Warsaw University of Technology

Master of Sciences Thesis

Optimal Control using the Adjoint Lattice Boltzmann Method

Author:
Grzegorz Gruszczyński

Supervisor:
dr hab. inż. Jacek Szumbarski

A thesis submitted in fulfilment of the requirements
for the degree of Master of Sciences
in the Mechanical Engineering

Faculty of Power and Aeronautical Engineering

June 16, 2016

Declaration of Authorship

I, Grzegorz Gruszczyński, declare that this thesis titled, 'Optimal Control using the Adjoint Lattice Boltzmann Method' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: _____

Date: _____

WARSAW UNIVERSITY OF TECHNOLOGY

Abstract

Faculty of Power and Aeronautical Engineering

Master of Sciences

Optimal Control using the
Adjoint Lattice Boltzmann Method

by Grzegorz GRUSZCZYŃSKI

The purpose of this thesis is to adopt the Lattice Boltzmann Method (LBM), which is gaining popularity in modelling of the fluid motions, to a flexible framework for solving an optimal control problems of discrete dynamical systems in an open-loop fashion. Adjoint formulation has been introduced in order to reduce computational cost required to find the gradient of the objective function having many control variables.
As a study case, the problem of optimal mixing of fluids having nonheterogeneous scalar fields is solved, since it often plays a crucial role in industrial process. It is usually desired to enforce a uniform distribution of the temperature or a chemical reactant in the domain using minimal amount of work during a prescribed time. This leads to the optimal control problem in the area of CFD.

The growing interest in the LBM is caused by its almost ideal scalability on parallel supercomputers, inherited time dependence, easy implementation and the ability of copying with complex geometries. In chapter 2, we have presented the way from Boltzmann transport equation, which emerges from kinetic theory of gases, to its discrete form being known as the Lattice Boltzmann equation. Next, using the Chapman-Enskog expansion, the recovery of the Navier-Stokes equations from the LBM has been highlighted. In the same chapter, the computational algorithm and commonly used boundary conditions are described. Basing on the dimensional analysis, we demonstrate a procedure of converting the physical units to the lattice units in which the LB simulations are performed. Next, to prove the validity of the LBM solver a set of separate benchmark cases including shedding of von Karman vortices, advection, diffusion, wall shear force and conservation of the passive scalar has been covered. The second part of the thesis is focused on the adjoint (dual) method. It starts from the general adjoint formulation for finding the gradient of the objective function subjected to nonlinear equality constrains. From that, it follow why the cost of solving the dual problem does not depend on the number of control variables. Later on, an adjoint for a discrete dynamical system and its application to the LBM is presented. Since the adjoint can be formulated for both continuous equations or their discretized form a brief discussion of these approaches is provided. In the present work, the discrete adjoint is utilized, because it can be easily automated using Tapenade which is an Automatic Differentiation tool for Source Code Transformation. The numerical scheme for the adjoint problem preserves the properties of the original system, thus it can be incorporated into the standard LBM solver keeping already mentioned benefits of this method. To have a baseline for optimization, several cases showing relation between work performed to obtain a mixed fluid have been calculated. Finally, the results of optimization are presented.

To sum up the above, an optimization technique applicable to wide range of CFD challenges is proposed. Presented framework is flexible enough to be adopted for other, even more complex, flow design problems like topology optimization.

Acknowledgements

I would like to thank my supervisor Jacek Szumbarski for all his time, Łukasz Łaniewski-Wołłk for consultations and strong support with programming and configuring the TCLB software, Wojciech Regulski and Michał Dzikowski for discussions concerning the LBM. Special thanks go out to my fiancée and my family for all their support.

Contents

List of Figures

List of Tables

Abbreviations

CFD	**C**omputational **F**luid **D**ynamics
HPC	**H**igh **P**erformance **C**omputing
NS	**N**avier **S**tokes
LBM	**L**attice **B**oltzmann **M**ethod
CPU	**C**entral **P**rocessing **U**nit
GPU	**G**raphical **P**rocessing **U**nits
AD	**A**utomatic **D**ifferentiation
BC	**B**oundary **C**onditions
IC	**I**nitials **C**onditions

List of the most important symbols

e_i	lattice velocity vector associated with the i-th distribution function	-
c_S	lattice speed of sound	-
t	lattice time	lt
Δt	lattice time step	-
Δx	lattice position step - next node	-
$\boldsymbol{u}$	lattice fluid velocity vector	$\frac{lu}{lt}$
f_i	lattice density distribution function	-
T	lattice passive scalar	-
$\mathbf{q}$	lattice passive scalar flux	-
w_i	weights for the equilibrium distribution function	-
$\boldsymbol{x}$	position of the lattice node	lu
$\mathbb{C}(\boldsymbol{f})$	LBM collision operator	-
$\mathbb{S}(\boldsymbol{f})$	LBM streaming operator	-

Greek letters

μ	fluid dynamic viscosity	$\frac{kg}{m \cdot s}$
ν	fluid kinematic viscosity	$\frac{m}{s^2}$
λ	thermal diffusivity	$\frac{m^2}{s}$
ν_{LB}	fluid kinematic viscosity in lattice units	$\frac{lu^2}{lt}$
λ_{LB}	thermal diffusivity in lattice units	$\frac{lu^2}{lt}$
θ_i	lattice passive scalar distribution function	-
ρ	lattice fluid density	-
τ	lattice relaxation time	-
Ω	collision frequency	-

Chapter 1

Introduction

1.1 Motivation

In many industrial processes mixing of fluid having nonheterogeneous scalar fields play a crucial role. For instance, it is often desired to enforce an uniform distribution of the temperature or a chemical reactant in the domain using minimal amount of work during a prescribed time. This leads to the optimal control problem in the area of CFD. The Lattice Boltzmann Method is gaining popularity in modelling of the unsteady fluid motions. It is characterized by an inherited time dependence, easy implementation and almost ideal scalability on a parallel supercomputers.

The adjoint formulation allows to obtain the open-loop solution of the optimal flow control problem posed in a discrete form. Strictly speaking, the discrete adjoint approach is applied in this work, since it permits to take advantage of the Automatic Differentiation (AD) technique which not only simplifies the gradient computation but also provides a flexible framework for various BC or collision operators of arbitrary complexity without additional derivations by hand. Going into more details, the solution of an adjoint system gives the sensitivity of the objective function to each control variable. In context of the optimal control, it means that the sensitivity to instantaneous variations of a control function at all time instants (or steps) can be obtained. It should be stressed out that the numerical cost of solving the dual system does not depend on the amount of control variables. This leads to fundamental time savings comparing to more naive methods like finite differences. Once the gradient at each time step is obtained, a standard optimisation techniques can be applied to modify the control variables and minimize the objective

function. The final (optimized) control function will describe the movement of the actuating device during the whole time of simulation in an explicit form, which means that the solution is in an open-loop form. On the other hand, there are closed-loop systems with a measuring device which provides feedback for the controller, but they are beyond of scope of this thesis.

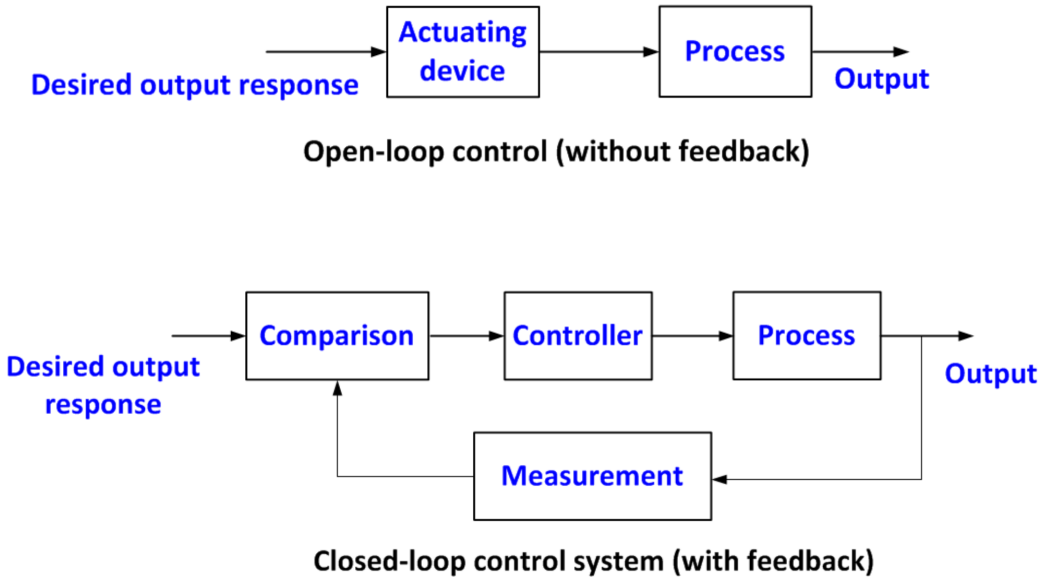


FIGURE 1.1: System control: open vs closed loop

The optimalization using the LBM has been already adressed by several authors also with the adjoint approach. However, there is not much in the context of optimal control. Simple case has been studied by Hekmat and Mirzaei [8, 9] and a more complex 3D problem by Krause, Thater, and Heuveline [11],Krause and Heuveline [10].
Except [10], the adjoint formulation has been usually derived by hand for a specific model.

1.2 Optimal mixing - problem statement

In this section the mathematical formulation of the problem is presented.

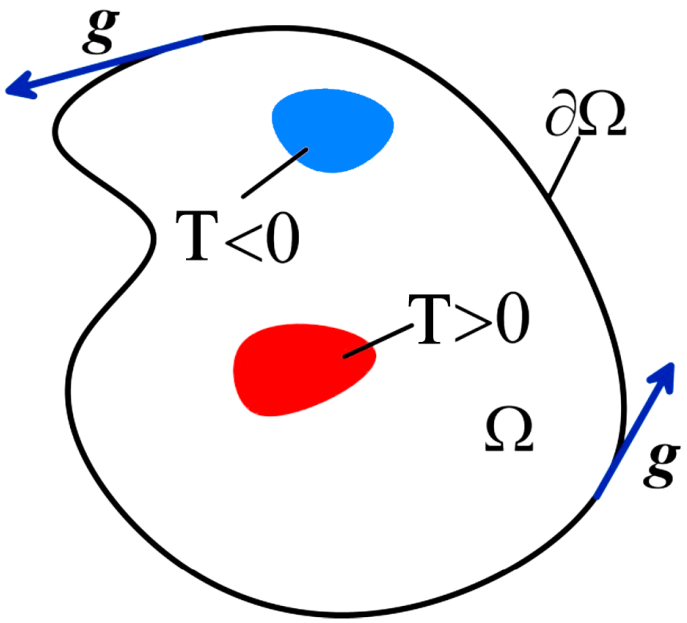


FIGURE 1.2: Flow domain

Consider a container Ω filled with a viscous, incompressible, Newtonian fluid. Points of the boundary of the container $\partial\Omega$ can move in the direction tangent to the boundary with an arbitrary velocity. Thus, the shape of the container remains unchanged. In the initial time instant, the field of a passive scalar is highly inhomogeneous. The boundary is impermeable for this field, i.e., the flux of the passive scalar through the boundary is equal zero. The task is to find such time dependent boundary velocity so that - after prescribed time $\widehat{t}$ - the spatial distribution of the passive scalar is as close to the uniform as possible. At the same time, we want to keep under control the mechanical work needed to perpetuate the fluid within this time interval.

Primal problem

Movement of the fluid is described by the Navier-Stokes equation and the continuity equation. In case of the incompressible fluids they can be written as:

$$\begin{cases} \partial_t \boldsymbol{u} + (\boldsymbol{u} \cdot \nabla)\boldsymbol{u} = -\nabla p + \nu \Delta \boldsymbol{u} \\ \nabla \cdot \boldsymbol{u} = 0 \end{cases} \tag{1.1}$$

Boundary and Initial Conditions for the fluid are:

$$\boldsymbol{u}\Big|_{\partial\Omega} = \boldsymbol{g} \quad ; \quad \boldsymbol{g} \cdot \boldsymbol{n}\Big|_{\partial\Omega} = 0 \quad ; \quad \boldsymbol{u}\Big|_{t=0} = 0$$

The transport of the passive scalar is governed by the advection - diffusion equation:

$$\partial_t T + \boldsymbol{u} \cdot \nabla T = \lambda \nabla^2 T \tag{1.2}$$

Boundary and Initial Conditions for the passive scalar are:

$$\frac{\partial}{\partial \boldsymbol{n}} T\Big|_{\partial\Omega} = 0 \quad ; \quad T\Big|_{t=0} = T_0$$

The goal is to find an extremum of the functional:

$$J[\boldsymbol{u}, T] = \underbrace{\int_\Omega [T(t_{end}) - \overline{T}]^2 d\boldsymbol{x}}_{I_1} + \varepsilon \underbrace{\left[\int_0^{t_{end}} \left(\int_{\partial\Omega} 2\nu \boldsymbol{u} \cdot \boldsymbol{D_u} \boldsymbol{n} dS \right) dt \right]^2}_{I_2} \tag{1.3}$$

where:

$$\begin{aligned} \boldsymbol{D_u} = \frac{1}{2}(\nabla \boldsymbol{u} + \nabla^T \boldsymbol{u}) \quad &- \text{ deformation rate tensor} \\ \overline{T} = \frac{1}{|\Omega|} \int_\Omega T d\boldsymbol{x} \quad &- \text{ average value of the passive scalar} \\ \varepsilon \quad &- \text{ weight coefficient} \end{aligned}$$

Note that due to the BC for T, the total amount of the passive scalar in Ω does not change in time, hence $\overline{T}$ is a fixed value.

Interpretation of the integrals:

I_1 - deviation of T from its equilibrium state $\overline{T}$ (ideally mixed fluid)

I_2 - work done by the boundary (wall) in order to impose fluid motion

Chapter 2

Lattice Boltzmann Method - theory

2.1 Introduction

2.1.1 Phase space

In physics, the phase space is the space of all possible physical states of a system. The physical state means both the position and the velocity/momentum of the object. Obviously, to determine the future of the system it is required to know its current physical state. In a 3D world each particle has 9 degrees of freedom (DOF): 3 coordinates + 3 velocities + 3 angular velocities. Graphical representation of even such a simple system is impossible, however the general concept is of a great importance. An example of a trivial 1D system is presented on a graph 2.1.

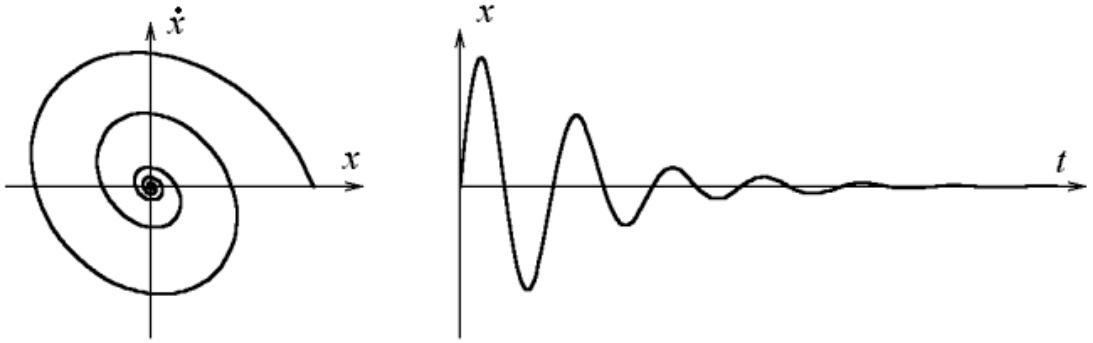


FIGURE 2.1: Phase space (on the left) of a 1DOF, damped oscillator

2.1.2 Distribution functions

The distribution function F describes the probability of finding a particle in the phase space. In a real systems, F depends on time because the particles are moving and changing their locations. For a 3D Cartesian coordinate system, the distribution function can be written in general form as:

$$F = F(t, x, y, z, u_x, u_y, u_z) \tag{2.1}$$

Basic information describing a physical system can be obtained from the distribution functions.

$$\begin{aligned} N &= \int_{R^3}\int_{R^3} F d\boldsymbol{u} d\boldsymbol{x} && \text{- Total number of particles.} \\ N_{expected} &= \int_{\Omega}\int_{R^3} F d\boldsymbol{u} d\Omega && \text{- Expected number of particles in the region } \Omega \\ \boldsymbol{u}_{avg}(\boldsymbol{x}) &= \int_{R^3} F\boldsymbol{u} d\boldsymbol{u} && \text{- Average velocity at a given point} \end{aligned}$$

2.1.3 Evolution of the distribution functions

Following the derivation given in [18], let us consider a system in which particles are moving, but not interacting with each other. The system is subjected to the force field $\mathbf{g}$. The natural, spontaneous motion in the phase space can be viewed as a evolution of the distribution function and is called streaming [22] - $\mathbb{S}$. In an infinitesimally small volume of the phase space $d\mathbf{x}d\mathbf{u}$ it can be written as:

$$F_{no\,collisions}(t + dt, \mathbf{x} + d\mathbf{x}, \mathbf{u} + d\mathbf{u}) d\mathbf{x}d\mathbf{u} = F_{no\,collisions}(t, \mathbf{x}, \mathbf{u}) d\mathbf{x}d\mathbf{u} \tag{2.2}$$

Next, the collisions between particles will cause them to fall from one part of the phase space to the other and vice versa. This flow through the phase space occurs at rate $\mathbb{C}^-(F)d\mathbf{x}d\mathbf{u}$ and $\mathbb{C}^+(F)d\mathbf{x}d\mathbf{u}$ respectively.
Taking that into account, the evolution of the distribution function is:

$$F(t + dt, \mathbf{x} + d\mathbf{x}, \mathbf{u} + d\mathbf{u}) d\mathbf{x}d\mathbf{u} = F(t, \mathbf{x}, \mathbf{u}) d\mathbf{x}d\mathbf{u} + \underbrace{\left[\mathbb{C}^+(F) - \mathbb{C}^-(F)\right]}_{\mathbb{C}(F)} d\mathbf{x}d\mathbf{u}dt \tag{2.3}$$

The collision term $\mathbb{C} = \mathbb{C}(F)$, also known as collision integral, is probably the most important issue in lattice Boltzmann modelling. It will be discussed in details later on. Now, let us expand the distribution function F into the Taylor series:

$$F(t + dt, \mathbf{x} + d\mathbf{x}, \mathbf{u} + d\mathbf{u}) = F(t, \mathbf{x}, \mathbf{u}) + \frac{\partial F}{\partial t}dt + \nabla_{\mathbf{x}} F d\mathbf{x} + \nabla_{\mathbf{u}} F d\mathbf{u} \tag{2.4}$$

Plugging it into 2.3 gives:

$$\left[F(t, \mathbf{x}, \mathbf{u}) + \frac{\partial F}{\partial t}dt + \nabla_{\mathbf{x}} F d\mathbf{x} + \nabla_{\mathbf{u}} F d\mathbf{u}\right] d\mathbf{x}d\mathbf{u} = \left[F(t, \mathbf{x}, \mathbf{u}) + \mathbb{C}(F)dt\right] d\mathbf{x}d\mathbf{u}$$

The classical Newtonian laws of mechanics describes velocity as $\mathbf{u} = \frac{d\mathbf{x}}{dt}$ and acceleration as $\frac{d\mathbf{u}}{dt} = \frac{\mathbf{g}}{m}$ allows us to reformulate the transport equation in a following manner:

$$\frac{\partial F}{\partial t} + (\mathbf{u} \cdot \nabla_{\mathbf{x}})F + (\frac{\mathbf{g}}{m} \cdot \nabla_{\mathbf{u}})F = \mathbb{C}(F) \tag{2.5}$$

The Boltzmann equation can be viewed as a substantial derivative (of an intensive quantity F) which is equal to the collision term $\mathbb{C}$ applied to the distribution finction F.

2.1.4 Collision operator $\mathbb{C}(F)$

The collision operator describes interactions between colliding particles. In general, it has to fulfil two constraints (see [3]):

1. $\mathbb{C}(F)$ has to obey the conservation laws of mass, linear momentum and energy. This means, that the following equalities must hold:

$$\iint \mathbb{C}(F)\psi_k(\mathbf{u})d\mathbf{u}d\mathbf{x} = 0 \qquad \text{for k = 0,1,2,3,4}$$

Functions $\psi_k(\mathbf{u})$ are known as collision invariants. They refer to the total flux of mass ($\psi_0 = 1$), linear momentum ($\psi_1, \psi_2, \psi_3 = \mathbf{u}$) and kinetic energy ($\psi_4 = \mathbf{u}^2$) across the phase space volume $d\mathbf{u}d\mathbf{x}$.

