy comparison between hand written adjoint differentiation and fully automatic differentiation. The readers familiar with traditional adjoint analysis applied to transient topology optimization problems will identify immediately the similarities between the tape (the storage object in AD) and the storage of the forward solution in transient optimization problems. In the following Section 3 the applicability of AD is demonstrated for more complex optimization of transient fluid mechanics problems (Nørgaard et al. 2016), where the explicit form of the Jacobian of the state equations including the boundary conditions is practically impossible to be derived by hand. The advantages and the disadvantages of AD are discussed and demonstrated in details, and finally the article is completed with a topology optimized example of a fluid device for oscillatory fluid input.

2 Application to transient wave propagation problems

The goal in this section is to demonstrate AD for well known one dimensional wave propagation problem (Dahl et al. 2008; Lazarov et al. 2011). The aim of the example is to compare and discuss the applicability of AD for complex topology optimization problems where significant amount of time is spend on derivation and implementation of sensitivities. The optimization problem is given as

$$\begin{aligned} \min_s : \quad & J(s, \mathbf{u}) = \int_0^T z(s, \mathbf{u}) dt, \\ \text{s.t.} : \quad & \mathbf{r}(t, s, \mathbf{u}, \dot{\mathbf{u}}, \ddot{\mathbf{u}}) = 0, \quad t \in [0, T], \\ & g_i(s, \mathbf{u}, \dot{\mathbf{u}}, \ddot{\mathbf{u}}) \leq 0, \quad i \in \{1, \dots, N_g\}, \\ & s \in \mathcal{D}_{\text{ad}}, \end{aligned} \tag{9}$$

where, $\mathbf{r}(t, s, \mathbf{u}, \dot{\mathbf{u}}, \ddot{\mathbf{u}}) = 0$ is the discrete form of the considered linear elastic state problem written in residual form, \mathbf{u} is a vector with nodal displacements, $\dot{\mathbf{u}}$ is a vector with nodal velocities, $\ddot{\mathbf{u}}$ is a vector with nodal accelerations, s is the design vector with relative element densities, $J(s, \mathbf{u})$ is the objective function and $g_i(\cdot), i \in \{1, \dots, N_g\}$ is a set of additional constraints. The residual form is given as

$$\mathbf{r}(t, s, \mathbf{u}, \dot{\mathbf{u}}, \ddot{\mathbf{u}}) = \mathbf{f}(t) - [\mathbf{M}(s)\ddot{\mathbf{u}} + \mathbf{C}(s)\dot{\mathbf{u}} + \mathbf{K}(s)\mathbf{u}], \tag{10}$$

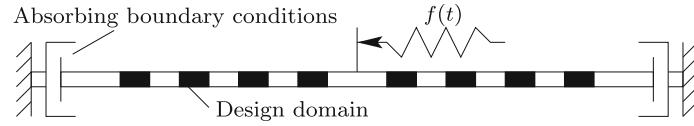
where the mass, damping and stiffness matrices $\mathbf{M}(s)$, $\mathbf{C}(s)$, and $\mathbf{K}(s)$ are obtained by standard finite element assembly procedures. For every element the local matrices are obtained using linear interpolation between the matrices for two different materials, i.e.,

$$\mathbf{M}(s)_e = (1 - s_e)\mathbf{M}_0 + s_e\mathbf{M}_1, \tag{11}$$

$$\mathbf{C}(s)_e = (1 - s_e)\mathbf{C}_0 + s_e\mathbf{C}_1, \tag{12}$$

$$\mathbf{K}(s)_e = (1 - s_e)\mathbf{K}_0 + s_e\mathbf{K}_1. \tag{13}$$

Fig. 1 Optimization setup. Absorbing boundary conditions are applied at both ends of the wave guide



An external excitation is applied as a time dependent nodal force in the middle of the computational domain

$$f(t) = \begin{cases} \cos(2\pi f_c(t - t_0))e^{-\delta(\frac{t}{t_0}-1)^2}, & t \geq 0, \\ 0, & t < 0, \end{cases} \quad (14)$$

where t_0 is the center of the wave packet in the time domain, f_c is the central frequency, and δ defines the bandwidth (Dahl et al. 2008). The excitation generates two Gaussian wave packets propagating towards the two ends of the wave guide. The set up is shown in Fig. 1. The selected objective is to minimize an integral of the squared displacements in a region of the design domain for a selected time interval. The optimization results in periodic band-gap structures as demonstrated in Dahl et al. (2008) and shown in Fig. 1, with a period depending on the wavelength of the waves propagating through the wave guide. As these results are well known and investigated in details in the literature, study and discussion of the optimized design will be omitted here and the focus will be shifted on the sensitivity analysis.

The gradients of the objective in (9) can be obtained using adjoint analysis as shown in Dahl et al. (2008), and are given as

$$\int_0^T \frac{\partial z(s, \mathbf{u})}{\partial s_e} dt = \int_0^T \lambda^T \left[\frac{\partial \mathbf{M}(s)}{\partial s_e} \ddot{\mathbf{u}} + \frac{\partial \mathbf{C}(s)}{\partial s_e} \dot{\mathbf{u}} + \frac{\partial \mathbf{K}(s)}{\partial s_e} \mathbf{u} \right] dt, \quad (15)$$

where the Lagrange multipliers vector $\lambda(t) = \bar{\lambda}(T - \tau)$ is obtained as the solution of the following equation

$$\mathbf{M}\ddot{\bar{\lambda}} + \mathbf{C}\dot{\bar{\lambda}} + \mathbf{K}\bar{\lambda} = \frac{\partial z(\tau, \mathbf{u})}{\partial \mathbf{u}}, \quad \tau \in [0, T], \quad (16)$$

with initial conditions $\bar{\lambda} = 0$ and $\dot{\bar{\lambda}} = 0$ and $\tau = T - t$.

2.1 Time integration

As no analytic solution to (10) exists in the general case, the vectors of displacements, velocities and accelerations are obtained numerically at discrete time steps. Here the time derivatives are computed based on finite difference scheme and at the n^{th} time step they are given as

$$\dot{\mathbf{u}}_n = \frac{\mathbf{u}_{n+1} - \mathbf{u}_{n-1}}{2\Delta t}, \quad (17)$$

$$\ddot{\mathbf{u}}_n = \frac{\mathbf{u}_{n+1} - 2\mathbf{u}_n + \mathbf{u}_{n-1}}{\Delta t^2}. \quad (18)$$

Inserting (11) and (14) in (10) and rearranging the terms results in

$$\begin{aligned} & \left(\frac{1}{\Delta t^2} \mathbf{M} + \frac{1}{2\Delta t} \mathbf{C} \right) \mathbf{u}_{n+1} \\ & = \mathbf{f}_n - \left(\frac{2}{\Delta t^2} \mathbf{M} + \mathbf{K} \right) \mathbf{u}_n - \left(\frac{1}{\Delta t^2} \mathbf{M} - \frac{1}{2\Delta t} \mathbf{C} \right) \mathbf{u}_{n-1}. \end{aligned} \quad (19)$$

The above equation provides the solution at time t_{n+1} using the system response at time steps n and $n - 1$. The integration starts with $\mathbf{u}_0 = 0$ and $\mathbf{u}_{-1} = 0$. The time step is chosen based on the Courant-Friedrichs-Lewy (CFL) condition

$$\Delta t \leq \Delta t_c = \frac{\Delta x}{c}, \quad (20)$$

where Δx is the distance between the finite element nodes and c is the wave speed. The same scheme is applied for solving the adjoint (16). The second derivative for the sensitivity analysis at $t = 0$ is computed as $\ddot{\mathbf{u}} = \mathbf{M}^{-1} \mathbf{f}(0)$.

The Lagrange multipliers sequence can be obtained from (16) by stepping backward in time. First, the forward solution is computed for each discrete point $\mathbf{u}_0, \mathbf{u}_1, \dots, \mathbf{u}_{N_s-1}, \mathbf{u}_{N_s}$, and as second step the adjoint equation is computed with right hand sides depending on the forward solution. As final step the sensitivities are evaluated based on (15). For more details the interested readers are referred to Dahl et al. (2008); Elesin et al (2012, 2014); Lazarov et al. (2011).

The great advantage of the adjoint approach, compared to for example finite difference derivatives, is that all sensitivities can be evaluated by solving the adjoint problem once. As noted above, however, the same is true of the reverse mode of automatic differentiation, since the objective function is of type $J : \mathbb{R}^{N_d} \rightarrow \mathbb{R}$. For the numerical implementation, the discrete form of the objective J is

$$J = \int_0^T z(s, \mathbf{u}) \approx \sum_i z(s, \mathbf{u}(t_i)) \Delta t. \quad (21)$$

The discretization (19) is sufficiently simple that it can be evaluated as a simple stencil type computation, as is illustrated in Algorithm 1.

Algorithm 1 Evaluating the wave propagation objective

```

 $u_{-1} = \mathbf{0}, u_0 = \mathbf{0}.$ 
for all  $t \in \{0, \Delta t, 2\Delta t, \dots, T\}$  do
  for all  $e \in \{0, \dots, N_e - 1\}$  do
    Compute local value  $\mathbf{u}(x_e, t_i)$  by (19).
  end for
   $J \leftarrow J + z(s, \mathbf{u})\Delta t.$ 
end for

```

The key feature of Algorithm 1 is that it is relatively simple to implement without the need of any external linear algebra libraries. This makes it very simple to automatically differentiate the evaluation of the objective function. Using the operator overloading approach, the problem can be simply differentiated in a black box manner without the need of deriving and implementing the adjoint method. As an aside, note that even if a linear algebra library was required, C++ libraries such as Eigen (Guennebaud et al. 2010) support linear algebra on arbitrary numeric types, and thus would allow automatic differentiation as well.

In order to allow the application of operator overloading AD, our implementation of Algorithm 1 was converted into a C++ template, thus allowing the implementation to use the numeric types provided by CoDiPack. After this, the code must perform some calls to the tape type provided by CoDiPack, before and after the call to the function evaluating Algorithm 1. These additional calls add very little code and are described in the CoDiPack tutorial (CoDiPack website 2016). The differentiation procedure using CoDiPack yields identical sensitivities to those yielded by evaluating the adjoint expression (15). Comparative performance measures are shown in Table 1. The code was compiled with GCC 5.4.0 with `-O2`, and the performance was measured on an Intel Core i7-3720QM processor.

As expected the memory and the computational time grow proportional to the number of the time steps for the AD and the hand coded example. Comparing the performance of AD with CoDiPack to the hand coded adjoint, CoDiPack is roughly 1.5 to 1.8 times slower. Considering that the development time needed to obtain the derivative code is essentially zero, this seems like a modest price to pay, though this would of course depend on the specifics of the problem and the performance requirements. The memory requirements of the AD solution, however, is more than an order of magnitude greater than the hand coded equivalent. This is because the tape structure implemented by CoDiPack must store, in addition to all solution states $\mathbf{u}_0, \mathbf{u}_1, \dots, \mathbf{u}_{N_s}$, all operations needed by the computation in order to reverse them. Whether this memory requirement is an intractable issue depends on the nature and size of the problem one wishes to solve. For a research problem such as this, the memory requirement of AD is available on many

modern personal computers, and the advantage of being able to differentiate a code without spending any significant time deriving, implementing, and debugging an adjoint solver can hardly be emphasized enough. Even in cases where AD does not scale to the desired problem size, it would still be a useful tool for prototyping the optimization problem and verifying the derivation and implementation of an adjoint code. As a final point, note that the large memory footprint of CoDiPack is due to the fact that we are solving a transient problem. This means that the tape structure redundantly stores the time stepping operations once for each time step taken. Thus, steady state type problems would require significantly less memory. In addition, CoDiPack is actively developed and future optimizations might address this issue.

In the above discussion we considered a problem which readily lent itself to black box automatic differentiation. In the next section, we will consider a more complicated example, in which automatic differentiation is applied to an already existing parallel code.