2. $\mathbb{C}(F)$ can not decrease entropy.
In statistics, the probability distribution function which leads to maximum entropy at the equilibrium is known as the Maxwell-Boltzmann distribution:

$$F^{eq}(\boldsymbol{x}, \boldsymbol{u}, t) = N\left(\frac{m}{2\pi k_B T}\right)^{3/2} exp\left[-\frac{m}{2k_B T}(\mathbf{u} - \overline{\mathbf{u}})^2\right] \tag{2.6}$$

$$\overline{\mathbf{u}} = \frac{1}{N}\int F(\mathbf{x}, \mathbf{u}, t)\mathbf{u}d\mathbf{u} \qquad \text{- mean velocity}$$

The quantity H is defined as:

$$H(t) = \iint F(\boldsymbol{x}, \boldsymbol{u}, t)\, ln\Big(F(\boldsymbol{x}, \boldsymbol{u}, t)\Big)d\mathbf{u}d\mathbf{x}$$

It is linked with the entropy S by the following formula:

$$\frac{dS}{dt} = -\frac{dH}{dt}$$

The Boltzmann's H-theorem (1872) states that each function $F(\boldsymbol{x}, \boldsymbol{u}, t)$ which satisfies the Boltzmann equation fulfils the Entropy principle:

$$\frac{dH}{dt} \leq 0$$

2.2 From Boltzmann equation to Lattice Boltzmann equation

The key points of derivation of the Lattice Boltzmann equation from the Boltzmann equation are be presented below (details can be found in [4, p. 160-163]).

2.2.1 Boltzmann equation:

Skipping the force term, the Bolztmann transport eq. 2.5 reduces to:

$$\frac{\partial F}{\partial t} + (\mathbf{u} \cdot \nabla)F = \mathbb{C}(F) \tag{2.7}$$

The left hand side of the 2.7 reminds the NS equation. The key step is to define the operator which will recover the physics of the real fluid. The most widely known approximation was proposed in 1954 by Bhatnagar, Gross and Krook (BGK). It states that each collision changes the distribution function $F(\mathbf{x}, \mathbf{u})$ by an amount proportional to the departure from the equilibrium $F^{eq}(\mathbf{x}, \mathbf{u})$, i.e., such value of F that $\mathbb{C}(F^{eq}) = 0$.

$$\mathbb{C}(F) = -\frac{1}{\tau}(F - F^{eq}) \qquad - \textit{BGK approximation}$$

The τ takes a role of a relaxation parameter and its value depends on the viscosity of fluids. The quantity $\Omega = \frac{1}{\tau}$ is called the collision frequency.

2.2.2 Boltzmann equation - BGK approximation:

Applying the BGK approximation $\left(\mathbb{C}(F) = -\frac{1}{\tau}(F - F^{eq})\right)$ to the Boltzmann equation leads to:

$$\frac{\partial F}{\partial t} + (\mathbf{u} \cdot \nabla)F = -\frac{1}{\tau}(F - F^{eq}) \tag{2.8}$$

2.2.3 Discrete Boltzmann equation:

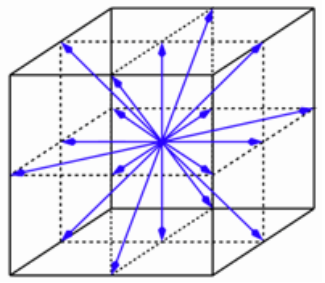


FIGURE 2.2: Discrete directions on a 3D lattice

Next step is to introduce discrete directions, see 2.2. The distribution functions F_i are defined separately for each direction and satisfy equation:

$$\frac{\partial F_i}{\partial t} + (\mathbf{u} \cdot \nabla)F_i = -\frac{1}{\tau}(F_i - F_i^{eq}) \tag{2.9}$$

The lattice tensors theory state that such a degenerate micro-scale form of the velocity field can fully reflect the isotropic behavior of the physical system. On a simple D2Q9 lattice (2 dimensions, 9 discrete density functions) it can be showed in a following manner:

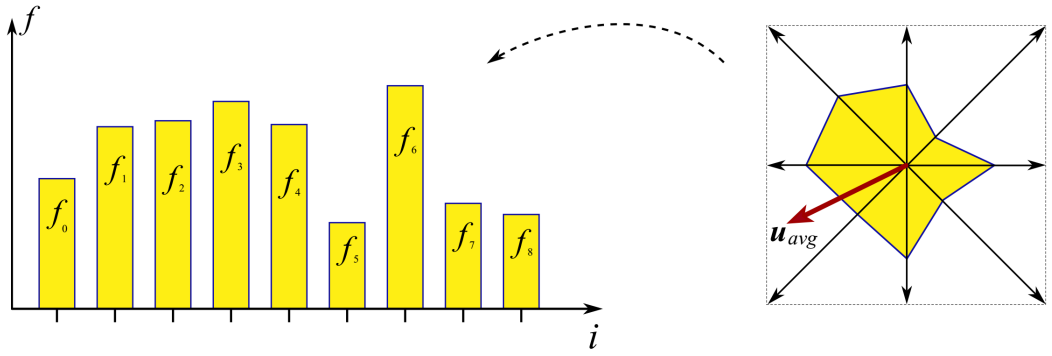


FIGURE 2.3: Conceptual view of the density functions and large-scale average velocity (from Regulski [18])

2.2.4 Non-dimensional discrete Boltzmann equation:

The discrete Boltzmann equation can be non-dimensionalized by introducing characteristic scales:

L - characteristic length

U - characteristic velocity

t_c - time between collisions

n_r - reference density

ϵ - ratio of mean free path to characteristic length (Knudsen number)

Dimensionless quantities:

$$\boldsymbol{e_i} = \frac{\boldsymbol{u_i}}{U}, \quad \widehat{\Delta} = L\Delta, \quad \widehat{t} = \frac{t \cdot U}{L}, \quad \widehat{\tau} = \frac{\tau}{t_c}, \quad f_i = \frac{F_i}{n_r}, \quad \epsilon = t_c \frac{U}{L}$$

Plugging these allows to present the dimensionless Boltzmann equation as:

$$\frac{\partial f_i}{\partial t} + (\mathbf{u} \cdot \widehat{\nabla}) f_i = \frac{1}{\widehat{\tau}\epsilon}(f_i - f_i^{eq}) \tag{2.10}$$

2.2.5 Discretized Boltzmann equation:

Discretization of equation 2.10 leads to:

$$\begin{aligned}\frac{f_i(\boldsymbol{x},t+\Delta\widehat{t})-f_i(\boldsymbol{x},t)}{\Delta\widehat{t}} &+ e_{ix}\frac{f_i(\boldsymbol{x}+\Delta\widehat{\boldsymbol{x}},t+\Delta\widehat{t})-f_i(\boldsymbol{x},t+\widehat{t})}{\Delta\widehat{\boldsymbol{x}}}+\\ &+ e_{iy}\frac{f_i(\boldsymbol{x}+\Delta\widehat{\boldsymbol{y}},t+\Delta\widehat{t})-f_i(\boldsymbol{x},t+\widehat{t})}{\Delta\widehat{\boldsymbol{y}}}+\\ &+ e_{iz}\frac{f_i(\boldsymbol{x}+\Delta\widehat{\boldsymbol{z}},t+\Delta\widehat{t})-f_i(\boldsymbol{x},t+\widehat{t})}{\Delta\widehat{\boldsymbol{z}}}=-\frac{1}{\widehat{\tau}\epsilon}(f_i-f_i^{eq})\end{aligned}$$

Now, let us select such discretization step that the lattice spacing divided by the time step will be equal the lattice velocity $\frac{\Delta\widehat{\mathbf{x}}}{\Delta\widehat{t}}=\mathbf{e}_i$. Luckily, some terms cancel each other and the method becomes explicit:

$$\begin{aligned}&\frac{f_i(\boldsymbol{x},t+\Delta\widehat{t})-f_i(\boldsymbol{x},t)}{\Delta\widehat{t}}+\frac{f_i(\boldsymbol{x}+\boldsymbol{e}_i\Delta\widehat{t},t+\Delta\widehat{t})-f_i(\boldsymbol{x},t+\widehat{t})}{\Delta\widehat{\boldsymbol{t}}}=\\ =\;&\frac{f_i(\boldsymbol{x}+\boldsymbol{e}_i\Delta\widehat{t},t+\Delta\widehat{t})-F_i(\boldsymbol{x},t+\widehat{t})}{\Delta\widehat{\boldsymbol{t}}}=-\frac{1}{\widehat{\tau}\epsilon}(f_i-f_i^{eq})\end{aligned}\tag{2.11}$$

2.2.6 Lattice Boltzmann equation:

Finally, let us choose $\Delta t=t_c$ and multiply the discretized equation 2.11 by $\widehat{t}$:

$$\underbrace{f_i(\boldsymbol{x}+\boldsymbol{e_i}\Delta\boldsymbol{x},t+\Delta t)-f_i(\boldsymbol{x},t)}_{Streaming}=\underbrace{-\frac{1}{\tau\epsilon}(f_i-f_i^{eq})}_{Collision}\tag{2.12}$$

The LB method could be viewed as a particular discretization for the lattice Boltzmann equation.

2.2.7 Derivation of equilibrium distribution f_i:

The distribution functions f_i can be expressed as a sum of f_i^{eq} and perturbations f_i':

$$f_i = f_i^{eq} + f_i' \qquad assuming: \quad f_i' \ll f_i^{eq}$$

It is intuitive, that the lattice velocity moments over f_i shall match the velocity moments over the Maxwell distribution. If we will take a closer look, we will notice that the odd moments cancels out:

$$\sum_i f_i^{eq} \mathbf{e}_i = 0$$
$$\sum_i f_i^{eq} \mathbf{e}_i \mathbf{e}_i \mathbf{e}_i = 0$$

while the even moments remains:

$$\sum_i f_i^{eq} = \int w_B(u) d\mathbf{u} = \rho$$
$$\sum_i f_i^{eq} \mathbf{e}_i \mathbf{e}_i = \int w_B(u) \mathbf{u}\mathbf{u} d\mathbf{u} = \rho \frac{k_B T}{m}$$
$$\sum_i f_i^{eq} \mathbf{e}_i \mathbf{e}_i \mathbf{e}_i \mathbf{e}_i = \int w_B(u) \mathbf{u}\mathbf{u}\mathbf{u}\mathbf{u} d\mathbf{u} = 3\rho \left(\frac{k_B T}{m}\right)^2$$

The Maxwell distribution is given by:

$$w_B(\boldsymbol{x}, \boldsymbol{u}, t) = \rho \left(\frac{m}{2\pi k_B T}\right)^{D/2} exp\left[-\frac{m}{2k_B T} u^2\right]$$

where:

- D - dimension
- m - particle mass
- u - particle speed
- k_B - Boltzmann constant
- T - temperature

For the D2Q9 lattice the non-negative solution (to assure positive mass density) is:

$$f_i^{eq}(\boldsymbol{x},t) = w_i\rho(\boldsymbol{x},t)\left[1+3\frac{\boldsymbol{e}_i\boldsymbol{u}}{e^2}+\frac{9}{2}\frac{(\boldsymbol{e}_i\boldsymbol{u})^2}{e^4}-\frac{3}{2}\frac{\boldsymbol{u}^2}{e^2}\right] \tag{2.13}$$

where:

$$\begin{cases} w_i = 4/9 & i=0 \\ w_i = 1/9 & i=1,2,3,4 \\ w_i = 1/36 & i=5,6,7,8 \\ \frac{e^2}{3} = \frac{k_BT}{m} & \end{cases}$$

From the basic thermodynamics relation, one can recognise the pressure as $p = \frac{k_BT}{m}\rho$ It is interesting to observe that the speed of sound is:

$$c_S = \sqrt{\frac{dp}{d\rho}} = \frac{e}{\sqrt{3}} \tag{2.14}$$

2.2.7.1 Limitations of LBM

Analysis shows that LBM resembles Navier–Stokes equations for incompressible flow for low Mach number. It can be mathematically proved [16, p. 69-70], that the error in LBM is of order of Ma^2.

It is usually assumed that the flow can be considered as incompressible if $Ma < 0.3$, then the lattice velocity is limited to:

$$u_{max} < 0.3\frac{e}{\sqrt{3}} \approx 0.17e\Big|_{e=1} = 0.17$$

However, as a 'rule of thumb' it is recommended to use even lower velocities: $u_{max} < 0.1$

2.3 From LB to Navier Stokes equations

In this section we will discuss the way of deriving the Navier-Stokes equations from the Boltzmann equation.

2.3.1 Chapman-Enskog expansion

The Chapman-Enskog expansion is a method of multiscale expansion of both f and t (but not $\mathbf{x}$) in the powers of a small parameter ϵ. The parameter ϵ represents a small quantity and helps to keep track of the relative orders of magnitude of the various terms. In case of the Chapman-Enskog expansion the Knudsen number can be used, i.e., $\epsilon = K_n$. Let us expand the distribution function and partial derivatives:

$$f_i = f_i^{(0)} + \epsilon f_i^{(1)} + \epsilon^2 f_i^{(2)} + \mathcal{O}(\epsilon^3) \qquad where\, f_i^{(0)} = f_i^{eq} \tag{2.15}$$

Following scaling for time and space derivatives is introduced:

$$\begin{aligned} \partial_t &= \epsilon \partial_t^{(1)} + \epsilon^2 \partial_t^{(2)} \\ \partial_{x_\alpha} &= \epsilon \partial_{x_\alpha}^{(1)} \end{aligned} \tag{2.16}$$

The small perturbations $f_i^{(1)}, f_i^{(2)}, ...$ do not contribute to the mass and momentum density:

$$\begin{cases} \sum_i f_i^{(m)} = 0 & for\, m = 1, 2, ... \\ \sum_i \mathbf{e}_i f_i^{(m)} = 0 & for\, m = 1, 2, ... \end{cases}$$

Let us also expand a part of the Bolztmann kinetic equation 2.12 into a Taylor series:

$$\begin{aligned} f_i(\boldsymbol{x} + \boldsymbol{e_i}\Delta\boldsymbol{x}, t + \Delta t) &= f_i(\boldsymbol{x}, t) + \Delta t \partial_t f_i + \underbrace{\Delta t e_i}_{\Delta \mathbf{x}_i} \partial_{x_\alpha} f_i + \\ &+ \frac{(\Delta t)^2}{2}\left[\partial_t \partial_t f_i + 2e_{i\alpha}\partial_t \partial_{x_\alpha} f_i + e_{i\alpha}e_{i\beta}\partial_{x_\alpha}\partial_{x_\beta} f_i\right] + \mathcal{O}(\partial^3 f_i) \end{aligned} \tag{2.17}$$

The expansions 2.15, 2.16 and 2.17 have to be substituted into the Bolztmann kinetic equation 2.12:

$$0 = f_i(\boldsymbol{x} + \boldsymbol{e_i}\Delta\boldsymbol{x}, t + \Delta t) - f_i(\boldsymbol{x}, t) + \Omega(f_i - f_i^{eq}) \qquad \text{(2.12 revisited)}$$

This leads to:

$$
\begin{aligned}
0 \underbrace{=}_{2.17} \quad & f_i(\boldsymbol{x},t) + \Delta t \partial_t f_i + \Delta t e_i \partial_{x_\alpha} f_i + \frac{(\Delta t)^2}{2}\left[\partial_t \partial_t f_i + 2 e_{i\alpha}\partial_t \partial_{x_\alpha} f_i + e_{i\alpha}e_{i\beta}\partial_{x_\alpha}\partial_{x_\beta} f_i\right] \\
& + \mathcal{O}(\partial^3 f_i) - f_i(\boldsymbol{x},t) + \Omega(f_i - f_i^{eq}) \\
\underbrace{=}_{2.15,2.16} \quad & \epsilon \Delta t\left[\partial_t^{(1)} f_i^{(0)} + e_{i\alpha}\partial_{x_\alpha}^{(1)} f_i^{(0)}\right] \\
& + \epsilon^2 \Delta t\left[\partial_t^{(1)} f_i^{(1)} + \partial_t^{(2)} f_i^{(0)} + + e_{i\alpha}\partial_{x_\alpha}^{(1)} f_i^{(1)}\right] \\
& + \epsilon^2 \frac{(\Delta t)^2}{2}\left[\partial_t^{(1)}\partial_t^{(1)} f_i^{(0)} + \partial_t^{(2)} f_i^{(0)} + 2 e_{i\alpha}\partial_t^{(1)}\partial_{x_\alpha}^{(1)} f_i^{(0)} + e_{i\alpha}e_{i\beta}\partial_t^{(1)}\partial_{x_\alpha}^{(1)}\partial_{x_\beta}^{(1)} f_i^{(0)}\right] \\
& + + \epsilon\Omega f_i^{(1)} + \epsilon^2 \Omega f_i^{(2)} + \mathcal{O}(\epsilon^3)
\end{aligned}
$$

Next, the terms are sorted according to order of ϵ:

$$0 = \epsilon E_i^{(0)} + \epsilon^2 E_i^{(1)} + \mathcal{O}(\epsilon^3) \tag{2.18}$$

where:

$$E_i^{(0)} = \partial_t^{(1)} f_i^{(0)} + e_{i\alpha}\partial_{x_\alpha}^{(1)} f_i^{(0)} + \frac{\Omega}{\Delta t} f_i^{(1)} \tag{2.19}$$

$$
\begin{aligned}
E_i^{(1)} = \quad & \partial_t^{(1)} f_i^{(1)} + \partial_t^{(2)} f_i^{(0)} + e_{i\alpha}\partial_{x_\alpha}^{(1)} f_i^{(1)} \\
& + \frac{(\Delta t)^2}{2}\left[\partial_t^{(1)}\partial_t^{(1)} f_i^{(0)} + \partial_t^{(2)} f_i^{(0)} + + 2 e_{i\alpha}\partial_t^{(1)}\partial_{x_\alpha}^{(1)} f_i^{(0)} + e_{i\alpha}e_{i\beta}\partial_t^{(1)}\partial_{x_\alpha}^{(1)}\partial_{x_\beta}^{(1)} f_i^{(0)}\right] \\
& + \epsilon^2 \Omega f_i^{(2)}
\end{aligned} \tag{2.20}
$$

Mass and momentum flux are corresponing to the zeroth and first lattice velocity moments of $E_i^{(0)}$ and $E_i^{(1)}$. Analysis of the terms of first order in ϵ yields:

$$
\begin{aligned}
\sum_i E_i^{(0)} &= \sum_i \left[\partial_t^{(1)} f_i^{(0)} + e_{i\alpha}\partial_{x_\alpha}^{(1)} f_i^{(0)} + \frac{\Omega}{\Delta t} f_i^{(1)}\right] \\
&= \partial_t^{(1)}\rho + \partial_{x_\alpha}^{(1)} j_\alpha
\end{aligned} \tag{2.21}
$$

$$
\begin{aligned}
\sum_i e_{i\alpha} E_i^{(0)} &= \sum_i e_{i\alpha}\left[\partial_t^{(1)} f_i^{(0)} + e_{i\alpha}\partial_{x_\alpha}^{(1)} f_i^{(0)} + \frac{\Omega}{\Delta t} f_i^{(1)}\right] \\
&= \partial_t^{(1)} j_\alpha + \partial_{x_\alpha}^{(1)} \Pi_{\alpha\beta}^{(0)}
\end{aligned} \tag{2.22}
$$

The momentum density symbol is introduced as:

$$j_\alpha = \rho e_\alpha \tag{2.23}$$

and the momentum flux tensor in the first order of ϵ:

$$\Pi^{(0)}_{\alpha\beta} = \sum_i e_{i\alpha} e_{i\beta} f_i^{(0)} = \frac{1}{\rho} \begin{bmatrix} j_{11}^2 & j_{12} \\ j_{21} & j_{22}^2 \end{bmatrix} + p\delta_{\alpha\beta} \tag{2.24}$$

Finally, at the first order of ϵ we obtained the continuity equation:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot \mathbf{j} = 0 \tag{2.25}$$

and the Euler equation (in the incompressible limit $\rho = const$):

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} = -\frac{1}{\rho}\nabla p \tag{2.26}$$

Applying the same procedure to the terms of the second order in ϵ and summing them up with the first order terms leads to the Navier Stokes equation:

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} = -\frac{1}{\rho}\nabla p + \nu \nabla^2 \mathbf{u} \tag{2.27}$$

with the kinematic shear viscosity (D2Q9):

$$\nu_{LB} = \frac{2\tau - 1}{6}\frac{(\Delta x)^2}{\Delta t} = \frac{e^2}{3}(\tau - \frac{\Delta t}{2}) \tag{2.28}$$

2.3.2 Summary - interpretation of the distribution functions

The hydrodynamics is recovered from the fundamental properties of the distribution functions. To sum up, the first few moments of the equilibria f_i are:
density:

$$\rho = \sum_{i=0}^{N-1} f_i(\boldsymbol{x}, t) \tag{2.29}$$

momentum:

$$\boldsymbol{u} = \frac{1}{\rho} \sum_{i=0}^{N-1} f_i(\boldsymbol{x}, t) \boldsymbol{e_i} \tag{2.30}$$

momentum flux:

$$\boldsymbol{\Pi} = \sum_{i=0}^{N-1} f(\boldsymbol{x}, t) \boldsymbol{e_i} \otimes \boldsymbol{e_i} = \underbrace{\rho c_s^2}_{p} \mathbb{1} + \rho \boldsymbol{u} \boldsymbol{u} \tag{2.31}$$

local stress tensor:

$$\boldsymbol{\sigma} = \sum_{i=0}^{N-1} \left[f_i(\boldsymbol{x}, t) - f_i^{eq}(\boldsymbol{x}, t) \right] \boldsymbol{e_i} \otimes \boldsymbol{e_i} \tag{2.32}$$

Knowing the σ, a turbulence model (for example Smagorinsky) can be incorporated into the LBM by adding additional turbulent viscosity term to the relaxation factor during the collision step. For a D2Q9 lattice fluid viscosity is related to the relaxation time τ as:

$$\nu_{LB} = \frac{2\tau - 1}{6} \frac{(\Delta x)^2}{\Delta t} \tag{2.33}$$

The limit $\tau \to 0.5$ corresponds to the inviscid flow, while $\tau \to \infty$ represents the creeping (Stokes) flow. The last one is stable, however it may have issues with slow convergence rate. From the other hand, when $\tau \to 0.5$ stability issues appear if the lattice resolution is too coarse. This may be explained by the fact, that the velocity gradients can become very large and the model can not dissipate energy due to very low viscosity. The natural remedy is to increase grid density, which naturally needs more computational power. More advanced solution is to use a better collision operator. The so called MRT (Multiple

Relaxation Time) differs from the basic approach (BGK) in the way that in each lattice direction $\boldsymbol{e_i}$ a different relaxation parameter is used.