Table 1 Performance measurements comparing AD to the hand coded adjoint for the wave propagation problem

| | Automatic differentiation | Hand coded adjoint |
|-----------------|---------------------------|--------------------|
| 2500 timesteps | | |
| Memory | 1.0 Gb | – |
| Wall time | 6 s | 4.2 s |
| Relative time | 1.43 | 1 |
| 5000 timesteps | | |
| Memory | 2.1 Gb | 0.07 Gb |
| Wall time | 13 s | 8.5 s |
| Relative time | 1.52 | 1 |
| 10000 timesteps | | |
| Memory | 4.2 Gb | 0.15 Gb |
| Wall time | 26 s | 16 s |
| Relative time | 1.625 | 1 |
| 20000 timesteps | | |
| Memory | 8.5 Gb | 0.27 Gb |
| Wall time | 50.53 s | 32.46 s |
| Relative time | 1.55 | 1 |
| 30000 timesteps | | |
| Memory | 12.8 Gb | 0.43 Gb |
| Wall time | 88 s | 48 s |
| Relative time | 1.83 | 1 |

The number of elements is $N_e = 900$. The memory utilized for AD can be significantly reduced to the level of the hand coded adjoint by introducing checkpointing as discussed in Section 3.2

3 Application to lattice Boltzmann

In this section, we will focus on a more involved example of automatic differentiation applied to the lattice Boltzmann method (LBM). While there has been work to apply AD to an already existing parallel LBM code (Krause and Heuveline 2013), the LBM implementation presented here uses a few components from the open source topology optimization code presented by Aage et al. (2015), and thus uses the PETSc library (Balay et al. 2016a, b) for execution in parallel. In this case, neither source transformation (i.e. with Tapenade) nor operator overloading (i.e. with CoDiPack) are naively applicable, since they do not interface directly with PETSc. A different strategy than simple black box differentiation is required (Sagebaum et al. 2013). For lattice Boltzmann, it is possible to derive an adjoint method in which the local operations can be differentiated with AD. In this way, the AD code is only invoked within the main loop, which decouples the code from external library calls.

3.1 The lattice Boltzmann equation

The lattice Boltzmann method is a method for computing fluid flows based on kinetic theory, rather than continuum dynamics. A thorough introduction is beyond the scope of this paper, but the interested reader is referred to e.g. the book by Succi (2001). LBM is an explicit time-stepping method, based on the equation

$$f_\alpha(\mathbf{x}_i + \mathbf{e}_\alpha \Delta x, t + \Delta t) = \Omega[f(\mathbf{x}_i, t)], \quad (22)$$

$$\alpha \in \{0, \dots, N_v - 1\},$$

where $\mathbf{f} \in \mathbb{R}^{N_v}$ is a set of distribution values associated with a discrete set of particle velocities \mathbf{e}_α . The right hand side models particle collisions and is known as the *collision operator*. There are numerous different collision operators available in the literature (Bhatnagar et al. 1954; D’Humières 1994; Geier et al. 2006; Latt and Chopard 2006), and a large class of lattice Boltzmann models differ only in the collision operator, while the left-hand side—known as the *streaming step*—remains unchanged. The collision operator is in general highly non-linear in \mathbf{f} ; indeed, this is part of the reason automatic differentiation is attractive for the lattice Boltzmann method. For the purpose of density based topology optimization, (22) is modified as follows:

$$f_\alpha(\mathbf{x}_i + \mathbf{e}_\alpha \Delta x, t + \Delta t) = \tilde{\Omega}[f(\mathbf{x}_i, t), s(\mathbf{x}_i)], \quad (23)$$

$$\alpha \in \{0, \dots, N_v - 1\},$$

where $s(\mathbf{x}_i) \equiv s_i$ determines whether the grid point \mathbf{x}_i is a fluid or solid node. This modification of the collision step is to enforce an immersed no-slip boundary in the solid part of the domain. Again, numerous models to achieve this are

available in the literature (Ladd and Verberg 2001; Spaid and Phelan 1997; Zhu and Ma 2013), and the modification is typically orthogonal to the choice of “base” operator, leaving a high number of possibly combinations that are all valid collision operators.

The macroscopic variables of the flow governed by (23) can be computed by

$$\rho(\mathbf{x}_i, t) = \sum_\alpha f_\alpha(\mathbf{x}_i, t), \quad (24)$$

$$\rho(\mathbf{x}_i, t) \mathbf{u}(\mathbf{x}_i, t) = \sum_\alpha \mathbf{e}_\alpha f_\alpha(\mathbf{x}_i, t), \quad (25)$$

$$p(\mathbf{x}_i, t) = c_s^2 \rho(\mathbf{x}_i, t), \quad (26)$$

where ρ , p , \mathbf{u} are the macroscopic density, pressure, and velocity, respectively. The lattice Boltzmann method is a weakly compressible method, and the macroscopic pressure is proportional to the macroscopic density, with a proportionality constant equal to c_s^2 , the speed of sound squared. The numerical value of this constant depends on the choice of velocity discretization.

One attractive feature of the lattice Boltzmann method is that the algorithm has high spatial locality: the collision step requires only local information while the streaming step requires only nearest neighbor information. This makes it ideally suited for execution in parallel.

Time stepping in the LBM can be executed in either a *collide and stream* fashion, in which the collision step is executed followed by the streaming step, or conversely in a *stream and collide* fashion. For the purpose of topology optimization, we choose the stream and collide approach. The reason for this is that the objective is a function of the macroscopic values, which are evaluated during the collision step. Hence, by performing stream and collide from timestep n to $n + 1$, the macroscopic variables are also in the correct state at step $n + 1$. A function of the macroscopic variables can then be conveniently evaluated following the stream and collide procedure.

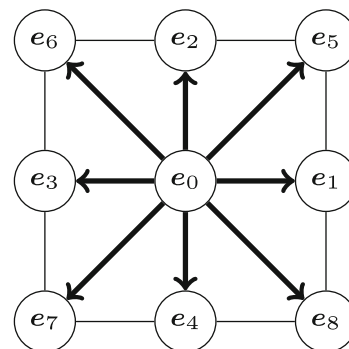


Fig. 2 The D2Q9 model

In residual form the LB scheme can be written:

$$R_{\alpha}^{\text{stream}}(\mathbf{x}_i, t) = f_{\alpha}^{\text{collision}}(\mathbf{x}_i + \mathbf{e}_{\alpha}, t + \Delta t) - f_{\alpha}^{\text{stream}}(\mathbf{x}_i, t) = 0, \quad (27a)$$

$$R^{\text{bc}}(\mathbf{x}_i, t) = \psi[f^{\text{stream}}(\mathbf{x}_i, t)] - f(\mathbf{x}_i, t) = 0, \quad (27b)$$

$$R^{\text{collision}}(\mathbf{x}_i, s_i, t) = \tilde{\Omega}[f(\mathbf{x}_i, t), s_i] - f^{\text{collision}}(\mathbf{x}_i, t) = 0. \quad (27c)$$

Above, f denotes the initial state of distributions at timestep t , $f^{\text{collision}}$ denotes the post-collision state, and f^{stream} denotes the post-streaming state. On interior nodes $f = f^{\text{stream}}$; on boundary nodes, however, there are unknown distribution values, which are computed in the boundary value step (27a). Here ψ simply denotes a generic boundary condition operator. For the LBM, there are many different operators available for different kinds of boundaries (Inamuro et al. 1995; Junk and Yang 2008; Latt et al. 2008; Zou and He 1997).