2.3.3 Lattice tensors

It is natural to ask what conditions must the lattice satisfy to be able to recover the physics of the flow governed by the Navier-Stokes equations.

Let us define the lattice tensors of rank n as:

$$L_{\alpha_1\alpha_2\ldots\alpha_n} = \sum_i e_{i\alpha_1} e_{i\alpha_2} \ldots e_{i\alpha_n} \tag{2.34}$$

where $e_{i\alpha_\nu}$ are cartesian components of the lattice velocities $\boldsymbol{e}_i$

Definition:
The tensor is called isotropic when it has the same components in all rotated coordinate systems.

It has been proven that the lattice tensor must be isotropic up to 4th rank (higher ranks consists of products of lower rank tensors [5]) to be able to recover the stress tensor in a real fluid [24]. More detailed discussion, covered with examples may be found in [4, pp. 90-112].
For us, it is enough to know that the D2Q4 lattice has a symmetric tensor of rank 2 and a non-symmetric tensor of rank 4 thus it fails to model real fluid, while a D2Q9 lattice has all tensors symmetric, thus it may be used to produce physical results. For less demanding problems like advection - diffusion a D2Q4 lattice is sufficient.

2.4 Unit conversion LB $\longleftrightarrow$ NS

The parameters have to be chosen in a way which will keep the dimensionless flow parameters (like Reynolds number) untaught. The reasoning of unit conversion is based on the dimensional analysis. The process is simple but inconvenient thus it is an obvious drawback of the LBM.

2.4.1 Unit conversion - example

Let us consider a flow around an obstacle. The physical dimensions are:

$v = 10^{-6}\,[\frac{m^2}{s}]$ - kinematic viscosity of water
$U = 7.5 \cdot 10^{-4}\,[\frac{m}{s}]$ - free flow velocity
$d = 0.2\,[m]$ - obstacle diameter

To convert the physical problem to the world of LB the Reynolds number must be preserved:

$$Re = Re_{LB} = 150 \Longleftrightarrow \frac{Ud}{v} = \frac{U_{LB} d_{LB}}{v_{LB}} \tag{2.35}$$

Let us impose $U_{LB} = 0.1$ and $v_{LB} = \frac{1}{6}$. Then the missing d_{LB} is:

$$d_{LB} = Re\frac{v_{LB}}{U_{LB}} = 150 \cdot \frac{\frac{1}{6}}{0.1} = 250\,[lu]$$

Additionally, let us find the lattice time step $[lt]$.
One of the possible choices of expressing the relation $[lt] \longleftrightarrow [s]$ is:

$$\frac{d}{U}\frac{U_{LB}}{d_{LB}}\left[\frac{m}{m/s}\right]\left[\frac{lu/lt}{lu}\right] = \left[\frac{s}{lt}\right] \tag{2.36}$$

$$1[lt] = \frac{d}{U}\frac{U_{LB}}{d_{LB}}[s] = \frac{0.2}{7.5 \cdot 10^{-4}}\frac{0.1}{250}[s] \simeq 0.11[s]$$

From the conduced analysis follows that the change of one parameter affects other parameters of simulation. It is impossible to control time and space density independently.

2.5 Computational Algorithm

The motion of the fluid and the passive scalar are solved on two separate lattices coupled by the velocity term.

The weight factors for D2Q9 (fluid) lattice are:

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

while for D2Q5 (passive scalar) lattice:

$$w_i = \begin{cases} 2/6 & i = 0 \\ 1/6 & i = 1, 2, 3, 4 \end{cases}$$

The collision frequency $\Omega = \frac{1}{\tau}$ where:

$$\tau = \begin{cases} \nu_{LB} \frac{3\Delta t}{(\Delta x)^2} + \frac{1}{2} & \text{for fluid} \\ \\ \lambda_{LB} \frac{3\Delta t}{(\Delta x)^2} + \frac{1}{2} & \text{for passive scalar} \end{cases}$$

2.5.1 Algorithm - fluid

The algorithm → momentum transfer:

1 Initialize f_i^{in}

2 Compute ρ, $\boldsymbol{u}(\boldsymbol{x},t)$

$$\rho = \sum_{i=0}^{8} f_i^{in}(\boldsymbol{x},t) \tag{2.37}$$

$$\boldsymbol{u}(\boldsymbol{x},t) = \frac{1}{\rho}\sum_{i=0}^{8} \boldsymbol{e}_i\, f_i^{in}(\boldsymbol{x},t) \tag{2.38}$$

3 Compute $f_i^{eq}(\boldsymbol{x},t)$

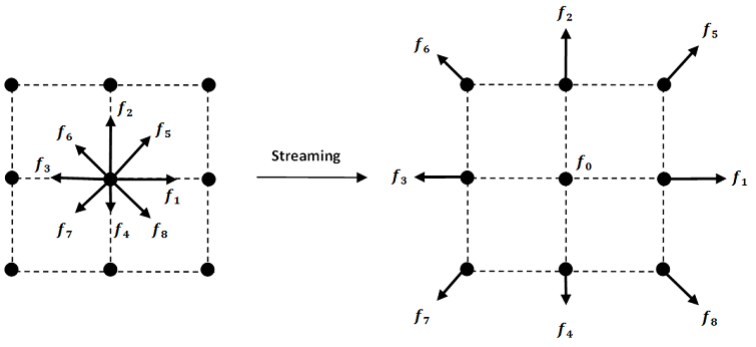
$$f_i^{eq}(\boldsymbol{x},t) = w_i\rho(\boldsymbol{x},t)\left[1 + 3\frac{\boldsymbol{e}_i\boldsymbol{u}}{e^2} + \frac{9}{2}\frac{(\boldsymbol{e}_i\boldsymbol{u})^2}{e^4} - \frac{3}{2}\frac{\boldsymbol{u}^2}{e^2}\right] \tag{2.39}$$

4 $\mathbb{C}$ - BGK Collision

$$f_i^{out}(\boldsymbol{x},t) = (1-\Omega_f)f_i^{in}(\boldsymbol{x},t) + \Omega_f f_i^{eq}(\boldsymbol{x},t) \tag{2.40}$$

5 $\mathbb{S}$ - Streaming

$$f_i^{in}(\boldsymbol{x}+\boldsymbol{e}_i, t+1) = f_i^{out}(\boldsymbol{x},t) \tag{2.41}$$



2.5.2 Algorithm - passive scalar

The algorithm → advection - diffusion of the passive scalar T:

The algorithm - passive scalar:

1 Initialize $\theta_i^{in}(\boldsymbol{x}, t)$

2 Compute $T(\boldsymbol{x}, t)$

$$T(\boldsymbol{x}, t) = \sum_{i=0}^{4} \theta_i^{in}(\boldsymbol{x}, t) \tag{2.42}$$

3 Compute $\theta_i^{eq}(\boldsymbol{x}, t)$

$$\theta_i^{eq}(\boldsymbol{x}, t) = T(\boldsymbol{x}, t) w_i \left[1 + 3\frac{\boldsymbol{e}_i \boldsymbol{u}}{e^2}\right] \tag{2.43}$$

4 $\mathbb{C}$ - BGK Collision

$$\theta_i^{out}(\boldsymbol{x}, t) = (1 - \Omega_\theta)\theta_i^{in}(\boldsymbol{x}, t) + \Omega_\theta \theta_i^{eq}(\boldsymbol{x}, t) \tag{2.44}$$

5 $\mathbb{S}$ - Streaming

$$\theta_i^{in}(\boldsymbol{x} + \boldsymbol{e}_i, t + 1) = \theta_i^{out}(\boldsymbol{x}, t) \tag{2.45}$$

The physical interpretation of the passive scalar would be an ink in a fluid.
To consider the passive scalar as a temperature an additional buoyancy term should be added to the Collision operator to couple both lattices.

2.5.3 Boundary conditions

2.5.3.1 Wall - bounce back

The on-grid bounce back does not distinguish the orientation of the boundaries. The implementation is pretty simple since and allows for simulating flows in complex geometries (like porous media). During the collision step the distribution function are reversed on the boundary node. It has been verified that this kind of implementation is only 1st order accurate.

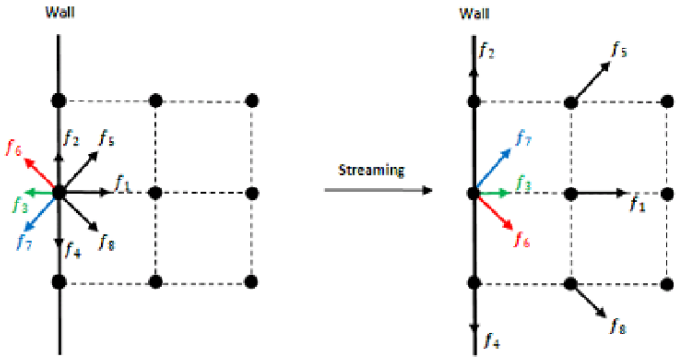


FIGURE 2.4: On-grid bounce back [2, p. 4]

The C++ implementation is straightforward and self-explanatory:

```
int oppositeF[9] = { 0, 3, 4, 1, 2, 7, 8, 5, 6 };

for (int i = 0; i < 9; ++i)
      fOut[i] = fIn[oppositeF[i]];
```

2.5.3.2 Symmetry

The symmetry BC is similar to the bounce back. It should be noted, that the last two pairs of indices have been exchanged ($7 \longleftrightarrow 8$ and $5 \longleftrightarrow 6$) comparing to the bounce back, which corresponds to the condition of slip on the symmetry node.

```
1 int oppositeF[9] = { 0, 3, 4, 1, 2, 8, 7, 6, 5 };
2
3 for (int i = 0; i < 9; ++i)
4       fOut[i] = fIn[oppositeF[i]];
```

2.5.3.3 Periodic

To accomplish the periodic BC it is enough to move the distributions f_i from one side of the domain to the other.

2.5.3.4 Velocity/Pressure BC

This particular BC was originally developed by Zou and He [25].
Reminding the equations 2.37 and 2.38:

$$\begin{cases} \rho = \sum_{i=0}^{8} f_i^{in}(\boldsymbol{x}, t) \\ \boldsymbol{u}(\boldsymbol{x}, t) = \dfrac{1}{\rho} \sum_{i=0}^{8} \boldsymbol{e}_i \, f_i^{in}(\boldsymbol{x}, t) \end{cases}$$

The idea of Zou and He is to formulate a linear system for solving f_i^{in} which comes out of the domain knowing the velocity/pressure at the boundary.

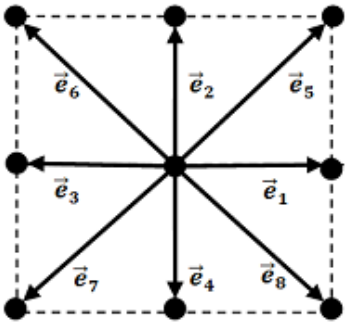


FIGURE 2.5: Discrete velocities [2, p. 2]

For example, let us consider a moving lid:

$$\begin{cases} f_7^{in} + f_4^{in} + f_8^{in} = \rho - (f_6^{in} + f_2^{in} + f_5^{in} + f_3^{in} + f_0^{in} + f_1^{in}) \\ f_1^{in} + f_5^{in} + f_8^{in} = \rho u_x + (f_3^{in} + f_6^{in} + f_7^{in}) \\ f_5^{in} + f_2^{in} + f_6^{in} = \rho u_y + (f_6^{in} + f_2^{in} + f_5^{in}) \end{cases} \tag{2.46}$$

Using two equations from the above, the ρ can be determined as:

$$\rho = \frac{1}{1 - u_x} \left[f_0^{in} + f_2^{in} + f_4^{in} + 2(f_3^{in} + f_6^{in} + f_7^{in}) \right] \tag{2.47}$$

To close the system Zou and He assumed that the bounce-back rule is still valid for the non-equilibrium part of the particle distributions f normal to the boundary:

$$f_2^{in} - f_2^{eq} = f_4^{in} - f_4^{eq} \tag{2.48}$$

Now, the system of equations is ready to be solved:

$$f_7^{in} = f_2^{in} - \frac{2}{3}\rho u_x \tag{2.49}$$

$$f_4^{in} = f_2^{in} + \frac{1}{2}(f_1^{in} - f_3^{in}) - \frac{1}{2}\rho u_x - \frac{1}{6}\rho u_y \tag{2.50}$$

$$f_8^{in} = f_6^{in} + \frac{1}{2}(f_3^{in} - f_1^{in}) + \frac{1}{2}\rho u_x - \frac{1}{6}\rho u_y \tag{2.51}$$

To compute the f_7^{out} the relaxation procedure is applied.

A remark regarding the passive scalar distribution function should be done. The wall is moving, thus the simple bounce-back BC is not enough. First the unknown distribution must be found. The condition of zero flux through the wall is met when the isolines of θ are perpendicular to the boundary:

$$0 = \int_{\partial\Omega} \mathbf{q} \cdot \mathbf{n} dS = \int_{\partial\Omega} -\nabla\theta \cdot \mathbf{n} dS \tag{2.52}$$

Similarly to density ρ and velocity $\mathbf{u}$, the passive scalar T and its flux q can be expressed as moments of θ:

$$\begin{cases} T = \sum_{i=0}^{4} \theta_i^{in}(\boldsymbol{x}, t) \\ \boldsymbol{q}(\boldsymbol{x}, t) = \dfrac{1}{\rho} \sum_{i=0}^{4} \boldsymbol{e}_i \, \theta_i^{in}(\boldsymbol{x}, t) \end{cases}$$

In the case of the moving lid on D2Q5 lattice:

$$\begin{aligned} 0 &= \int_{\partial\Omega} \mathbf{q} \cdot \mathbf{n} dS = \mathbf{q} \cdot \mathbf{n} \\ &= \sum_{i=0}^{4} e_i \theta_i^{in} \cdot \mathbf{n} \\ &= \theta_2^{in} \cdot [0, 1] + \theta_4^{in} \cdot [0, -1] \\ \theta_4^{in} &= \theta_2^{in} \end{aligned} \tag{2.53}$$

The outgoing unknown θ_4^{in} is equal to the incoming θ_2^{in} Ultimately, the relaxation procedure is applied, which results in θ_i^{out}.

Chapter 3

Lattice Boltzmann Method - Validation

To validate the computational code, a following set of tests has been performed.

3.1 von Karman vortex street

The von Karman vortex street is a complex phenomenon which has been excessively studied and covered with experimental data. It may be used as benchmark since the relation between blockage ratio, Reynolds and Strouhal number is known [19].

Geometry (in lattice units) and boundary conditions:

- lattice size: 2100x300
- obstacle (cylinder): d = 120, located in the 2/7 from the inlet
- velocity inlet (left) - Poiseuille profile with $u_{max} = 0.05$ ($u_{average} = \frac{2}{3}u_{max}$)
- pressure outlet (right)
- walls (top & bottom)
- lattice kinematic viscosity $\nu_{LB} = 0.02$
- velocity measurement point: 4/7 of the channel

The results obtained from LBM are in agreement with the experimental data:

$$Blockage\,Ratio = \frac{d_{obstacle}}{d_{channel}} = 0.4$$

$$Re_{LB} = \frac{u_{avg}d}{\nu_{LB}} = \frac{0.0(3) \cdot 120}{0.02} = 200$$

$$St = \frac{fd}{u_{avg}} = \frac{0.000111 \cdot 120}{0.0(3)} = 0.401$$

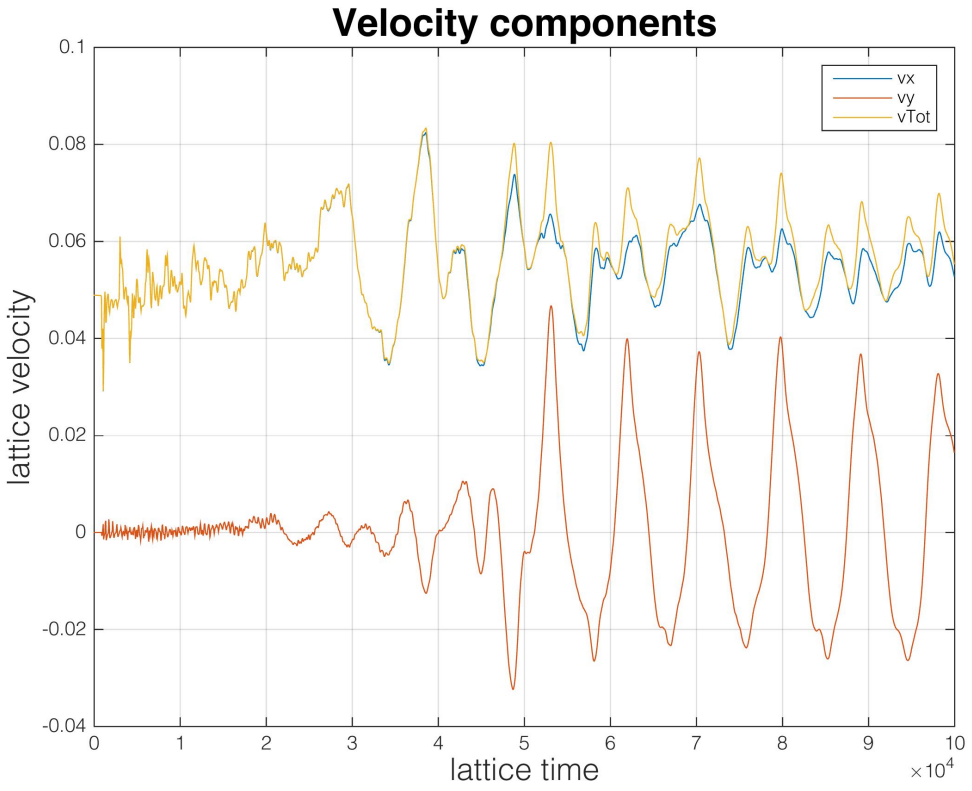


FIGURE 3.1: Velocity as function of time: v_x (blue), v_y (red), v_{total} (yellow)

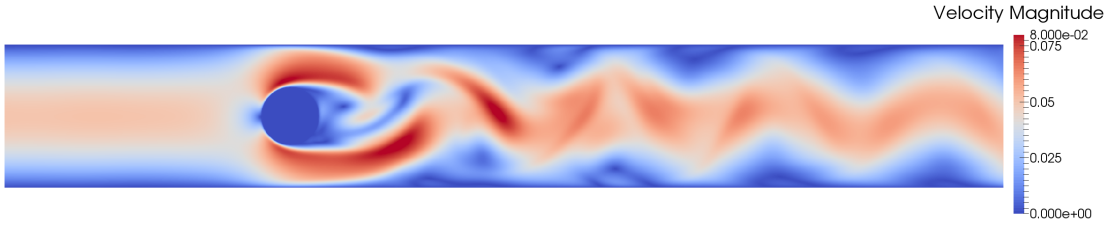


FIGURE 3.2: Velocity Contours

3.2 Passive Scalar - conservation

The conservation of the passive scalar was proved by summing up its intensity at each node at the beginning and at the end of the simulation:

$$\sum_{i=1}^{N} T_i \Big|_{t=0} = \sum_{i=1}^{N} T_i \Big|_{t=t_{end}}$$

As expected, its uniform distribution has not changed.

The whole control volume of fluid being at rest was initialized with a uniform distribution of θ. Then the fluid is accelerated by the moving wall. Little disturbance can be noticed if the fluid accelerates rapidly. This can be explained by the fact, that the LBM is of slightly compressible nature and that the equation being solved is ρT rather than T.

It is advisable to divide the result $T(\boldsymbol{x}, t) = \sum_{i=0}^{4} \theta_i^{in}(\boldsymbol{x}, t)$ by ρ to get the exact value during the post-processing. It is worth reminding that the BC for the passive scalar at the lid is a Zou-He velocity not bounce-back as stated in 2.5.3.4.

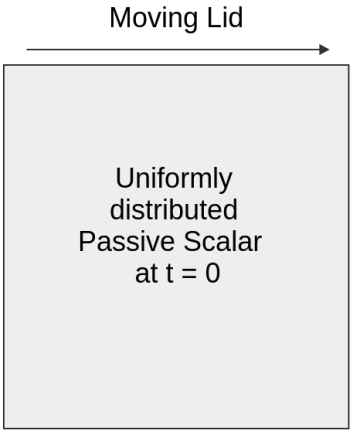


FIGURE 3.3:
Uniform distribution of passive scalar and its total amount is conserved

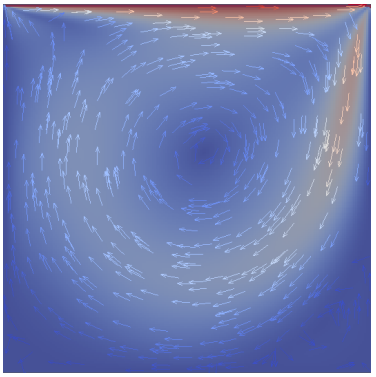


FIGURE 3.4:
Velocity Contours $\Big|_{t=t_{end}}$

3.3 Advection

In the LBM the advection is embedded in the advection-diffusion couple:

$$\frac{\partial T}{\partial t} + \mathbf{u}\frac{\partial T}{\partial t} = \lambda \frac{\partial^2 T}{\partial \mathbf{x}^2}$$

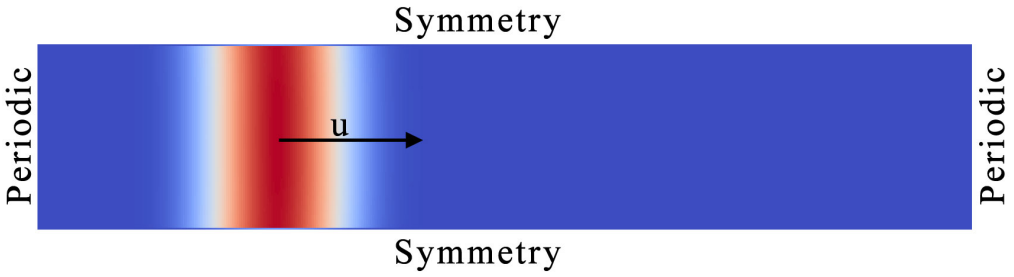


FIGURE 3.5: Experimental setup: flow is uniformly initialized with u = 0.1

The computational program was successfully validated by measuring the speed of advection of T. As expected, it is equal to the flow velocity:

$$u = \frac{\Delta x}{\Delta t} = \frac{256 - 116}{1800 - 400} = 0.1$$

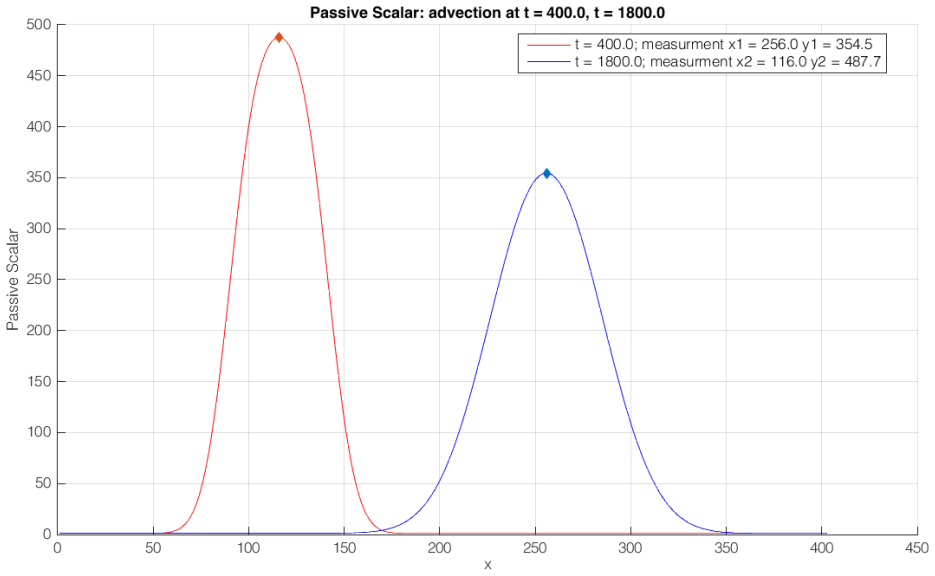


FIGURE 3.6: Advection, points of measurement are marked with diamonds

3.4 Diffusion

The differential equation describing the heat diffusion:

$$\frac{\partial T}{\partial t} = \lambda \frac{\partial^2 T}{\partial \mathbf{x}}$$

With BC:

$$T(x, t = 0) = \begin{cases} 0 & for\, x \in (-\infty, b) \\ T_{max} & for\, x \in (b, \infty) \end{cases}$$

The Gauss error function appears in solution of the PDE describing diffusion.

It is defined as:

$$erf(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt = \frac{2}{\sqrt{\pi}} \sum_{n=0}^{\infty} \frac{(-1)^n x^(2n+1)}{n!(2n+1)}$$

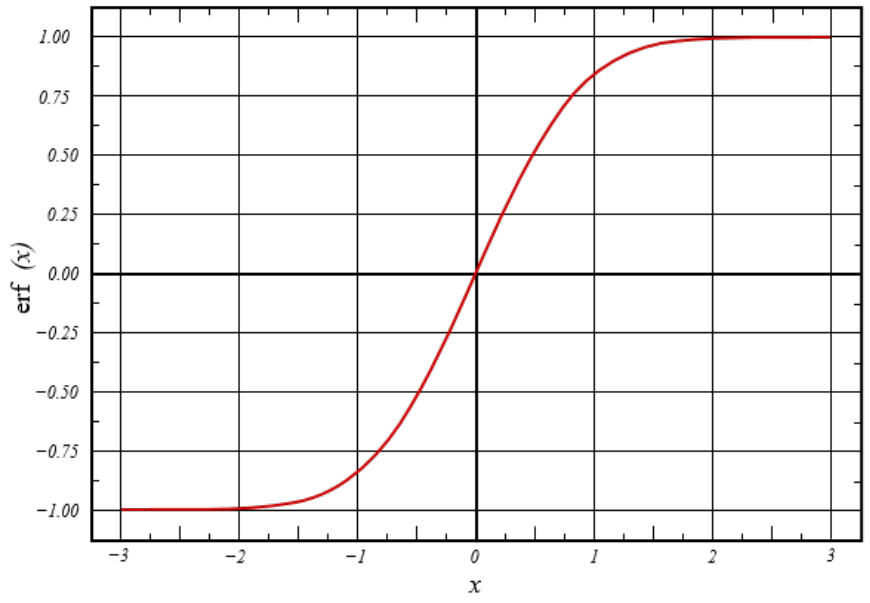


FIGURE 3.7: Error function

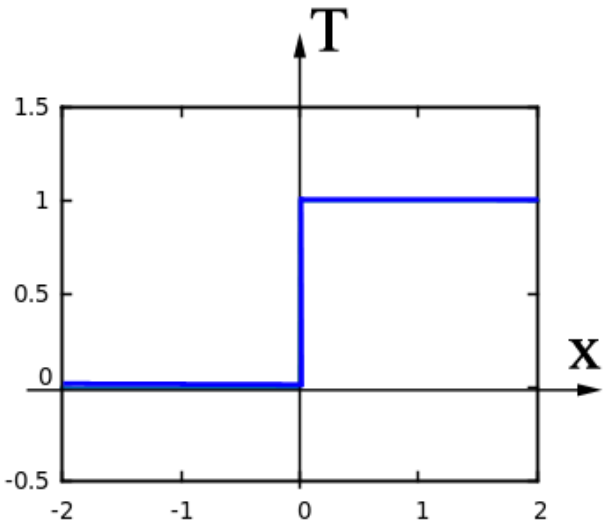


FIGURE 3.8: $T(x, t = 0)$, step at b = 0

A 1D diffusion equation with IC as on figure 3.8 has a solution of the form:

$$\theta(x,t) = \frac{T_{max}^2}{2} erf(\frac{x-b}{\sqrt{4\lambda t}})$$

The analytical solution is in excellent agreement with the LBM:

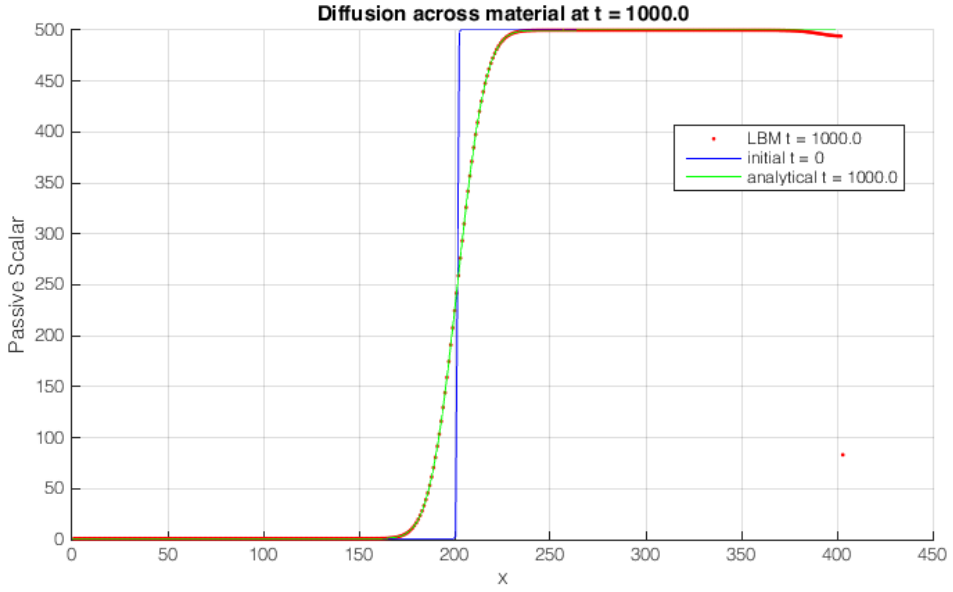


FIGURE 3.9: Diffusion

3.5 Wall shear force

The easiest way to validate shear force between the fluid and the wall is to compare it with the analytical solution of the Couette flow.

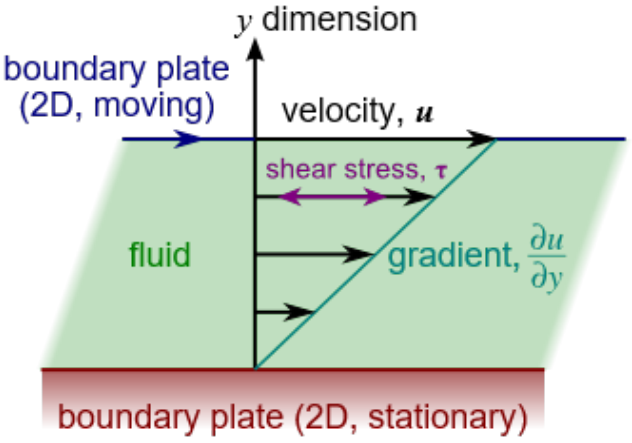


FIGURE 3.10: Couette flow

Boundary conditions:

$$\begin{cases} u(y=h) = u_{Lid} \\ u(y=0) = 0 \end{cases}$$

The analytical solution shows that the velocity profile is linear:

$$u(y) = u_{Lid}\frac{y}{h}$$

Knowing that, it is trivial to calculate the shear stress:

$$\tau = \mu \frac{du}{dy}\bigg|_{y=0}$$

To get the value of the force it is enough to integrate the shear stress at the wall surface:

$$F_D = \int \tau dx = \mu\, u_{Lid}\frac{\Delta x}{h}$$

In the LBM force is calculated by the momentum exchange method. The force transferred by the wall-node to the fluid-node can be expressed as:

$$\boldsymbol{F_D} = \boldsymbol{P}_{out} - \boldsymbol{P}_{in} \tag{3.1}$$

In case of the bottom wall the momentum exchange would be expressed as follows: Momentum of the incoming particles $\boldsymbol{p}_i = f_i \boldsymbol{e}_i$:

$$\boldsymbol{P}_{in} = \boldsymbol{p}_7 + \boldsymbol{p}_4 + \boldsymbol{p}_8 \tag{3.2}$$

Momentum of the outcoming particles (after the bounce back):

$$\boldsymbol{P}_{out} = \boldsymbol{p}_6 + \boldsymbol{p}_2 + \boldsymbol{p}_5 \tag{3.3}$$

The test has proved that the LB momentum exchange method is in perfect agreement with the analytical solution.

Chapter 4

Adjoint - introduction

As was mentioned at the beginning of chapter 1, the main motivation standing behind the adjoint technique is to reduce the computational cost of problems involving many control variables, since the cost of solving the dual system does not depend on their number.

There are two approaches to the adjoint technique. First, the continuous adjoint is based on the equation dual to the original problem (e.g., the Navier-Stokes equations). This is quite suitable, because usually both primal and dual equations are of similar nature. Thanks to this, in most cases, the discretization can be done by the same scheme as the original problem. Moreover, the possibility of physical insight into the detailed structure of the dual equation can lead to useful conclusions. Unfortunately, to obtain it, a lot of hand calculation involving high level mathematics has to be done.

On the other hand, the discrete adjoint work-flow is to formulate equations dual to already discretized original problem. The second approach can be easily automated (without loosing numerical accuracy), which makes it more flexible, easier to maintain and develop new add-ons. This is a huge advantage, because many of hand differentiated codes, used widely in commercial solvers, suffer from problem with maintenance, as the adjoint code has to be kept up-to-date with the primal [14].

More detailed discussion of continuous and discrete adjoint in terms of LBM can be found in [7].

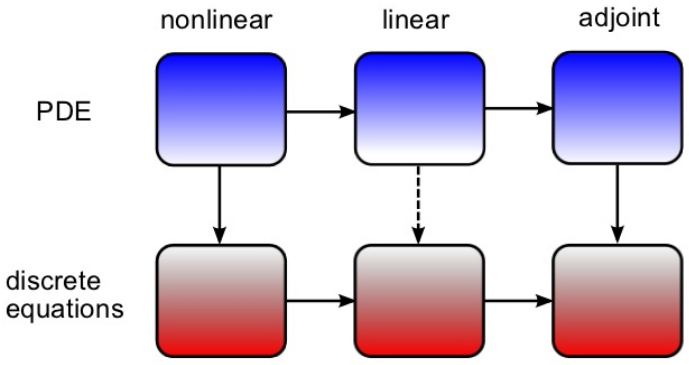


FIGURE 4.1: Continous vs Discrete adjoint

In this work the discrete adjoint (bottom approach from fig 4.1) is used.

4.1 Adjoint - basic concept

To show the basic concept, we will make a general adjoint formulation for finding the gradient of the objective function subjected to nonlinear equality constrains.

Let J be an objective function such that $J\left(Q(\alpha), \alpha\right)$ where:

α - control variables

Q - vector of state for which the residuals $R = R(Q(\alpha), \alpha) = 0$

We would like to find the derivative of the objective function:

$$\frac{dJ}{d\alpha} = \frac{\partial J}{\partial \alpha} + \underbrace{\left(\frac{\partial J}{\partial Q}\right)^T}_{h^T} \underbrace{\frac{\partial Q}{\partial \alpha}}_{U} = \frac{\partial J}{\partial \alpha} + h^T U \tag{4.1}$$

4.1.1 Tangent method

First, let us consider the naive way. The derivative of the residuals must be equal to 0, since the constrains have to preserved.

$$\begin{aligned} \frac{dR}{d\alpha} = 0 \quad &\Leftrightarrow \quad \frac{\partial R}{\partial \alpha} + \frac{\partial R}{\partial Q}\frac{\partial Q}{\partial \alpha} = 0 \\ &\Leftrightarrow \quad \underbrace{\frac{\partial R}{\partial Q}}_{A}\underbrace{\frac{\partial Q}{\partial \alpha}}_{U} = \underbrace{-\frac{\partial R}{\partial \alpha}}_{B} \end{aligned} \tag{4.2}$$

Finally, we arrive with the formula:

$$\frac{dJ}{d\alpha} = \frac{\partial J}{\partial \alpha} + h^T U \qquad \text{- where: } AU = B \tag{4.3}$$

The disadvantage is that to find the unknown matrix U the system of systems of linear equations $AU = B$ has to be solved. If the number of control variables (size of the α vector is k and the size of the vector of state is n, then determination of the U matrix requires solving k linear systems of size n. The remedy to the numerical cost lies in the dual method.