3.2 Automatic differentiation of lattice Boltzmann

As mentioned above, because the LB code relies on an external library, it is not feasible to differentiate the code in a black box manner. Instead, the discrete adjoint method is applied to obtain an adjoint lattice Boltzmann method in which the local collision step can be evaluated with automatic differentiation. A similar derivation was given by Łaniewski Wołstrok and Rokicki (2016).

Following Kreissl et al. (2011), we consider an objective function for unsteady flow of the following form

$$J = \sum_{t=0}^{N_t} z(t, \mathbf{f}_t, \mathbf{s}), \quad (28)$$

where N_t is the number of time steps, $\mathbf{f}_t = [f(x_0, t), f(x_1, t), \dots]$ is the vector of state variables (i.e. the LBM distributions) at timestep t , and $\mathbf{s} =$

$[s_0, s_1, \dots]$ is the vector of design variables. To derive the adjoint LBM, Lagrange multipliers are added to (28):

$$\hat{J} = \sum_{t=0}^{N_t} z(t, \mathbf{f}_t, \mathbf{s}) + \lambda_t^T R_t^{\text{stream}} + \sigma_t^T R_t^{\text{bc}} + \tau_t^T R_t^{\text{collision}}. \quad (29)$$

Taking the derivative with respect to the design variable s_i yields:

$$\frac{d\hat{J}}{ds_i} = \frac{\partial \hat{J}}{\partial s_i} + \sum_{t=0}^{N_t} \frac{\partial \hat{J}}{\partial \mathbf{f}_t} \frac{\partial \mathbf{f}_t}{\partial s_i} + \frac{\partial \hat{J}}{\partial f_t^{\text{collision}}} \frac{\partial f_t^{\text{collision}}}{\partial s_i} + \frac{\partial \hat{J}}{\partial f_t^{\text{stream}}} \frac{\partial f_t^{\text{stream}}}{\partial s_i}. \quad (30)$$

For an optimal design, we must have $d\hat{J}/ds_i = 0, \forall i$. Since each term in (30) is mutually independent, this implies that each summand must be zero.

From the residuals (27), we then have:

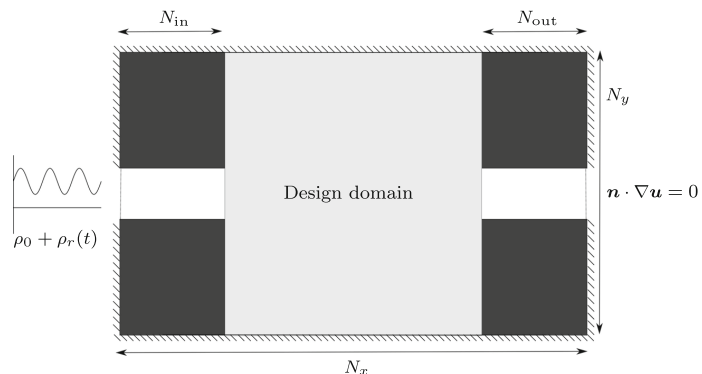
$$\sum_{t=0}^{N_t} \frac{\partial \hat{J}}{\partial \mathbf{f}_t} \frac{\partial \mathbf{f}_t}{\partial s_i} = \sum_{t=0}^{N_t} \left(\tau_t^T \frac{\partial \tilde{\Omega}}{\partial \mathbf{f}_t} - \sigma_t^T \mathbf{I} + \frac{\partial z}{\partial \mathbf{f}_t} \right) \frac{\partial \mathbf{f}_t}{\partial s_i} = 0, \quad (31)$$

where \mathbf{I} is the identity matrix. Since the collision Ω is purely local, this implies

$$\sigma(\mathbf{x}_i, t)^T = \tau(\mathbf{x}_i, t)^T \frac{\partial \tilde{\Omega}[f(\mathbf{x}_i, t), s_i]}{\partial f(\mathbf{x}_i, t)} + \frac{\partial z(t, \mathbf{f}_t, \mathbf{s})}{\partial f(\mathbf{x}_i, t)}. \quad (32)$$

This is the adjoint collision step. Notice that the first summand on the right-hand side of (32) is of the form (8), meaning that it can be evaluated exactly with one computation of the AD reverse mode.

Fig. 3 Computational domain for the pressure diode problem



Continuing, we further have:

$$\sum_{t=0}^{N_t} \frac{\partial \hat{J}}{\partial \mathbf{f}_t^{\text{stream}}} \frac{\partial \mathbf{f}_t^{\text{stream}}}{\partial s_i} = \sum_{t=0}^{N_t} \left(\boldsymbol{\sigma}_t^T \frac{\partial \psi}{\partial \mathbf{f}_t^{\text{stream}}} - \boldsymbol{\lambda}_t^T \right) \frac{\partial \mathbf{f}_t^{\text{stream}}}{\partial s_i}, \quad (33)$$

$$\boldsymbol{\lambda}(\mathbf{x}_i, t)^T = \boldsymbol{\sigma}(\mathbf{x}_i, t)^T \frac{\partial \psi[\mathbf{f}^{\text{stream}}(\mathbf{x}_i, t)]}{\partial \mathbf{f}^{\text{stream}}(\mathbf{x}_i, t)}, \quad (34)$$

which is the adjoint boundary step, assuming the boundary function ψ is purely local. This could again be evaluated by AD, which would be advantageous for complicated boundary conditions such as the regularized boundary conditions (Latt et al. 2008). For simpler boundary conditions such as those presented by Zou and He (1997), or the frequently applied “bounce back” no-slip condition, it is quite simple to derive this step by hand.

Differentiation of the final step leads to the adjoint streaming step, which was shown by Liu et al. (2014) to be given by

$$\boldsymbol{\tau}_\alpha(\mathbf{x}_i, t) = \boldsymbol{\lambda}_\alpha(\mathbf{x}_i - \mathbf{e}_\alpha, t - \Delta t), \quad (35)$$

that is, the adjoint streaming is backwards in time and in the opposite direction of the primal streaming. Finally, the sensitivities can be evaluated by

$$\frac{\partial \hat{J}}{\partial s_i} = \sum_{t=0}^{N_t} \frac{\partial z}{\partial s_i} + \boldsymbol{\tau}_i^T \frac{\partial \tilde{\Omega}}{\partial s_i} \quad (36)$$

$$= \sum_{t=0}^{N_t} \frac{\partial z}{\partial s_i} + \boldsymbol{\tau}(\mathbf{x}_i, t)^T \frac{\partial \tilde{\Omega}[\mathbf{f}(\mathbf{x}_i, t), s_i]}{\partial s_i}, \quad (37)$$

with the final equality again being due to the local nature of the collision operator Ω . This completes the adjoint lattice Boltzmann method, its implementation is summarized by pseudo code in Algorithm 2.