4.1.2 Dual Method

The key advantage of the dual method is that for a single objective function J, the dual system is always a single linear system of size n. In comparison, in the tangent method there were as many systems as the size of the control variables α vector. It should be emphasized, that the cost of solving the dual problem 4.4 does not depend on the size of $\boldsymbol{\alpha}$. Solution of the dual system allows to determine the sensitivity of J to all control variables. To accomplish this task, let us pick up such a vector v that:

$$h^T U = (A^T v)^T U = v^T A U = v^T B$$

Thus the v vector is a solution of a dual problem:

$$A^T v = h \Leftrightarrow \left(\frac{\partial R}{\partial Q}\right)^T v = \frac{\partial J}{\partial Q} \tag{4.4}$$

Finally, the differential of the objective can be written as:

$$\frac{dJ}{d\alpha} = \frac{\partial J}{\partial \alpha} + v^T \underbrace{\left(-\frac{\partial R}{\partial \alpha}\right)}_{B} \tag{4.5}$$

4.1.3 Lagrange multipliers

The dual (adjoint) method can be derived using the Langrange multipliers. First, we have to introduce the extended objective function as:

$$\widehat{J}(Q,\alpha) = J(Q,\alpha) - v^T R(Q,\alpha) \tag{4.6}$$

Observe, that for the vector of state $Q = Q(\alpha)$ functions $\widehat{J}$ and J are equal to each other:

$$\widehat{J}(Q,\alpha) = J(Q(\alpha),\alpha) - v^T \underbrace{R(Q(\alpha),\alpha)}_{=0} \tag{4.7}$$

Moreover, their differentials are also equivalent:

$$\left.\frac{d\widehat{J}}{d\alpha}\right|_{\alpha,Q(\alpha)} = \left.\frac{J}{d\alpha}\right|_{\alpha,Q(\alpha)} \tag{4.8}$$

Thus:

$$\begin{aligned}\frac{d\widehat{J}}{d\alpha} &= \frac{\partial J}{\partial \alpha} + \left(\frac{\partial J}{\partial Q}\right)^T \frac{\partial Q}{\partial \alpha} - v^T\left[\frac{\partial R}{\partial \alpha} + \frac{\partial R}{\partial Q}\frac{\partial Q}{\partial \alpha}\right] \\ &= \frac{\partial J}{\partial \alpha} - v^T\frac{\partial R}{\partial \alpha} + \left[\left(\frac{\partial J}{\partial Q}\right)^T - v^T\frac{\partial R}{\partial Q}\right]\frac{\partial Q}{\partial \alpha}\end{aligned} \tag{4.9}$$

Since the vector v can be chosen arbitrary, we will do it in a way which will allows us to eliminate the 'unwanted' term containing $\frac{\partial Q}{\partial \alpha}$ (the 'U' matrix). This can be accomplished when:

$$\left(\frac{\partial J}{\partial Q}\right)^T - v^T\frac{\partial R}{\partial Q} = 0 \Leftrightarrow \underbrace{\left(\frac{\partial R}{\partial Q}\right)^T}_{A^T} v = \underbrace{\frac{\partial J}{\partial Q}}_{h} \tag{4.10}$$

Having said that, the formula for $\frac{d\widehat{J}}{d\alpha}$ and at the same time for $\frac{dJ}{d\alpha}$ can be written as:

$$\frac{dJ}{d\alpha} = \frac{\partial J}{\partial \alpha} + v^T \underbrace{\left(-\frac{\partial R}{\partial \alpha}\right)}_{B} \tag{4.11}$$

4.1.4 Summary: primal and adjoint equations

Derivative of the objective function:

$$\frac{dJ}{d\alpha} = \frac{\partial J}{\partial \alpha} + \underbrace{\left(\frac{\partial J}{\partial Q}\right)^T}_{h^T} \underbrace{\frac{\partial Q}{\partial \alpha}}_{U}$$

Primal equation:

$$AU = B \qquad \Leftrightarrow \qquad \underbrace{\frac{\partial R}{\partial Q}}_{A} \underbrace{\frac{\partial Q}{\partial \alpha}}_{U} = \underbrace{-\frac{\partial R}{\partial \alpha}}_{B}$$

Dual equation:

$$A^T v = h \qquad \Leftrightarrow \qquad \underbrace{\left(\frac{\partial R}{\partial Q}\right)^T}_{A^T} v = \underbrace{\frac{\partial J}{\partial Q}}_{h}$$

Then, using basic properties of transposition, the computionaly expensive part ($h^T U$) of the functional differential $\frac{dJ}{d\alpha}$ can be computed as:

$$\left(\frac{\partial J}{\partial Q}\right)^T \frac{\partial Q}{\partial \alpha} = h^T U = (A^T v)^T U = v^T \underbrace{AU}_{B} = v^T B$$

4.2 Adjoint - application to LBM

4.2.1 Stationary solution

First, imagine a stationary flow solved with the LBM.

The goal is to find the derivative of the objective function $J\left(f(\alpha), \alpha\right)$:

$$\frac{dJ}{d\alpha} = \frac{\partial J}{\partial \alpha} + \underbrace{\left(\frac{\partial J}{\partial f}\right)^T}_{h^T} \underbrace{\frac{\partial f}{\partial \alpha}}_{U}$$

At equilibrium, the residuum $R\left(f(\alpha), \alpha\right) = 0$

$$R = f - \mathbb{C}(f(\alpha), \alpha)\mathbb{S} = 0$$

where $\mathbb{C}$ and $\mathbb{S}$ are Collision and Streaming operators, respectively.

The Streaming $\mathbb{S}$ does not depend on f neither on α, since its role is just to move the values distribution functions f from a given lattice node to neighbouring ones.

Differentiate R with respect to α as a control parameter:

$$\frac{dR}{d\alpha} = \frac{\partial f}{\partial \alpha} - \frac{\partial \mathbb{C}}{\partial f}\frac{\partial f}{\partial \alpha}\mathbb{S} - \frac{\partial \mathbb{C}}{\partial \alpha}\mathbb{S} = 0$$

Consider it as a set of equations.

Primal:

$$\underbrace{\left[I - \frac{\partial \mathbb{C}}{\partial f}\mathbb{S}\right]}_{A} \underbrace{\frac{\partial f}{\partial \alpha}}_{U} - \underbrace{\frac{\partial \mathbb{C}}{\partial \alpha}\mathbb{S}}_{B} = 0$$

Dual:

$$\underbrace{\left[I - \frac{\partial \mathbb{C}}{\partial f}\mathbb{S}\right]^T}_{A^T} v - \underbrace{\frac{\partial J}{\partial f}}_{h} = 0$$

To get v, the dual system has to be solved. Since the set of equations is large, an iterative approach is preferred:

$$\left[I - \mathbb{S}^T \left(\frac{\partial \mathbb{C}}{\partial f}\right)^T\right] v - \frac{\partial J}{\partial f} = 0 \Leftrightarrow v - \mathbb{S}^T \left(\frac{\partial \mathbb{C}}{\partial f}\right)^T v - \frac{\partial J}{\partial f} = 0 \tag{4.12}$$

Then a direct iterative scheme can be constructed in a following manner:

$$v_{n+1} = \mathbb{S}^T \left(\frac{\partial \mathbb{C}}{\partial f}\right)^T v_n + \frac{\partial J}{\partial f}$$

Now we are ready to plug in the solution to $\frac{dJ}{d\alpha}$:

$$\frac{dJ}{d\alpha} = \frac{\partial J}{\partial \alpha} + \underbrace{\left(\frac{\partial J}{\partial f}\right)^T}_{h^T} \underbrace{\frac{\partial f}{\partial \alpha}}_{U} = ... = \frac{\partial J}{\partial \alpha} + v^T b \tag{4.13}$$

4.2.2 Discrete dynamical system - general conception

Following the work of Łaniewski-Wołłk [12], let us define a discrete dynamical system:

$$\begin{bmatrix} u_{n+1} \\ g_{n+1} \end{bmatrix} = H(u_n, \alpha_n) \tag{4.14}$$

The system is analysed in a discrete time domain $t \in [0, 1, ..., N]$

The objective function is defined as a product of a weights h and a corresponding function of interest g:

$$J = \mathbf{h} \cdot \mathbf{g} = \sum_{n=1}^{N} h_n \cdot g_n \tag{4.15}$$

Now, using a formal differentiation parameter s, the derivative of the objective J can be expressed as:

$$\frac{d}{ds} J = \sum_{n=1}^{N} h_n \cdot \frac{\partial g_n}{\partial s} \tag{4.16}$$

In a variational form, one would write: $\delta J = \sum_{n=1}^{N} h_n \cdot \delta g_n$

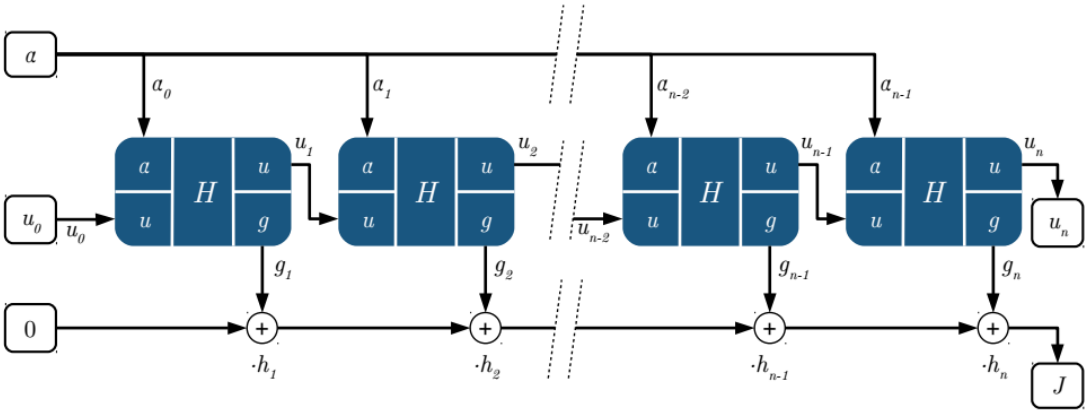


FIGURE 4.2: Discrete iterative process (from Łaniewski-Wołłk [12])

Theorem 4.1. *Let us define an adjoint equation as:*

$$\begin{bmatrix} v_{n-1} \\ \beta_{n-1} \end{bmatrix} = [dH]^T \begin{bmatrix} v_n \\ h_n \end{bmatrix} \tag{4.17}$$

If u is a solution of 4.14 and v is a solution of the adjoint equation with $v_N = 0$, then the differential of the objective function J can be calculated as:

$$\frac{d}{ds}J = v_0 \cdot \frac{\partial u_0}{\partial s} + \sum_{n=0}^{N-1} \beta_n \cdot \frac{\partial \alpha_n}{\partial s} \tag{4.18}$$

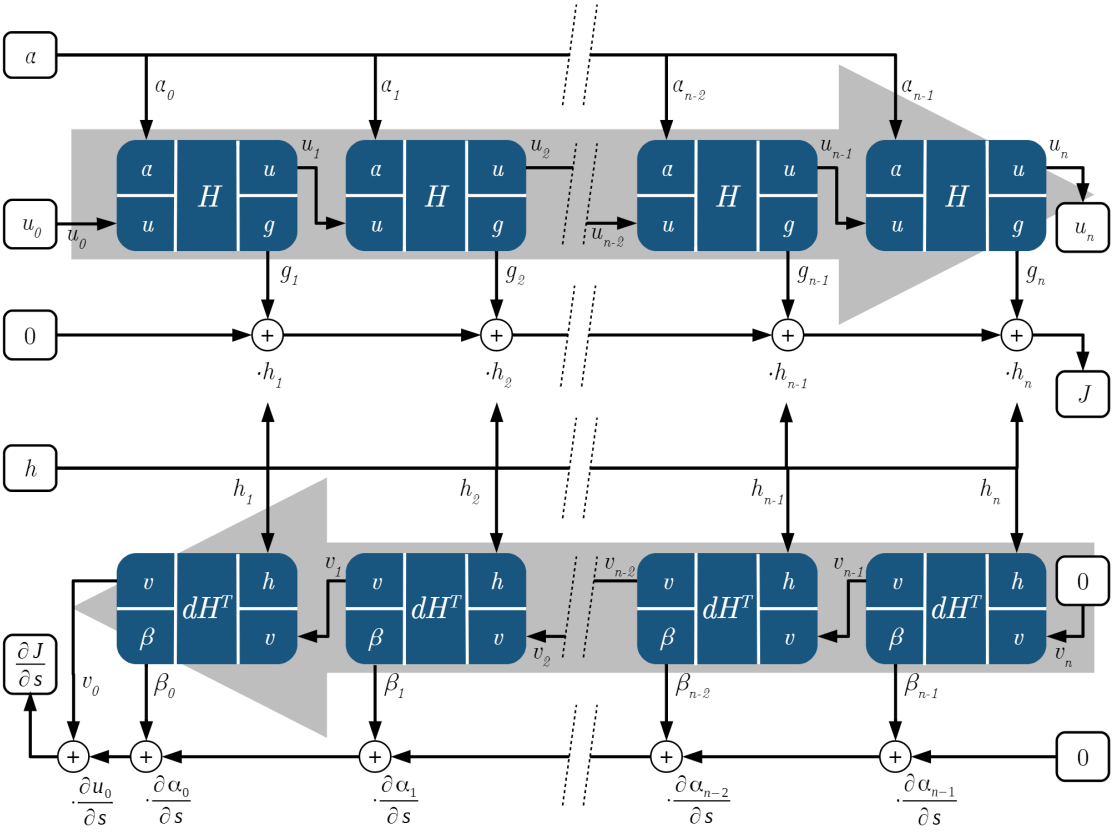


FIGURE 4.3: Discrete adjoint iterative process (from Łaniewski-Wołłk [12])

The proof of theorem 4.1 and remarks regarding $\frac{d}{ds}J$ are in the appendix B.

A simple, illustrative example of the application of the discrete adjoint is provided in the appendix A.

4.3 The objective function - LB approach

Before we formulate adjoint for the LBM, we have to specify how to calculate the objective function. As stated in the beginning of the thesis (see section 1.3), the objective J is given as:

$$J[\boldsymbol{u}, T] = \underbrace{\int_{\Omega} [T(t_{end}) - \overline{T}]^2 d\boldsymbol{x}}_{I_1 \text{ - mix quality}} + \varepsilon \underbrace{\left[\int_0^{t_{end}} \left(\int_{\partial\Omega} 2\nu \boldsymbol{u} \cdot \boldsymbol{D_u} \boldsymbol{n} dS \right) dt \right]^2}_{I_2 \text{ - work needed to impose motion}} \tag{4.19}$$

The mean $\overline{T} = \frac{1}{N} \sum_{i=1}^{N} T_i$, while the first integral I_1 is just a variance of T:

$$\begin{aligned} I_1 &= Var(T) = \sigma^2 = \int_{\Omega} [T(t) - \overline{T}]^2 d\boldsymbol{x} = \frac{1}{N} \sum_{i=1}^{N} (T_i - \overline{T})^2 \\ &= \frac{1}{N} \sum_{i=1} (T_i^2 - 2T_i\overline{T} + \overline{T}^2) = \frac{1}{N} (NT_i^2 - 2T_i \sum_i \overline{T} + \sum_i T_i^2) \\ &= \overline{T}^2 - 2\overline{T}^2 + \frac{1}{N} \sum_{i=1} T_i^2 \\ &= \frac{1}{N} \sum_{i=1}^{N} T_i^2 - \overline{T} \end{aligned} \tag{4.20}$$

The simplest way to evaluate the second integral I_2 is to use the momentum exchange method as stated in section 3.5.

4.4 Discrete dynamical system - adjoint for LBM

To utilize the adjoint equations derived so far for the discrete dynamical system we have to consider specific features of the LBM. Let us remind the dual equation 4.17, which has to be adopted for $\mathbb{C}$ and $\mathbb{S}$ operators:

$$\begin{bmatrix} f_{n+1} \\ g_{n+1} \end{bmatrix} = H(f_n, \alpha_n)$$

From a general perspective, the H operator is responsible for moving the system to the next state (it performs both collision and streaming) and assessing its value g at the same time. In case of optimal mixing, the vector g_k consists of two components: I_1 - mix quality and I_2 - work done during the k-th time step to impose the fluid motion.
Now, lets make a step further and denote x as an index of the lattice node and H^x as a local operator (acting on a given node). The collision operator $\mathbb{C}$ will be incorporated into H^x since it also acts locally (and may behave differently on each node depending on the BC).

$$\begin{bmatrix} f_{n+1}(x) \\ g_{n+1}(x) \end{bmatrix} = H^x(\mathbb{S}(f_n)(x), \alpha_n(x)) \text{ for all } x \in L \tag{4.21}$$

The dual system can be formulated as:

$$\begin{bmatrix} v_{n-1}(x) \\ \beta_{n-1}(x) \end{bmatrix} = [dH^x]^T \begin{bmatrix} (\mathbb{S}^T v_n)(x) \\ h_n(x) \end{bmatrix} \tag{4.22}$$

As defined in equation 4.15, the objective function is:

$$J = \mathbf{h} \cdot \boldsymbol{g} = \sum_{n=1}^{N} \sum_{x \in L} h_n(x) \cdot g_n(x)$$

Then, the differential of the objective function is expressed as [12]:

$$\frac{d}{ds} J = \sum_{n=1}^{N} h_n \cdot \frac{\partial g_n}{\partial s} = v_0 \cdot \mathbb{S} \frac{\partial f_0}{\partial s} + \sum_{n=0}^{N-1} \beta_n \cdot \frac{\partial \alpha_n}{\partial s} \tag{4.23}$$

4.5 Automatic Differentiation (AD)

To compute the derivatives the automatic differentiation is used. Automatic differentiation is not:

- Symbolic differentiation
- Numerical differentiation (the method of finite differences)

The AD can be realized in two ways, which are based on the interaction with the computer code.

4.5.1 Operator overloading (OO)

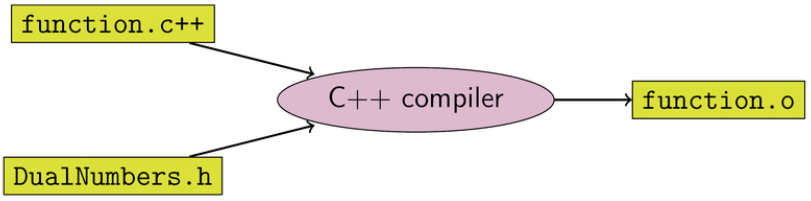


FIGURE 4.4: Operator overloading - idea

Operator overloading can be realized in object-oriented programming languages supporting it. The key is to change the basic data types and overload arithmetic operators to handle additional informations stored in the derived types. This method is relatively easy to implement. Next, the library containing derived data types can be imported and the variables of interest are ready to be substituted with the overloaded ones. This requires no change in the form or sequence of operations in the function to be differentiated. The nature of the computational process is similar to symbolic differentiation, the key is that the derivatives are evaluated 'on the fly' during the program flow and are stored as a numerical values, not a symbolic expressions. The drawback of this method is caused by compiler lags in code optimization and computational overhead associated with additional layer of abstraction.

4.5.2 Source code transformation (SCT)

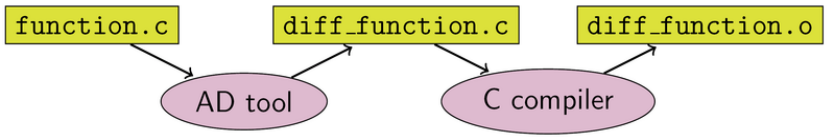


FIGURE 4.5: Source code transformation - idea

As the name indicates the source function is replaced by an automatically generated one which is able to calculate the derivative along with the original instructions.
SCT can be done with all programming languages and does not suffer from the run-time computational overhead caused by additional abstraction layer introduced in OO. Unfortunately software for SCT is hard to develop and may be incapable to handle code which is characterized by high level of abstraction, directives (like MPI) and complex interactions between classes. In other words, architecture of the program may limit the SCT to small local procedures.
In this work, the SCT is used to generate small local procedures which allows the code to be both efficient and parallel.

Chapter 5

Results

5.1 Programming and implementation

The calculations has been done by the in-house solver programmed by the author, but mostly on the open source TCLB solver developed at the faculty (mainly by Łaniewski-Wołłk, Dzikowski, et al. [15]). The TCLB solver is highly parallel and tuned to run on multi-GPU (nVidia) architectures. High performance is achieved due to special code generation techiques. To avoid overhead comming from handling with high level programming (polymorphic, etc.), the code is generated for each model (like optimal mixing on a D2Q9 lattice) separately. The model is described by a set of tables defining [14]:

- `Densities` - includes densities and streaming vectors - e.g., $\boldsymbol{e}_i$, f_i
- `Settings` - includes global setting like fluid viscosity v
- `Globals` - includes global integrals like total work performed by the moving lid
- `Quantities` - includes macroscoping quantities that are exported by the model e.g., u, ρ, T

To generate the code from this description the in-house, open source, template tool called R-Template is used [13]. It is capable of expanding the results (as a C code) of symbolic matrix multiplication. It originates from the R programming language[23]. The Automatic Differentiation is achieved using tool TAPENADE [6], which performs source code transformation.

Knowing the gradient, the optimization routine (method of moving asymptotes) is imported from NLopt [17], which is an open-source library for non-linear optimization.

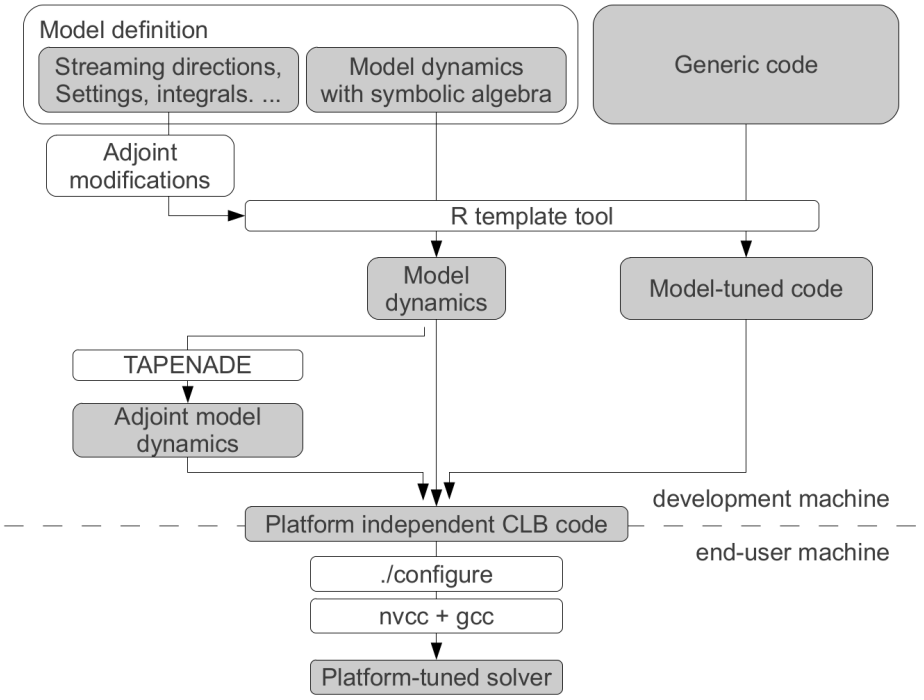


FIGURE 5.1: Code generation and compilation in TCLB [14]

Most of the simulations were performed on the PROMETHEUS cluster at the Cyfronet computing center (`https://kdm.cyfronet.pl/portal/Prometheus`).
More informations concerning parallel implementations of LBM for HPC may be found in [21, 20].