The adjoint lattice Boltzmann algorithm step backwards through time to evaluate the Lagrange multipliers and thus the sensitivities. Note that at each timestep, the primal vector \mathbf{f}_i must be known in order to evaluate the adjoint lattice Boltzmann step. As a consequence, the full time history of the primal solver must be available. Naively, this means that the full history must be stored in memory. While such a strategy is feasible for small problems, it does not scale well. As an alternative, parts of the history can be recomputed during the adjoint evaluation. With this strategy, only selected time steps are stored in memory. These time steps are typically referred to as *checkpoints*. The rest of the time steps are then recomputed starting from the nearest checkpoint as they are needed. The papers by Griewank and Walther (2000) and Wang et al. (2009) both describe provably optimal algorithms for checkpoint placement. With

Table 2 Numerical parameters for the example problem

| | | | |
|-----------------|------|--------------------|-------|
| N_x | 350 | N_y | 125 |
| N_{in} | 75 | N_t | 20000 |
| ρ_0 | 1 | $\Delta\rho$ | 0.01 |
| ω | 1000 | V_{fluid} | 0.6 |
| γ | 1 | Filter radius | 6 |

these algorithms, the cost of re-computation grows only logarithmically with the memory saved. For example, allocating 20 checkpoints for an objective requiring 200 time steps to evaluate reduces the memory requirement by an order of magnitude compared to the naive approach, but only increases the computational cost of the adjoint evaluation by a factor of log 10.

Algorithm 2 Adjoint lattice Boltzmann with AD

```

for all  $t \in \{N_t, \dots, 0\}$  do
  Obtain  $\mathbf{f}_t$  by reading from memory or performing necessary re-computation.
  for all  $\mathbf{x}_i, i \in \{0, \dots, N_x - 1\}$  do
    Compute adjoint collision step by (32).
    Add contribution to sensitivity  $dJ/ds_i$  by (36).
    if  $\mathbf{x}_i$  is a boundary node then
      Compute adjoint boundary conditions by (34).
    end if
    for all  $\mathbf{x}_i, i \in \{0, \dots, N_x - 1\}$  do
      Perform adjoint streaming by (35).
    end for
  end for
end for

```

Note that Algorithm 2 is executed in collide and stream order. This is a consequence of our choice of the stream and collide order for the primal solver. Had we chosen collide and stream for the primal solver, the adjoint algorithm would have to be executed in stream and collide order.

It should be emphasized that Algorithm 2 can be used to differentiate a large class of LB models, as long as the model follows the basic structure of a local collision step and a shifting streaming step. More complicated models which follow this basic structure include thermal lattice Boltzmann (Bartoloni et al. 1993; Guo et al. 2002; Mezrhab et al. 2010), as well as lattice Boltzmann for multi-component flow (Asinari 2006; Parker 2008).

3.3 An example problem

The main challenge in implementing the adjoint LBM introduced above is the evaluation of the adjoint collision step (32). Of course, the collision operator could

Fig. 4 Example results at different Reynolds numbers



be differentiated by hand, but as noted above, the equation can be evaluated by applying the reverse mode of automatic differentiation. In this section, we will test our implementation against an example problem, followed by an evaluation of the performance of different AD implementations.

For the sample problem, the collision operator Ω applied is the multiple relaxation time (MRT) operator (D’Humières 1994), operating on the common D2Q9 lattice (nine discrete velocities in two dimensions). For this lattice, the velocities are given by

$$\begin{aligned} & [e_0, e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8] \\ &= \frac{\Delta x}{\Delta t} \begin{bmatrix} 0 & 1 & 0 & -1 & 0 & 1 & -1 & -1 & 1 \\ 0 & 0 & 1 & 0 & -1 & 1 & 1 & -1 & -1 \end{bmatrix}, \end{aligned} \quad (38)$$

the set of velocities defined by the D2Q9 lattice is illustrated in Fig. 2.

In order to enforce the no-slip condition on the solid part of the domain, we use the partial bounce back collision operator introduced by Zhu and Ma (2013). In this model, the base collision operator Ω is modified to

$$\tilde{\Omega}[f(x_i, t), s_i]_\alpha = \Omega[f(x_i, t)] + \frac{1}{2}g(s_i) \times (\Omega[f(x_i, t)]_{-\alpha} - \Omega[f(x_i, t)]_\alpha), \quad (39)$$

where the index $-\alpha$ indicates the discrete velocity opposite to the index α , i.e. $e_{-1} = e_3$; the function $g(s_i)$ is continuous and satisfies $g(0) = 1$ and $g(1) = 0$, so that $s_i = 0$ corresponds to a solid node, while $s_i = 1$ corresponds to a fluid node. Here, we use the following convex function introduced by Borrvall and Petersson (2003):

$$g(s_i) = 1 - s_i \frac{1 + \gamma}{s_i + \gamma}, \quad (40)$$

where γ is an adjustable parameter which allows penalization of intermediate values of s_i . Increasing γ increases the penalization of intermediate values.

The example problem considered is an unsteady flow problem with an objective function of the form (28). The computational domain for the problem is shown in Fig. 3. The problem is inspired by the work on fluid diodes by Lin et al. (2015).

The computational domain consists of two narrow channels, the left side with prescribed density (and therefore pressure, since $\rho \propto p$ in LBM), and the right side with a

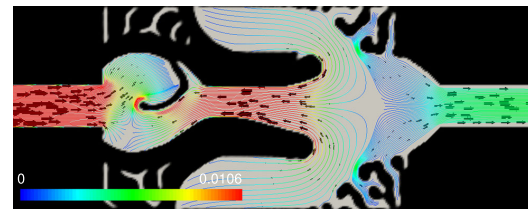
Neumann boundary on the velocity. The enforced density on the left is oscillating, with oscillations given by

$$\rho_{\text{oscillating}}(t) = \rho_0 + \rho_r(t) = \rho_0 + \Delta\rho \sin\left(\frac{2\pi t}{\omega}\right), \quad (41)$$