5.2 Reference cases

In order to obtain some reference data, to which the optimal solution can be compared, following cases were calculated:

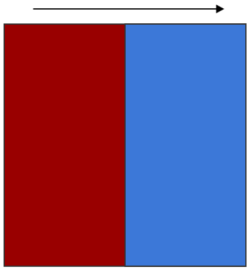


FIGURE 5.2: Initial conditions for the passive scalar distribution, fluid is at rest

Setup:

- Reference control function: $U_{lid} = u_0\, sin(t)$
- Lattice size (with walls): 128x128
- Lattice fluid viscosity : $\nu = 0.1$
- Lattice fluid thermal diffusivity: $\lambda = 0.005$
- passive scalar intensity: half domain $\pm 1 \Rightarrow$ average $\overline{T} = 0$

The mixture quality is defined in the equation 4.20.
Initially, for the fluid domain of size 126x127 (lid is included) it gives:

$$I_1 = \frac{1}{N}\sum_{i=1}^{N} T_i^2 - \overline{T} = \frac{1}{126 \cdot 127}\sum_{i=1}^{126\cdot 127} (\pm 1)^2 - 0 = 1$$

To find the range of Reynold's numbers in a flow driven by a variable input, the maximum allowed lattice velocity (constrain imposed by the model usability, see 2.2.7.1) is chosen as an upper limiting value:

$$Re_{max} = \frac{U_{lid}\, L}{\nu} = \frac{0.1 \cdot 126}{0.1} = 126$$

Thus $Re \in (0, 126)$.

$U_{lid} = u_0\, sin(t)$			Objective			
u_0	$t \in$		50k time steps		100k time steps	
			mix quality	work	mix quality	work
0	————		0.593	—	0.432	—
0.025	$0 \div 2\pi$		0.241	599	0.045	1198
0.025	$0 \div 4\pi$		0.388	597	0.107	1197
0.050	$0 \div 2\pi$		0.089	2393	0.013	4787
0.050	$0 \div 4\pi$		0.167	2389	0.032	4785
0.075	$0 \div 2\pi$		0.051	5374	0.005	10753
0.075	$0 \div 4\pi$		0.108	5367	0.015	10748
0.1	$0 \div 2\pi$		0.032	9536	0.002	19076
0.1	$0 \div 4\pi$		0.074	9535	0.007	19071
0.1	const		0.011	19046	0.001	38102

TABLE 5.1: Reference data

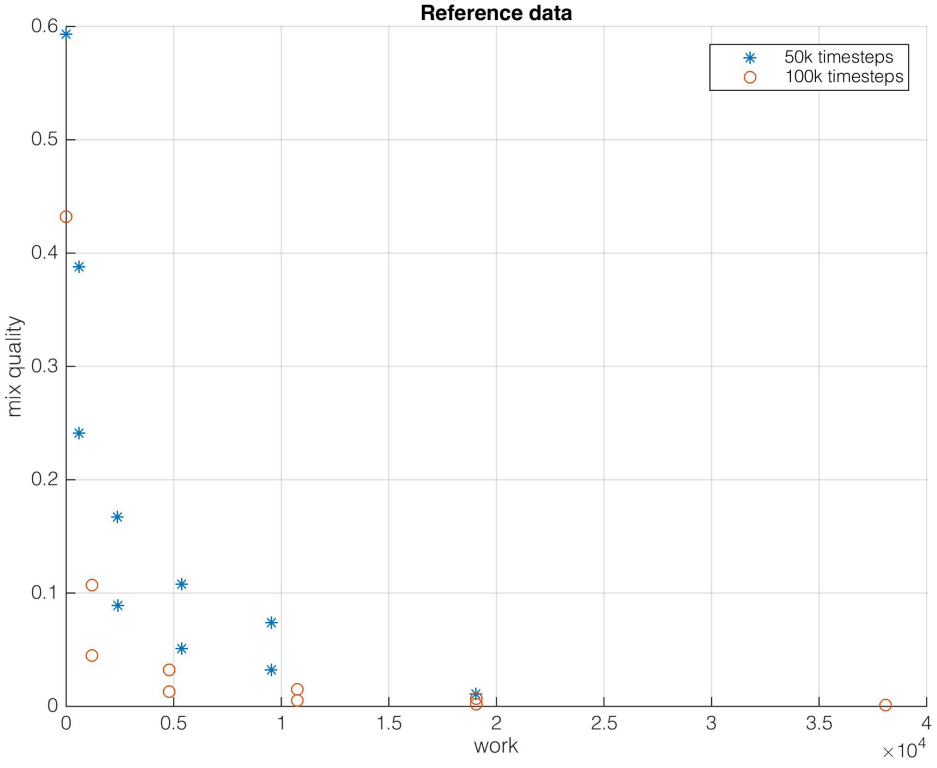


FIGURE 5.3: Scatter plot of the reference data

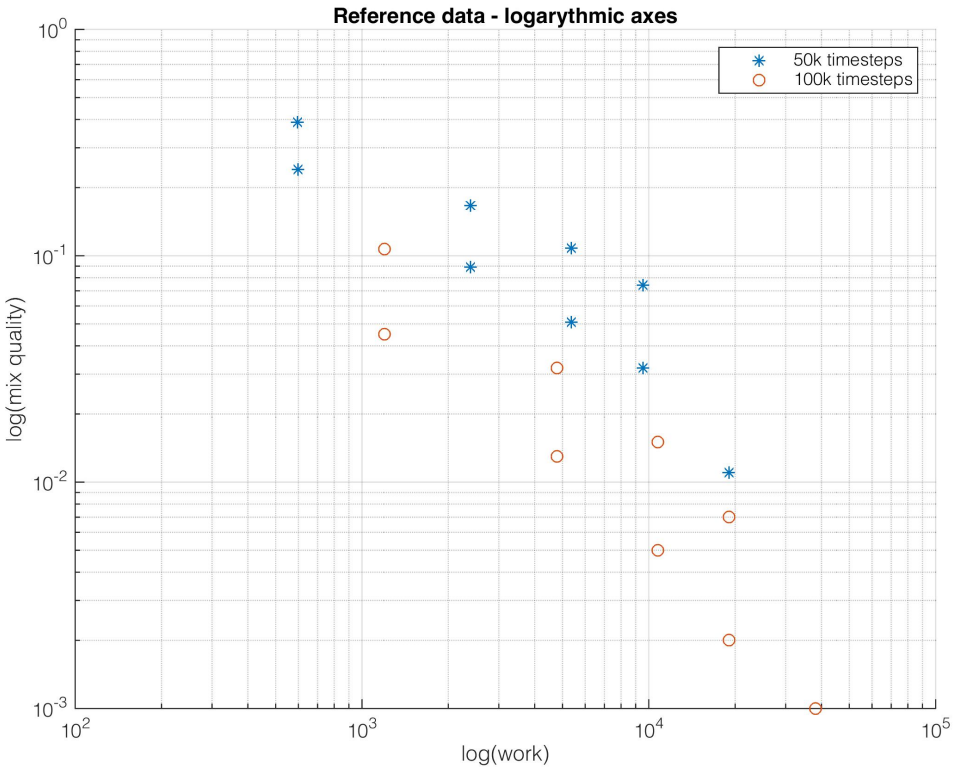


FIGURE 5.4: Scatter plot of the reference data with logarithmic axes

5.2.1 Selected case study

Conditions:

$U_{lid} = 0.1sin(t)$ where: $t \in (0 \div 2\pi)$, 50k time steps

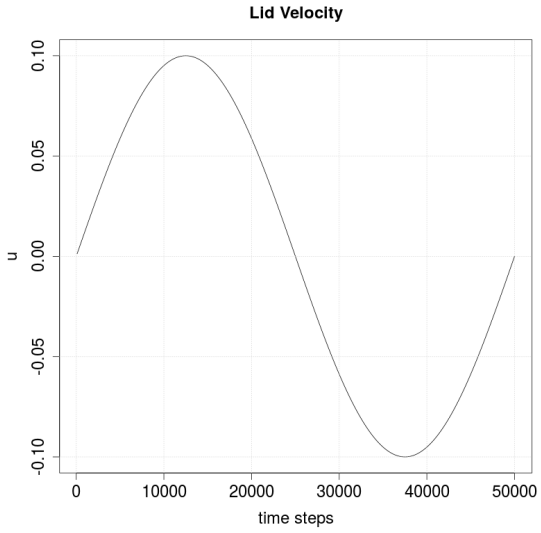


FIGURE 5.5: Lid Velocity

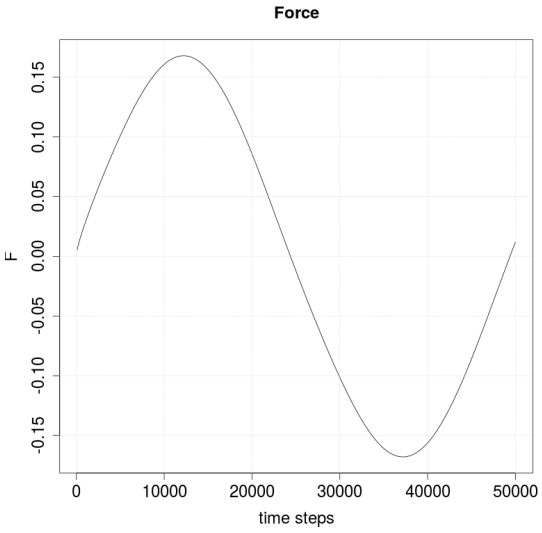


FIGURE 5.6: Force

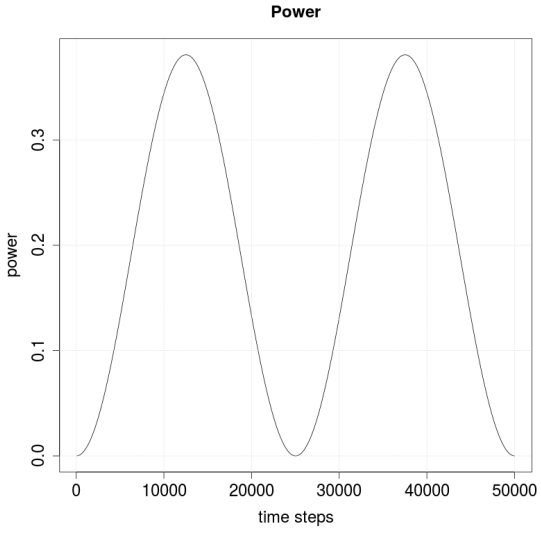


FIGURE 5.7: Power

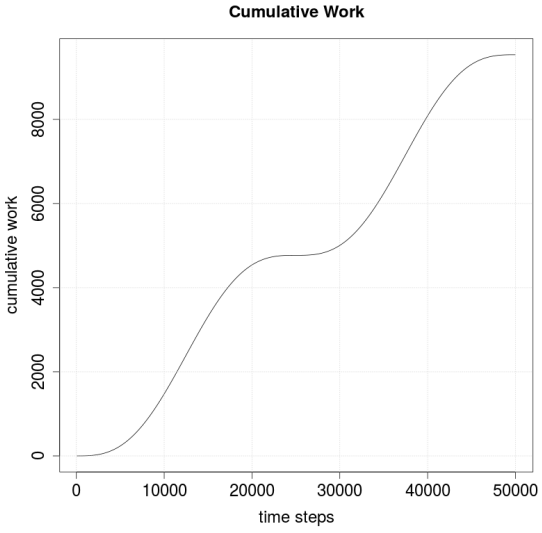


FIGURE 5.8: Cumulative Work

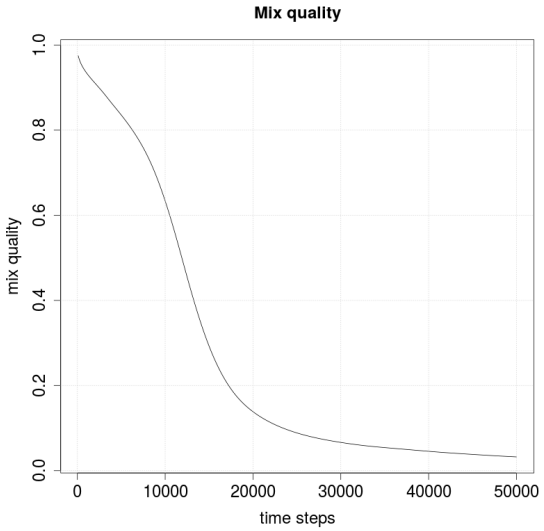
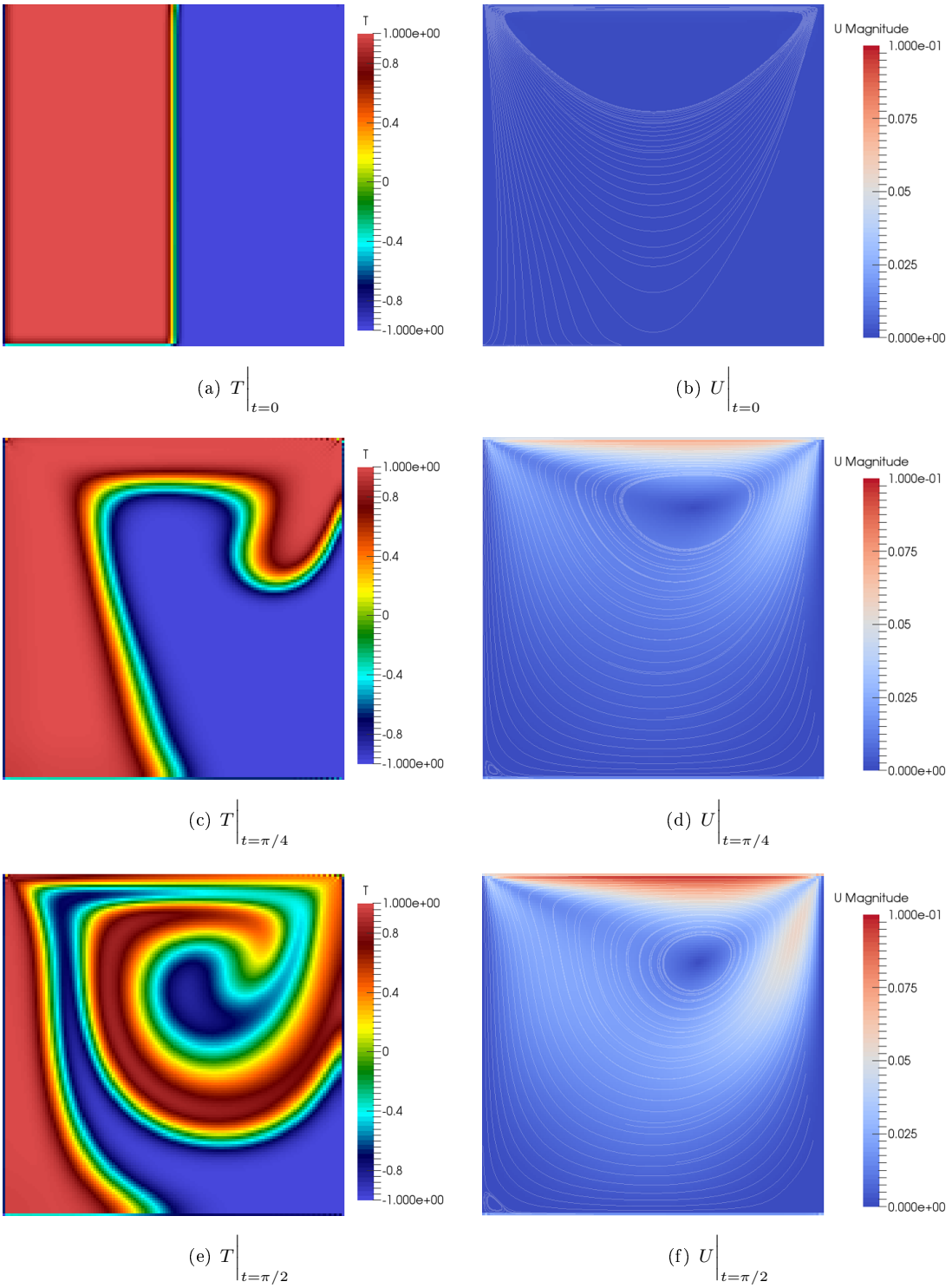