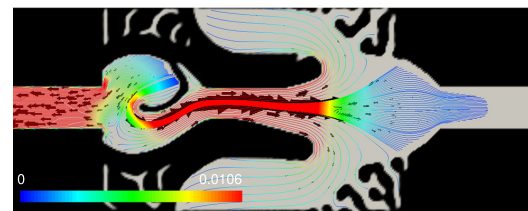
here, $\Delta\rho$ is the amplitude of the oscillation, and ω is the period. We now seek to maximize the average outflow at the right end, subject to a volume constraint on the amount of fluid in the design domain. That is, the optimization problem is formulated as:

$$\begin{aligned} \min_{\vec{s}} J &= -\frac{1}{N_t} \sum_{t=0}^{N_t} \bar{u}_x, \\ \text{s.t. } &\begin{cases} \frac{1}{N_s} \sum_i s_i - V_{\text{fluid}} \leq 0, \\ \vec{f}_i \text{ satisfies (27),} \end{cases} \end{aligned} \quad (42)$$

where V_{fluid} is the allowed fraction of fluid in the design domain, and \bar{u}_x is the spatially averaged x -component of the velocity at the right outlet. In order to regularize the design, and obtain a fully black and white solution, the projection filter (Guest et al. 2004) is applied. To compute the Reynolds number, the characteristic length is defined as $L = N_{\text{in}}$, and the characteristic velocity is taken to be



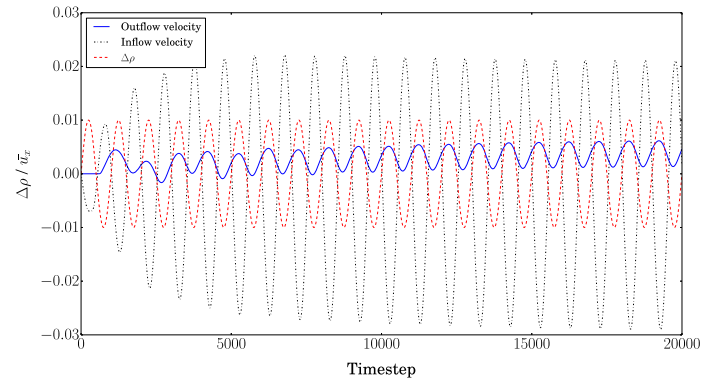
(a) $\rho_r > 0$.



(b) $\rho_r < 0$.

Fig. 5 Sample streamlines during the inflow and outflow phase for the result at $Re = 250$

Fig. 6 Average outflow velocity of the optimized design at $Re = 250$ as a function of time. Also shown are the average inflow velocity, and the density variation $\Delta\rho$



$u_{\text{characteristic}} = 0.01$. The choice of characteristic velocity is somewhat arbitrary, since no set velocity is directly imposed anywhere in the domain, but agrees well with the observed order of magnitude of velocities in the final designs. The remaining numerical parameters used are listed in Table 2.

Two example designs at different Reynolds numbers are shown in Fig. 4. Both have the same basic structure, but higher Reynolds number results in slightly more intricate side channels in the final design. In order to better understand the working principle of the designs, sample streamlines are shown in Fig. 5, both for the case of the oscillatory term in (41) being negative ($\rho_r > 0$), and positive ($\rho_r < 0$). From the figure, it is observed that even though the oscillating pressure on the left side results in fluid periodically flowing both in and out at the boundary, the right boundary only ever acts as an outflow. It appears that the side “arms” of the design act as a deposit for fluid during the outflow phase of the left boundary; this deposited fluid then flows towards the desired outlet when the pressure oscillations reverse. In Fig. 6, the average outflow is plotted as a function of time. It is observed that the cyclic behaviour observed in Fig. 5 does indeed repeat throughout the whole time history.

3.4 Performance of AD implementation

To close this section, the performance of different AD implementations will be reported. The performance is measured according to the following methodology: since reverse

AD is applied only in the adjoint collision step (32), we will only measure the computational time of this step. The adjoint collision step is implemented in a simple C++ for loop, no attempts have been made at optimization for memory accesses. The performance metric will be the average collisions per second (CPS) in a single iteration of the example problem presented above. Since the adjoint collision step is purely local, we will consider only the single core performance and thus ignore any parallel message passing overhead. The performance is measured on an Intel Xeon X5660 processor.

In addition to the MRT collision operator used above, we will consider the commonly used Bhatnagar-Gross-Krook (BGK) collision operator Bhatnagar et al. (1954), as well as the more recent cascaded collision operator by Geier et al. (2006). Both operator overloading and source transformation implementations of the adjoint collision (32) will be considered. For operator overloading, CoDiPack will be used. For source transformation, the online tool Tapenade will be used. For Tapenade, two versions will be considered: the “raw” source transformation output, and a version of the source transformed output which has been hand optimized. All kernels have been compiled with GCC 4.8.5 with $-O3$. The results of the performance measurements are listed in Table 3.

As is apparent from Table 3, unsurprisingly, the best performance also comes from the implementation which requires the most effort. While the CoDiPack implementation cannot compete with Tapenade in terms of speed, it should once again be reiterated that using Tapenade involves

Table 3 Results of performance measurements for adjoint LBM with AD. Higher CPS (collisions per second) is better

| Problem size: 350×125 , 20000 timesteps | BGK | MRT | Cascaded |
|--|-------------------------|-------------------------|-------------------------|
| CoDiPack | 1.12×10^6 CPS | 0.631×10^6 CPS | 0.481×10^6 CPS |
| Tapenade | 4.18×10^6 CPS | 4.32×10^6 CPS | 1.17×10^6 CPS |
| Tapenade (optimized) | 12.27×10^6 CPS | 7.56×10^6 CPS | 4.23×10^6 CPS |

a trade-off between implementation time and running time. Even if a good optimized collision routine is implemented with the help of Tapenade, any changes in the source code for the primal collision step will not be reflected in the adjoint code. Conversely, with CoDiPack, any optimizations made to the primal collision source code will immediately result in better adjoint performance with no additional implementation effort.

4 Discussion and conclusion

In this paper, we have demonstrated the application of automatic differentiation to two different classes of problems for topology optimization. While the AD promise of completely black box differentiation of numerical codes is certainly tantalizing, achieving this does require that the code has been written with the application of AD in mind. For codes where this is not the case, some additional implementation work will be necessary. At best, it is simply a matter of parametrizing core routines to accept generic numeric types (e.g. turning core routines into templates). For more complicated codes, which might have external dependencies which are unrealistic or even impossible to modify, a significantly greater implementation effort could be required. Whether this time investment is worth it will of course be project dependent.