FIGURE 5.9: Mix quality

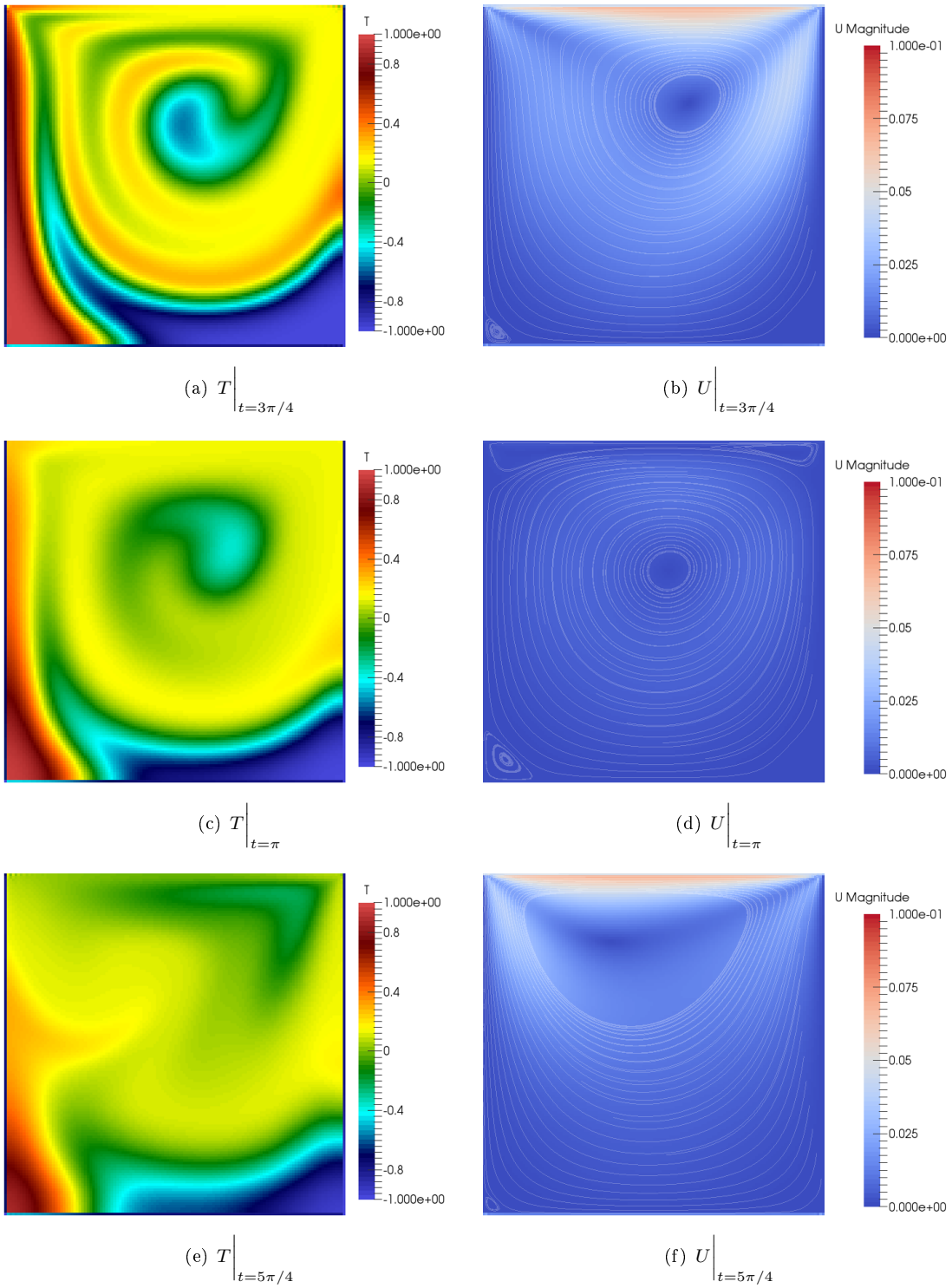


(a) $T\Big|_{t=0}$ (b) $U\Big|_{t=0}$

(c) $T\Big|_{t=\pi/4}$ (d) $U\Big|_{t=\pi/4}$

(e) $T\Big|_{t=\pi/2}$ (f) $U\Big|_{t=\pi/2}$

FIGURE 5.10: History of passive scalar (left) and velocity (right) fields part I

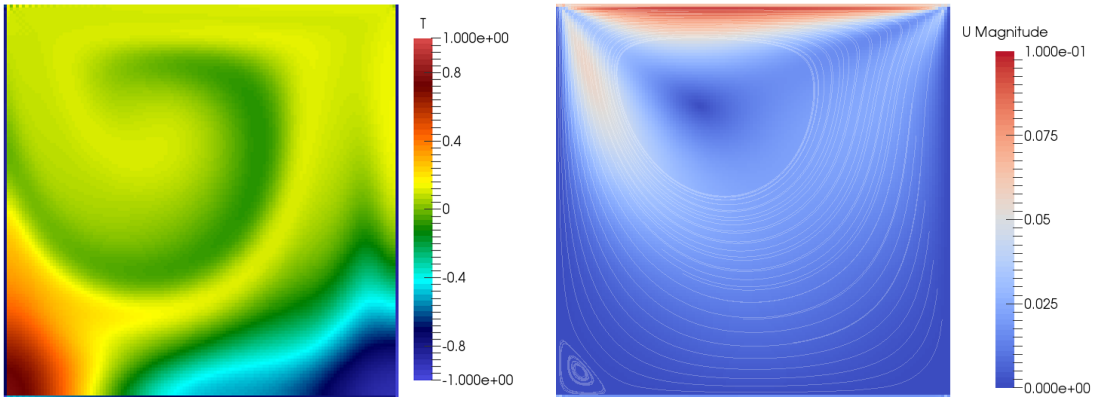


(a) $T\Big|_{t=3\pi/4}$ (b) $U\Big|_{t=3\pi/4}$

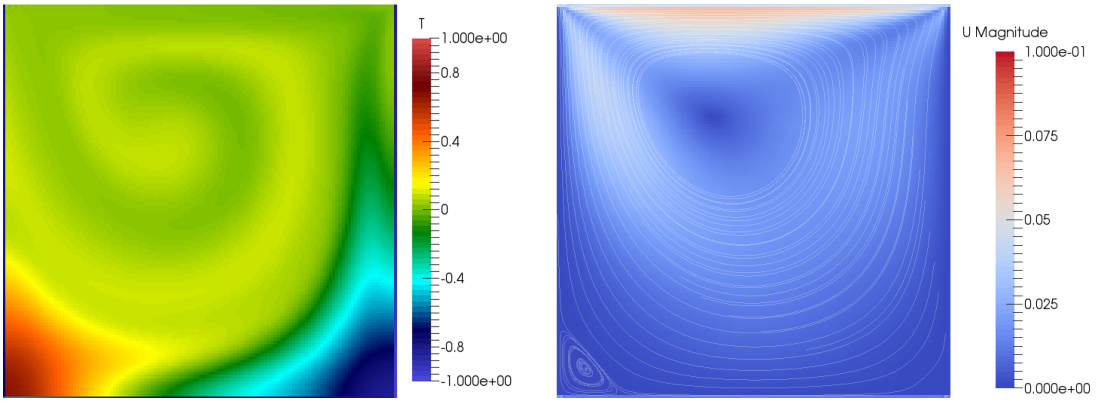
(c) $T\Big|_{t=\pi}$ (d) $U\Big|_{t=\pi}$

(e) $T\Big|_{t=5\pi/4}$ (f) $U\Big|_{t=5\pi/4}$

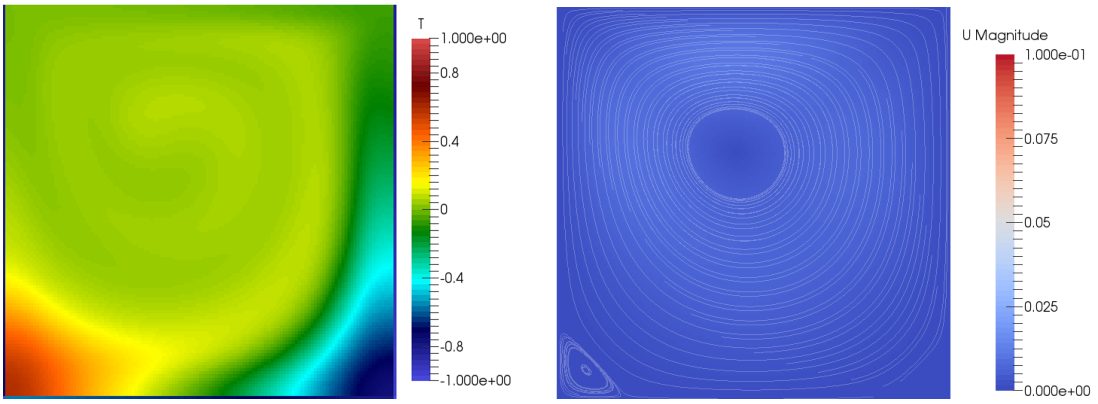
FIGURE 5.11: History of passive scalar (left) and velocity (right) fields part II



(a) $T\Big|_{t=3\pi/2}$ (b) $U\Big|_{t=3\pi/2}$



(c) $T\Big|_{t=7\pi/4}$ (d) $U\Big|_{t=7\pi/4}$



(e) $T\Big|_{t=2\pi}$ (f) $U\Big|_{t=2\pi}$

FIGURE 5.12: History of passive scalar (left) and velocity (right) fields part III

5.3 Optimization results

5.3.1 Selected case study

For the same conditions as in previous section 5.2.1, the function controlling lid velocity U_{lid} has been optimized. To regulate the influence of work and mix quality in the objective, the work is multiplied by a coefficient $\varepsilon = 1E-5$. It has been experimentally checked that this value of ε corresponds to almost equal weight of I_1 and I_2 in the final objective J:

$$J = \underbrace{I_1}_{mix\,quality} + \varepsilon \underbrace{I_2}_{work}$$

Let us remind the initial conditions: $U_{lid} = 0.1sin(t)$ where: $t \in (0 \div 2\pi)$, 50k time steps.

5.3.1.1 Objective function in subsequent iterations

The amount of computations is high, thus the number of optimization cycles is set a priori to 20. Fortunately, the figure 5.13 shows that the optimal value of objective is almost constant after 20 cycles. The solution is obtained based on a gradient algorithm, thus it can not be guaranteed that the global minimum has been found.

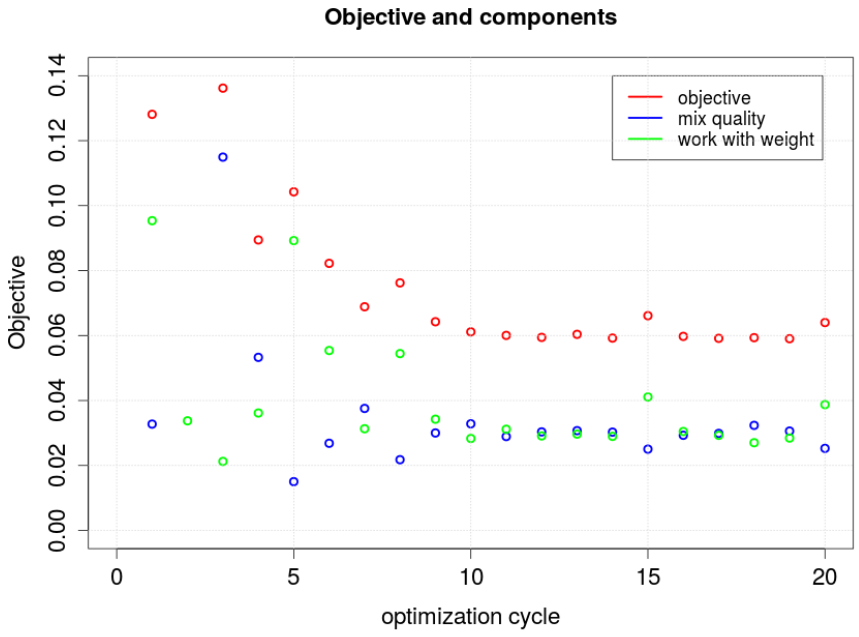


FIGURE 5.13: Objective and its components
- final values from each optimization cycle

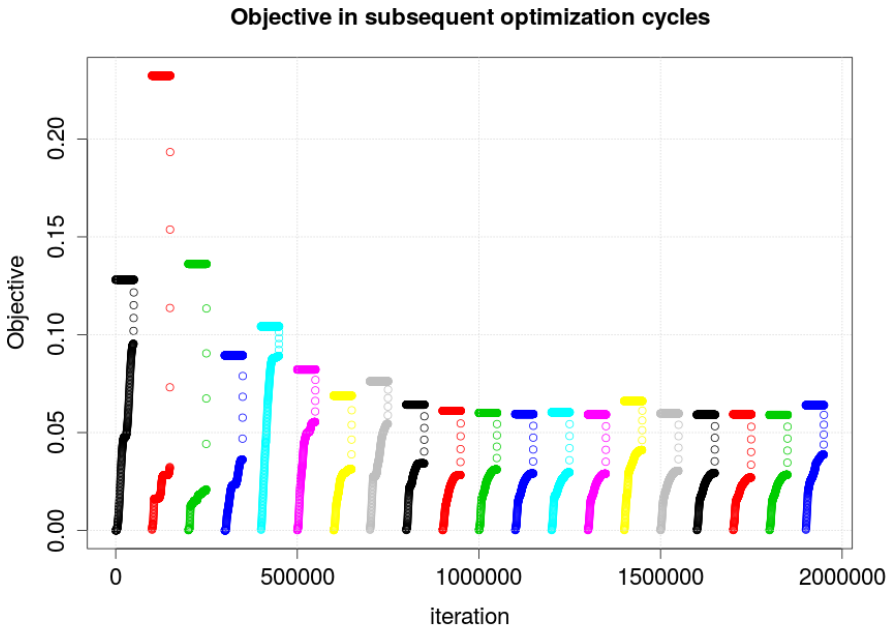
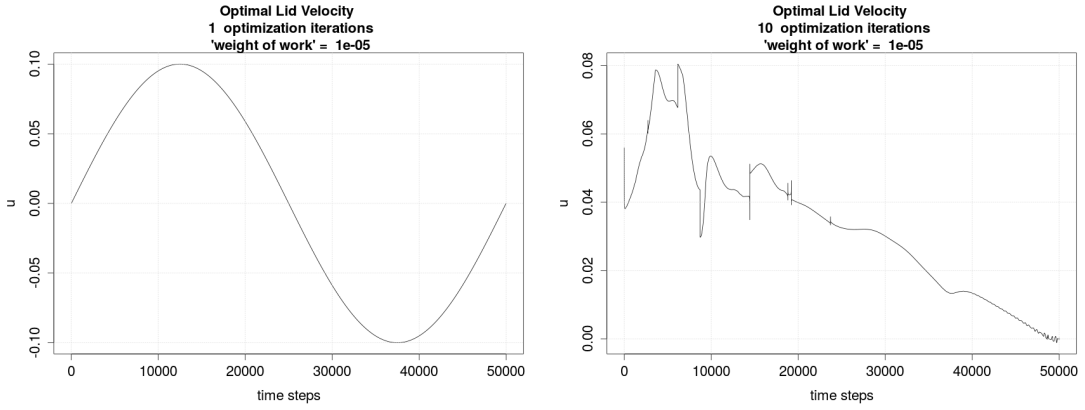


FIGURE 5.14: Objective, the work is taken into account with weight ε = 1E-5

Figure 5.14 illustrates the evolution of the objective in each iteration thorough 20 optimization cycles. Each bar represent another optimization cycle. For a given bar there is 50k steps forward and the same amount backward (segment with constant value at the figure) in the adjoint mode, together 100k iterations per one optimization cycle.

5.3.1.2 Optimal Lid velocity in subsequent iterations

The maximum value of the lid velocity is constrained to ± 0.1 since it is the limit of applicability of LBM (for higher the method does not converge to Navier-Stokes equations). To limit the numerical oscillations at the end of the control the mix quality from last 2% of time domain has been taken into the objective. Further smoothing would require additional technique like averaging the gradient between neighbouring time steps.



(a) $U_{Lid}\Big|_{n=1}$ - initial control

(b) $U_{Lid}\Big|_{n=10}$

(c) $U_{Lid}\Big|_{n=20}$

(d) $U_{Lid}\Big|_{n\in(1,2,3\ldots20)}$

FIGURE 5.15: History of U_{Lid} optimization

To summarize the 'Selected case study', the control velocity given by 5.15(c) shall be applied in order to realize the optimal mixing is this particular case.

5.4 Pareto Frontier

In multi - objective optimization the optimal decision has to be taken in presence of trade-offs between conflicting objectives. For a given system, the Pareto frontier is the set of optimal solutions being equally good. It is impossible to make any of the sub-objectives better without making at least one worse.

To investigate the influence of mix quality and work on the objective function a series of additional optimization experiments was conducted.

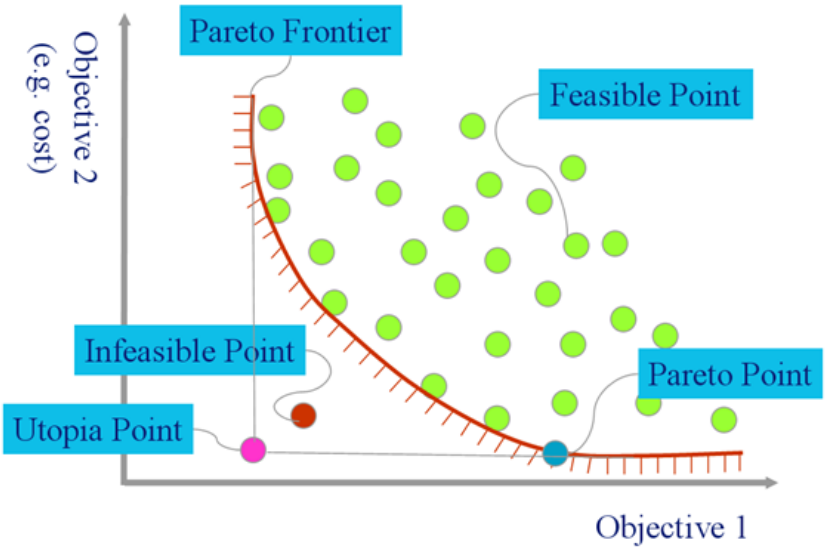
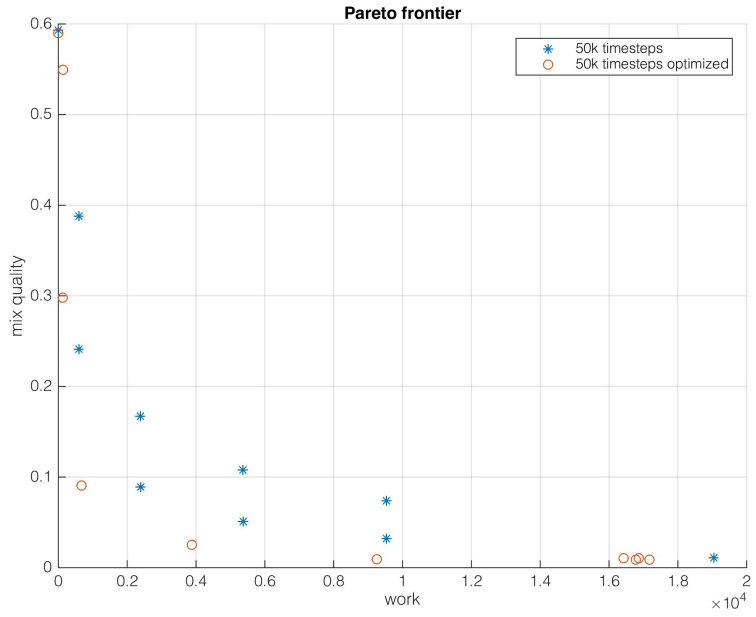


FIGURE 5.16: Pareto frontier [1].

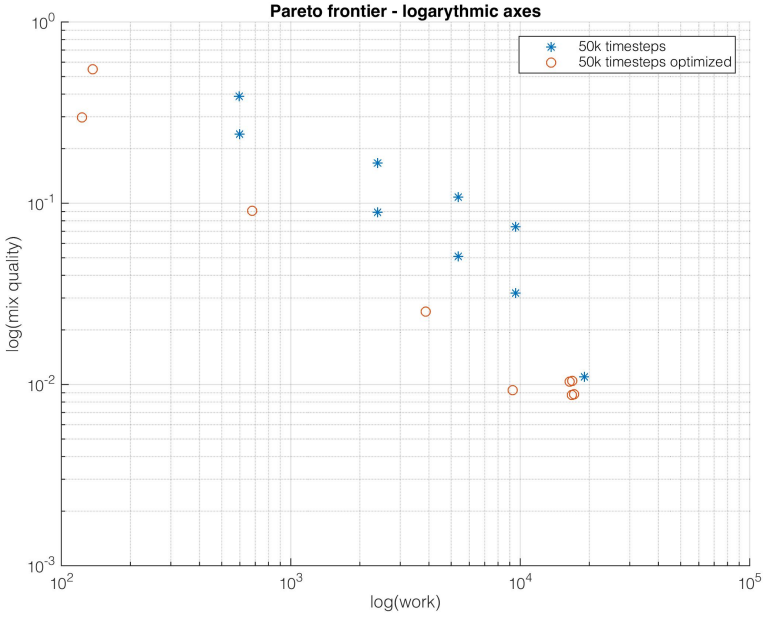
The table 5.2 shows the results of running the optimization procedure with different work weights ε.

$U_{lid} = u_0\, sin(t)$			**Objective**	
u_0	$t \in$		50k time steps	
			mix quality	work
0	————		0.593	—
0.025	$0 \div 2\pi$		0.241	599
0.025	$0 \div 4\pi$		0.388	597
0.050	$0 \div 2\pi$		0.089	2393
0.050	$0 \div 4\pi$		0.167	2389
0.075	$0 \div 2\pi$		0.051	5374
0.075	$0 \div 4\pi$		0.108	5367
0.1	$0 \div 2\pi$		0.032	9536
0.1	$0 \div 4\pi$		0.074	9535
0.1	const		0.011	19046
After optimization IC: $U_{lid} = 0.1\, sin(t)\, t \in [0 \div 2\pi]$				
ε (work weight)			mix quality	work
0			0.009	17173
1E-9			0.009	16781
1E-8			0.010	16426
1E-7			0.010	16861
1E-6			0.009	9256
1E-5			0.025	3873
1E-4			0.091	678
1E-3			0.298	123
1			0.550	137
1E+3			0.590	1

TABLE 5.2: Optimization results



(a) normal axes



(b) logarithmic axes

FIGURE 5.17: Scatter plot of the optimization results

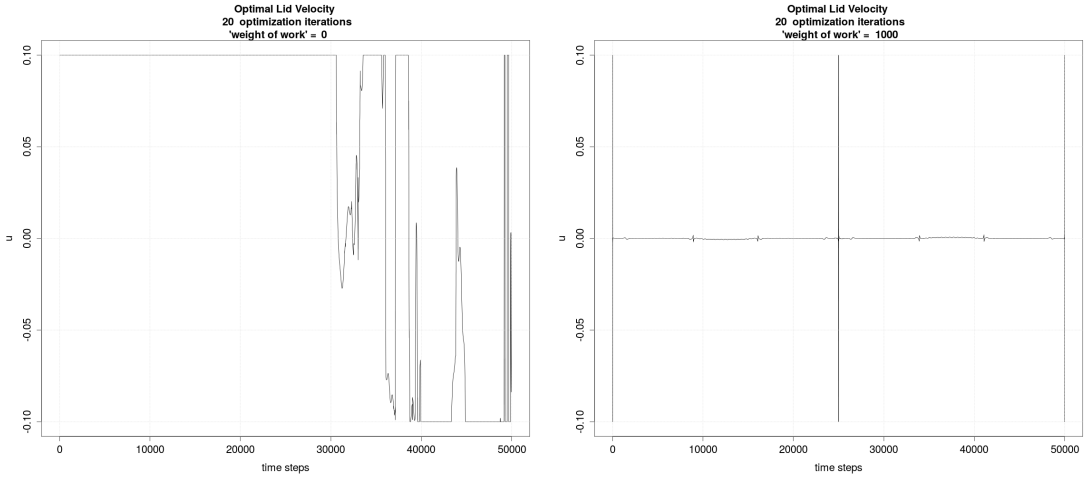
Figures 5.17(b) and 5.17(a) show that the asymptotes of the Pareto friontier are situated below the reference values, therefore we can conclude that the optimization is successful.