While the above considerations does limit the applicability of AD to some extent, many research codes are developed from scratch in order to solve a single well-defined problem. In these cases, getting the derivatives of a function for “free” can greatly decrease the time required to solve a particular problem; even in cases where black box differentiation is not possible, AD might be still be applicable with a bit more up front work. This was demonstrated in the lattice Boltzmann example above. Here, some work was required to derive and implement the AD supported adjoint method, but once this was done, it became possible to differentiate any lattice Boltzmann type method with little additional work.

The final point to consider is the issue of performance. In both problems presented, there is a trade-off between performance and development time; in both cases it is possible to improve performance by implementing a hand tuned adjoint code (either by derivation or by optimization of the output from Tapenade). However, even if these performance improvements were strictly necessary in order to solve the problem within a realistic time, the less performant version would still be useful for prototyping and validation. During development of the optimized Tapenade routines for lattice Boltzmann, the CoDiPack adjoint collision implementation was used as a reference known to give the correct answer. This greatly eased development, since any mistakes

introduced during the tuning of the code were immediately caught.

As with all things in software development, automatic differentiation is a technique which comes with advantages and disadvantages. In the view of the authors, it is a powerful tool that can be used to great advantage in many types of problems in topology optimization, and should be considered as a useful supplement to hand derived adjoints.

Acknowledgements The first author would like to acknowledge the generous help and fruitful discussion offered by Emre Özkaya and Tim Albring at TU Kaiserslautern. The first and the last authors acknowledge the financial support received from the TopTen project sponsored by the Danish Council for Independent Research (DFR-4005-00320).

References

- Aage N, Andreassen E, Lazarov BS (2015) Topology optimization using PETSc: an easy-to-use, fully parallel, open source topology optimization framework. *Struct Multidiscip Optim* 51(3):565–572. doi:[10.1007/s00158-014-1157-0](https://doi.org/10.1007/s00158-014-1157-0)
- Albring T, Sagebaum M, Gauger N (2015a) Development of a consistent discrete adjoint solver in an evolving aerodynamic design framework. *AIAA* 2015-3240
- Albring T, Zhou B, Gauger N, Sagebaum M (2015b) An aerodynamic design framework based on algorithmic differentiation. *ERCOTAC Bulletin* 102:10–16
- Albring T, Sagebaum M, Gauger NR (2016) Efficient aerodynamic design using the discrete adjoint method in su2. In: 17th AIAA/ISSMO multidisciplinary analysis and optimization conference
- Asinari P (2006) Semi-implicit-linearized multiple-relaxation-time formulation of lattice Boltzmann schemes for mixture modeling. *Phys Rev E* 73(5):056705. doi:[10.1103/PhysRevE.73.056705](https://doi.org/10.1103/PhysRevE.73.056705)
- AutoDiff website (2016) [Autodiff.org](http://www.autodiff.org): Community portal for automatic differentiation. <http://www.autodiff.org>, accessed: 2016-10-18
- Balay S, Abhyankar S, Adams MF, Brown J, Brune P, Buschelman K, Dalcin L, Eijkhout V, Gropp WD, Kaushik D, Knepley MG, McInnes LC, Rupp K, Smith BF, Zampini S, Zhang H, Zhang H (2016a) PETSC users manual. Tech. Rep. ANL-95/11 - revision 3.7, argonne national laboratory, <http://www.mcs.anl.gov/petsc>
- Balay S, Abhyankar S, Adams MF, Brown J, Brune P, Buschelman K, Dalcin L, Eijkhout V, Gropp WD, Kaushik D, Knepley MG, McInnes LC, Rupp K, Smith BF, Zampini S, Zhang H, Zhang H (2016b) PETSC Web page. <http://www.mcs.anl.gov/petsc>
- Bartoloni A, Battista C, Cabasino S, Paolucci P, Pech J, Sarno S, Todesco G, Torelli M, Tross W, Vicini P, Benzi R, Cabibbo N, Massaioli F, Tripiccone R (1993) LBE Simulations of Rayleigh-Benard convection on the APE100 parallel processor. *Int J ModPhys C Phys Comput* 4(5):993–1006. doi:[10.1142/S012918319300077X](https://doi.org/10.1142/S012918319300077X)
- Bendsøe MP, Sigmund O (2004) *Topology optimization: theory, methods and applications*. Springer
- Bhatnagar PL, Gross EP, Krook M (1954) A model for collision processes in gases. I: Small amplitude processes in charged and neutral one-component systems. *Phys Rev* 94(3)
- Borrvall T, Petersson J (2003) Topology optimization of fluids in stokes flow. *Int J Numer Methods Fluids* 41(1):77–107. doi:[10.1002/flid.426](https://doi.org/10.1002/flid.426)

- CoDiPack website (2016) Codipack—code differentiation package. <http://www.scicomp.uni-kl.de/software/codi/>, accessed: 2016-10-18
- Dahl J, Jensen JS, Sigmund O (2008) Topology optimization for transient wave propagation problems in one dimension. *Struct Multidiscip Optim* 36:585–595
- D’Humières D (1994) Generalized lattice Boltzmann equations. *Prog Astronaut Aeronaut* 159:450–458
- Elesin Y, Lazarov B, Jensen J, Sigmund O (2012) Design of robust and efficient photonic switches using topology optimization. *Photonics Nanostruct Fundam Appl* 10(1):153–165. doi:10.1016/j.photonics.2011.10.003
- Elesin Y, Lazarov B, Jensen J, Sigmund O (2014) Time domain topology optimization of 3d nanophotonic devices. *Photonics Nanostruct Fundam Appl* 12(1):23–33. doi:10.1016/j.photonics.2013.07.008
- Geier M, Greiner A, Korvink JG (2006) Cascaded digital lattice Boltzmann automata for high Reynolds number flow. *Phys Rev E* 73(6):066,705. doi:10.1103/PhysRevE.73.066705
- Griewank A, Walther A (2000) Algorithm 799: revolve: an implementation of checkpointing for the reverse or adjoint mode of computational differentiation. *Acem Trans Math Softw* 26(1):19–45. doi:10.1145/347837.347846
- Griewank A, Walther A (2008) Automatic differentiation of algorithms. SIAM
- Guennebaud G, Jacob B et al. (2010) Eigen v3. <http://eigen.tuxfamily.org>
- Guest JK, Prévost JH, Belytschko T (2004) Achieving minimum length scale in topology optimization using nodal design variables and projection functions. *Int J Numer Methods Eng* 61(2):238–254. doi:10.1002/nme.1064
- Guo Z, Shi B, Zheng C (2002) A coupled lattice BGK model for the Boussinesq equations. *Int J Numer Methods Fluids* 39(4):325–342. doi:10.1002/flid.337
- Hascoët L, Pascual V (2013) The tapenade automatic differentiation tool: principles, model, and specification. *ACM Transactions Mathematical Software* 39(3), doi:10.1145/2450153.2450158
- Hogan RJ (2014) Fast reverse-mode automatic differentiation using expression templates in C++. *ACM Trans Math Softw (toms)* 40(4):1–16. doi:10.1145/2560359
- Inamuro T, Yoshino M, Ogino F (1995) A non-slip boundary-condition for lattice boltzmann simulations. *Phys Fluids* 7(12):2928–2930. doi:10.1063/1.868766
- Junk M, Yang Z (2008) Outflow boundary conditions for the lattice Boltzmann method. *Progress in Computational Fluid Dynamics* 8(1–4):38–48. doi:10.1504/PCFD.2008.018077
- Krause MJ, Heuveline V (2013) Parallel fluid flow control and optimisation with lattice Boltzmann methods and automatic differentiation. *Computers and Fluids* 80:28–36. doi:10.1016/j.compfluid.2012.07.026
- Kreissl S, Pingen G, Maute K (2011) Topology optimization for unsteady flow. *Int J Numer Methods Eng* 87(13):1229–1253. doi:10.1002/nme.3151
- Ladd A, Verberg R (2001) Lattice-Boltzmann simulations of particle-fluid suspensions. *J Stat Phys* 104(5–6):1191–1251. doi:10.1023/A:1010414013942
- Latt J, Chopard B (2006) Lattice Boltzmann method with regularized pre-collision distribution functions. *Math Comput Simul* 72(2–6):165–168. doi:10.1016/j.matcom.2006.05.017
- Latt J, Chopard B, Malaspina O, Deville M, Michler A (2008) Straight velocity boundaries in the lattice Boltzmann method. *Phys Rev E* 77(5):056,703. doi:10.1103/PhysRevE.77.056703
- Lazarov B, Matzen R, Elesin Y (2011) Topology optimization of pulse shaping filters using the hilbert transform envelope extraction. *Struct Multidiscip Optim* 44:409–419. doi:10.1007/s00158-011-0642-y
- Lin S, Zhao L, Guest JK, Weihs TP, Liu Z (2015) Topology optimization of fixed-geometry fluid diodes. *J Mech Des* 137(8):081,402. doi:10.1115/1.4030297
- Liu G, Geier M, Liu Z, Krafczyk M, Chen T (2014) Discrete adjoint sensitivity analysis for fluid flow topology optimization based on the generalized lattice Boltzmann method. *Computers and Mathematics With Applications* 68(10):1374–1392. doi:10.1016/j.camwa.2014.09.002
- Mezrhab A, Moussaoui MA, Jami M, Naji H, Bouzidi M (2010) Double MRT thermal lattice Boltzmann method for simulating convective flows. *Phys Lett A* 374(34):3499–3507. doi:10.1016/j.physleta.2010.06.059
- Nemili A, Özkaya E, Gauger NR, Kramer F, Höll T, Thiele F (2014) Optimal design of active flow control for a complex high-lift configuration. In: Proceedings of 7th AIAA flow control conference, 2014-2515
- Nørgaard S, Sigmund O, Lazarov B (2016) Topology optimization of unsteady flow problems using the lattice Boltzmann method. *J Comput Phys* 307:291–307. doi:10.1016/j.jcp.2015.12.023
- Parker J (2008) A novel lattice Boltzmann method for treatment of multicomponent convection, diffusion and reaction phenomena in multiphase systems. PhD thesis, Oregon State University
- Sagebaum M, Gauger NR, Naumann U, Lotz J, Leppkes K (2013) Algorithmic differentiation of a complex C++ code with underlying libraries. *Procedia Computer Science* 18:208–217. doi:10.1016/j.procs.2013.05.184
- Sigmund O, Maute K (2013) Topology optimization approaches. *Struct Multidiscip Optim* 48(6):1031–1055. doi:10.1007/s00158-013-0978-6
- Spaid M, Phelan F (1997) Lattice boltzmann methods for modeling microscale flow in fibrous porous media. *Phys Fluids* 9(9):2468–2474. doi:10.1063/1.869392
- Succi S (2001) The lattice Boltzmann equation for fluid dynamics and beyond. Oxford University Press
- Tapenade website (2016) Tapenade on-line automatic differentiation engine. <http://www-tapenade.inria.fr:8080/tapenade/index.jsp>, accessed: 2016-10-18
- Wang Q, Moin P, Iaccarino G (2009) Minimal repetition dynamic checkpointing algorithm for unsteady adjoint calculation. *SIAM J Sci Comput* 31(4):2549–2567. doi:10.1137/080727890
- Łaniewski Wołstrok L, Rokicki J (2016) Adjoint lattice Boltzmann for topology optimization on multi-gpu architecture. *Computers and Mathematics With Applications* 71(3):833–848. doi:10.1016/j.camwa.2015.12.043
- Zhou BY, Albring T, Gauger NR, Illario da Silva CR, Economon TD, Alonso JJ A discrete adjoint approach for jet-flap interaction noise reduction. In: Proceedings of 58th AIAA/ASCE/AHS/ASC structures, structural dynamics, and materials conference. AIAA SciTech Forum, AIAA, 2017-0130
- Zhu J, Ma J (2013) An improved gray lattice Boltzmann model for simulating fluid flow in multi-scale porous media. *Adv Water Resour* 56:61–76. doi:10.1016/j.advwatres.2013.03.001
- Özkaya E, Hay JA, Gauger NR, Schönwald N, Thiele F (2016) A two-level approach for design optimization of acoustic liners. In: Proceedings of 9th international conference on computational fluid dynamics, ICCFD9-2016-184
- Zou Q, He X (1997) On pressure and velocity boundary conditions for the lattice Boltzmann BGK model. *Phys Fluids* 9(6):1591–1598. doi:10.1063/1.869307

DTU Mechanical Engineering
Section of Solid Mechanics
Technical University of Denmark

Nils Koppels Allé, Bld. 404
DK-2800 Kgs. Lyngby
Denmark
Phone (+45) 4525 4250
Fax (+45) 4593 1475
www.mek.dtu.dk
ISBN: 978-87-7475-506-7

DCAMM
Danish Center for Applied Mathematics and Mechanics

Nils Koppels Allé, Bld. 404
DK-2800 Kgs. Lyngby
Denmark
Phone (+45) 4525 4250
Fax (+45) 4593 1475
www.dcam.dk
ISSN: 0903-1685