5.4.0.1 Behaviour of control velocity for extreme values of weight factor ε.

Below, control velocity for asymptotes of Parento froniter are shown.

The figure 5.18(a) corresponds to control which does not care for the amount of work required to impose the upper wall movement. This leads to behaviour similar to 'full ahead - full astern' also known as 'bang-bang' control known from Minimum Time Control theory.

On the other hand, the figure 5.18(b) represents extremely high price of work which results in diffusive mixing without fluid motion. Both of these 'extreme controls' agrees with engineering intuition.



(a) $U_{Lid}\Big|_{\varepsilon=0}$ (b) $U_{Lid}\Big|_{\varepsilon=1E+3}$

FIGURE 5.18: History of U_{Lid} optimization for extreme values of ε.

Vertical peaks can be explained by the numerical discrepancies occurring in the gradient optimization scheme. Apart from gradient averaging technique, an additional penalty term associated with rapid acceleration could be added to objective function in order to smooth the control.

Chapter 6

Summary

The general framework concerning adjoint method for discrete dynamical system has been presented. Specific features of the LBM such as locality of the collision operator and reversibility of streaming allows to utilize the adjoint concept in a parallel manner. Moreover, the same framework can be used to solve even more complex problems like topology optimization.

As mentioned before, the optimal control solution is obtained based on a gradient algorithm, thus it can not be guaranteed that the global minimum has been found. We did not investigate the influence of starting point for the optimization algorithm (the initial control) neither its stability. Finally, the main goal of providing a solution for a fluid optimal control problem has been achieved.

As a benchmark, we have solved problem of mixing in a lid driven cavity. The objective function is composed from conflictive components being highest quality at minimal work. This leads to the set of solutions know in a form of Pareto Frontier. We have shown that that the value of objective has been successfully minimized.

Current state of the work:

The TCLB solver is under development at the Division of Aerodynamics of the Faculty of Power and Aeronautical Engineering, Warsaw University of Technology as an open source project. It can be found and downloaded from:

`https://github.com/CFD-GO/TCLB/`

Appendix A

Adjoint - an example of a discrete dynamical system

To better understand the formulas, let us consider a simple example of a cannon ball shot. Knowing position of the cannon and the target, the goal is to hit it. The control variables are initial velocity and an adjustable drag coefficient.

Equation of motion:

$$m\ddot{\boldsymbol{x}} = C_t\dot{\boldsymbol{x}}^2 + m\boldsymbol{g} \tag{A.1}$$

$$F_d = \underbrace{\frac{1}{2}\rho S C_D}_{C_t}\dot{\boldsymbol{x}}^2 \tag{A.2}$$

Since the drag coefficient is adjustable (like a parachute) we can write it as function of a control parameter $C_t = C_t(\alpha)$. In a similar manner an afterburner for rocket engine can be inserted and controlled.

Acceleration:

$$\ddot{\boldsymbol{x}} = \begin{bmatrix}\ddot{x}\\ \ddot{y}\end{bmatrix} = \begin{bmatrix}\dfrac{C_t(\alpha)}{m}(\dot{x}_n^2+\dot{y}_n^2)\dfrac{-\dot{x}}{\sqrt{\dot{x}_n^2+\dot{y}_n^2}}\\ \dfrac{C_t(\alpha)}{m}(\dot{x}_n^2+\dot{y}_n^2)\dfrac{-\dot{y}}{\sqrt{\dot{x}_n^2+\dot{y}_n^2}} - g\end{bmatrix} \tag{A.3}$$

For the seek of simplicity we will use Euler discretization scheme.

Velocity:

$$\dot{\boldsymbol{x}} = \begin{bmatrix} \dot{x}_{n+1} \\ \dot{y}_{n+1} \end{bmatrix} = \begin{bmatrix} \dot{x}_n + \dfrac{C_t(\alpha)}{m}\sqrt{\dot{x}_n^2 + \dot{y}_n^2}(-\dot{x}_n)\Delta t \\ \dot{y}_n + \dfrac{C_t(\alpha)}{m}\sqrt{\dot{x}_n^2 + \dot{y}_n^2}(-\dot{x}_n)\Delta t - g\Delta t \end{bmatrix} \tag{A.4}$$

Position:

$$\boldsymbol{x} = \begin{bmatrix} x_{n+1} \\ y_{n+1} \end{bmatrix} = \begin{bmatrix} x_n + \dot{x}_n\Delta t \\ y_n + \dot{y}_n\Delta t \end{bmatrix} \tag{A.5}$$

Objective function:

$$J = \boldsymbol{h}\cdot\boldsymbol{g} = \sum_{n=1}^{N} h_n\, g_n = \sum_{n=1}^{N}\left(h_{x_n}\underbrace{\sqrt{(x_n - x_N)^2}}_{L_{xn}} + h_{y_n}\underbrace{\sqrt{(y_n - y_N)^2}}_{L_{yn}}\right) \tag{A.6}$$

Where the weights vector $h_k = \begin{bmatrix} h_{x_k} \\ h_{y_k} \end{bmatrix} = \left[\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \end{bmatrix}, ..., \begin{bmatrix} 1 \\ 1 \end{bmatrix}\right]$, because only the final position of the ball matters.

The g_k is a vector containing the distance in the x and y direction from the target at the k-th time step: $g_k = \begin{bmatrix} L_{xk} \\ L_{yk} \end{bmatrix}$

The primal $\rightarrow$ adjoint formulation:

$$\begin{bmatrix} u_{n+1} \\ g_{n+1} \end{bmatrix} = H(u_n, \alpha_n) \xrightarrow{adjoint} \begin{bmatrix} v_{n-1} \\ \beta_{n-1} \end{bmatrix} = [dH]^T \begin{bmatrix} v_n \\ h_n \end{bmatrix}$$

Operator H:

$$\begin{bmatrix} u_n \begin{bmatrix} \dot{x} \\ \dot{y} \\ x \\ y \end{bmatrix} \\ \alpha_n \end{bmatrix}_{5\times 1} \xrightarrow[n=n+1]{H} \begin{bmatrix} u_{n+1} \begin{bmatrix} \dot{x} \\ \dot{y} \\ x \\ y \end{bmatrix} \\ g_{n+1} \begin{bmatrix} L_x \\ L_y \end{bmatrix} \end{bmatrix}_{6\times 1} \tag{A.7}$$

Jacobian - dH:

$$dH = \begin{bmatrix} \frac{\dot{x}_{n+1}}{\dot{x}_n} & \frac{\dot{x}_{n+1}}{\dot{y}_n} & \frac{\dot{x}_{n+1}}{x_n} & \frac{\dot{x}_{n+1}}{y_n} & \frac{\dot{x}_{n+1}}{\alpha_n} \\ \frac{\dot{y}_{n+1}}{\dot{x}_n} & \cdots & & & \frac{\dot{y}_{n+1}}{\alpha_n} \\ \frac{x_{n+1}}{\dot{x}_n} & \cdots & & & \vdots \\ \frac{y_{n+1}}{\dot{x}_n} & \cdots & & & \vdots \\ \frac{L_{xn+1}}{\dot{x}_n} & \cdots & & & \vdots \\ \frac{L_{yn+1}}{\dot{x}_n} & \cdots & & & \frac{L_{yn+1}}{\alpha_n} \end{bmatrix}_{6\times 5} \tag{A.8}$$

Appendix B

Adjoint equation for discrete dynamical system - Proof

The proof of theorem 4.1 comes from the work of Łaniewski-Wołłk [12].

Proof. Let us differentiate the dynamical system 4.14 with respect to s:

$$\begin{bmatrix} u_{n+1} \\ g_{n+1} \end{bmatrix} = H(u_n, \alpha_n) \xrightarrow{\frac{d}{ds}} \begin{bmatrix} \frac{\partial u_{n+1}}{\partial s} \\ \frac{\partial g_{n+1}}{\partial s} \end{bmatrix} = [dH] \begin{bmatrix} \frac{\partial u_n}{\partial s} \\ \frac{\partial \alpha_n}{\partial s} \end{bmatrix}$$

Multiplying by $\begin{bmatrix} v_{n+1} \\ h_{n+1} \end{bmatrix}$ and using 4.17:

$$\begin{aligned} \begin{bmatrix} v_{n+1} \\ h_{n+1} \end{bmatrix}^T \begin{bmatrix} \frac{\partial u_{n+1}}{\partial s} \\ \frac{\partial g_{n+1}}{\partial s} \end{bmatrix} &= \begin{bmatrix} v_{n+1} \\ h_{n+1} \end{bmatrix}^T [dH] \begin{bmatrix} \frac{\partial u_n}{\partial s} \\ \frac{\partial \alpha_n}{\partial s} \end{bmatrix} \\ &= \left([dH]^T \begin{bmatrix} v_{n+1} \\ h_{n+1} \end{bmatrix} \right)^T \begin{bmatrix} \frac{\partial u_n}{\partial s} \\ \frac{\partial \alpha_n}{\partial s} \end{bmatrix} \\ &= \begin{bmatrix} v_n \\ \beta_n \end{bmatrix}^T \begin{bmatrix} \frac{\partial u_n}{\partial s} \\ \frac{\partial \alpha_n}{\partial s} \end{bmatrix} \\ v_{n+1} \cdot \frac{\partial u_{n+1}}{\partial s} + h_{n+1} \cdot \frac{\partial g_{n+1}}{\partial s} &= v_n \cdot \frac{\partial u_n}{\partial s} + \beta_n \cdot \frac{\partial \alpha_n}{\partial s} \end{aligned}$$

Taking the sum of both sides from n = 0,...,N-1 gives:

$$\sum_{n=0}^{N-1} v_{n+1} \cdot \frac{\partial u_{n+1}}{\partial s} + \sum_{n=0}^{N-1} h_{n+1} \cdot \frac{\partial g_{n+1}}{\partial s} = \sum_{n=0}^{N-1} v_n \cdot \frac{\partial u_n}{\partial s} + \sum_{n=0}^{N-1} \beta_n \cdot \frac{\partial \alpha_n}{\partial s}$$

Renumeration and further transformation goes as follows:

$$\sum_{n=1}^{N} v_n \cdot \frac{\partial u_n}{\partial s} + \sum_{n=1}^{N} h_n \cdot \frac{\partial g_n}{\partial s} = \sum_{n=0}^{N-1} v_n \cdot \frac{\partial u_n}{\partial s} + \sum_{n=0}^{N-1} \beta_n \cdot \frac{\partial \alpha_n}{\partial s} \quad // - \sum_{n=1}^{N-1} v_n \cdot \frac{\partial u_n}{\partial s}$$

$$v_N \cdot \frac{\partial u_N}{\partial s} + \sum_{n=1}^{N} h_n \cdot \frac{\partial g_n}{\partial s} = v_0 \cdot \frac{\partial u_0}{\partial s} + \sum_{n=0}^{N-1} \beta_n \cdot \frac{\partial \alpha_n}{\partial s}$$

The last time slice is $N-1$ thus the term $\frac{\partial u_N}{\partial s}$ is unknown. This leads to the assumption of $v_N = 0$, which allows us to get rid of the unknown term and finish the proof:

$$\frac{d}{ds} J = \sum_{n=1}^{N} h_n \cdot \frac{\partial g_n}{\partial s} = v_0 \cdot \frac{\partial u_0}{\partial s} + \sum_{n=0}^{N-1} \beta_n \cdot \frac{\partial \alpha_n}{\partial s}$$

□

Remarks regarding $\frac{d}{ds}J$:

Let us analyse special cases of the general formula 4.18:

$$\frac{d}{ds}J = v_0 \cdot \frac{\partial u_0}{\partial s} + \sum_{n=0}^{N-1} \beta_n \cdot \frac{\partial \alpha_n}{\partial s}$$

Case I What should be the α_k at the particular k-th step?

$\Longrightarrow s = s_k = \alpha_k$

$$\frac{d}{ds}J = v_0 \underbrace{\frac{\partial u_0}{\partial s}}_{=0} + \sum_{n=0}^{N-1} \beta_n \underbrace{\frac{\partial \alpha_n}{\partial s}}_{=\delta_{nk}} = \beta_n \qquad - \textit{Interpretation of } \beta$$

Case II: What should be the α for each step?

$\Longrightarrow \mathbf{s} = [s_0 = \alpha_0,\ s_1 = \alpha_1,\ ...,\ s_{N-1} = \alpha_{N-1}]$

$$\frac{d}{ds}J = v_0 \underbrace{\frac{\partial u_0}{\partial \mathbf{s}}}_{=0} + \sum_{n=0}^{N-1} \beta_n \frac{\partial \alpha_n}{\partial \mathbf{s}} = \boldsymbol{\beta} = [\beta_0, \beta_1, ..., \beta_{N-1}]$$

Case III Same $\alpha = const$ during all steps

$\Longrightarrow s = const = \alpha_k$ for k = 0, 1, ..., N-1

$$\frac{d}{ds}J = v_0 \underbrace{\frac{\partial u_0}{\partial s}}_{=0} + \sum_{n=0}^{N-1} \beta_n \underbrace{\frac{\partial \alpha_n}{\partial s}}_{=1} = \sum_{n=0}^{N-1} \beta_n$$

Case IV Initial conditions

$\Longrightarrow s = u_0$

$$\frac{d}{ds}J = v_0 \underbrace{\frac{\partial u_0}{\partial s}}_{=1} + \sum_{n=0}^{N-1} \beta_n \underbrace{\frac{\partial \alpha_n}{\partial s}}_{=0} = v_0$$

Case V Let $\alpha = \alpha(n)$ be a polynomial depending on iteration number 'n' with constant coefficients 's'. Find the optimal value of coefficients.

$\Longrightarrow$ for example: $\alpha(n) = s_0 + s_1 n + s_2 n^2$

$$\begin{cases} \frac{d}{ds_0} J = v_0 \underbrace{\frac{\partial u_0}{\partial s_0}}_{=0} + \sum_{n=0}^{N-1} \beta_n \underbrace{\frac{\partial \alpha_n}{\partial s_0}}_{=1} = \sum_{n=0}^{N-1} \beta_n \\ \\ \frac{d}{ds_1} J = v_0 \underbrace{\frac{\partial u_0}{\partial s_0}}_{=0} + \sum_{n=0}^{N-1} \beta_n \underbrace{\frac{\partial \alpha_n}{\partial s_0}}_{=n} = \sum_{n=0}^{N-1} n\beta_n \\ \\ \frac{d}{ds_1} J = v_0 \underbrace{\frac{\partial u_0}{\partial s_0}}_{=0} + \sum_{n=0}^{N-1} \beta_n \underbrace{\frac{\partial \alpha_n}{\partial s_0}}_{=n^2} = \sum_{n=0}^{N-1} n^2\beta_n \end{cases}$$

Bibliography

[1] Noesis Solutions. URL: http://www.noesissolutions.com/Noesis/design-optimization/optimize/multi-objective-optimization.

[2] Y. B. Bao and J. Meskas. *Lattice Boltzmann Method for Fluid Simulations*. Tech. rep. Department of Mathematics, Courant Institute of Mathematical Sciences, 2011. URL: http://www.cims.nyu.edu/~billbao/report930.pdf.

[3] C. Cercignani. *The Boltzmann Equation and Its Application*. Springer, 1988.

[4] Dieter A. Wolf Gladrow. *Lattice Gas Cellular Automata and Lattice Boltzmann Models*. Springer, 2000.

[5] H. and B. S. Jeffreys. *Methods of Mathematical Physics*. Cambridge University Press, 1956.

[6] L. Hascoet and V. Pascual. "The Tapenade Automatic Differentiation tool: principles, model, and specification". *ACM Transactions on Mathematical Software, Association for Computing Machinery* (2013). URL: https://hal.inria.fr/hal-00913983.

[7] M. H. Hekmat and M. Mirzaei. "A comparison of the continuous and discrete adjoint approach extended based on the standard lattice Boltzmann method in flow field inverse optimization problems". *Springer-Verlag* Acta Mech 227.DOI 10.1007/s00707-015-1509-x (2015), pp. 1025–1050.

[8] M. H. Hekmat and M. Mirzaei. "Development od discrete adjoint approach based on the lattice boltzmann method." *Adwanced in Mechanical Engineering* 6 (2014), p. 230854. DOI: 10.1155/2014/230854. URL: http://ade.sagepub.com/content/6/230854.full.

[9] M. H. Hekmat and M. Mirzaei. "Extraction of macroscopic and microscopic adjoint concept using a lattice boltzmann method and discrete approach". *Physical Review E* 9.1 (2015), p. 013303.

[10] M. J. Krause and V. Heuveline. "Parallel Fluid Flow Control and Optimisation with Lattice Boltzmann Methods and Automatic Differentiation." *Computers & Fluids* 80 (2013), pp. 28–36.

[11] M. J. Krause, G. Thater, and V. Heuveline. "Adjoint-based fluid flow control and optimisation with lattice Boltzmann methods". *Computers & Mathematics with Applications* 65.6 (2013), pp. 945–960.

[12] Ł. Łaniewski-Wołłk. "Adjoint Lattice Boltzmann for Topology Optimalization and Optimal Control on multi-GPU Architecture". PhD thesis. Warsaw Universisty of Technology, Faculty of Power and Aeronautical Engineering, 2016 (in preparation).

[13] Ł. Łaniewski-Wołłk. *R-Template Tool*. 2014. URL: https://github.com/llaniewski/rtemplate.

[14] Ł. Łaniewski-Wołłk and J. Rokicki. "Adjoint Lattice Boltzmann for Topology Optimization on multi-GPU architecture". *Computers & Mathematics with Applications* 71.3 (2016), pp. 833–848.

[15] Ł. Łaniewski-Wołłk, M. Dzikowski, et al. *TCLB*. C-CFD Group at Warsaw University of Technology. 2012. URL: https://github.com/CFD-GO/TCLB.

[16] A. A. Mohamad. *Lattice Boltzmann Method Fundamentals and Engineering Applications with Computer Code*. Springer, 2011.

[17] *NLopt*. URL: http://ab-initio.mit.edu/wiki/index.php/NLopt.

[18] W. Regulski. "Simulation of flow past an obstacle in a two-dimensional channel with the use of the Lattice Boltzmann Method". MSc thesis. Warsaw Universisty of Technology, Faculty of Power and Aeronautical Engineering, 2010.

[19] W. Regulski and J. Szumbarski. "Numerical simulation of confined flows pas obstacles, the comparative study of Lattice Boltzmann ans Spectral Element Methods". *Archives of Mechanics* 64.4 (2012), pp. 423–456.

[20] M. Schonherr et al. "Multi-thread implementations of the lattice Boltzmann method on non-uniform grids for CPUs and GPUs". *Computers and Mathematics with Applications Volume 61, Issue 12* (2011).

[21] F. Schornbaum and U. Rude. "Massively Parallel Algorithms for the Lattice Boltzmann Method on non-uniform Grids". *SIAM Journal on Scientific Computing* 38(2) (2015), pp. 96–126. DOI: 10.1137/15M1035240. URL: http://arxiv.org/pdf/1508.07982.pdf.

[22] M.C. Sukop and D. T. Thorne. *Lattice Boltzmann Modeling. An Introduction for Geoscientists and Engineers.* Springer, 2006.

[23] R Core Team. *R: A language and Enviroment for Statistical Computing.* 2013. URL: https://www.r-project.org/.

[24] S. Wolfram. "Cellurar Automata Fluid 1: Basic Theory". *Journal of statistical physics* 45.3-4 (1986), pp. 471–526.

[25] Q. Zou and X. He. "Pressure and Velocity boundary conditions for the lattice Boltzmann". *Physics of Fluids* 9.6 (1997), pp. 1591–1598.